HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY FOR HIGH QUALITY TRAINING

GRADUATION THESIS

RESEARCH AND DESIGN A DRIVER DROWSINESS DETECTION SYSTEM

STUDENT: MAI ĐUC TUNG

STUDENT ID: 19145187

STUDENT: PHAN QUOC BAO

STUDENT ID: 19145136

MAJOR: AUTOMOTIVE ENGINEERING

SUPERVISOR: NGUYỄN THIỆN DINH, MS.

Ho Chi Minh City, July 2023

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY FOR HIGH QUALITY TRAINING

GRADUATION THESIS

RESEARCH AND DESIGN A DRIVER DROWSINESS DETECTION SYSTEM

STUDENT : MAI DUC TUNG

STUDENT ID: 19145187

STUDENT : PHAN QUOC BAO

STUDENT ID: 19145136

MAJOR: AUTOMOTIVE ENGINEERING

SUPERVISOR: NGUYỄN THIỆN DINH, MS.

Ho Chi Minh City, July 2023

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

Ho Chi Minh City, August , 2022

GRADUATION PROJECT ASSIGNMENT

Student name: Phan Quốc Bảo	Student ID: 19145136
Student name: Mai Đức Tùng	Student ID: 19145187
Major: Automotive Engineering Technology	Class: 19145CLA
Advisor: Nguyễn Thiện Dinh, MS	Phone number:
Date of assignment: February, 2023	Date of submission: July, 2023

1. Project title: Research and design a driver drowsiness detection

2. Initial materials provided by the advisor:

.....

.....

.....

3. Content of the project:

.....

.....

.....

4. Final product:

- 01 Graduation Thesis.
- Upload the materials, thesis, PowerPoints on Google Drive.

CHAIR OF THE PROGRAM	**ADVISOR**
(Sign with full name)	*(Sign with full name)*

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

Ho Chi Minh City, July, 2023

ADVISOR'S EVALUATION SHEET

Student name: Phan Quốc Bảo Student ID: 19145136

Student name: Mai Đức Tùng Student ID: 19145187

Major: Automotive Engineering Technology

Project title: Research and design a driver drowsiness detection system.

Advisor: Nguyễn Thiện Dinh, MS.

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

6. Mark:……(*in words*:.....)

Ho Chi Minh City, July , 2023

ADVISOR

(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

Ho Chi Minh City, July , 2023

PRE-DEFENSE EVALUATION SHEET

Student name: Phan Quốc Bảo Student ID: 19145136

Student name: Mai Đức Tùng Student ID: 19145187

Major: Automotive Engineering Technology

Project title: Research and design a driver drowsiness detection system.

Name of Reviewer:.....

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

6. Mark:…….(*in words*:.....)

Ho Chi Minh City, July , 2023

REVIEWER

(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

EVALUATION SHEET OF
DEFENSE COMMITTEE MEMBER

Student name: Phan Quốc Bảo Student ID: 19145136

Student name: Mai Đức Tùng Student ID: 19145187

Major: Automotive Engineering Technology

Project title: Research and Design a driver drowsiness detection system.

Name of Defense Committee Member:

.....

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

5. Mark:…….(*in words*:.....)

Ho Chi Minh City, July , 2023

COMMITTEE MEMBER

(Sign with full name)

Disclaimer

The authors, Mai Duc Tung and Phan Quoc Bao confirm that the work presented in this thesis is my own.

All the data and statistics in the thesis are reliable and are not published in any previous studies or research. Where information has been derived from other sources, I confirm that it has been indicated in the thesis.

Acknowledgements

Firstly, we would like to express our tremendous gratitude to all the teachers of High Quality Training faculty who gave us lots of lessons not only about major knowledge but also soft knowledge.

Besides, we are so grateful to our supervisor, MSc Nguyen Thien Dinh for his significant guide and support during our thesis. He gave us useful advice for our thesis and helped us to propose a paper that is very meaningful to us.

Our friends and our colleagues also contributed a lot to our project. Without their help and encouragement in challenging situations, we cannot finish our work on time.

Throughout the thesis, we learned and achieved much knowledge which would become a strong base for our future. However, there are still errors existing in our work due to limitations in experience. We are looking forward to hearing feedback from the council to improve our report.

Finally, we would like to say thanks to our parents who gave us the opportunity to finish our study journey in HCMUTE.

Student

Mai Đức Tùng

Phan Quốc Bảo

Contents

List of Figures

List of Tables

List of Acronyms

CAN – Controller Area Network

ECU – Electronic Control Unit

OBD – Onboard Diagnostics

FPS – Frame per second

EAR – Eye Aspect Ratio

PERCLOS – Percentage of Eyelid Closure over the pupil

IDE – Integrated Development Environment.

Abstract

Due to multiple reports all around the world, drowsiness driving is a contributing factor to many crashes and fatalities each year. Therefore, an intelligent system implemented in vehicles is necessary to avoid this status. This project illustrated a module named “Driver Drowsiness Detection” that is auto-activated if the vehicle is running over a certain speed. With the appearance of “Driver Drowsiness Detection”, the drivers will be alerted in situations that are supposed to be drowsy or sleeping which can lead to a dangerous accident. This is a vision-based system and retrieves signal directly from automobiles network via OBD2 connector for additional functions we deployed in this project. The system is modelized with some electrical device and mainly processed in an embedded computer as an actual application. When the drowsy driver is detected according to the closure of driver’s eyes, the Driver drowsiness detection will alarm by an audio voice and text notification. The system operates properly especially in daytime with a certain rate of accuracy but need to be improved and developed so that it is consistent and accurate any different environment.

INTRODUCTION

1. Reasons to choose the topic

Technology has become vital in every aspect of our lives and it is true for the automobile industry. More and more systems are designed and implemented in traffic vehicles for better operations and safety functions at many levels such as ABS(Anti-locking brake system), ESP(Electronics Stability Program), Pre-collision Warning, Lane Keeping and so on. The advancement of technology has provided us with the means to develop effective detection systems. With the advent of machine learning algorithms, computer vision, and physiological sensors, it is now possible to accurately identify signs of driver drowsiness. By leveraging these technological advancements, we can create a system that detects early signs of fatigue and alerts the driver, potentially preventing accidents and saving lives.

Driver drowsiness is a critical safety concern on the roads. Fatigue-related accidents have proven to be a significant cause of injuries and fatalities worldwide. A drowsy driver is more prone to lapses in attention, slower reaction times, and impaired decision-making, increasing the risk of accidents. By developing a driver drowsiness detection system, we aim to address this pressing issue and contribute to road safety. According to National Traffic Safety Committee, drowsy-related traffic accidents account for 30% of all traffic accidents in a year. 3.354/5.637 traffic accidents in first 6 months in 2022 resulted from driver drowsiness and fatigue [1]. The impacts of such events are very tremendous and serious in regards of economics and society. The current solution suggested by experts is building more rest areas meeting both quantity and quality for drivers.

The urgency of this topic is emphasized by the increasing number of incidents related to driver drowsiness. Long working hours, demanding schedules, and the prevalence of shift work contribute to a higher likelihood of fatigue among drivers. Additionally, the rise in mobile device usage and distractions further exacerbate the risks associated with drowsy driving. Addressing this issue promptly is crucial to ensure the safety of drivers and passengers alike. Legal and regulatory bodies are recognizing the importance of driver drowsiness detection systems. Many countries have implemented or are considering legislation mandating the use of such systems in commercial vehicles with EU General Safety Regulation (GSR), all new vehicles must have a driver drowsiness detection system from July 2024 [2]. By proactively researching and designing an effective solution, we can contribute to compliance with these regulations and facilitate the adoption of drowsiness detection technology in the automotive industry.

Therefore, we decided to choose the topic: “Research and design driver drowsiness detection system” to propose a solution for the mentioned problems.

2. Research object

The thesis is carried out with following purposes:

- Understand an overview application of computer vision in automobiles.
- Take advantage of signals from OBD2 gate in automobiles.
- Understand communication of embedded computer and electrical devices.
- Use programming language to build an intelligent application.
- Validate an external system in a particular automobile model.

3. Research area

Our thesis is limited with warning alarm for drivers but not interfere to any internal system of vehicles. Besides, we only apply an available AI model but not build a new one.

4. Estimate result

- Understand the principles of computer vision and image processing applications in automobile industries.
- Understand drowsiness detection methods based on driver behavior and vehicle state.
- Understand the operation of CAN bus communication and retrieve signal from OBD2 for external usage.
- Design an external system to test and validate results.
- Being a base for further project in the future about similar idea.

5. Research method

- Research on published paper and thesis.
- Research through collected online document.

6. Structure of the thesis

In order to achieve the mention objectives, the thesis was organized with the following contents:

Chapter 1: Overview

Chapter 2: Basic theory

Chapter 3: Drowsiness detection method

Chapter 4: Hardware and software overview

Chapter 5: Experiment and results

Chapter 6: Conclusion and recommendation

CHAPTER 1: OVERVIEW

1.1 Domestic research

Due to a graduation thesis of Do Van Linh-a student in HCMUTE, the software is executed in Lab-view platform and divide different level of drowsiness alert according to velocity [3]: The researcher consulted theories on face recognition and its applications, as well as studied sleep warning systems in luxury car manufacturers. They analyzed traffic accident statistics and causes. The approach involved studying LabVIEW programming and face recognition, incorporating facial recognition into driving essentials, and collecting relevant information. The researcher synthesized and designed a demo model for the system. The results included one testable demo model for vehicles and accumulated research materials during the thesis implementation.

Another thesis of Thai Thi Hoa Van in Da Nang University focus mainly the algorithm of drowsiness detection with Haar Cascade method [4]: This thesis focuses on addressing driver fatigue and sleepiness problems by monitoring the driver's eye state using human face recognition. The project aims to contribute to practical applications in Vietnam, where fatigue-related accidents are common in long-distance transportation. The missions include studying face recognition methods, eye and mouth state monitoring algorithms, and developing fatigue detection algorithms using Python and OpenCV. The theoretical aspects involve exploring general approaches to solving driver fatigue issues, face recognition methods using OpenCV, and algorithms for eye and face detection. In practice, the project involves creating a demo program for detecting driver drowsiness from videos or live camera feeds.

The master thesis of Nguyen Dinh Quan in HCMUTE applied the EEG signals to detect drowsy drivers [5]: This thesis focuses on Brain-Computer Interface (BCI), which enables the interaction between the human brain and computers. The BCI system utilizes electrical signals from the brain, processed by computer programming, to facilitate cognitive activities and control peripheral devices. The main objective is to design a real-time BCI system with an application that detects drowsiness using simulated brain waves and alerts the driver. EEG signals are collected using the Emotiv EPOC headset for user identification and warnings.

1.2 Foreign research

Robert Chen-Hao Chang, et al [6] introduced a system based on a combination of percentage of eyelid closure over the pupil over time (PERCLOS) and Facial Physiological Signal: This study proposes a drowsiness detection system that combines heart rate

variability (HRV) analysis and percentage of eyelid closure over time (PERCLOS) to enhance accuracy and robustness. The algorithm performs LF/HF ratio from HRV status judgment, eye state detection, and drowsiness judgment. A near-infrared webcam is used for non-contact measurement and to overcome limitations of wearable devices and low-light conditions. The appropriate RGB channel is selected for HRV analysis. The drowsiness detection system achieves a sensitivity of 88.9%, specificity of 93.5%, positive predictive value of 80%, and system accuracy of 92.5% using 10 awake and 30 sleepy samples. Electroencephalography is used for validation of the proposed method's reliability.

Anjith George [7] designed and implemented a real time algorithm for eye tracking and PERCLOS measurement to estimate alertness of drivers on a single board computer as a main processor even at night with a support of near-infrared LED.

Tayyaba Azim, et al [8] proposed an automatic fatigue detection of drivers through yawning analysis. The face is detected through Viola-Jones method in a video frame. Then, a mouth window is extracted from the face region, in which lips are searched through spatial fuzzy c-means (s-FCM) clustering : This paper presents a non-intrusive fatigue detection system based on video analysis of drivers, specifically targeting the detection of yawning as an indicator of fatigue. The system utilizes face detection and mouth region extraction techniques to identify and analyze the degree of mouth openness. Persistent yawning triggers an alarm to alert the non-vigilant driver. Real data experiments were conducted under different lighting conditions, race, and gender to validate the system's performance.

Jyotsna Gabhane, et al [9] designed an anti-drowsiness goggle for driver. The IR sensor on the google is used to check eyes closure time and the buzzer will blow if closure time exceed set-up threshold : It emphasizes the impact of drunk driving accidents on company owners, leading to liability and potential economic losses. The proposed solution introduces an adaptive driver and company owner alert system, along with a driving behavior application, to mitigate these risks.

1.3 Commercial product

1.3.1 Smart Eye’s driver monitoring system

Smart Eye’s Driver Monitoring System software has been installed in more than 1,000,000 cars on roads globally – saving lives every day[10]. Smart Eye’s Driver Monitoring Systems uses sensors, such as in-car cameras, computer vision, and artificial intelligence to bring insight into the driver’s state and behavior.

Figure 1.1: Smart Eye's Monitoring system

Smart Eye's AI-based DMS technology enables a wide variety of features for improved road safety and driver convenience powered by Affectiva's Emotion AI to capture nuanced emotions, reactions, and facial expressions in real time.

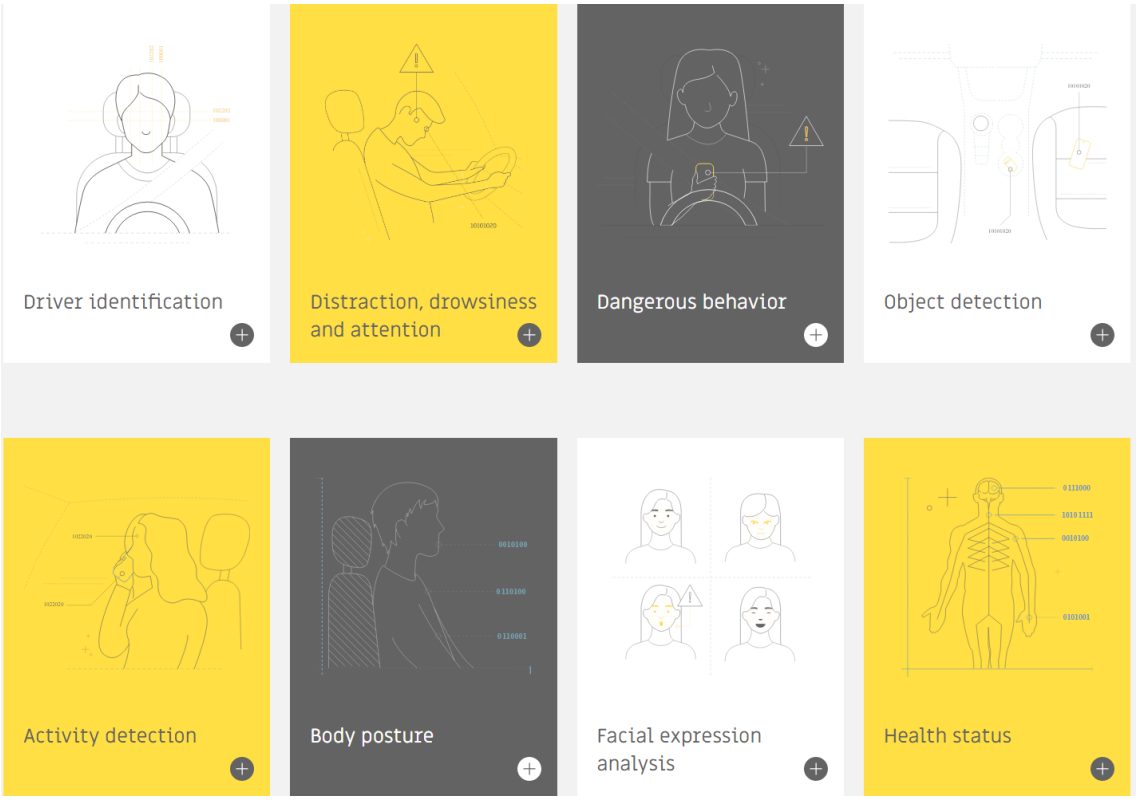

Figure 1.2: Smart Eye's Monitoring system modules.

1.3.2 Bosch interior monitoring system

The interior monitoring systems from Bosch increase safety for all vehicle occupants by using innovative sensor systems for the vehicle interior, critical situations such as

distraction and drowsiness so that they are detected at an early stage and the driver is warned accordingly [11]. The system consists of following modules:

The cabin sensing radar recognizes the presence of living creatures in the vehicle based on extremely small movements and vital signs. The radar locates the occupants in the vehicle combined with the camera-based data to enable passive safety systems such as seat belt reminders and automatic airbag suppression. The radar is usually mounted in the overhead console of the roof lining.

Driver monitoring camera: applied scientific criteria such as gaze direction, eye opening, and posture to detect whether the driver is distracted or drowsy with the help of artificial intelligence. The camera can be installed in different locations in the cockpit such as the steering column, in the central display or on the A-pillar.

Figure 1.3: Bosch’s camera setup

Occupant monitoring camera provides a larger view angle and also captures occupants in the passenger and rear seats. The camera also detects subjects such as a phone in the driver’s hand or a handbag left on the rear sear seat to execute corresponding warnings.

The steering angle sensor: the drowsiness detection algorithm analyzes the driver’s steering behavior and detects changes from long journey times and driver drowsiness. Even minor changes in steering behavior can indicate the signs of diminishing concentration. The frequency of these steering corrections and other parameters such as journey duration, turn signal activation and time of day are used to calculate a fatigue index. If this index exceeds a specific value, the driver is warned for a break in the display instrument.

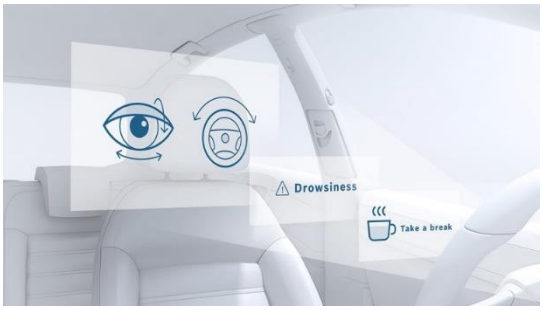

Figure 1.4: Bosch drowsiness detection

1.3.3 Mercedes's Attention Assist

Attention Assist analyzes driving behavior in the first few minutes according to 70 parameters. Then the system recognized drivers' drowsiness and fatigue due to complicated algorithms while considering external factors such as road condition, crosswinds, and interaction with vehicle controls.

The mentioned system activated at above 60km/h shows drive length on the display system and if the attention level is low, there are visual notification and audible warning. The driver can also set different modes of Attention Assist: Standard, Sensitive due to length of journey.

Figure 1.5: Mercedes Attention Assist display.

CHAPTER 2: BASIC THEORY

2.1 Introduction to computer vision

Computer vision[12] is a section of AI (Artificial Intelligence) that enables computers and system to obtain meaningful data from digital images, videos and other visual inputs to take actions or recommend based on acquired data.

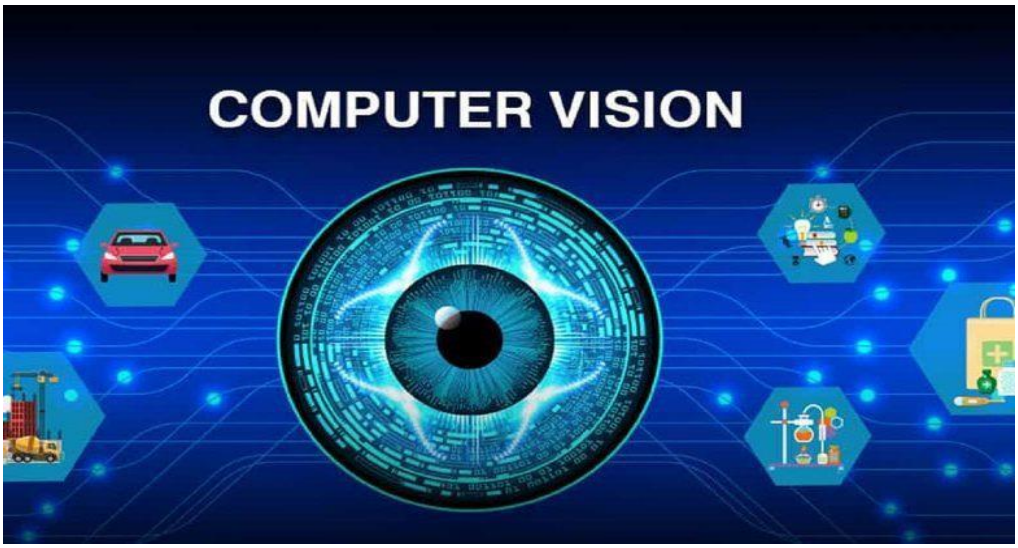

Figure 2.1: Computer vision applications

2.1.1 Computer vision principles

Computer vision involves the emulation of human visual perception, which comprises three distinct sequential stages analogous to the human visual process. These stages are acquisition (which involves the simulation of the eye and is considered challenging), description (which entails simulating the visual cortex and is deemed highly challenging), and understanding (which involves simulating the remaining aspects of the brain and is considered the most arduous task).

Acquisition, focusing on the simulation of the eye, has witnessed notable advancements. By developing sensors and image processors akin to the human eye, significant progress has been achieved. Modern camera technology enables the capture of thousands of images per second with remarkable accuracy over long distances. Nevertheless, even the most sophisticated camera sensors alone are incapable of autonomously detecting a ball. Hence, the primary hurdle lies in the limitations of hardware, emphasizing the paramount importance of software in this context.

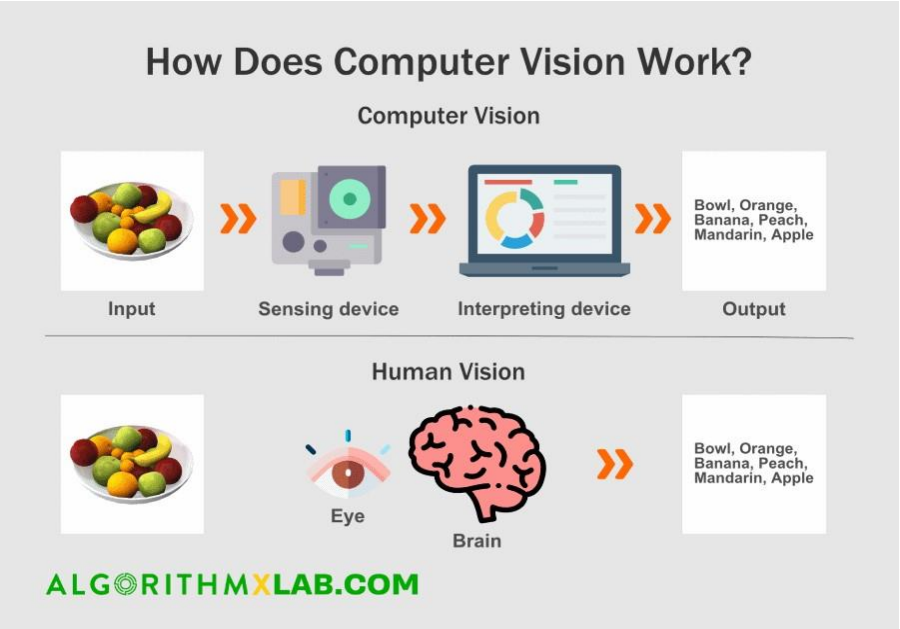

Figure 2.2: Computer vision principles

Description, the subsequent stage, encompasses an array of visual processes performed by the brain, predominantly at the cellular level. Billions of cells collaboratively engage in pattern recognition and signal processing. For instance, when a disparity along a line is detected (e.g., changes in angle, speed, or direction), a group of neurons promptly communicates this variance to another set. Initial research in computer vision indicated that neural networks exhibit an exceedingly intricate nature, rendering top-down comprehension and comprehensive object description infeasible. The immense difficulty arises from the need to describe objects from multiple perspectives, accounting for variations in color, motion, and other attributes. This endeavor becomes even more daunting when considering the substantial data requirements, even comparable to the cognitive capabilities of an infant. Consequently, a bottom-up approach, mirroring the intricacies of brain functioning, holds greater promise. Recent years have witnessed a surge in research and implementation of such brain-inspired systems, with the process of pattern recognition continually advancing and yielding further progress.

Understanding, the final stage, entails the development of a system capable of recognizing an apple from any angle or in diverse scenarios, whether static or in motion. However, the system's capabilities fall short when it comes to recognizing an orange or providing a comprehensive definition of what an apple is, including aspects such as edibility, size, and usage. To address this limitation, a complete system necessitates an operational framework analogous to the remaining functionalities of the brain. This includes the integration of short-term and long-term memory, data derived from sensory inputs, attention mechanisms, perception abilities, and knowledge accumulated through interactions with the surrounding environment. These intricate components operate within a complex

network of interconnected neurons, surpassing any existing computational architecture in complexity.

2.1.2 Applications

Defect detection: Among the various applications of computer vision, the detection of defects holds considerable prominence. Traditionally, the identification of faulty components relied on the expertise of designated supervisors, thereby limiting their ability to oversee an entire system's operational process. However, computer vision has revolutionized this domain by enabling comprehensive scrutiny for even the minutest imperfections, including metal cracks, paint defects, and substandard prints, all of which exhibit sizes smaller than 0.05mm.

Autonomous operation: The realm of autonomous vehicle technology has witnessed significant advancements in recent years. Through the utilization of artificial intelligence (AI), vast quantities of data collected from millions of drivers have been analyzed, enabling the acquisition of insights from driving behaviors. This invaluable information facilitates the automatic identification of lanes, estimation of road curves, detection of potential hazards, and interpretation of signals and traffic signs. Such sophisticated AI algorithms have played a pivotal role in enhancing the autonomous capabilities of vehicles, thereby propelling the field forward.

Figure 2.3: Object detection in automobile

Fire detection: The multifaceted application of computer vision extends to the realm of security, wherein drones, or Unmanned Aerial Vehicles (UAVs), can exploit computer vision systems to augment human capacity in identifying wildfires through the utilization of infrared (IR) images. Leveraging advanced algorithms, these systems scrutinize characteristics within video images, such as motion and brightness, to detect the presence of fire. By employing targeted extraction methods, these algorithms facilitate the

discernment of distinctive patterns and enable differentiation between fires and other types of movement that may be prone to misinterpretation as fire occurrences.

Facial recognition: The inception of facial recognition technology dates to 2011 when Google showcased the possibility of developing a face detector solely using unlabeled images. This breakthrough innovation entailed designing a system capable of autonomously learning to detect images of cats without explicit instructions regarding feline features. In contemporary times, smartphones equipped with high-quality cameras have harnessed the power of computer vision for identification purposes. Within the security sector, computer vision techniques are employed to identify criminals and forecast crowd movements during emergency situations.

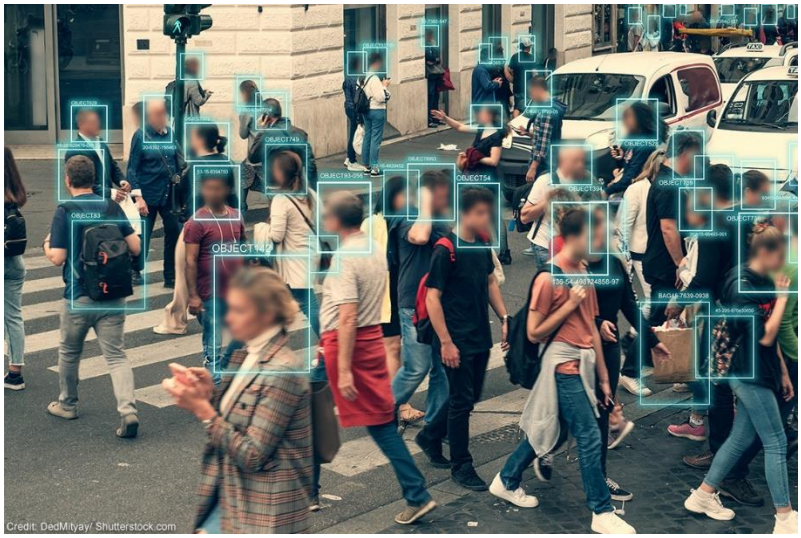

Figure 2.4: Object detection in social facilities

2.2. Introduction to image processing

Image processing is a subset of computer vision. It mainly focuses on processing the raw input images to enhance them or preparing them to do other tasks.

2.2.1 Image processing terminology

Picture element: Commonly referred to as a pixel, represents a minute color unit constituting a digital image. Each pixel possesses a specific geographic coordinate within the image, corresponding to a small fraction of the overall visual composition. The visual clarity of a photograph is directly influenced by the quantity of pixels it encompasses.

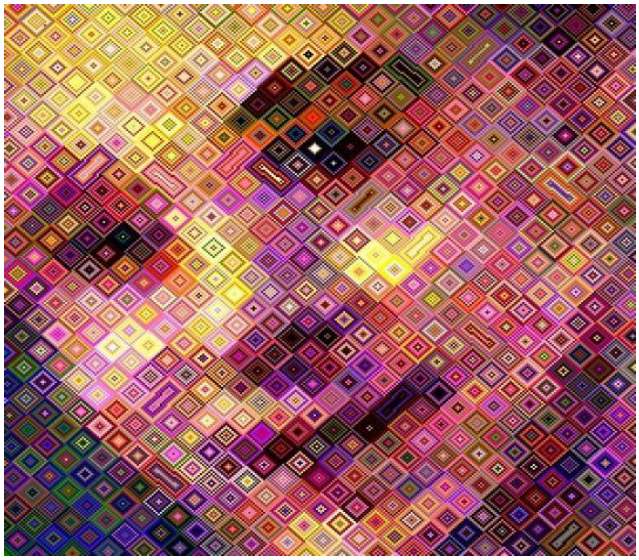

Figure 2.5: Image created from elements.

Image resolution: Serves as an indicator of the amount of information encompassed within an image file, as it pertains to the number of pixels comprising the display on a screen. The resolution of an image is conventionally measured in pixels or megapixels, with 1 megapixel equivalent to 1 million pixels. Each individual pixel occupies a relative size of 0.26x0.35 units. The depiction of pixels on a screen is quantified in terms of pixels per inch (PPI).

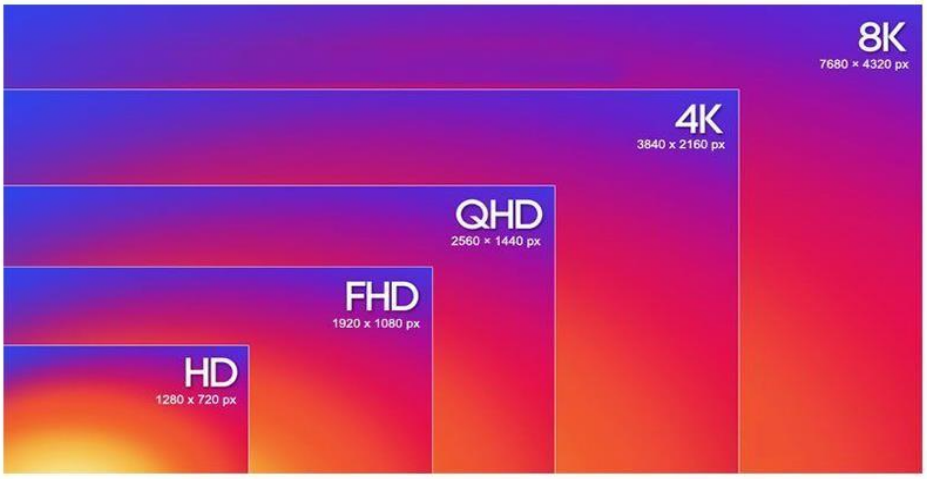

Figure 2.6: Image resolution comparison

The resolution of a photograph is denoted by the total number of pixels it comprises. Calculating the resolution of an image involves multiplying the number of pixel columns by the number of pixel rows, followed by dividing the resultant product by one million. For instance, an image with dimensions of 1920x1080 encompasses a total of 2,073,600 pixels, corresponding approximately to 2 megapixels.

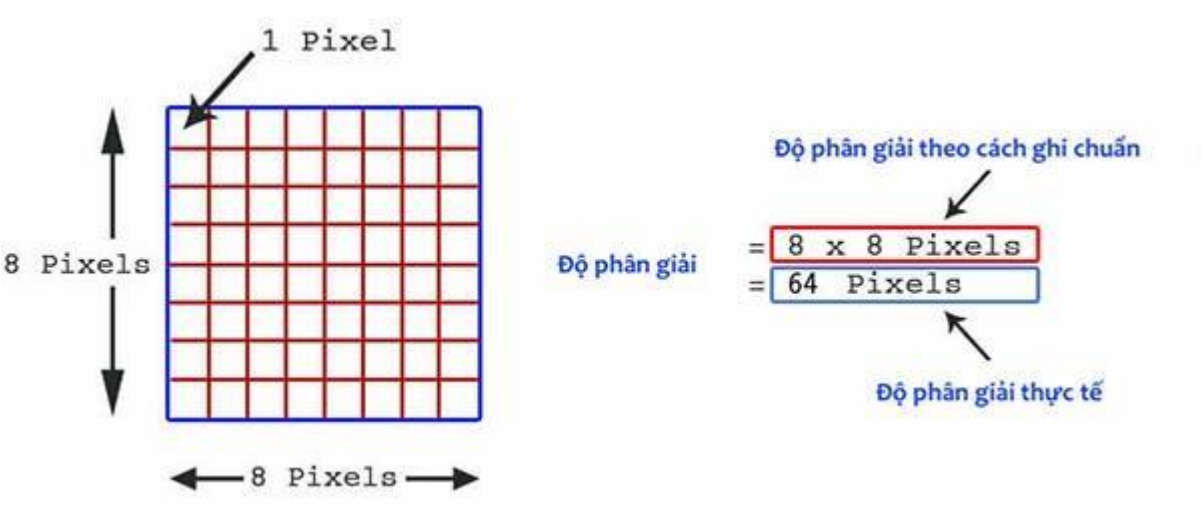

Figure 2.7: Resolution calculation

Gray level of the image: pertains to a monochromatic representation, commonly referred to as a grayscale image. In such images, each pixel assumes a value ranging from 0 to 255. A pixel value of 0 signifies a dark or black pixel, while a value of 255 indicates a light or white pixel. The gray level attribute of an image enables the depiction of varying shades of gray, representing different levels of brightness within the image.

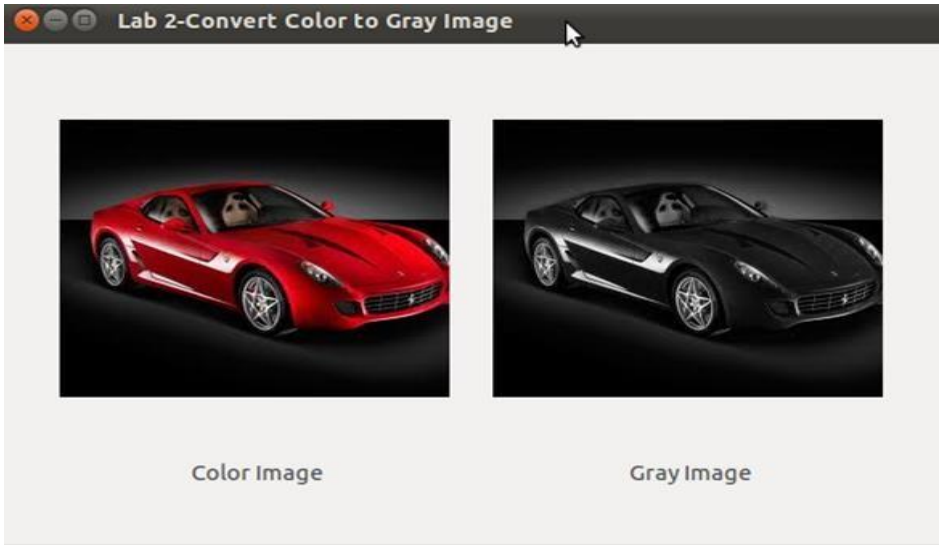

Figure 2.8: Image after converted to gray image

A binary image is a digital image, where each pixel is represented by a value of 0 (white) or 1 (black).

Figure 2.9: Binary image

The creation of a binary image involves the conversion of a grayscale image based on a predefined threshold. The process typically entails selecting a specific threshold value, against which the grayscale image is compared. The resulting binary image classifies pixels into two categories based on this threshold: pixels with values below the threshold are designated as white, while pixels with values above the threshold are designated as black (or vice versa, depending on the chosen convention). This transformation facilitates the segmentation of regions of interest within the image, enhancing the distinction between foreground and background elements.

Figure 2.10: Three types of images

Color channel: Color channels are fundamental elements in the representation of colors within digital images. They play a vital role in conveying and manipulating the diverse hues and tones present in an image. In the realm of digital imaging, color channels refer to the separate intensity values that define the contribution of each primary color, namely Red, Green, and Blue, to the overall color composition of an image.

Color channels facilitate the analysis and decomposition of color information contained within an image. In the prevalent RGB color model, which finds extensive usage in digital

imaging, each pixel comprises three distinct color channels: red, green, and blue. These channels determine the intensity of their respective primary color at that pixel. Through the combination of different intensities across these three-color channels, a broad spectrum of colors can be faithfully represented and reproduced on digital displays.

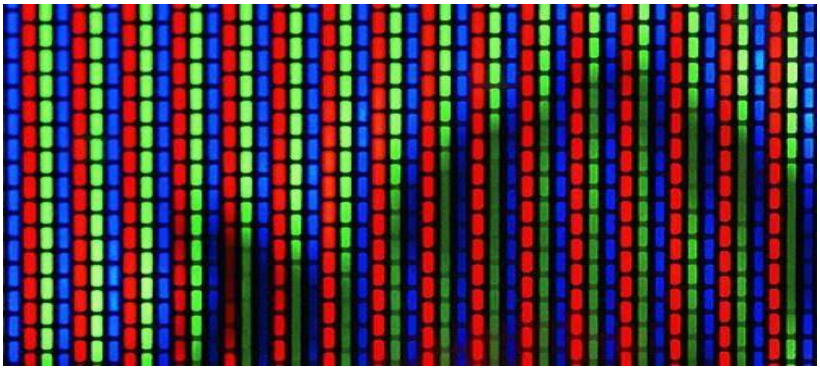

Figure 2.11: Pixel in RGB channel

Each color channel represents the luminance or brightness value associated with its corresponding primary color. The red channel indicates the intensity of red within the image, the green channel signifies the intensity of green, and the blue channel denotes the intensity of blue. Together, these three channels form the foundation for representing and reproducing the entire gamut of colors within an image.

Color channels provide flexibility in image processing and editing endeavors. They permit selective adjustments of individual color components, enabling enhancements such as color correction, color grading, and the application of effects based on specific channels. By manipulating the intensity values of color channels, it becomes possible to alter the overall color balance, emphasize or de-emphasize specific colors, or create captivating visual effects.

Beyond the RGB color model, alternative color spaces and models may employ varying sets of color channels. For instance, the CMYK color model, widely utilized in print and graphic design, employs four color channels: cyan, magenta, yellow, and black. Similarly, the HSV (Hue, Saturation, Value) color model employs distinct channels to represent key aspects of color, including hue, saturation, and brightness.

Proficiency in comprehending and working with color channels is vital in diverse fields such as photography, computer graphics, image processing, and computer vision. By manipulating color channels, professionals can finely adjust the appearance and quality of images, extract specific color information, and undertake advanced techniques for image analysis and manipulation.

HSV channel:(Hue, Saturation, Value) channel is extensively employed in image editing, image analysis, and computer vision applications. This color space utilizes three key parameters to describe colors:

H(Hue) :It represents the color itself, referring to the specific color region within the color model. Hue is typically represented as a numerical value ranging from 0 to 360 degrees, encompassing the entire color spectrum. This parameter enables the identification and differentiation of distinct hues, such as red, blue, green, etc.

Color	Angle
Red	0-60
Yellow	60-120
Green	120-180
Cyan	180-240
Blue	240-300
Deep Red	300-360

Table 2.1: Color angle distribution

S (Saturation): The parameter of saturation (S) in the HSV (Hue, Saturation, Value) color channel signifies the degree of vividness or intensity of a color and is expressed as a value ranging from 0 to 100 percent. By manipulating the saturation level, one can introduce varying amounts of gray within a color. A lower saturation value approaching zero yields a desaturated effect, resulting in the introduction of more gray tones. Alternatively, in certain contexts, saturation can be interpreted within a range of 0 to 1, where 0 represents grayscale and 1 represents the purest form of the primary color.

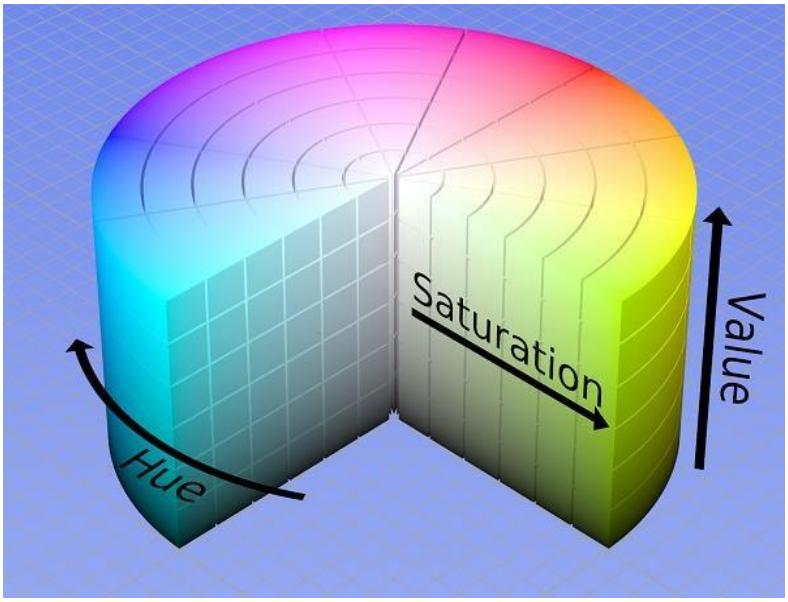

Figure 2.12: HSV channel

CMYK color space: The CMYK color space is a prevalent color model employed in the field of printing and graphic design. It is rooted in the subtractive color model, which entails the blending of different pigments to create a wide spectrum of colors. CMYK refers to the four primary ink colors utilized in printing: Cyan, Magenta, Yellow, and Key (Black).

Within the CMYK color space, each color is defined by the proportion of ink in each primary color component. Cyan represents the blue-green hues, Magenta embodies the purplish-red tones, Yellow signifies the yellows, and Key (Black) denotes the black shade. By manipulating the percentages of these primary colors, an extensive array of colors can be achieved.

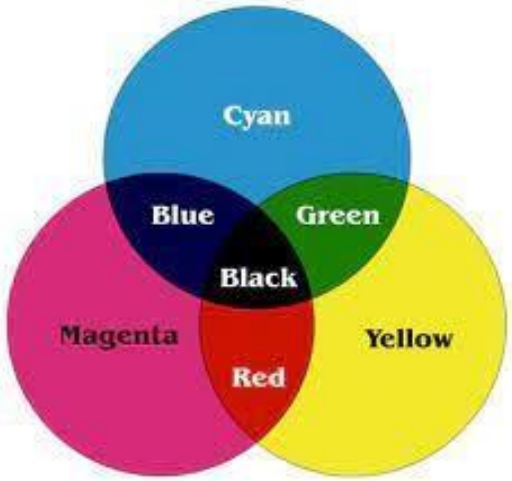

Figure 2.13: CMYK channel

The CMYK color space finds its primary application in printing due to its effectiveness in reproducing a broad gamut of colors on printed materials. It provides meticulous control over color reproduction and ensures faithful representation of colors in the final print. The inclusion of the black (Key) component enhances contrast and detail in printed materials while reducing excessive ink consumption.

It is essential to distinguish the CMYK color space from the RGB color space commonly employed in electronic displays like computer monitors and televisions. RGB employs the primary colors Red, Green, and Blue in an additive color model.

In essence, the CMYK color space serves as a subtractive color model extensively employed in the printing industry. By combining the primary ink colors of Cyan, Magenta, Yellow, and Key (Black), it offers a comprehensive range of colors. This color space guarantees accurate color reproduction in printed materials, affording precise control over color appearance in the final output.

Definition of digital image: A digital image is a representation of visual information that is encoded and stored in a digital format. It consists of a two-dimensional grid of discrete elements called pixels, which collectively form the visual content of the image. Each pixel

corresponds to a specific color or grayscale value, capturing the intensity and chromatic information of the image. Digital images are typically generated through the process of image acquisition, where analog signals from a camera or scanner are converted into digital data using analog-to-digital conversion techniques. Once in a digital form, images can be stored, transmitted, and processed using a wide range of computational algorithms and methods. Digital images have found extensive applications in various fields, including photography, computer graphics, medical imaging, and scientific research, facilitating the capture, analysis, and representation of visual data with precision and flexibility.

Black and white images, in contrast, are limited to a binary representation comprising only two colors: black and white. These images are characterized by dividing the grayscale spectrum into L discrete levels. When employing an 8-bit encoding scheme (B = 8 bits) to represent the black and white levels, the relationship between the number of levels (L) and the number of bits is governed by the formula: L = 2^B (resulting in L = 2^8 = 256)

Frame per second (FPS): is the frequency of consecutive images captured or displayed.

2.2.2 Image processing overview

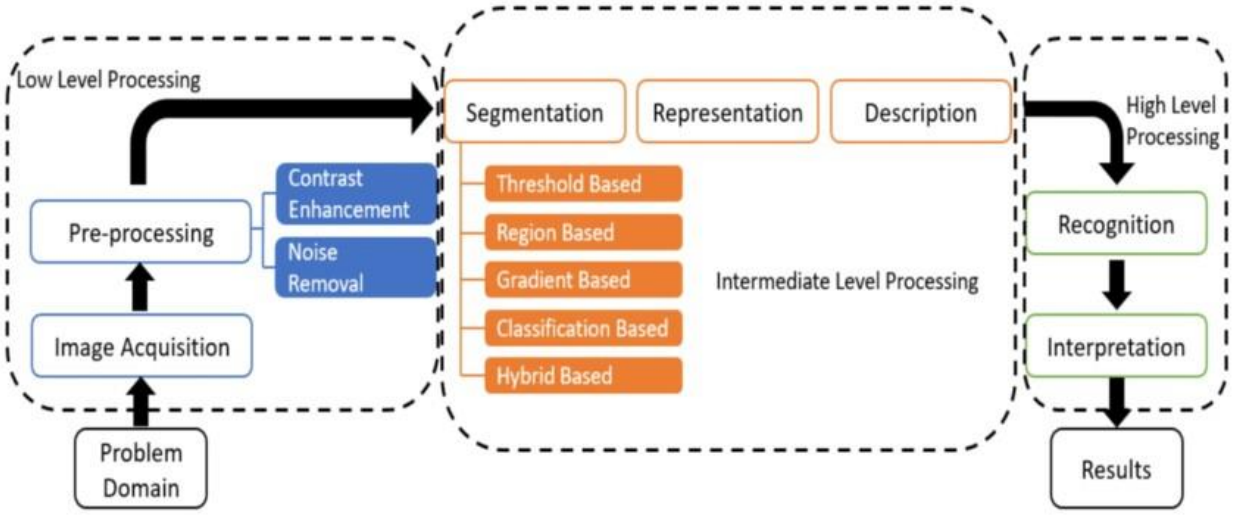

Figure 2.14: Image processing overview

Image Acquisition

Image acquisition serves as the initial stage in the realm of image processing. This step involves the utilization of a sensor, such as a camera, to capture the image, which is subsequently digitized if the output of the sensor is not already in digital format, achieved through the employment of an analog-to-digital converter.

Image preprocessing

Following image acquisition, image preprocessing plays a crucial role in enhancing the quality of the acquired image, particularly in scenarios where low contrast noise may be

present. The primary objective of the preprocessing stage is to filter out noise and amplify contrast, thereby yielding a clearer and sharper image.

Image segmentation:

Image segmentation entails the intricate process of partitioning an input image into distinct regions, thereby facilitating subsequent analysis and image recognition endeavors. For instance, in the context of mail classification, it becomes imperative to segment sentences, words, addresses, or individuals' names into separate units, such as words, letters, or numerical barcodes, to enable accurate identification. This segmentation phase represents the most intricate aspect of image processing, encompassing inherent complexities that may lead to errors and a loss of image precision.

Image presentation:

Once segmentation is completed, the resultant image output comprises the pixels pertaining to the segmented regions along with associated code denoting neighboring areas. It becomes necessary to transform this data into an appropriate format to enable further computer-based processing. The process of selecting features that effectively represent an image is known as feature extraction. This selection process aims to separate image features by quantifying informational content or establishing a basis for distinguishing different feature classes within the received image. For instance, in character recognition tasks on envelopes, describing the characteristics of each character becomes instrumental in distinguishing one character from another.

Image Recognition and Interpretation:

Image recognition and interpretation encompass the identification of images, typically achieved through a comparative analysis against reference samples that have been previously learned or saved. Interpretation, in turn, involves the judgment of meaning based on the outcomes of identification. For instance, a sequence of digits and dashes on a letter envelope can be interpreted as a phone code. Numerous methodologies exist for classifying images, each offering unique approaches and techniques.

Knowledge base:

The knowledge base surrounding image processing is characterized by the intricate nature of image contours, brightness, and density. The abundance of pixels and the image capture environment introduce complexities such as noise. In various stages of image processing and analysis, mathematical methods are often employed to simplify the procedures and facilitate ease of processing. Moreover, there is a growing interest in emulating the human perception and processing of images, prompting the integration of human-like intellectual methods into various processing steps.

2.3 Human face recognition algorithm

2.3.1 Haar Cascade method

The Haar Cascade method [14] is an object detection algorithm that leverages machine learning. It was developed by Viola and Jones in 2001 and has since become widely used for detecting objects in images and video streams. This method is particularly effective in detecting objects with different orientations and sizes.

Firstly, the algorithm utilizes Haar-like features, which are rectangular features that capture intensity variations between adjacent regions of an image. These features can represent edges, corners, or other visual patterns. To efficiently compute these features, the algorithm transforms the original image into an integral image. This allows for fast calculations of pixel intensity sums over rectangular regions, which speeds up the feature computation process.

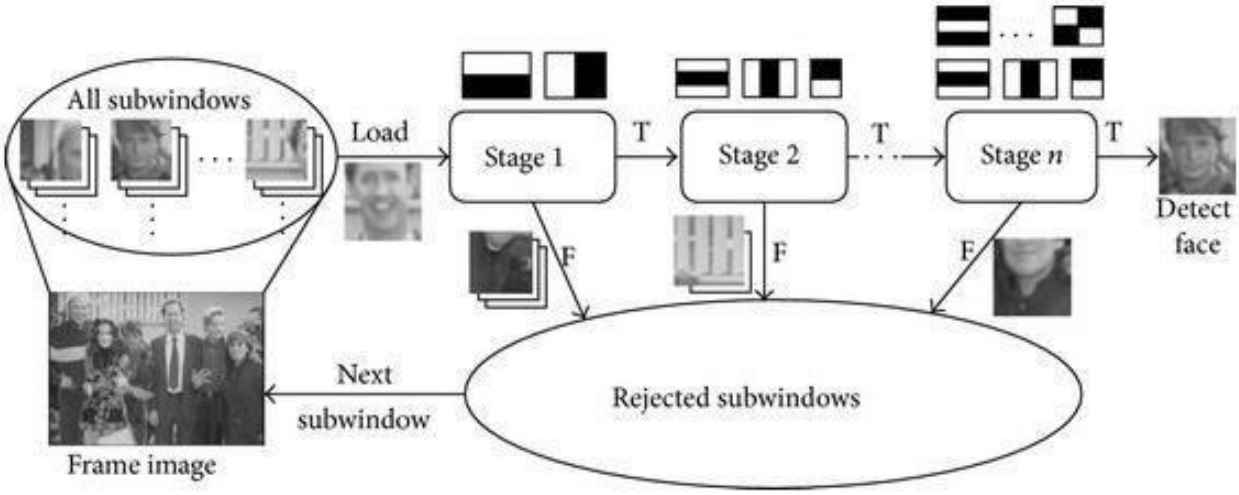

Figure 2.15: Process for recognizing a face using the Haar Cascade method

During the training phase, the algorithm learns the characteristics of the object to be detected. It uses a large dataset of positive images that contain instances of the object and negative images that do not. By combining multiple weak classifiers, a strong classifier is constructed using the AdaBoost algorithm.

AdaBoost is an iterative algorithm that trains weak classifiers on different subsets of the training data. It assigns higher weights to misclassified examples, which forces subsequent weak classifiers to focus on those examples. Ultimately, the weak classifiers are combined to form the final strong classifier.

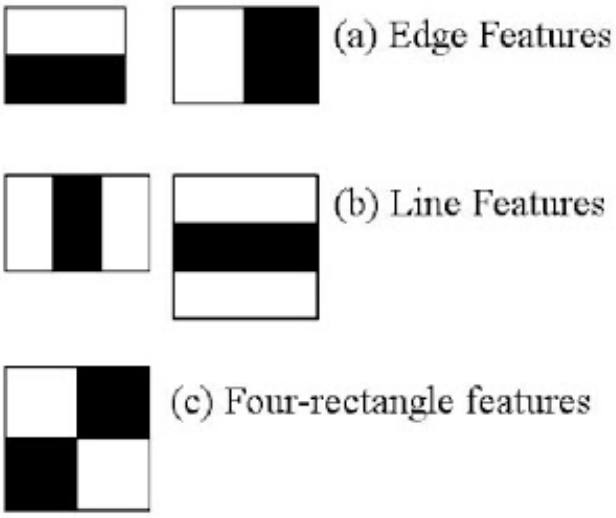

Figure 2.16: Based on the angles to determine the rectangle.

The strong classifier, known as a cascade classifier, consists of multiple stages, each containing several weak classifiers. During object detection, the cascade classifier analyzes different regions of the image at various scales. It progressively rejects regions that are unlikely to contain the object, which accelerates the detection process.

The Haar Cascade method adopts a sliding window approach, where a small window slides across the image at different scales. At each position, the cascade classifier evaluates the window to determine if it contains the object. This approach enables the detection of objects at various positions and scales within the image.

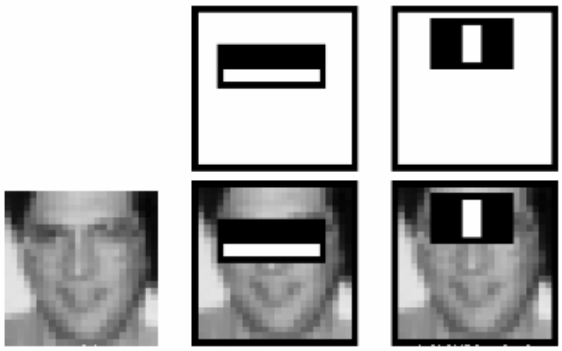

Figure 2.17: Human faces detection

When a potential object region is identified by the cascade classifier, further analysis is conducted to validate the detection. Additional filters or criteria may be applied to ensure accurate detection.

The Haar Cascade method has proven successful in various object detection tasks, such as face detection, pedestrian detection, and general object recognition. It strikes a balance between accuracy and computational efficiency, making it suitable for real-time applications.

The OpenCV library provides an implementation of the Haar Cascade method, enabling developers to utilize this technique for object detection in their applications.

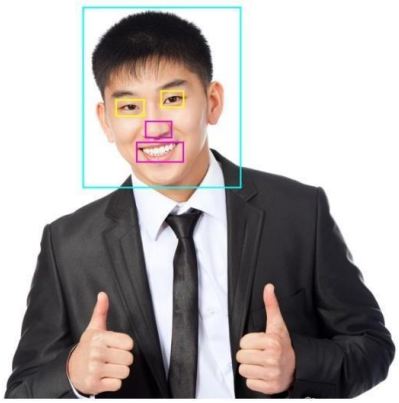

Figure 2.18: Face recognition results with different rectangles responsible for identifying eyes, nose and mouth objects.

2.3.2 Yolo method

The YOLO (You Only Look Once) method [15] is a widely used object detection algorithm in the field of computer vision. It takes a unique approach compared to traditional methods by treating object detection as a regression problem. Here's a breakdown of how the YOLO method works:

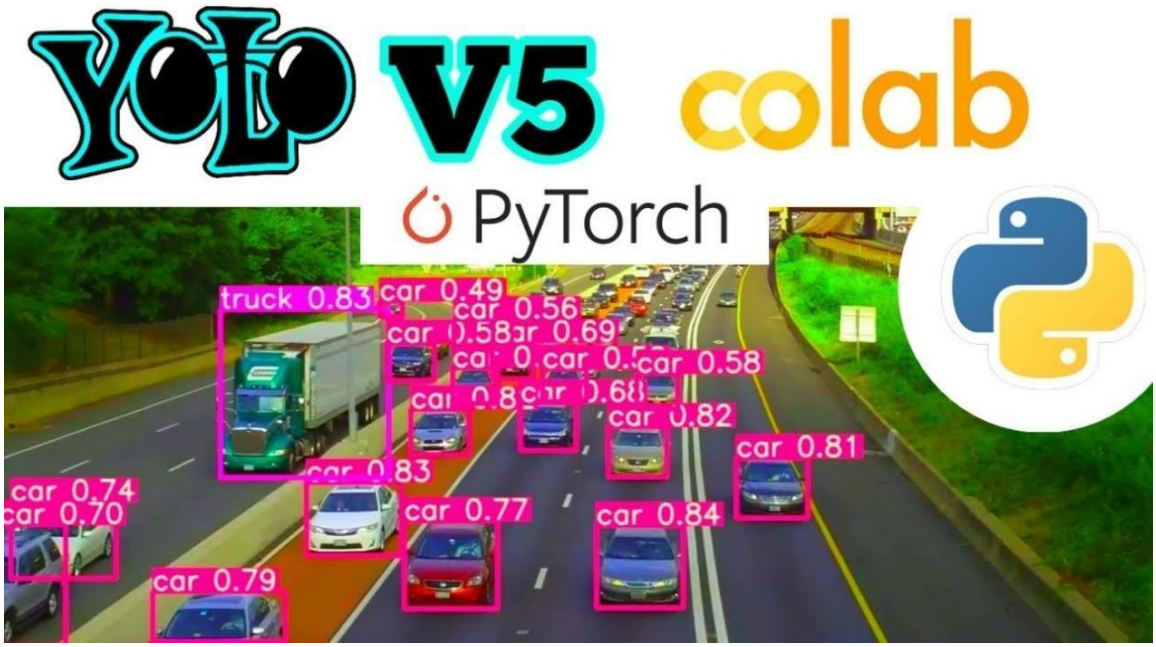

Figure 2.19: Yolov5 object detection

Instead of relying on sliding windows or region proposal techniques, YOLO divides the input image into a grid of cells. Each cell is responsible for detecting objects within its boundaries. To estimate object bounding boxes and class probabilities, YOLO utilizes anchor boxes. These predefined bounding boxes come in various shapes and sizes and are associated with specific object types.

Unlike multi-stage methods, YOLO performs object detection in a single pass. It predicts class probabilities and bounding box coordinates simultaneously for all objects within each cell. The prediction output of YOLO includes bounding boxes, class probabilities, and confidence scores. The confidence score reflects the likelihood of an object's presence within a bounding box, considering both the predicted class probability and bounding box accuracy. To enhance precision and eliminate duplicate detections, YOLO employs a technique called non-maximum suppression. This step suppresses overlapping bounding boxes based on their confidence scores, retaining the most accurate and confident detections.

Figure 2.20: Results after training for model YoloV5

During training, YOLO learns from a labeled dataset where objects are annotated with class labels and bounding box coordinates. The training process involves optimizing the network to minimize the difference between predicted and ground truth values. Over time, YOLO has evolved with different architecture variants, including YOLOv1, YOLOv2 (or YOLO9000), YOLOv3, and YOLOv4. These iterations introduced improvements in network architecture, feature extraction, and training strategies, leading to improved detection performance. Thanks to its real-time object detection capabilities and high accuracy, the YOLO method has gained significant popularity. It finds applications in diverse domains such as autonomous driving, surveillance systems, and video analysis.

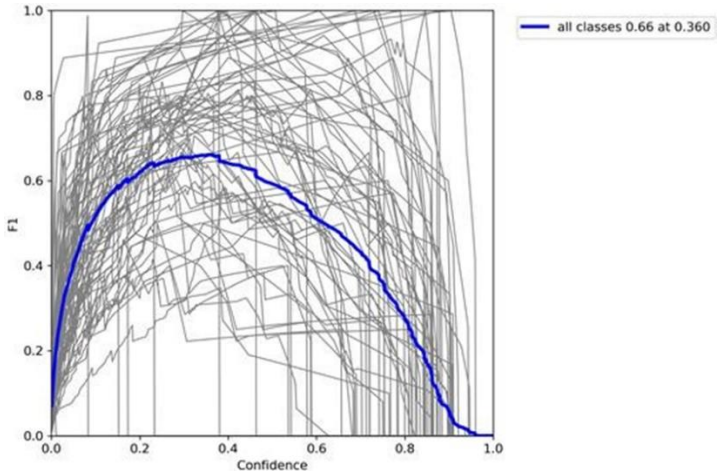

Figure 2.21: Graph results after training the YoloV5 model.

Open-source implementations of YOLO are readily available, enabling developers to leverage this algorithm for their object detection tasks. These implementations often come with pre-trained models on large-scale datasets, facilitating quick deployment and adaptation for specific applications.

2.1.3 Dlib algorithm

"Deep learning metric"[16] is the abbreviation for the face recognition model dlib. HOG (Histogram of Oriented Gradients) and SVM (Support Vector Machine) are two techniques that Dlib uses to improve recognition performance. can be used in real-time systems and has a relatively short running time. Recently, Dlib has added new features that make it simple for CNN network to recognize faces.

- Accuracy: Accuracy is a simple metric that measures the proportion of accurately predicted instances versus all instances. It provides a broad picture of model performance; however, it might not be appropriate for datasets with unbalanced classes.
- Precision: Out of all positive predictions made by the model, precision quantifies the percentage of real positive predictions. It aids in assessing the model's capacity to avoid false positives, which is especially important in situations when false alarms may have serious repercussions.
- Recall (Sensitivity/True Positive Rate): Recall calculates the proportion of accurate predictions among all occurrences of positive data that really occurred. It highlights the model's sensitivity in detecting the target class and shows how well it can recognize positive examples.
- F1 Score: The harmonic mean of recall and precision is the F1 score. It offers a fair evaluation by taking into account both metrics. The F1 score is particularly useful when working with datasets that are unbalanced since it takes false positives and false negatives into consideration.

- Area Under the ROC Curve (AUC-ROC): This statistic is frequently used for binary classification jobs. It stands for the receiver operating characteristic (ROC) curve's area under the curve. The true positive rate is plotted against the false positive rate on the ROC curve, which provides information on how well the model performs at various classification levels.
- Mean Average Precision (mAP): mAP is frequently used in tasks involving instance segmentation and object detection. It determines the mean across all classes after computing the average precision for each class. mAP provides a comprehensive evaluation of detection performance across many classes while accounting for both precision and recall.
- Mean Intersection over Union (mIOU): Semantic segmentation tasks frequently make use of mIOU. It calculates the average across all classes after measuring the overlap between expected and ground truth masks for each class. The effectiveness of segmentation outcomes is assessed by mIOU, which measures how accurately the model captures object boundaries.

The right metrics must be chosen based on the particular deep learning task at hand and the required evaluation standards. These metrics are essential in assisting academics and practitioners in comprehending the benefits and drawbacks of their models and in providing direction for improving performance.

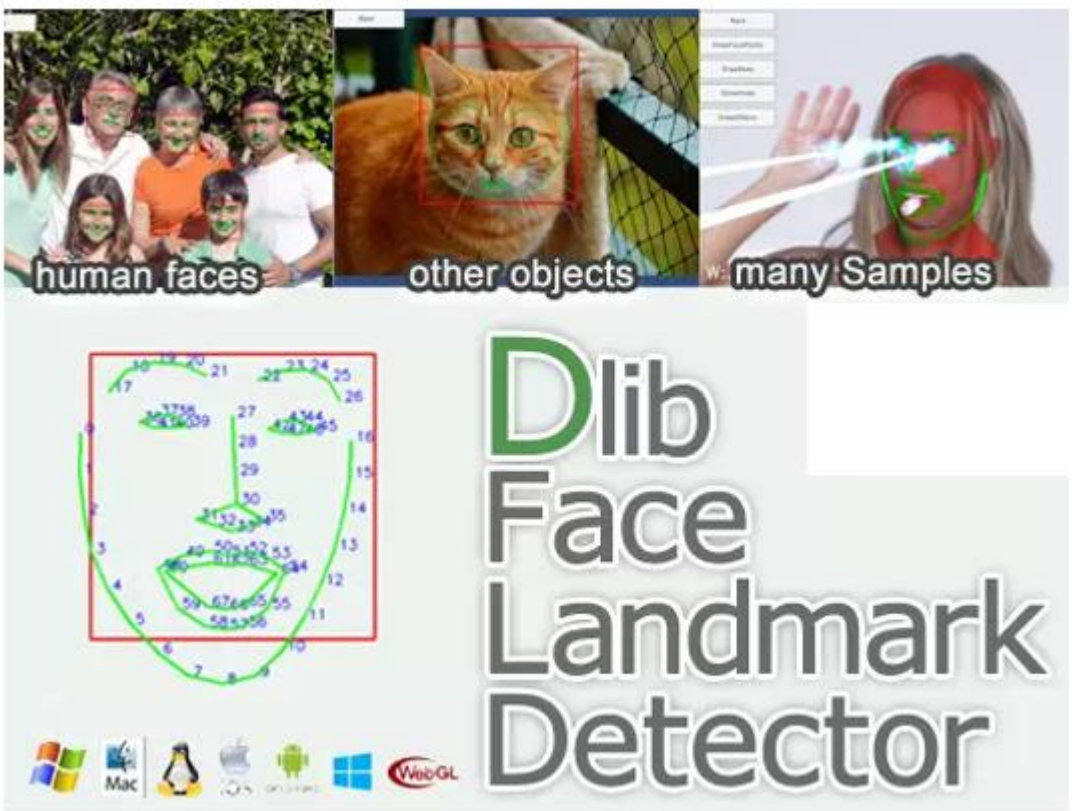

Figure 2.22: Dlib applications

Knowledge-based identification method: This method identifies a face in a picture by using human knowledge about faces (For example, a face in an image). will need to have eyes, a nose, and a mouth, and the space between them frequently needs to adhere to particular restrictions. The challenge with this method is that we will need to create a set of rules. The rule set won't operate if it's too broad or too restrictive because it will cause misidentification or non-recognition.

Feature-based technique: This method builds a model, then trains it to serve as a classifier to identify data. What parts of a face are visible in frames that were taken from the original picture. Timing is this technique's main flaw. This issue arises since the classifier requires us to extract several sections from one image.

The position of a face in an image is determined using a comparison between our provided face photographs (Feature template) and this approach called "Template-Matching," which crops the frames. Although Template-Matching is simple to apply, it shares the same timing issues as the Feature-Base approach.

Technique-based identification (Appearance-Base): Dlib will employ this method, which directly identifies the locations of sections in the image with faces by combining morphological techniques with analysis from a machine-leaning model. Later, we'll go into more detail regarding this approach.

2.2 Controller Area Network

2.2.1 CAN bus history.

The early beginnings of CAN-bus communication were presented by the German automotive company, Bosch, at the Society of Automotive Engineers congress in Detroit in 1986 [17]. Prior to this, vehicles primarily operated using a point-to-point wiring system. Each essential component was directly wired to the next. The increasing complexity of the sensors, actuators, and electronic control units (ECUs) present within modernized vehicles contributed to a high noise environment and added excess weight to the vehicle. By implementing the CAN-bus as a two-wired pair, the signals become more robust against the noisy environment and dramatically decrease the weight of the vehicle by simplifying the wired path between the various components [17].

2.2.2 CAN bus overview

A Controller Area Network (CAN) bus is a communication standard used to allow micro-controllers and other devices to communicate within a vehicle. The CAN-bus links various electronic control units (ECUs) within the vehicle and allows for more efficient and faster data transfer, in comparison to standard single strand transmission wires, by prioritizing which signals are processed per each ECU. Each main function of the vehicle corresponds to a specific ECU [18]. For modern vehicles, there can be close to 100 distinct ECUs, 50 actuators, and nearly 250 sensors relating to the overall functionality of the car [19].

The vehicle is shown using a communication method without the CAN-bus. There are large amounts of wires contributing to the total weight of the vehicle. Additionally, this wiring contributes to added noise in the signals due to the large noise environment developed within the vehicle from engine noise, wind resistance, radio interferences, and other added factors. This minimizes the wiring in the vehicle and is better optimized for noise, due to the two-wire pair of the CAN line. Additionally, the reduced wiring lowers the total weight of the vehicle which minimizes fuel consumption based on the reduction of torque in the wheels and motor.

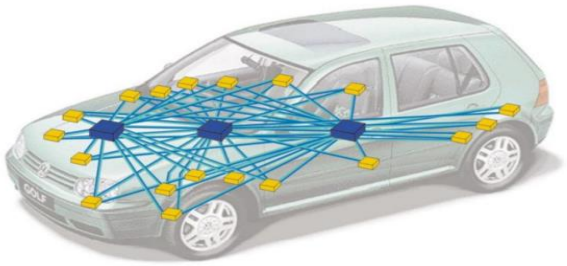

Figure 2.23: Car without CAN bus system

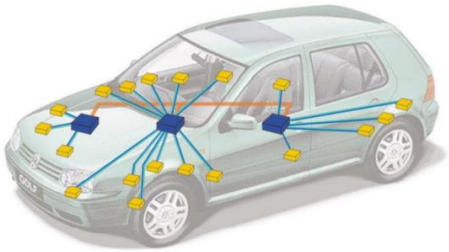

Figure 2.24: Car with CAN bus system

In Figure 2.16, the blue blocks represent ECU's, and the yellow blocks are sensors and actuators. It is the ECU's job to translate data from the sensors and actuators onto the CAN-bus line. In application, the ECU for the driver's door is separate from the cruise control ECU. When the driver opens the door of the vehicle, the door ECU sends a message onto the CAN-bus line which transmits the signal to all other ECUs, including the cruise control ECU. However, the cruise control ECU, despite recognizing that there is a message, does not do anything because the message ID of the door ECU does not match that of the cruise control ECU. Only messages directed towards the cruise control ECU can be interpreted by the cruise control ECU.

CAN bus has following advantages in comparison to others protocol:

- Simple and low cost: ECUs communicate via a single CAN system instead of via direct complex analogue signal lines - reducing errors, weight, wiring and costs.
- Fully centralized: the CAN bus provides 'one point-of-entry' to communicate with all network ECUs - enabling central diagnostics, data logging and configuration.
- Extremely robust: the system is robust towards electric disturbances and electromagnetic interference - ideal for safety critical applications.
- Efficient: CAN frames are prioritized by ID so that top priority data gets immediate bus access, without causing interruption of other frames.

2.2.3 CAN bus physical and data link layer (OSI)

The controller area network is described by a data link layer and physical layer. In the case of high-speed CAN, ISO 11898-1 describes the data link layer, while ISO 11898-2 describes the physical layer. The role of CAN is often presented in the 7-layer OSI model as per the illustration.

- The CAN bus physical layer defines things like cable types, electrical signal levels, node requirements, cable impedance etc. For example, ISO 11898-2 dictates a number of things, including below:
- Baud rate: CAN nodes must be connected via a two-wire bus with baud rates up to 1 Mbit/s (Classical CAN) or 5 Mbit/s (CAN FD)
- Cable length: Maximal CAN cable lengths should be between 500 meters (125 kbit/s) and 40 meters (1 Mbit/s)
- Termination: The CAN bus must be properly terminated using a 120 Ohms CAN bus termination resistor at each end of the bus

The CAN-bus system is primarily composed of 3 types: Drive-train CAN-bus, Convenience CAN-bus, and Infotainment CAN-bus. The transmission of the Drive-train CAN-bus is at a higher speed (500 kbps) in comparison to the low speed of the Convenience and Infotainment CAN-bus (100 kbps). The Drive-train CAN-bus contains ECUs relating to systems such as: engine control unit, brake control unit, steering angle sensor, and airbag control unit. The Convenience CAN-bus contains ECUs relating to systems such as: climate control unit, tire pressure check, and driver door control unit. The Infotainment CAN-bus contains ECUs relating to systems such as: radio, navigation, and phone interface box.

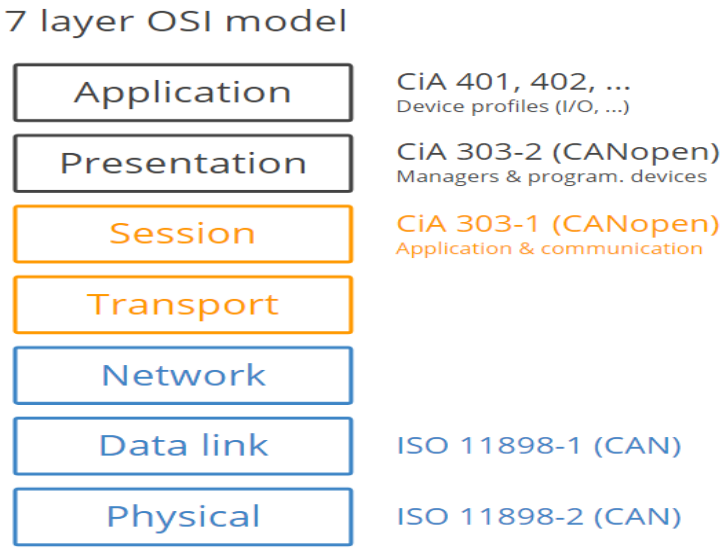

Figure 2.25: 7 layer OSI model of CAN

2.2.4 CAN bus operation

Every module in the network includes a CAN chip with a CAN controller and a CAN transceiver. The transceiver has the capability to transmit and receive data. The controller takes binary data from the microprocessor, converts it, and sends it to the transceiver. The transceiver then converts the binary data into a voltage range, which represents the signal voltage observed on the network.

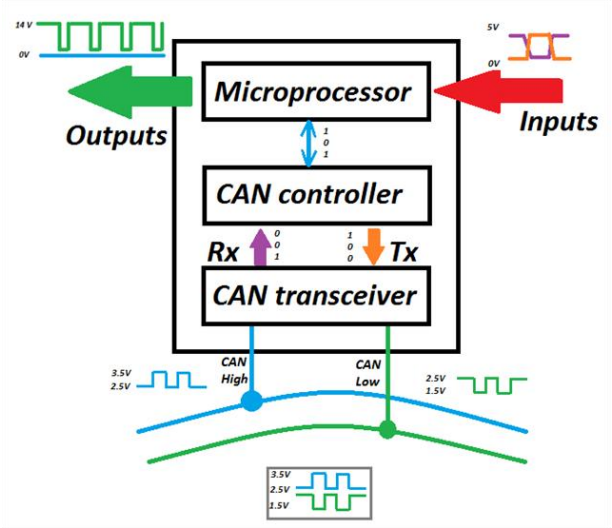

Figure 2.26: CAN bus operation illustration.

A message on the CAN-bus consists of two main components: the message ID and the message data. The message ID corresponds to the specific ECU that the signal is sourced from. The ECUs communicate with each other via the CAN-bus line. The message data is the specific content message generated by the ECU. Within the vehicle, there is a gateway control unit. The gateway navigates and regulates the sending and receiving of messages along the CAN-bus line and serves as the common node for all messages to pass through.

2.2.5 On-Board Diagnostic (OBD-II)

Beginning in 1962, each vehicle in the U.S. was designed with an on-board diagnostic (OBD-II) system. The OBD allows the vehicle to self-diagnose and provide status reports to a universal output port commonly presented on the bottom left section of the driver's cabin. It contains 16 pins and utilizes various tools within the vehicle in order to provide the driver with further information on the vehicle's status. CAN-H and CAN-L are critical in diagnostics and are the main form of communication within the vehicle. Although there are various other vehicle communication protocols, such as: LIN, FlexRay, K-line, ethernet, and MOST. In this project, we mainly focus on obtaining data via OBD2 protocol.

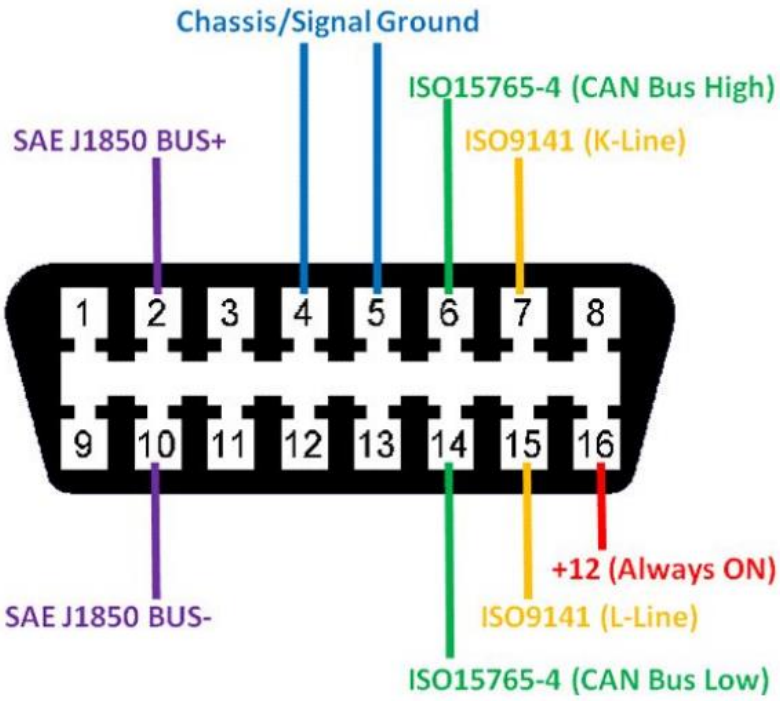

Figure 2.27: OBD2 connector

SAE J1962: This standard defines the physical connector used for the OBD2 interfacing, i.e., the OBD2 connector. The standard describes both the vehicle OBD2 connector, and the connector used by the external test equipment (e.g., an OBD2 scanner or OBD2 data logger). In particular, the standard dictates the location and access to the OBD2 connector.

SAE J1979: The SAE J1979 standard describes the methods for requesting diagnostic information via the OBD2 protocol. It also includes a list of standardized public OBD2 parameter IDs (OBD2 PIDs) that automotive OEMs may implement in cars (though they are not required to do so). Vehicle OEMs may also decide to implement additional proprietary OBD2 PIDs beyond those outlined by the SAE J1979 standard.

SAE J1939: The J1939 standard describes the data protocol used for heavy-duty vehicle communication. While OBD2 PID information is only available on-request by OBD2 test equipment, the J1939 protocol is used in most heavy-duty vehicles as the basic means for communicating CAN traffic - meaning data is broadcast continuously.

ISO 11898: This standard describes the CAN bus data link layer and physical layer, serving as the basis for OBD2 communication in most cars today.

ISO 15765-2: The ISO-TP standard describes the 'Transport Layer', i.e., how to send data packets exceeding 8 bytes via CAN bus. This standard is important as it forms the basis for Unified Diagnostic Services (UDS) communication, which relies on sending multiframe CAN data packets.

To get started recording OBD2 data, it is helpful to understand the basics of the raw OBD2 message structure. In simplified terms, an OBD2 message consists of an identifier and data. Further, the data is split in Mode, PID and data bytes (A, B, C, D) as below [20].

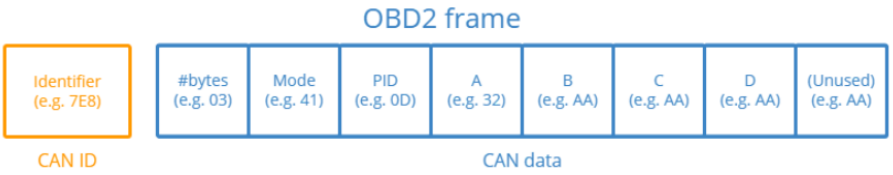

Figure 2.28: OBD2 data frame

Identifier: For OBD2 messages, the identifier is standard 11-bit and used to distinguish between "request messages" (ID 7DF) and "response messages" (ID 7E8 to 7EF). Note that 7E8 will typically be where the main engine or ECU responds at.

Length: This simply reflects the length in number of bytes of the remaining data (03 to 06). For the Vehicle Speed example, it is 02 for the request (since only 01 and 0D follow), while for the response it is 03 as both 41, 0D and 32 follow.

Mode: For requests, this will be between 01-0A. For responses the 0 is replaced by 4 (41, 42, …, 4A). There are 10 modes as described in the SAE J1979 OBD2 standard. Mode 1 shows Current Data and is e.g. used for looking at real-time vehicle speed, RPM etc. Other modes are used to e.g. show or clear stored diagnostic trouble codes and show freeze frame data

PID: For each mode, a list of standard OBD2 PIDs exist - e.g. in Mode 01, PID 0D is Vehicle Speed. Each PID has a description and some have a specified min/max and conversion formula. The formula for speed is e.g. simply A, meaning that the A data byte (which is in HEX) is converted to decimal to get the km/h converted value (32 becomes 50 km/h above). About RPM (PID 0C), the formula is (256*A + B) / 4.

A, B, C, D: These are the data bytes in HEX, which need to be converted to decimal form before they are used in the PID formula calculations. Note that the last data byte (after Dh) is not used.

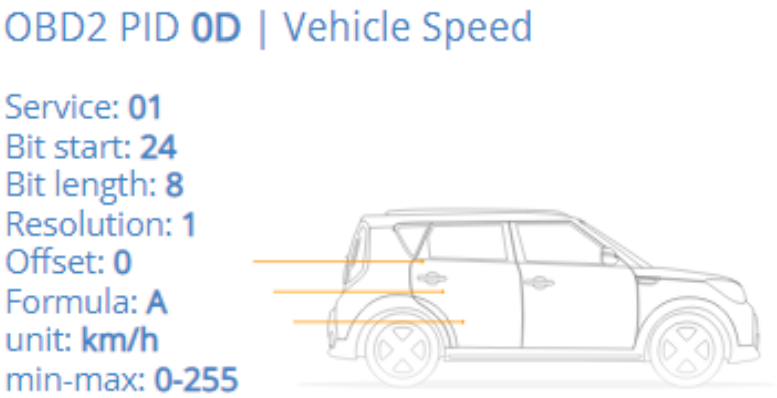

Figure 2.29: An example about OBD2 data

CHAPTER 3: DRIVER DROWSINESS DETECTION METHOD

3.1 Drowsiness definition

Drowsiness is a state of feeling sleepy or tired, and it often results in reduced ability to stay alert and perform tasks that require concentration or physical coordination. It is a natural response to the body's need for rest, and it can be caused by a variety of factors, such as inadequate sleep, sleep disorders, medications, alcohol, or illness.

Drowsiness can manifest as a feeling of lethargy or fatigue, heavy eyelids, difficulty staying awake, yawning, and reduced reaction time. It can be a temporary state that is relieved by rest or sleep, or it can be a chronic condition that requires medical attention. Excessive drowsiness can be dangerous, as it can lead to accidents while driving or operating heavy machinery. It is important to address the underlying causes of drowsiness and seek medical attention if it persists or interferes with daily life.

Figure 3.1: Drowsy driver

3.2 Drowsiness detection

Drowsiness detection is the process of identifying when a person is experiencing drowsiness or sleepiness, usually using technology such as sensors or cameras.

Drowsiness detection systems typically use various sensors and algorithms to monitor the user's vital signs, behavior, and other indicators of sleepiness, such as eye movements, head position, and facial expressions. For example, some systems use cameras to track eye movements and determine when the user is experiencing microsleeps or prolonged periods of drowsiness. Other systems may use wearable devices to measure the user's heart rate variability, body temperature, or other physiological signals that are indicative of drowsiness.

Once a drowsiness detection system identifies that the user is becoming drowsy, it can alert the user to take a break, rest, or take other corrective measures to prevent accidents. These systems are commonly used in transportation industries, such as aviation, trucking, and public transportation, but they can also be used in other settings where drowsiness can be a safety risk, such as medical settings or workplaces with heavy machinery.

Drowsiness detection methods can be divided into two major categories: Detection by measuring and observing the driver physiological symptoms and conditions and detection by measuring the vehicle variables and states, which are caused by the control actions of the driver. The latter obviously is still dependent on the drivers' condition and control action, but it does not require any direct measurement or monitoring of the driver. Each method has advantages and shortcomings.

3.2.1 Driver-based approach

Drowsiness detection can be conducted by measuring and observing the driver physiological symptoms [21]:

Eye closure: Monitoring driver eyes is one of the most successful techniques for detecting drowsiness and is studied by many researchers. Different techniques have been used to track the eyelid closures like Electrooculography (EOG) to detect eye movements, or the angle of inclination of eye corners to track the eyelid closures. Above of these, PERCLOS is the only valid psychophysiological measurement of alertness. PERCLOS is the percentage of eyelid closure over the pupil over time and reflects slow eyelid closures rather than blinks. A PERCLOS drowsiness metric was established in a 1994 driving simulator study as the proportion of time in a minute that the eyes are at least 80 percent closed. FWHA and NHTSA consider PERCLOS to be among the most promising known real-time measures of alertness for in-vehicle drowsiness-detection systems.

Electroencephalogram (EEG): EEG recorded from the human scalp is the most important physiological indicator of the central nervous system activation and alertness. In time domain, commonly used EEG measures include average value, standard deviation, and sum of squares of EEG amplitude, while in frequency domain energy content of each band, mean frequency, and center of gravity of the EEG spectrum are commonly used.

Facial expressions and body posture: trained observers could rate the drowsiness level of drivers based on video images of driver faces. There are some vision-based systems developed to extract facial expressions automatically but there is little evidence about the accuracy and robustness of such systems.

Other physiological conditions have also been monitored that include changes in heartbeat, blood pressure, skin electrostatic potential, and body temperature but all with limited success.

3.2.2 Vehicle-based approach

Frequency of steering wheel: any steering wheel pass across zero degree is counted as a reversal. Sleep-deprived drivers have lower frequency of steering reversals.[21]

Steering correction: this data can describe that when a driver is drowsy or falling asleep his/her steering behavior becomes more erratic, that is more frequent steering maneuvers during driving.

Vehicle steering has been approved by many studies as a characteristic variable which can predict driver drowsiness.

- **Micro-corrections**: These adjustments can be traced as small amplitude oscillations in steering wheel angle plots which keep a vehicle in the center of a lane (lane keeping adjustments).
- **Macro-corrections:** Besides lane keeping adjustments, drivers may make large steering adjustments to negotiate a curve or change lanes. Research indicates that a sleepy driver has larger amplitude steering corrections (over steering) and less frequent micro-corrections.

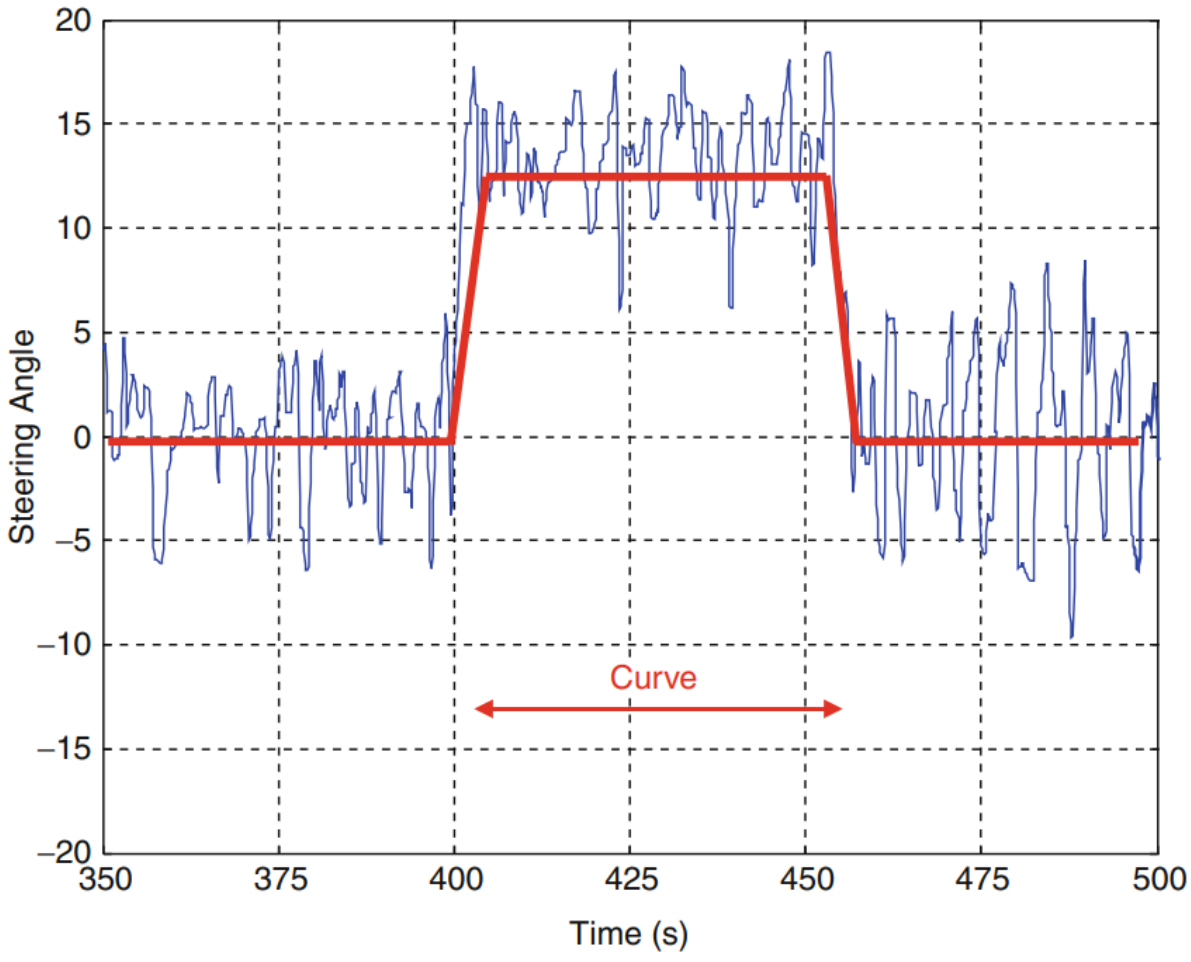

Figure 3.2: Micro and macro-adjustments in a steering wheel angle waveform while a curve

- **Frequency of steering wheel**: any steering wheel pass across zero degree is counted as a reversal. Sleep-deprived drivers have lower frequency of steering reversals.
- **Steering correction**: this data can show that when a driver is drowsy or falling asleep his/her steering behavior becomes more erratic, that is, more frequent steering maneuvers during the drive.

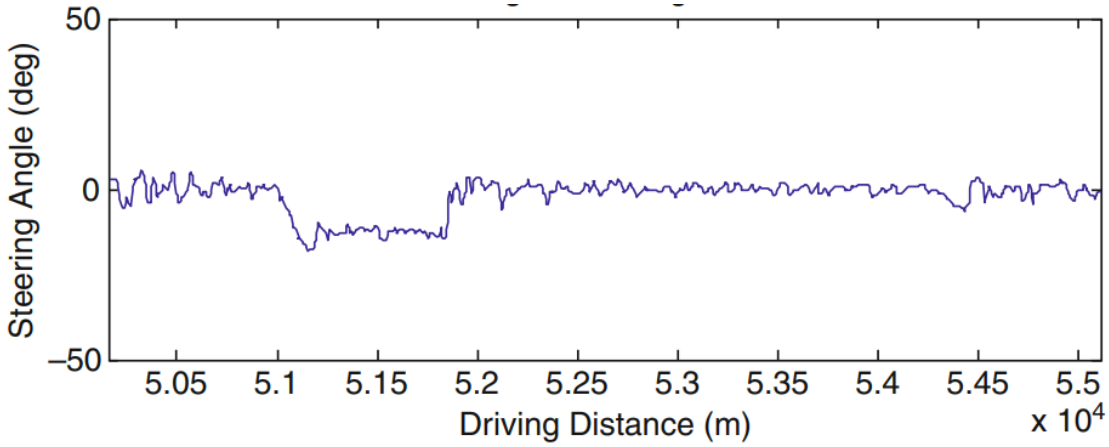

Figure 3.3: Steering angle of the alert driver

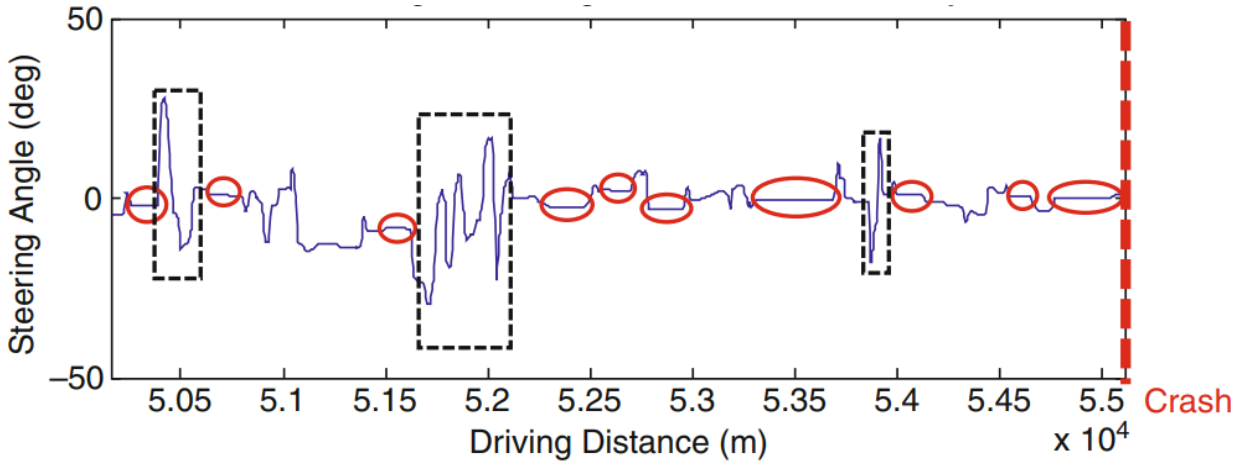

Figure 3.4: Steering angle of the drowsy driver

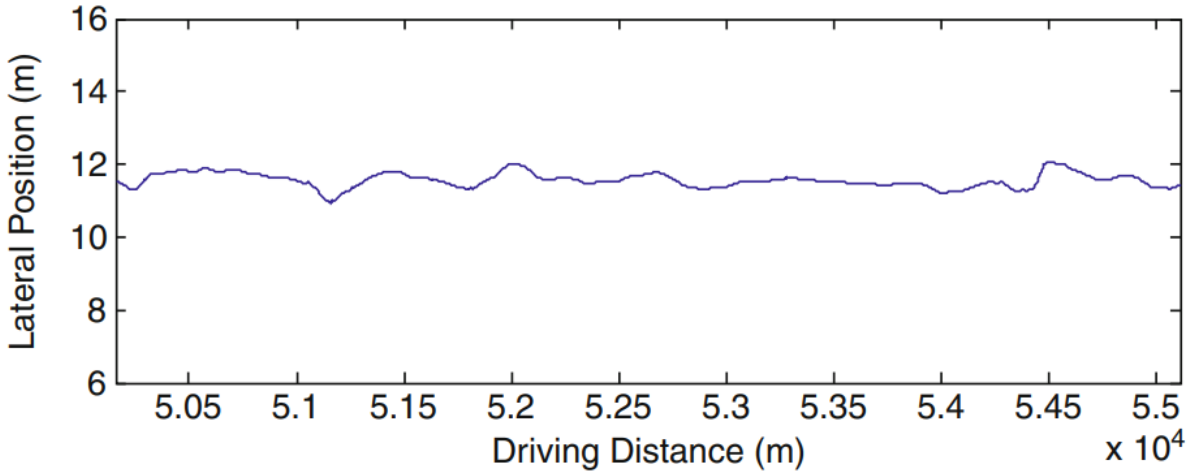

Figure 3.5: Lateral position of the alert driver

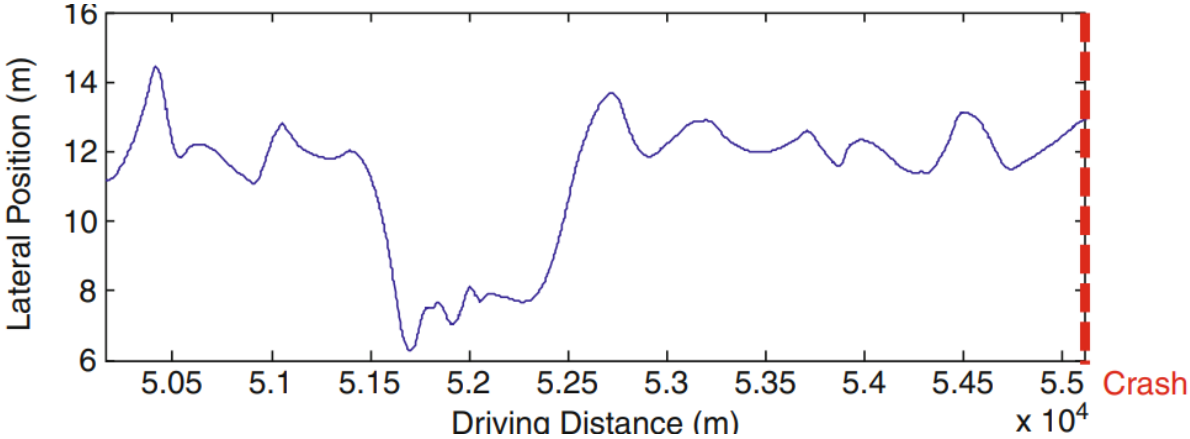

Figure 3.6: Lateral position of the drowsy driver.

The above figures are measurement results in the same situation with 2 types of drivers: alert and drowsy. We can conclude the sleepy driver has less steering frequency and higher steering amplitude than alert drivers.

Steering Velocity: Drowsiness and sleep deprivation decreases steering velocity and increases standard deviation of steering velocity.

Moreover, there are some other vehicle state data that can indicate the status of the driver:

- Vehicle speed
- Lateral position
- Yaw, brake, acceleration

	Advantages	Disadvantages
Vehicle based approach	- No need to integrate external sensors, devices.	- Complicated algorithm
Driver based approach	- Direct interact with driver	- Low-Medium accuracy due to multiple environment variables - Require external device/sensors

Table 3.1: Methods comparison

In this thesis, we only use the vision method to detect drowsiness because of some limitations in behavior analysis. This method is based on the most critical symptom of drowsiness which is micro-sleep. Micro sleep refers to a brief episode of sleep that occurs involuntarily and lasts for a few seconds to a few minutes. During a micro sleep, a person's brain enters a state of sleep while they may appear to be awake.

The below figure is the algorithm that we apply for the drowsiness detection system.

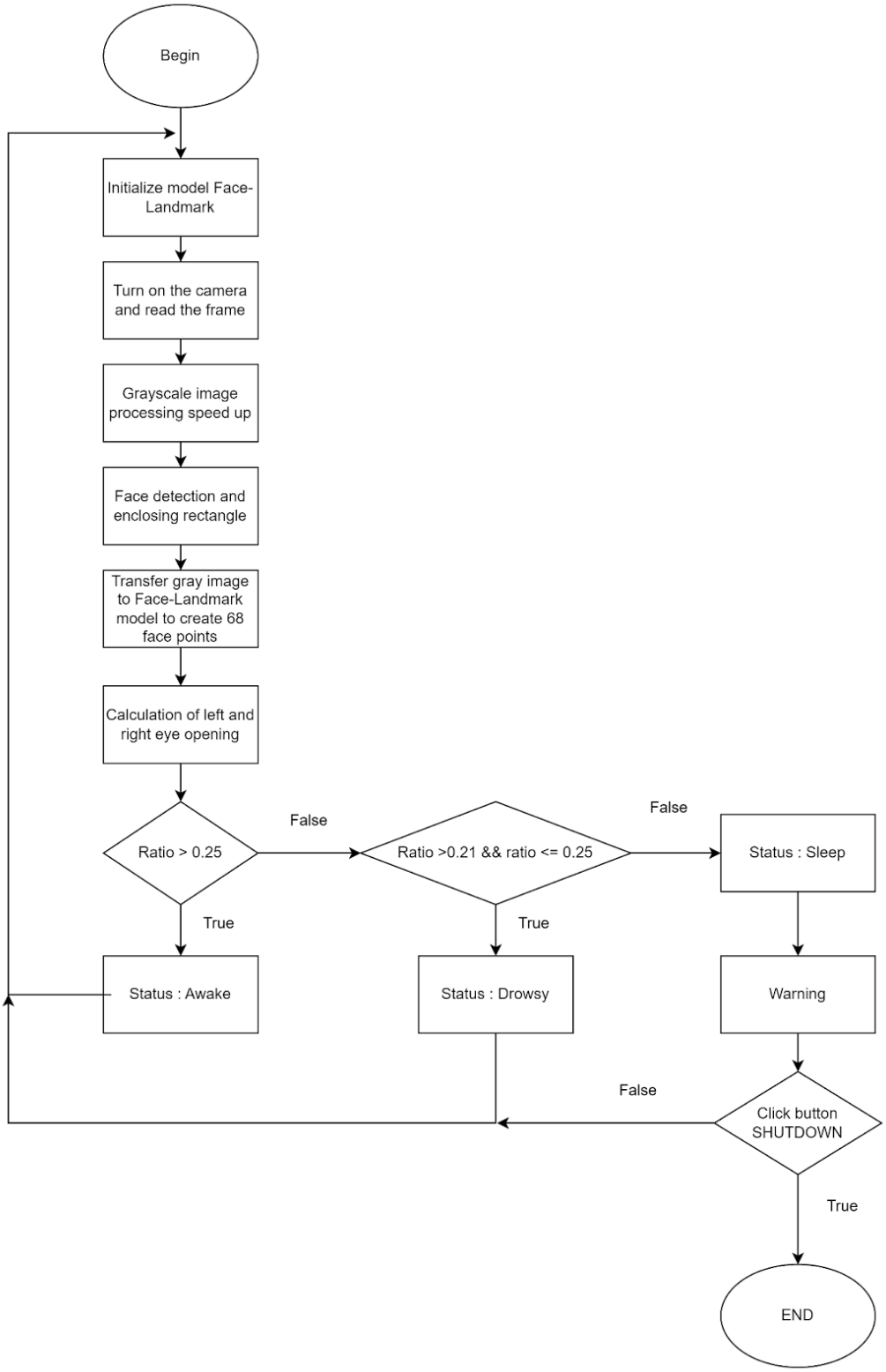

Figure 3.7: Drowsiness detection algorithm

Initialize model Face Landmark: Initializing the face detector and landmark detector.

Turn on the camera and read the frame: To acquire the driver's face as the input of the system.

Grayscale image processing speed up: the acquired images must be converted into gray scale for faster processing without losing necessary data.

Face detection and enclosing rectangle: create a rectangular frame around the recognized face for later calculation.

Transfer gray image to Face-Landmark model to create 68 face points: generate 68 points of face defining face parts for calculation.

Calculation of left and right eyes opening evaluate the openness of both left and right eyes. If the specified threshold is exceeded will return to values: Awake-2; Drowsy-1; Sleep-0.

Warning: If the sleep state is long enough, a audio warning sound will be activated to alert the driver to wake up.

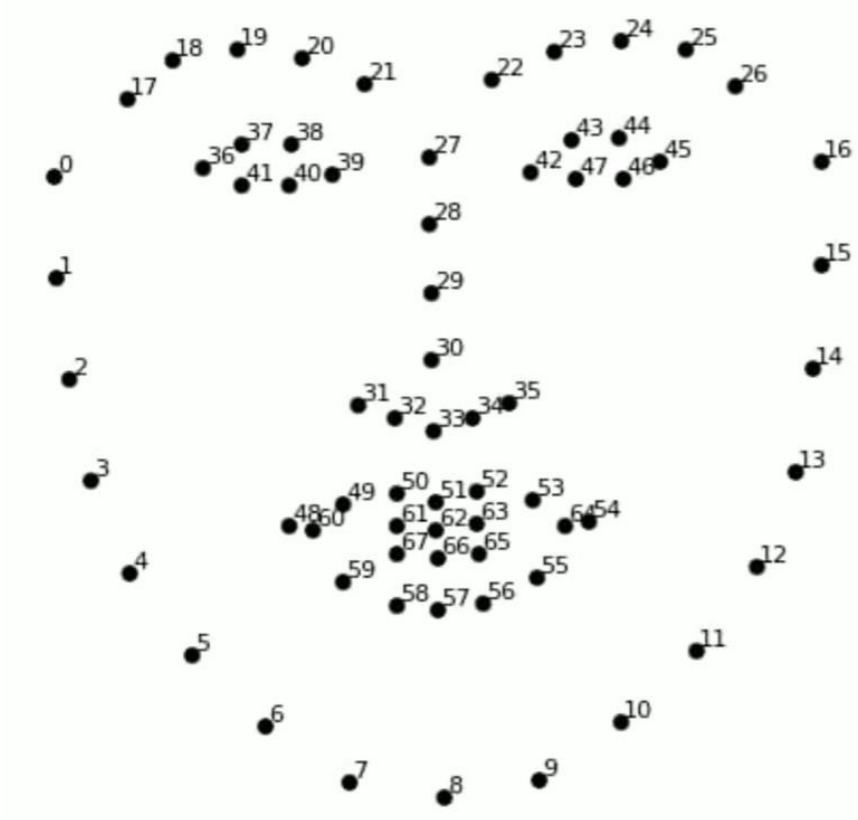

Figure 3.8: Face landmark detector

The points in the left eye's image are 36,37,38,41,40,39, while the points in the right eye's image are 42,43,44,47,46,45.

A variable called Eyes Aspect Ratio (EAR) is used to estimate eye opening state [22]. For example, the equation 1 show us how to calculate EAR of left eye:

$$EAR = \frac{||P38 - P40|| + ||P37 - 41||}{2||P39 - P36||} \quad (1)$$

with 41, 42, 36, ..., are the points illustrated in the figure and the letter "P" indicates for the word "point". Similar calculation is applied to find EAR of right eye.

CHAPTER 4: HARDWARES AND SOFTWARE OVERVIEW

4.1 Hardware overview

4.1.1 Raspberry pi model 4

Raspberry Pi is a series of small, low-cost single-board computers developed by the Raspberry Pi Foundation. The first Raspberry Pi was released in 2012, and since then, several models have been released with increasing capabilities and performance.

The Raspberry Pi 4 single-board computer developed by the Raspberry Pi Foundation. It is the fourth iteration in the Raspberry Pi series and was released in June 2019. The Raspberry Pi 4 has several improvements over its predecessor, the Raspberry Pi 3, including a more powerful processor, support for dual 4K displays, faster Ethernet and Wi-Fi connectivity, and increased memory options. It is designed for use in a variety of projects, including home automation, media centers, robotics, and education. The Raspberry Pi 4 runs on a variety of operating systems, including Raspbian, Ubuntu, and several other Linux-based distributions, as well as Windows 10 IoT Core.

The Raspberry Pi is used in a variety of projects, from home automation to media centers to robotics. Its versatility and affordability make it popular among hobbyists and professionals alike, and it has been used in a wide range of applications, including education, research, and industrial control.

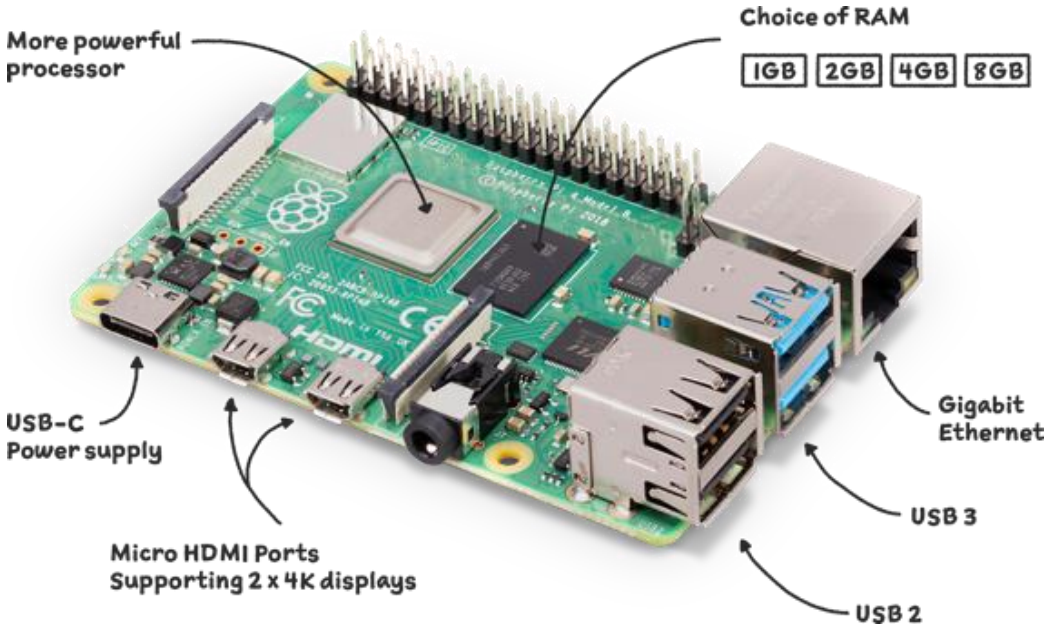

Figure 4.1: Raspberry pi architecture

Processor	Broadcom BCM2837B0, quad-core A53 (ARMv8) 64-bit SoC @1.4GHz
RAM	1GB LPDDR2 SDRAM
Connection	2.4GHz and 5GHz IEEE 802.11 b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0, USB: 4 x 2.0
Video and sound	1 full-sized HDMI, MIPI DSI Display, MIPI CSI Camera, stereo output and composite video 4 pins.
Multimedia	H.264, MPEG-4 decode (1080p30), H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
Storage	MicroSD
Power supply	5V/2.5A DC from USB port; 5V DC from GPIO pins; Power over Ethernet (PoE)

Table 4.1 Raspberry pi configuration

4.1.2 Monitor

In this thesis, a monitor is used to display a driver's face to evaluate the accuracy of the system. A 7-inch Universal Portable Touch Monitor, HDMI Port, 1080×1920 Full HD, IPS Screen, Optical Bonding/AF Coating Toughened Glass Panel, Various Systems & Devices Support Including Raspberry Pi, Jetson Nano, PC.

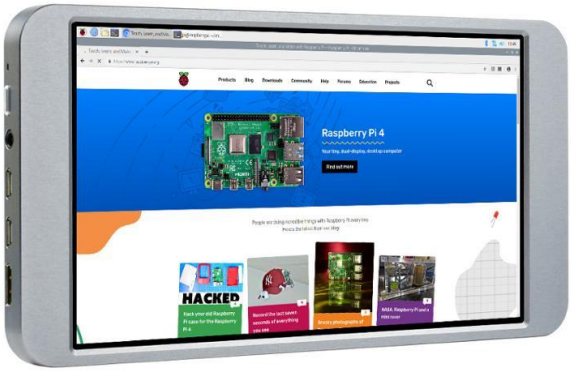

Figure 4.2: 7-inch Universal Portable Touch Monitor

4.1.3 Camera

The Raspberry Pi camera is a camera module specifically designed to be used with the Raspberry Pi single-board computer.

The Raspberry Pi camera is popular among hobbyists and makers who use it for a wide range of projects, including time-lapse photography, motion detection, home security systems, and more. The camera module can be controlled using various programming languages, including Python and C++, and there are many third-party libraries and applications available to extend its functionality.

Figure 4.3: Camera Raspberry Pi V2 IMX219 8MP

Name	Camera Raspberry Pi V2 IMX219 8MP
Camera Resolution	8 megapixels; 3280 x 2464 p
Video Resolution	HD 1080p30, 720p60 and 640x480p90 video
Size	25mm x 23mm x 9mm
Weight	3g
Connection	Ribbon cable

Table 4.2 Camera specifications

4.1.4 Power supplier

In this project, we use a 5V-battery to power others device. However, for better design in automobiles the hardware should be powered with an available source.

Figure 4.4: USB port in vehicle

4.1.5 CAN bus shield

The CAN bus shield module is used to obtain the necessary signals. In the figure below, the module is connected with Arduino via SPI protocol.

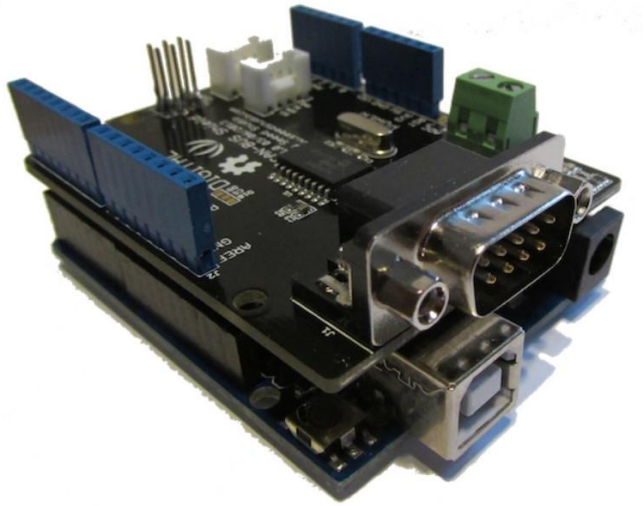

Figure 4.5: CAN bus shield

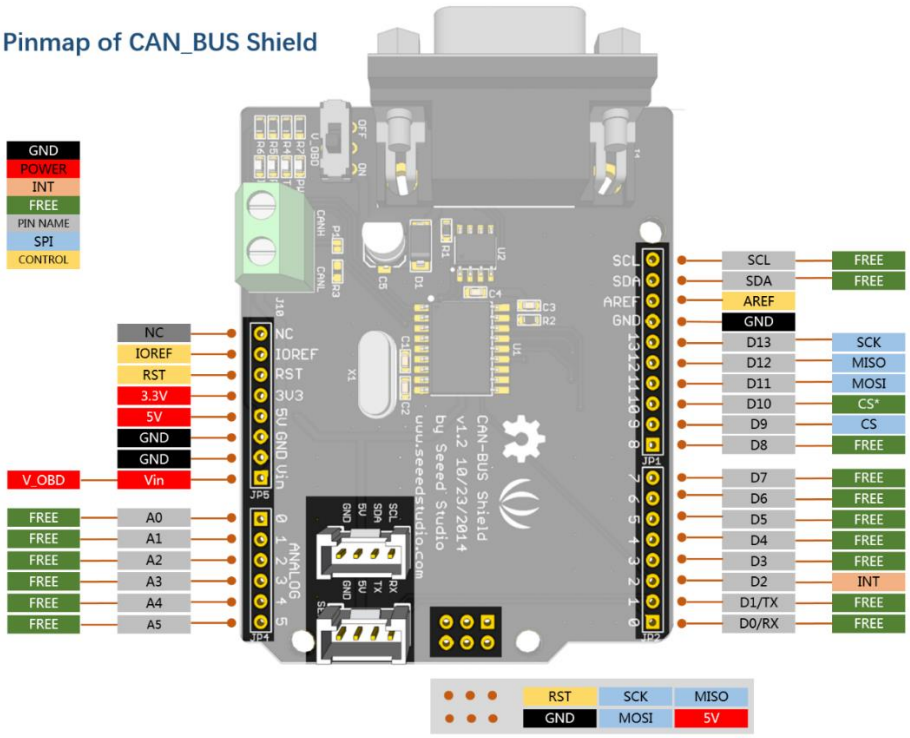

Figure 4.6 : CAN bus shield pin-out

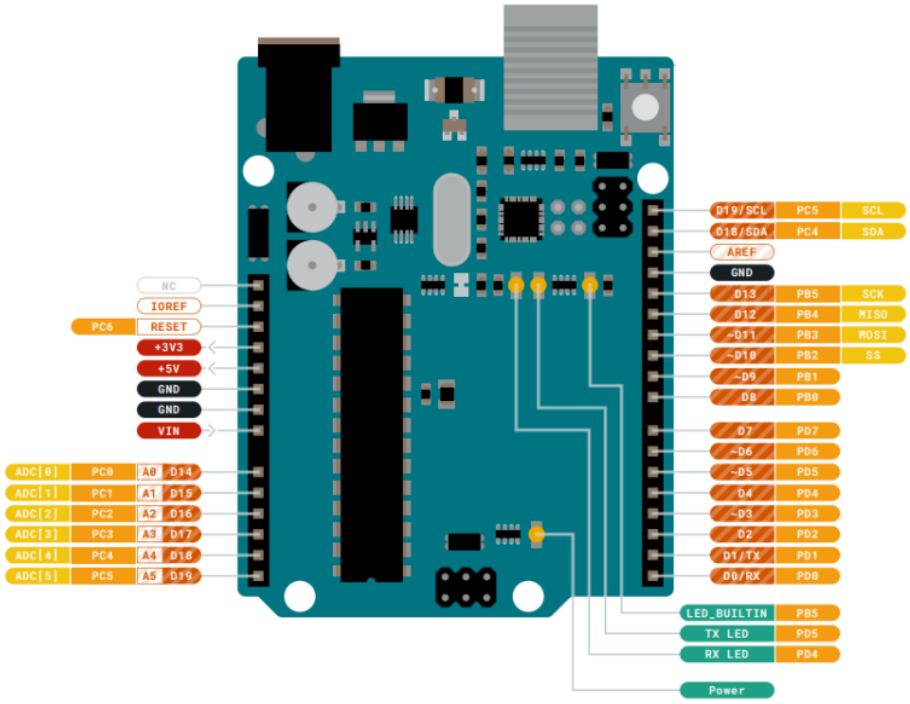

Figure 4.7: Arduino Uno pin-out

Microcontroller	ATmega328P
Operating voltage	5V
Input voltage	7-12V
Limit voltage	6-20V
Digital I/O pins	14(6 PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC current per I/O pins	20mA
DC current for 3.3V pins	50mA
Flash memory	32KB
SRAM	2KB
EEPROM	1KB
Clock speed	26MHz
LED built in	13
Length	68.6mm
Width	53.4mm

Table 4.3: Arduino Uno specifications

This CAN-BUS Shield adopts MCP2515 CAN Bus controller with SPI interface and MCP2551 CAN transceiver to give CAN-BUS capability.

MCP2515 is a CAN peripheral expansion module for microcontrollers that do not incorporate this modern communication standard. The MCP2515 uses the SPI interface, so any microcontroller can communicate with it through SPI protocol or even using regular IO pins.

Voltage	4.75-5.25V
Power	4.5V DC
Current	5mA
Static	1uA
Data speed	1Mbps
Data field	0-8 byte
Size	4.4cm x 2.8 cm
Temperature	-40 C-85 C

Table 4.4 CAN bus shield specifications

4.2 Software overview

4.2.1 Programming language

Python is a high-level, interpreted programming language that was first released in 1991. It is designed to be easy to read and write, with a simple syntax that emphasizes code readability and reduces the cost of program maintenance. Python is an object-oriented language, which means that it supports concepts such as classes, objects, and inheritance. It is also a dynamically-typed language, which means that the type of a variable is determined at runtime rather than at compile time. Python has become very popular in recent years, and is widely used for a variety of purposes including web development, scientific computing, data analysis, machine learning, and more. One of the reasons for its popularity is its large and supportive community, which has created a vast ecosystem of third-party libraries and tools that make it easy to get started with Python and to extend its capabilities.

For drowsiness detection, in comparison with multiple programming languages Python is one of the best ones which is easy to write and read. Besides, it support us to use various AI library, epsecially OpenCV and Dlib.

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library that provides developers with a wide range of tools and algorithms for processing and analyzing images and videos. It was first released in 2000 by Intel Corporation and has since been maintained by a community of developers.

OpenCV is written in C++ and can be used with a variety of programming languages including Python, Java, and MATLAB. It provides a number of built-in algorithms and

functions for performing tasks such as image and video processing, object detection and recognition, feature extraction, motion analysis, and more.

One of the key advantages of OpenCV is its speed and efficiency, which make it suitable for use in real-time applications such as robotics, surveillance, and autonomous vehicles. OpenCV also includes a number of machine learning algorithms, including support for deep learning frameworks such as TensorFlow and PyTorch, making it a powerful tool for developing computer vision and machine learning applications.

Figure 4.8: Python logo

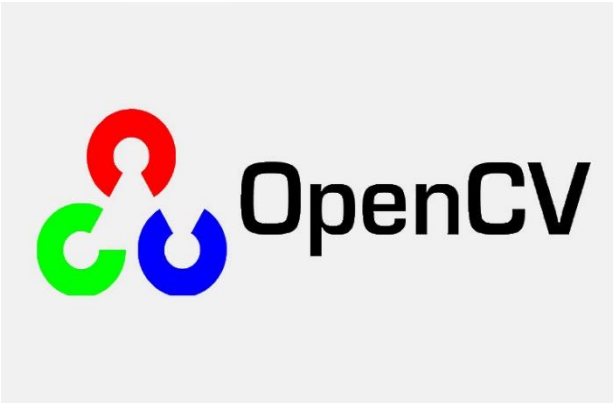

Figure 4.9: OpenCV logo

4.2.2 Integrated Development Environment

With Raspberry pi programming, we use Visual Studio Code. Visual Studio Code, also known as VS Code, is a free and open-source code editor developed by Microsoft. It is a cross-platform editor, meaning it can run on multiple operating systems, including Windows, macOS, and Linux. Visual Studio Code is widely used by developers for coding, debugging, and deploying applications.

Visual Studio Code supports a wide variety of programming languages, including JavaScript, Python, Java, C++, and many others. It includes features such as syntax highlighting, code completion, debugging tools, version control integration, and extensions that allow developers to customize the editor to their specific needs.

One of the key features of Visual Studio Code is its ease of use and accessibility. It has a simple and intuitive user interface that allows developers to quickly navigate and edit code. Additionally, it is highly customizable and can be configured to suit the preferences of individual users.

Visual Studio Code is popular among developers because it is lightweight, fast, and efficient. It is widely used in web development, data science, and other fields where coding is an essential part of the work.

Figure 4.10 : Visual studio code logo

About CAN bus shield, we utilize a popular Arduino IDE. The Arduino Integrated Development Environment (IDE) is a software platform that is used to write and upload code to Arduino boards. Arduino is an open-source platform that provides a wide range of microcontroller-based development boards that are used in a variety of applications, including robotics, automation, and Internet of Things (IoT) projects.

The Arduino IDE is free and open-source software that is designed to be user-friendly and easy to use. It includes a code editor, a compiler, and a uploader that allows users to write code in a simple programming language based on C/C++. The IDE also includes a range of libraries and examples that can be used to develop projects quickly and efficiently.

The Arduino IDE supports a wide range of Arduino boards, from simple microcontrollers to more advanced boards with built-in Wi-Fi and Bluetooth capabilities. The platform is highly versatile and can be used in a wide range of applications, from hobbyist projects to commercial products.

One of the key features of the Arduino IDE is its simplicity and ease of use. It is designed to be accessible to people with little or no programming experience, and its user-friendly interface and extensive documentation make it an ideal platform for beginners. Additionally, the Arduino community is large and active, providing support and resources for users at all levels of experience.

Figure 4.11: Arduino IDE logo

4.2.3 Raspberry Pi OS

Raspberry Pi (RasPi) is a small and affordable single-board computer that has gained immense popularity for its versatility and wide range of applications. As an operating system, RasPi provides a platform for running various software and applications on the Raspberry Pi hardware.

The RasPi operating system is based on the Linux kernel and offers a choice of several operating systems tailored specifically for the Raspberry Pi. The most widely used and supported operating system for RasPi is Raspbian, which is a Debian-based distribution optimized for the Raspberry Pi's architecture. Raspbian provides a user-friendly interface and includes a suite of pre-installed software, making it accessible for beginners and enthusiasts alike.

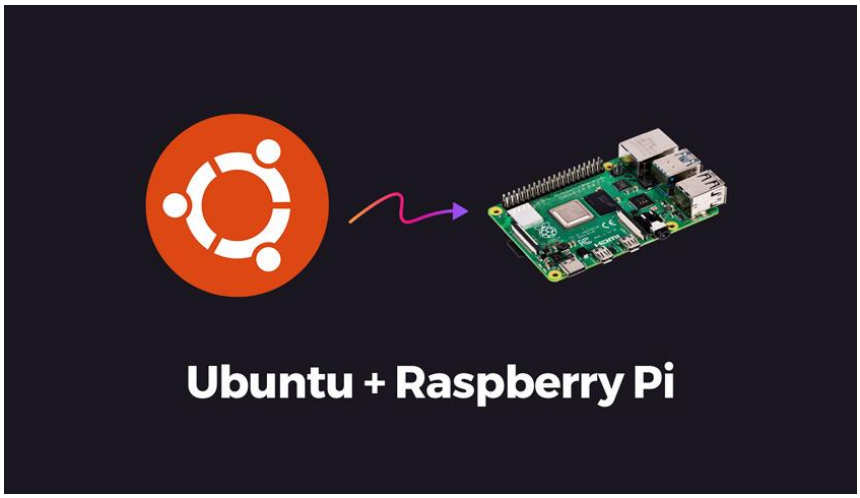

Figure 4.12: Raspberry Pi OS

Apart from Raspbian, there are other operating systems available for RasPi, such as Ubuntu Mate, Arch Linux, and Windows 10 IoT Core. These operating systems cater to different user preferences and requirements, offering a range of features and compatibility options.

RasPi's operating system provides a platform for a diverse set of applications and projects. It supports programming languages like Python, C/C++, and Java, enabling users to develop and execute their own software projects. The RasPi operating system also supports various multimedia applications, making it ideal for media centers, retro gaming consoles, and home automation systems.

Moreover, RasPi's operating system allows for easy integration with external devices and sensors, enabling users to create Internet of Things (IoT) projects and build prototypes for innovative ideas. Its GPIO (General Purpose Input/Output) pins provide a means for connecting and controlling external electronic components, making it a versatile platform for electronics enthusiasts and makers. The RasPi operating system provides a robust and flexible platform for running software and applications on the Raspberry Pi. Its Linux-based architecture, diverse operating system options, and extensive community support

make it an accessible and powerful tool for a wide range of projects, from educational initiatives to hobbyist endeavors and even professional applications.

CHAPTER 5: EXPERIMENT AND RESULTS

5.1 EXPERIMENT

5.1.1 System design

After referring to lots of drowsiness detection systems, we finally come up with a completed one. The below picture is complicated system that can integrate to multiple system of the vehicle [21].

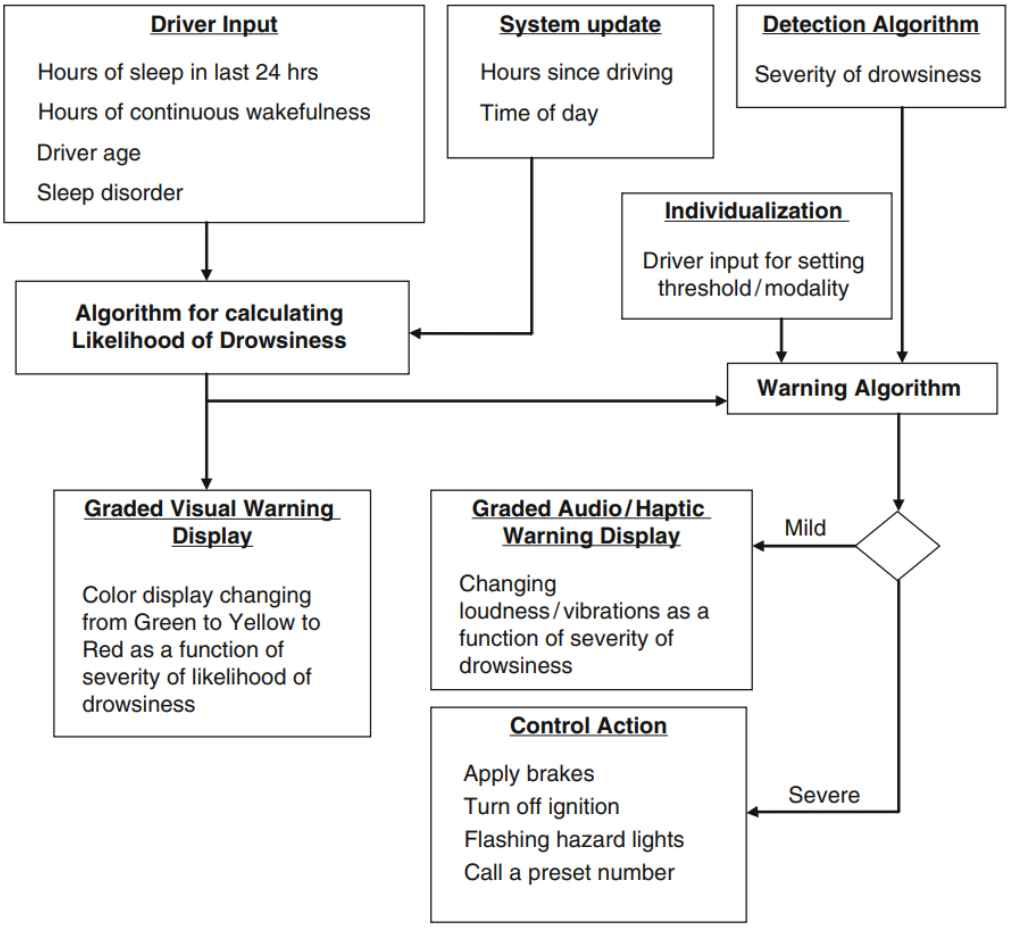

Figure 5.1: Schematic of a drowsy driver warning system

Because of some limitations of knowledge, we propose a simpler system design with mentioned hardware in the previous chapter.

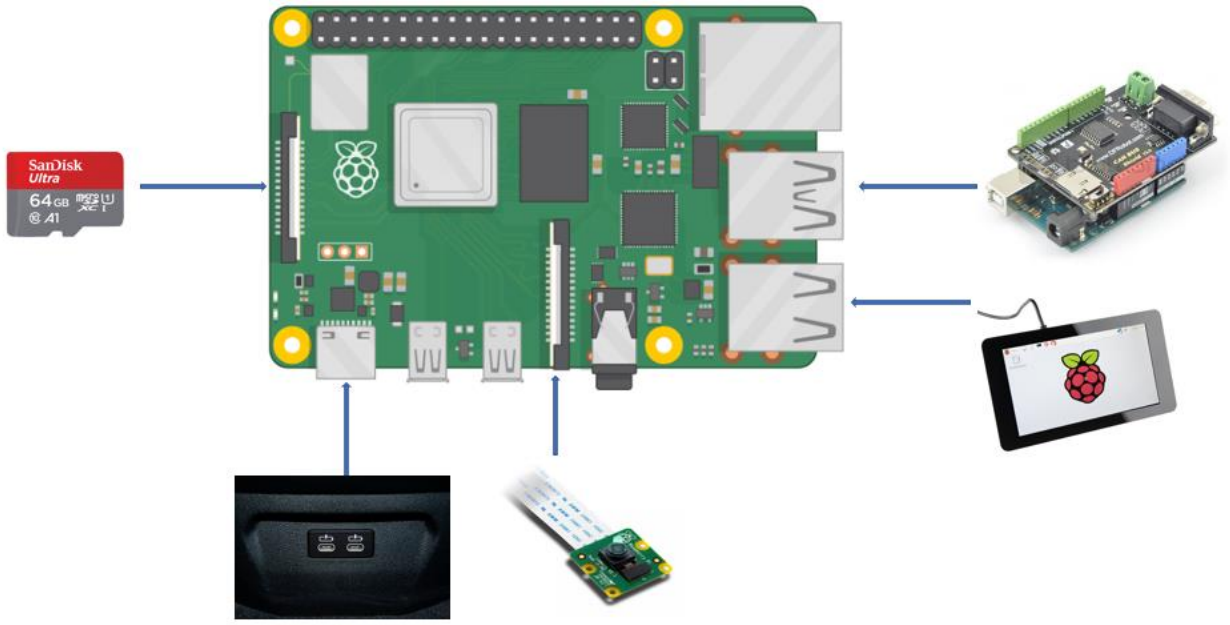

Figure 5.2: Proposed driver drowsiness detection system

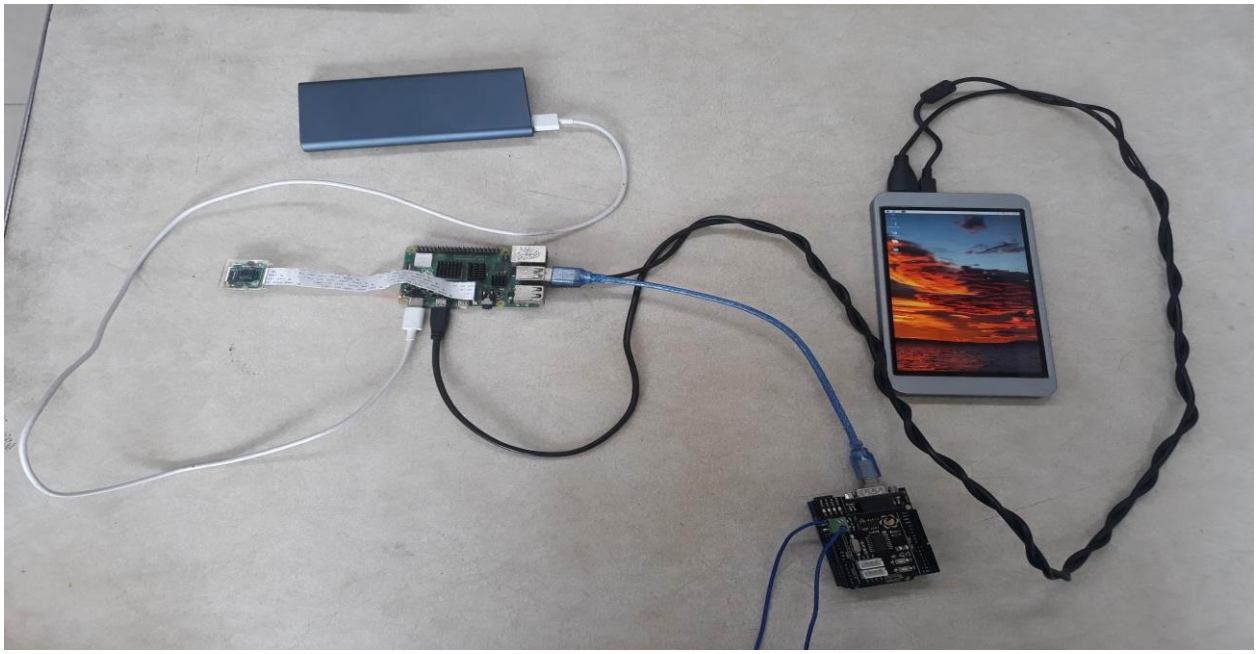

Figure 5.3: Real-life hardware connection

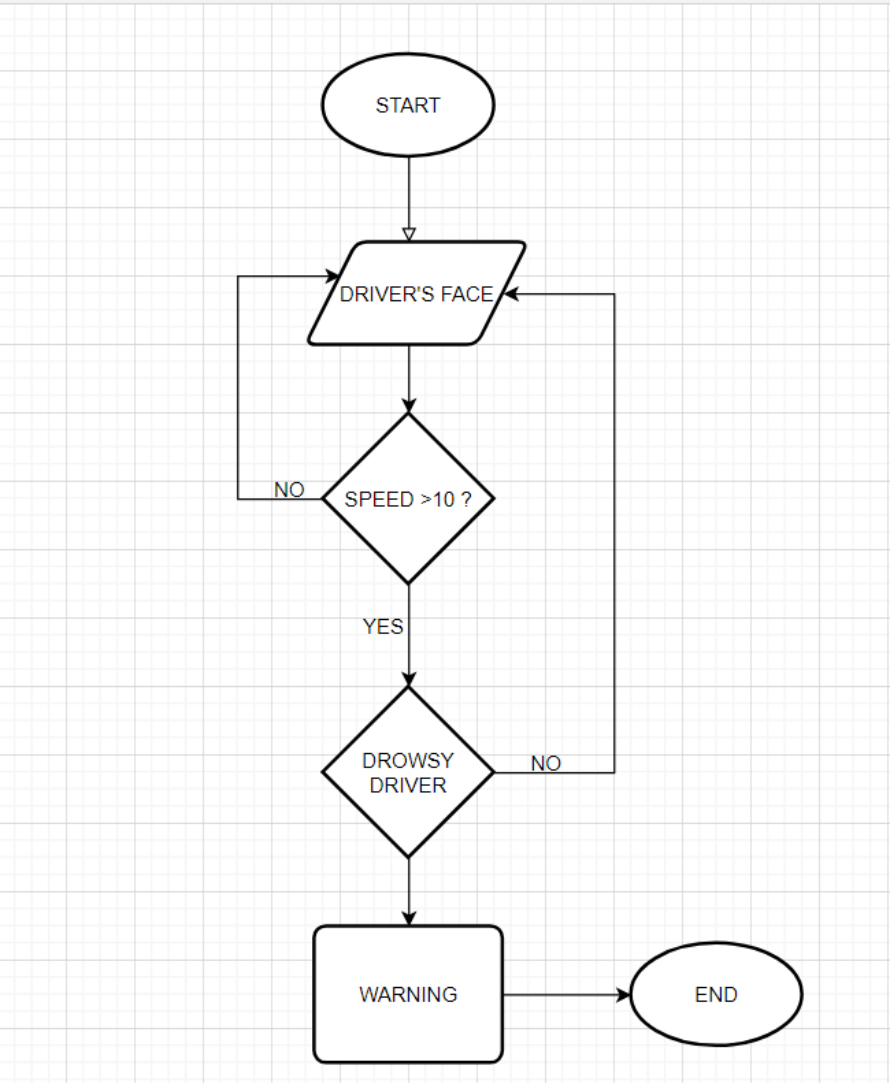

Figure 5.4 : System schematic diagram

5.1.2 System requirements

Hardware setup is one of major challenges for any vision-based systems in automobile. There are some issues that must be considered in this paper, as well.

- Vehicle ego-motion: a vision system must be robust when the vehicles move. A high-speed driving or a sharp turn can be a reason why input images get blurred.
- Illumination condition: illumination is a major issue because humans hardly control it. For example, weather conditions, the sun positions, and artificial light sources such as headlights or streetlamps highly affect scene illumination.
- Real-time adaptation: an advanced driver assistance system must meet a level of real-time adaption due to its importance.

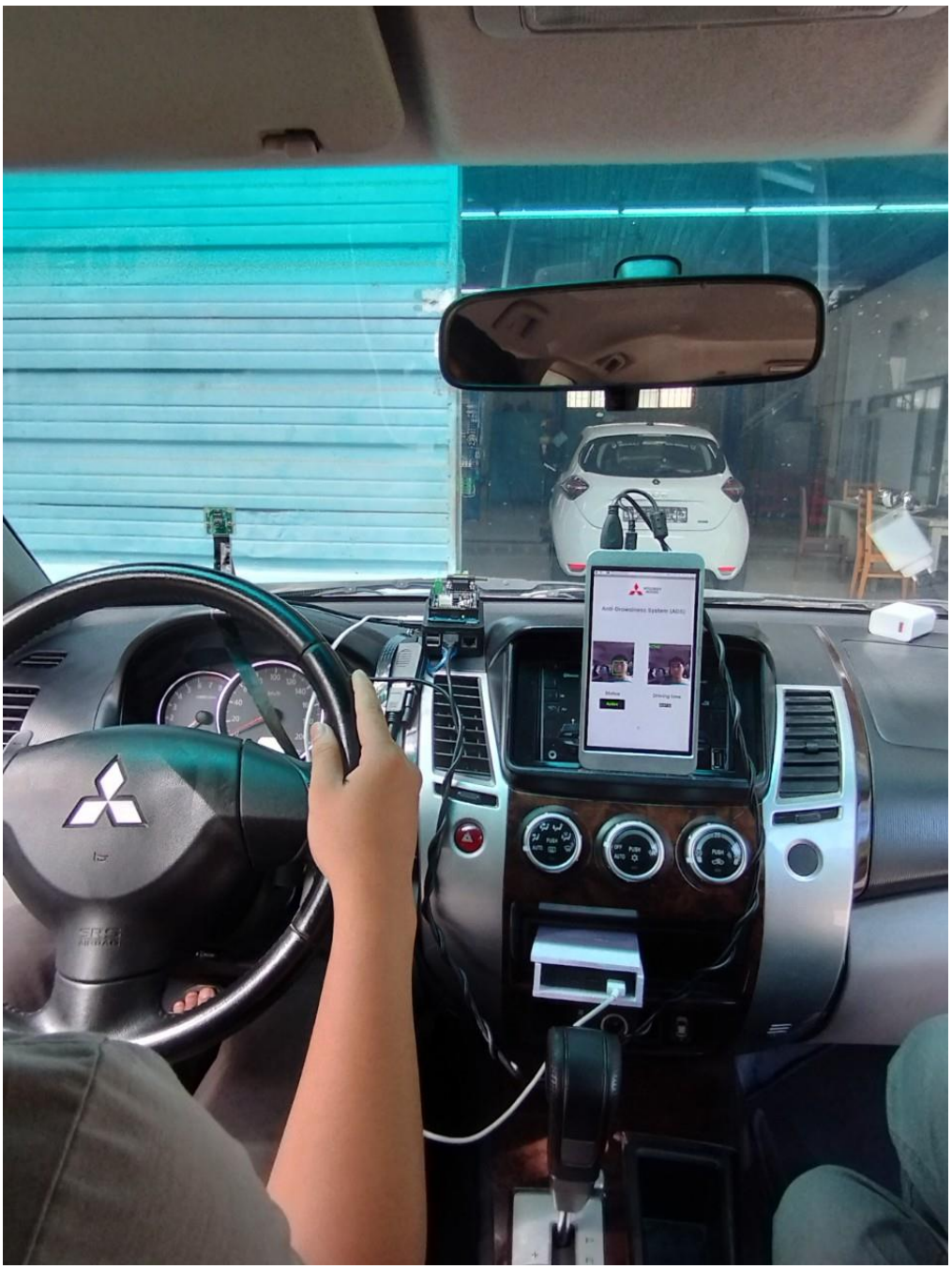

Figure 5.5: Actual setup on vehicle

5.2 Test results on the computer and vehicle

In situations where there are sufficient lighting conditions and the individual is wearing glasses, the ability to detect sleep is significantly reduced to a delay of less than 1-2 seconds. The combination of adequate lighting and glasses enables the face to be readily recognized, allowing for a swift transition into an active state without any noticeable delay.

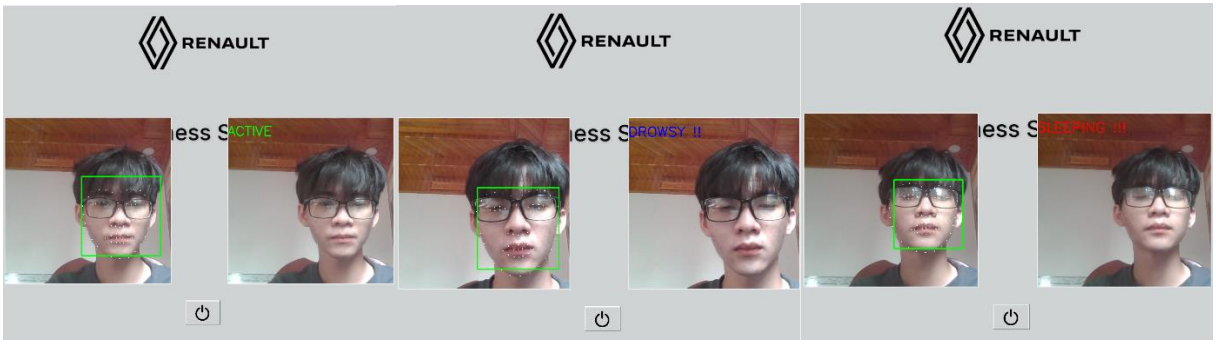

Figure 5.6 : Detect in good illumination conditions and wear glasses.

Under optimal lighting conditions and with the aid of glasses, the detection of sleep is not impeded by any significant delay. However, due to the transitional nature between an active and sleep state, this detection process typically occurs rapidly and seamlessly, often resulting in an immediate transition into the sleep state.

In scenarios where there are optimal lighting conditions and the individual is not wearing glasses, the system exhibits a remarkable ability to recognize facial features and promptly detect the active status without any noticeable delay. The absence of glasses does not hinder the face detection process, allowing for seamless identification and confirmation of an active state.

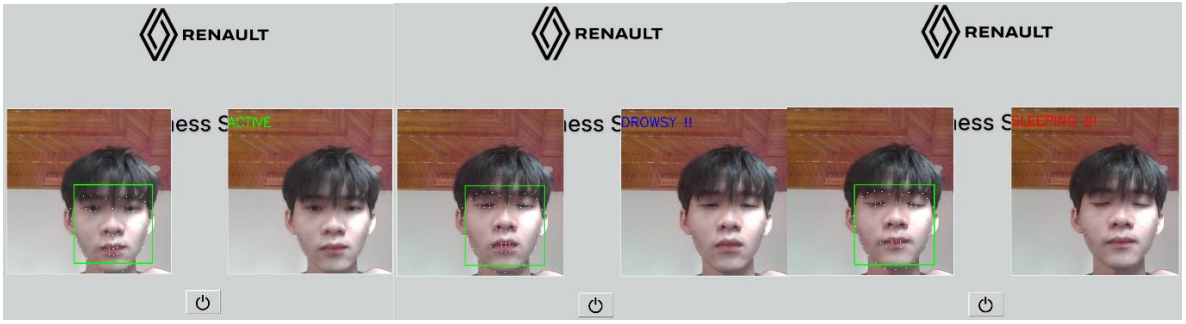

Figure 5.7: Detect in well-lit conditions and without glasses.

When the lighting conditions are favorable and glasses are not worn, the system demonstrates an efficient capacity to detect the sleep state in a prompt manner. Within a mere 1-2 seconds of the driver closing their eyes, the system initiates an immediate alert to notify the individual of their drowsiness. This proactive response ensures the safety of the driver by promptly addressing the onset of drowsiness and encouraging appropriate corrective actions.

In challenging circumstances where the surrounding ambient light is minimal, such as in a dark environment, the system demonstrates its capability to detect the driver's face and accurately determine their status. In this scenario, the primary source of illumination for face recognition comes from the light emitted by the laptop screen, which is directed towards the driver's face.

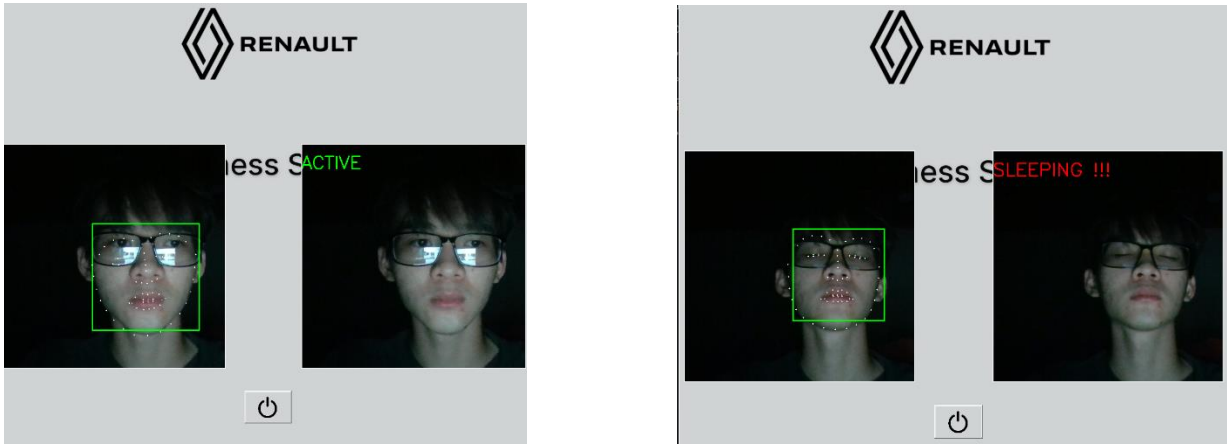

Figure 5.8: Detection in low light

While the process of recognizing the active state may experience a slight delay of approximately 1-2 seconds due to the intricacies of face search processing under low-light conditions, the system's ability to identify the driver's face remains highly effective. The presence of the laptop screen's light significantly aids in illuminating the driver's face, enabling the system to successfully identify the sleep state with relative ease.

Despite the darkness of the environment, the light emitted by the computer screen serves as a valuable resource, allowing the system to effectively distinguish between active and sleep states. As a result, the system's proficiency in identifying the sleep state remains largely unaffected even in dark conditions, exhibiting performance comparable to that in well-lit environments.

Figure 5.9: Anti-drowsy device on the market

For products on the market such as Device Against Drowsiness While Driving as pictured above. Wearing in the ear causes discomfort to the driver, although the ability to detect drowsiness is quite good when the driver lowers his face, it will vibrate. However, there

are quite a few disadvantages such as: This device can fall out ears when the driver is dozing and when the driver is facing the sky, it is almost impossible to wake the driver up.

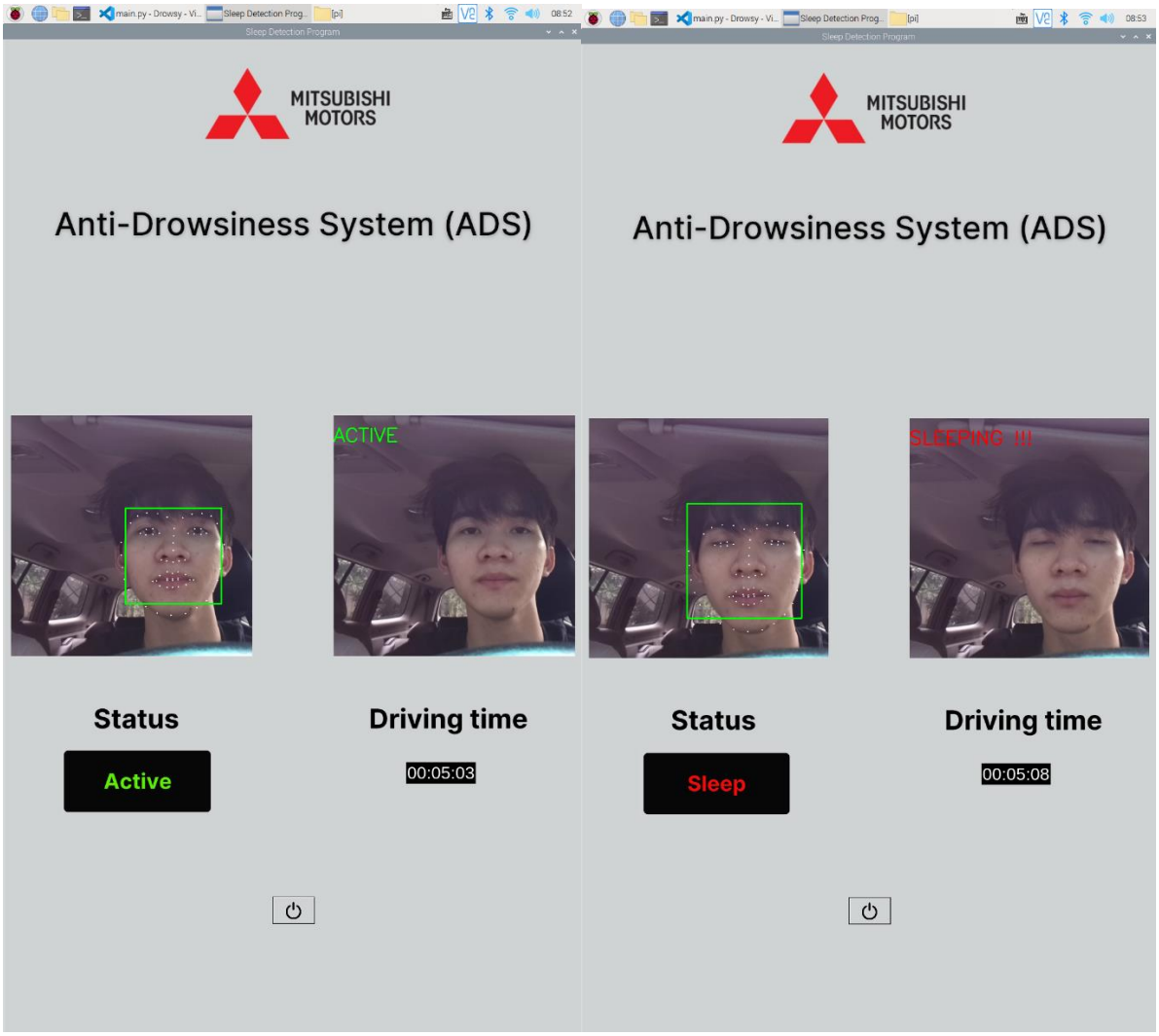

Figure 5.10: Testing on car

Upon conducting the test, several noteworthy conclusions have been drawn. The evaluation of the computer's ability to detect and identify drowsiness revealed exceptional speed and effectiveness owing to its superior processing power. Furthermore, the computer exhibited a higher frames-per-second (FPS) rate, resulting in a seamless test execution devoid of any noticeable delays. However, when the same test was conducted on the Raspberry Pi 4 embedded computer, performance lags became evident. This disparity in performance can be attributed to various factors. Notably, the dissimilarity in processing power and computational capabilities between the two devices plays a significant role. Computers are equipped with advanced and potent hardware components, including swifter processors and increased RAM capacity, enabling them to handle complex tasks with enhanced efficiency. Conversely, the Raspberry Pi, while renowned for its flexibility and compactness, possesses relatively constrained hardware resources, thereby leading to slower execution times for tasks that demand substantial computational resources.

Consequently, optimizing the code or exploring alternative methodologies better aligned with the Raspberry Pi's capabilities becomes imperative to achieve satisfactory performance levels. As a prospective avenue for future development, in circumstances where economic feasibility permits, replacing the primary processor, the Raspberry Pi, with a Jetson Nano could be considered. Such a substitution would likely yield improvements in processing power and computational capacity, potentially alleviating the performance discrepancies observed during the test. The test results on the vehicle, although the recognition is slower than on the computer, still meet the conditions to warn the driver in time.

5.3 Research results

5.3.1 Building an interface that displays user warnings for Tkinter

In this topic, in addition to interacting via voice, users can also interact via touch screen. In the project using 7 inch touch screen. The program will calculate to display the appropriate status for the driver's current condition and at the same time provide audio feedback. To design the user interface in a programming language, we choose the Tkinter library to design. .Because Tkinter is cross-platform compatible and works on major operating systems like Windows, macOS and Linux. This allows developers to create GUI applications that can run seamlessly on different platforms without significant modification. Importantly, it is easy to learn and use as Tkinter has a simple and straightforward API. Relatively easy for beginners to learn and use. Its documentation is extensive and provides clear examples, making it accessible to developers of different skill levels.

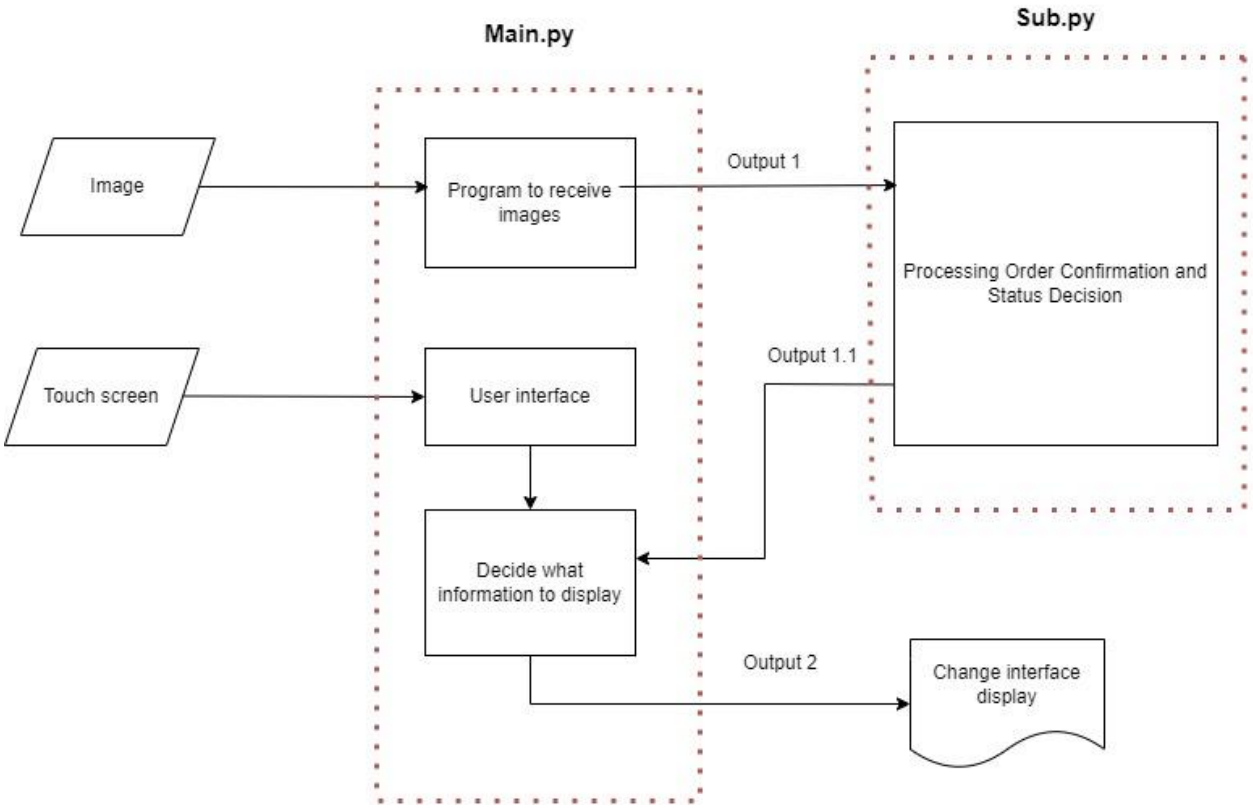

Figure 5.11: Flowchart of user interface algorithm

In addition to design background images and status information, we use Lunacy software to design. Due to its user-friendliness: Lunacy has an intuitive and user-friendly interface that is easy to navigate and use. easy to understand. Its interface closely resembles popular design tools, making it familiar and accessible to both beginners and experienced designers. The tool offers a clean workspace and a host of useful features to streamline the design process.

First, since the size of the 7 inch touch screen is 1920x1080, we will create a panel with the same size .Next, design the Logo section and the information to be displayed such as the location of the status and the driving time.

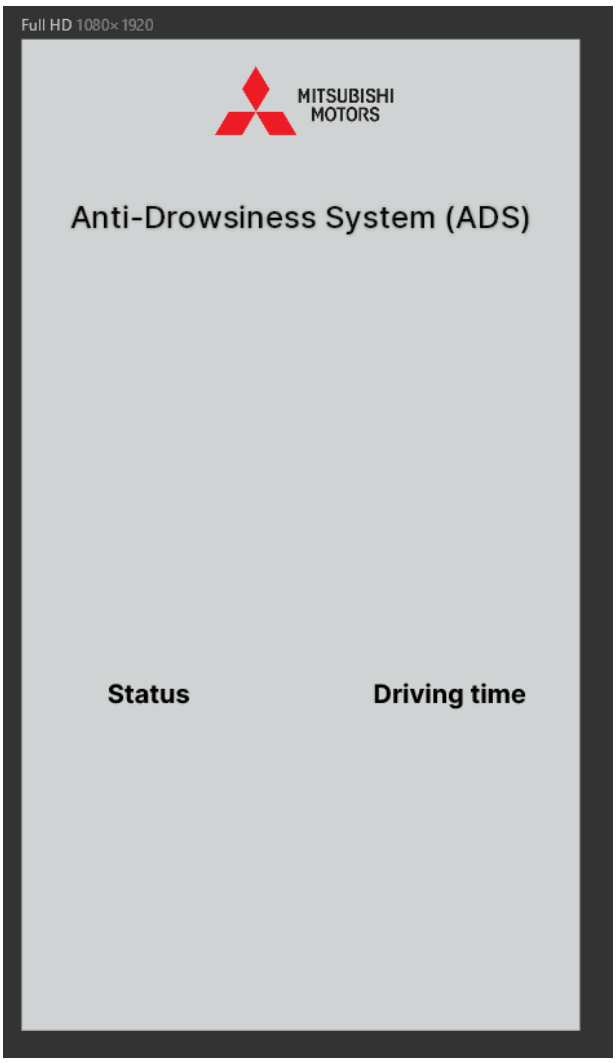

Figure 5.12 : Main interface

```
1.1   from tkinter import *
1.2   from PIL import ImageTk, Image
1.3   import PIL.Image , PIL.ImageTk
1.4   window = Tk()
1.5   window.title("Sleep Detection Program")
1.6   background_path =
Image.open(r"/home/pi/Documents/Drowsy/Photo/BSC37.jpeg")
1.7   background= ImageTk.PhotoImage(background_path)
1.8   window.geometry('1080x1920')    #Can change length and width
1.9   label1 = Label(window,image=background)
1.10  label1.place(x=0,y=0)
1.11  window.mainloop()
```

Line of code 1.4 initializes a window and assigns it to the window variable to manage. The line code1.5 names the window. The 1.6 line of code leads to the background image. The 1.8 line of code specifies the size of the window.

Next, we design the active states including: Active ,Drowsy ,Sleep.

Figure 5.13 :Display status

To be able to display these states, we use the Label function and the syntax used with the icon is an image. The following code passes the path of each state to be displayed into a class called PhotoImage to create an image object that Tkinter can understand.

1.12 img_active = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/Active.png"))

1.13 img_drowsy = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/Drowsy.png"))

1.14 img_sleep = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/Sleep.png"))

1.15 img_none = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/None.png"))

1.16 label_status = Label(window,image=img_none)

1.17 label_status.place(x=110,y=1325)

Then to be able to display the interface according to each state that the function in the program gives will be done by the following code

1.18 label_status.config(image=img_active)

1.19 label_status.config(image=img_drowsy)

1.20 label_status.config(image=img_sleep)

The program will first display the status of None if the driver has not been identified. After identification, depending on the driver's status, a decision will be displayed to display the appropriate status.

5.3.2 Get image from CSI Camera on Raspberry Pi and use model to detect

Image recognition and image processing are considered as an important part of the driver's state recognition process. In the project we choose to use CSI Camera to be able to receive images of the subject, because it is designed specially designed to communicate directly with the connector on the Raspberry Pi board. This allows for direct and seamless communication between the camera and the Raspberry Pi, ensuring optimal performance and compatibility. In addition, compatibility and compactness are also an aspect. widely available and supported by various libraries and software frameworks, including OpenCV and Dlib. This makes it easy to find documentation, examples, and community support for integrating . CSI cameras are typically compact in size, making them suitable for applications with limited space or where portability is required. Overall, choosing a CSI camera for Raspberry Pi provides a seamless and efficient solution. to maximize image quality, compatibility, and performance, making it the ideal choice for various computer vision and imaging applications.

Before installing the library for the environment, you need to install PyPi. This is a library manager for Python. For Python environments 3.4 or later or 2.7.9. or later, PyPi is built in. To install PyPi in Raspberry Pi if Python 2.x with command sudo apt-get install python-pip and for Python 3.x with command sudo apt-get install python3-pip. Allows installing, removing and upgrading libraries with the pip command. With Python 2.x it is recommended to use pip while Python 3 users use pip3 when running the pip command.

To be able to open and read images from the camera, it is necessary to install the OpenCV library, which stands for Open Source Computer Vision Library is a popular open source computer vision and image processing library that supports many languages. Various programmers like C++ ,Python and Java .To install the OpenCV library for Python on Raspberry Pi with the command $ pip install opencv-python .

To be able to recognize facial points we use the file shape_predictor_68_face_landmarks.dat. It is a pre-trained model used in face analysis and computer vision tasks. It is specially designed to detect and locate 68 landmarks on the human face.Start with OpenCV library to get images via Camera:

2.1 video = VideoStream(src=0).start() #Initializing the camera and taking the instance

2.2 frame = video.read() #Contains readable images

2.3 face_frame = video.read()

Next, upload the trained model file :

2.4 face_detect = dlib.get_frontal_face_detector() #Initializing the face detector and landmark detector

2.5 landmark_detect = dlib.shape_predictor(r"/home/pi/Documents/Drowsy/Models/shape_predictor_68_face_landmarks.dat")

5.3.3 Calculation method and recognition of drowsy drivers

After detecting user's eyes, we generate the code that determine the EAR with the formula mentioned in chapter 3.

There is a function programmed to calculate the Euclidian distance between two points in 2D plane thanks to Numpy available function "linalg.norm".

3.1 def compute(self,ptA,ptB):

3.2 return np.linalg.norm(ptA - ptB)

Next, we apply this Euclidian distance formula with 6 points in each eye to calculate EAR. In this function a, b, c, d, e, f are eye points in 68 face landmarks figure.

3.3 def blinked(self,a,b,c,d,e,f):

3.4 up = self.compute(b,d) + self.compute(c,e)

3.5 down = self.compute(a,f)

*3.6 ratio = up/(2.0*down)*

3.7 print(ratio)

3.8 #Checking if it is blinked

3.9 if (ratio > 0.25):

3.10 return 2

3.11 elif (ratio > 0.21 and ratio <= 0.25):

3.12 return 1

3.13 else:

3.14 return 0

With some experiments and research, we conclude that the more the eye are close, the smaller EAR is. Therefore, we set boundaries 0.21, 0.25 to indicate different status of driver:

- EAR > 0.25: active (indicating the eye is open)
- 0.21<EAR<0.25: drowsy (indicating the eye is relatively close)
- EAR<0.21: sleep (indicating the eye is totally close)

```
3.15  if(left_blink== 0 or right_blink==0):
3.16            self.sleep+=1
3.17            self.drowsy=0
3.18           self.active=0
3.19            if(self.sleep>10):
3.20               self.check = 0
3.21        elif(left_blink==1 or right_blink==1):
3.22            self.sleep=0
3.23            self.active=0
3.24           self.drowsy+=1
3.25            if(self.drowsy>6):
3.26               self.check = 1
3.27        else:
3.28            self.drowsy=0
3.29           self.sleep=0
3.30            self.active+=1
3.31           if(self.active>6):
3.32               self.check = 2
```

When the program recognizes the eyes status with EAR value of both eyes, it will count the number of consecutive frames that EAR stays in the range as illustrated. If the frame number exceed the limit we set up, the "flag" mode will be triggered and warn the driver:

- Sleep: mode 0
- Drowsy : mode 1
- Active : mode 2

To avoid while driving, the driver sleeps in positions that the camera cannot detect, inside the program is integrated with a function that if the driver cannot be detected within a certain period of time, it will automatically emits a warning sound to wake the driver to help them focus more.

```
3.33  if len(faces) > 0:
3.34  self.start_time = None
3.35  self.no_face_duration = 0
3.36  elif self.start_time is None:
```

3.37 self.start_time = time.time()

3.38 else:

3.39 self.no_face_duration = time.time() - self.start_time

3.40 if self.no_face_duration >= 5 and speed == 1:

3.41 if (self.last_alert is None) or ((datetime.datetime.utcnow() - self.last_alert).total_seconds() > self.alert_each):

3.42 self.last_alert = datetime.datetime.utcnow()

3.43 t1 = Thread(target=self.no_Face_Alert)

3.44 t1.start()

From lines of code 3.33 to 3.39 used to detect the driver's face for the first time, the variable start_time will have the value None and no_face_duration = 0 , if the face is not detected, the variable start_time will have the current time value and variable no_face_duration stores the time when the face is not found. From the line of code 3.40 to 3.44 is used to see if the driver's face is not detected in 5 seconds combined with the vehicle rolling, it will emit a warning sound.

5.3.4 Method to calculate driving time

According to this study, driving time awareness is crucial for a variety of compliance, efficiency, and safety-related reasons. Monitoring one's driving time aids in avoiding driver weariness. The danger of an accident rises when a person drives for extended periods of time without getting enough rest. Drivers and fleet managers should make sure that drivers have enough breaks and rest periods to avoid fatigue-related issues by being aware of driving times.

Many countries also have rules governing the most hours that commercial drivers may drive. These laws, like the Hours of Work (HOS) rule, are meant to protect public safety on the roads by preventing excessive driver weariness. Drivers and fleet managers may assure compliance with these rules and stay out of trouble by keeping an eye on driving times.

*4.1 seconds = hours * 3600*

4.2 start_time = time.time() #Get current time

4.3 if (speed > 10):

4.4 elapsed_time = int(time.time() - start_time)

4.5 hours_elapsed = elapsed_time // 3600

4.6 minutes_elapsed = (elapsed_time % 3600) // 60

4.7 seconds_elapsed = (elapsed_time % 3600) % 60

4.8 time_str = f"{hours_elapsed:02d}:{minutes_elapsed:02d}:{seconds_elapsed:02d}"

4.9 countdown_label.config(text=time_str) # Time display on interface

4.10 time.sleep(1) # Update the timer once every second

4.11 if elapsed_time >= seconds:

4.12 break

4.13 countdown_label.config(text="Time's up!") #Notification timed out

4.14 playsound(r"/home/pi/Documents/Drowsy/Audio/time_4.mp3") #Warning

To be able to do this we have set up a timer of 4 hours, the value of 4 hours will be calculated in seconds by multiplying by 3600. Elapsed_time calculates the difference between the current time, hours_elapsed calculates the total number of hours elapsed by performing integer division `//`, removing the remainder and returning the quotient ,minutes_elapsed math the remainder after dividing the elapsed time by 3600 seconds (seconds in an hour)) the result is then divided by 60 to get the number of minutes ,seconds_elapsed calculates the number of seconds elapsed after counting both hours and minutes. It uses the `%` operator twice to get the remainder after dividing the elapsed time by both 3600 seconds (hours) and 60 seconds (minutes). This counter only counts when the vehicle's speed is greater than 10.

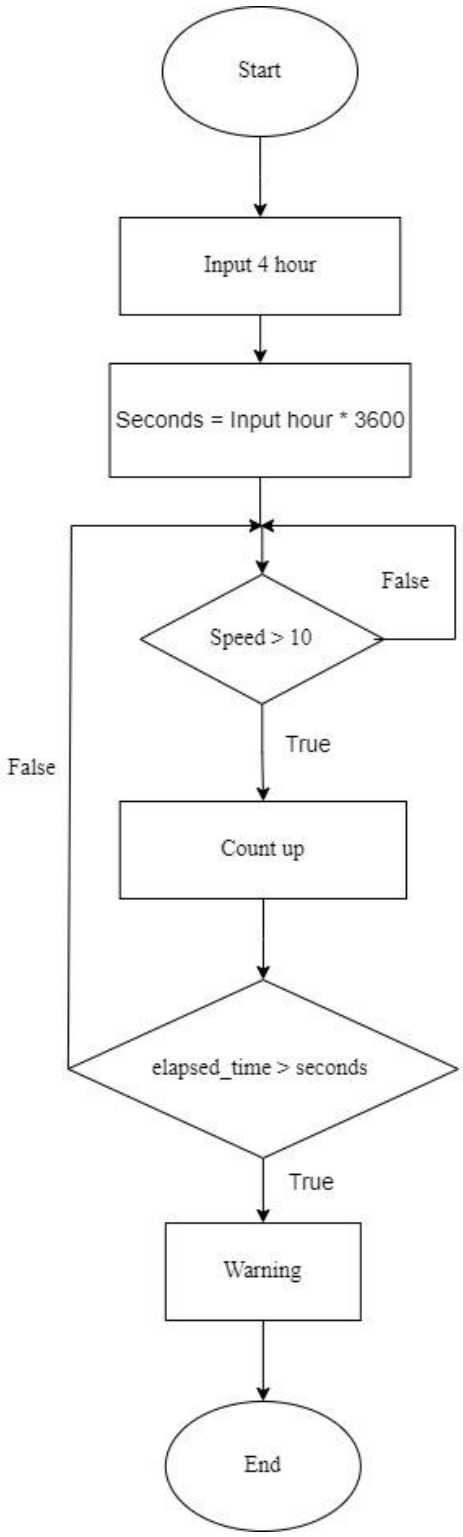

Figure 5.14: Driving time algorithm diagram.

5.3.5 Make a warning sound and play the sound in the program

In this project to be able to wake the driver we use audible warnings to ensure safety.Acoustic alarms provide immediate and noticeable warnings to the driver, collecting draw their attention to potential hazards or critical situations on the road. It also helps to reduce distractions caused by non-visual communication to the driver, keeping them from

having to take their eyes off the road or shift their attention to the display. By providing auditory cues , the driver can focus on driving while receiving important notifications or warnings .

To create audio files during running of the program in this topic, use the gTTS (Google text to speech) library. This library converts a text string into a .mp3 audio file. Install the gTTS library in Python $ pip install gTTS.

5.1 *from gtts import gTTS*

5.2 *text = "Wake up!"# Text to convert to audio*

5.3 *text2audio= gTTS(text)# Create a gTTS object*

5.4 *text2audio.save("Sleep.mp3")# Save the audio file*

In this code to pass a text string "Wake up!" into a .mp3 file. The 5.1 line of code is used to declare the gTTS library, the 5.2 line of code defines the text string to be converted to audio, the 5.3 line of code is the function to convert the text to an mp3 audio file, and the 5.4 line of code will save the mp3 file with the name “Sleep.mp3” .

Also, to be able to attract attention as well as wake up the driver with this audio file is not enough. So we need to add some background sound[23] combined with the above audio file by use Adobe Premiere Pro software, to be able to create an audio file capable of alerting the driver.

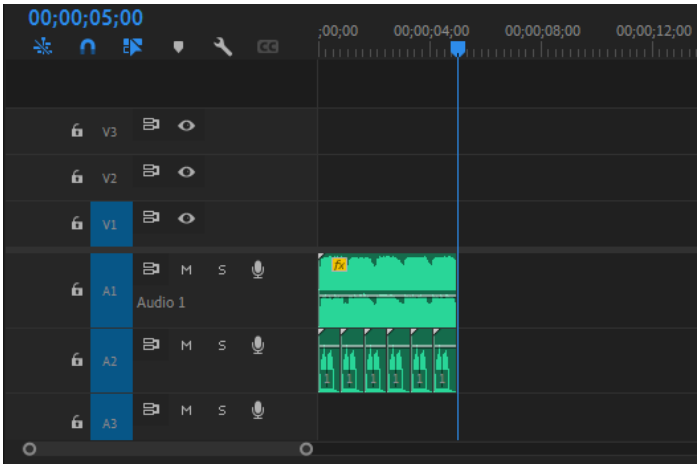

Figure 5.15 :Create sound files capable of waking up

To play audio files using the playsound library, it is necessary to install the library in

Python with command $ pip install playsound

5.5 *from playsound import playsound*

5.6 *playsound("Audio/Wakeup.mp3")*

The code plays an audio file from the "Audio/Wakeup.mp3" path.

5.3.6 Automatically run the program when starting the Raspberry Pi and Shutdown the program

Automatically running python at Raspian OS startup is really necessary for convenience and eliminates unnecessary accessories like keyboard and mouse to be able to run commands manually in terminal.

In order for the Anti-Drowsiness system program to run automatically when powered on, there are many methods on the Raspian operating system. This rc.local method is the simplest and easiest to install to execute a script during system startup and can also be run as root . However, the biggest drawback is that rc.local runs before the Raspbian operating system runs, so the user cannot interact with the user interface in other words, the user interface is not visible on the desktop. Systemd method here is how new to run automatically with the program on Linux it is definitely more complicated than the previous methods, because it runs before the desktop graphics program and the complication is waiting until you have access and the another process or just restart your program over and over until it works. So the systemd method is a powerful way to create and manage services that run in the background. Finally, autostart this method is specially designed for GUI applications because it will automatically run the program after the operating system. The operation is finished so that the user can interact with the interface, the disadvantage is that it requires manual configuration in the user's autostart directory.

In this research topic, using the autostart method is the most optimal because this topic is related to graphics and user interaction. To be able to do this, we will first go to the /home/pi/ directory. config/ and create directory autostart

Then create a text file open as a text editor and add the command lines as shown in Figure 5.16 below save and close the window.In there with Exec= python3 /home/pi/Documents/Drowsy/main.py here is the path to the main.py file that needs to run automatically at startup . With python3 in the path so as not to confuse the program running with any other Python version.

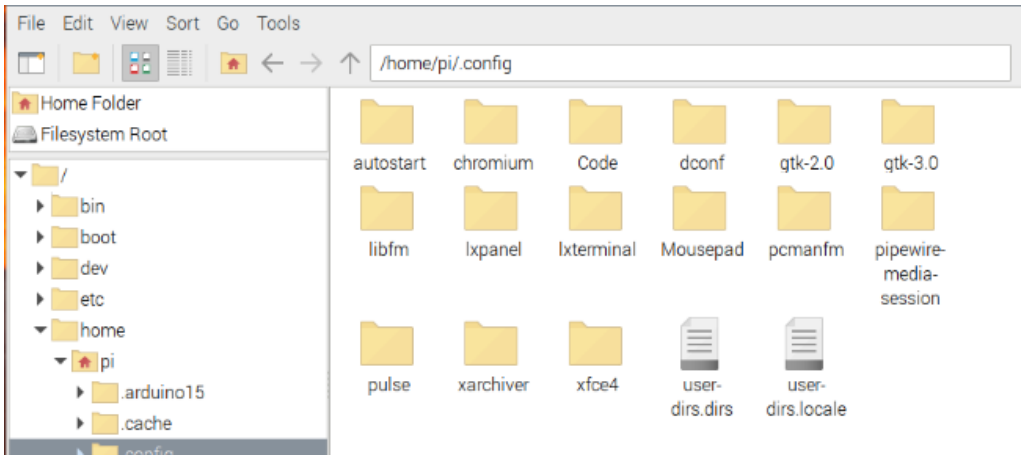

Figure 5.16: Create folder autostart.

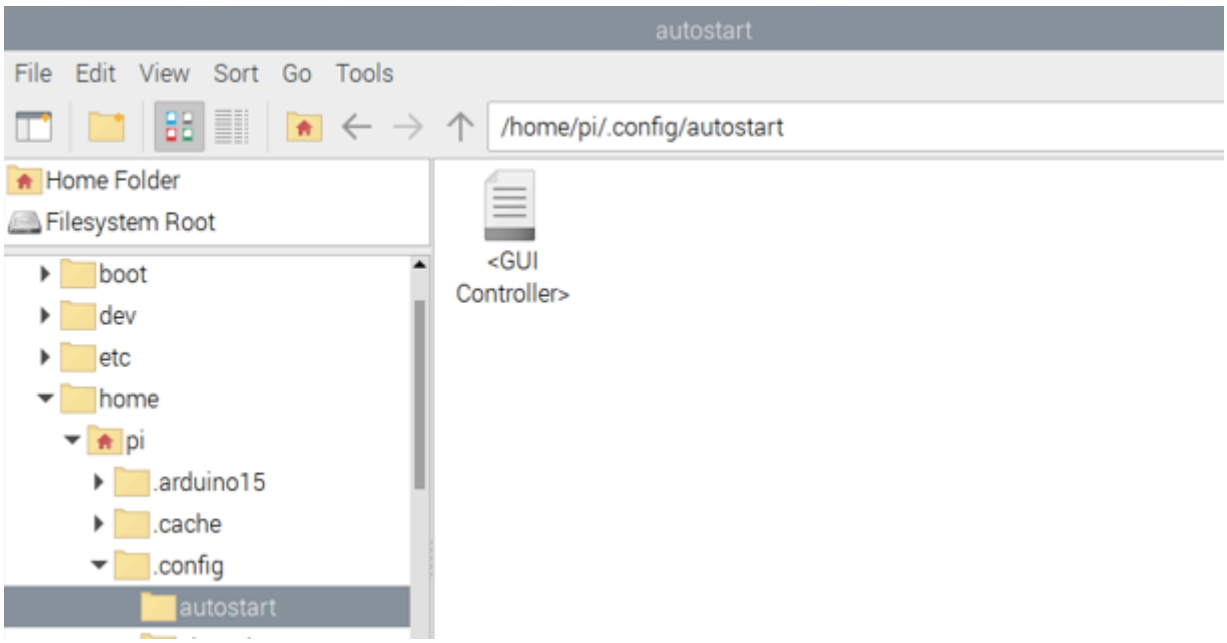

Figure 5.17: Create GUI Controller

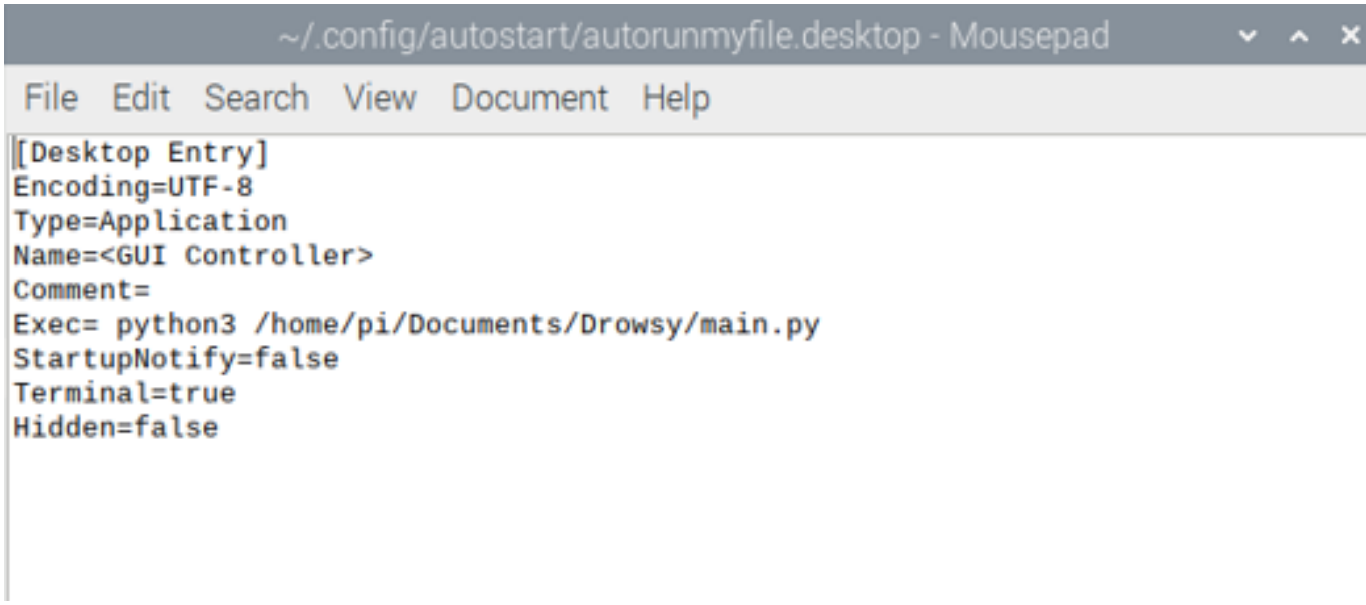

Figure 5.18: Autorun program main.py

To be able to shutdown the program, the user can press the shutdown button by interacting on the screen interface. At this point, the Raspberry pi will shut down with the program and the system will restart until the Raspberry pi has power again.

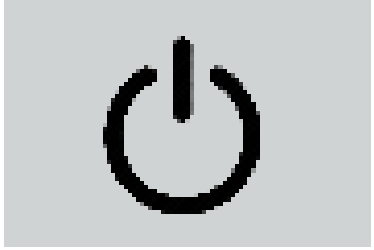

Figure 5.19: Shutdown button

5.3.7 Communication method between Raspberry Pi and Arduino and method of reading speed signal through CAN module

In this topic to be able to get the speed signal we use a CAN module, from which Arduino will send data to the Raspberry Pi 4 Model b. Although the embedded computer has GPIO pins to run the program, it does not. used to interact directly with the CAN module without the addition of an Arduino. The Raspberry Pi is a powerful computer capable of running complex operating systems and executing high-level programming languages, while the Arduino is a microprocessor. The controller is designed for real-time control and communication with various sensors and actuators. By combining the two, one can leverage the computing power of the Raspberry Pi and the I/O capabilities of the Arduino to create more complex and interactive projects.

To be able to communicate between Raspberry and Arduino, in this topic use COM port with pySerial library. This is the module that provides access to the serial ports. It provides backends for Python running on Windows, OSX, Linux... (can be shown to any POSIX-compliant system) and IronPython. This Serial module automatically selects the compatible backend. To install the library $ pip install pyserial.

7.1 import serial

7.2 ser = serial.Serial('/dev/ttyACM0',115200, timeout=1.0)

7.3 time.sleep(3)

7.4 ser.reset_input_buffer()

7.5 if (ser.in_waiting>0):

7.6 speed = int(ser.readline().decode())

In this code two-way communication between Raspberry Pi and Arduino. The 8.1 line of code declares the serial library to communicate between Raspberry (Python language) and Arduino. Line 8.2 declares the USB port on the Raspberry in use and the communication speed 115200 bauds. Line 8.4 clears the serial interface input buffer represented by the variable `ser`. Line 8.5 checks if there is any data waiting to be read in the serial port's input buffer. Line 8.6 reads a line of text from the serial port represented by ser and assigns it to the speed variable.

To acquire the CAN signal with CAN bus shield, we need to connect 2 port CANH and CANL to OBD2 connector in automobile. In this thesis, we conduct this job in Mitsubishi Pajero 2014.

Model name	Mitsubishi Pajero 2014

Engine type	6B31
Number and type of cylinder	V6
Fuel system	Gasoline indirect injection
Horse power net	219HP
Torque net	281Nm

Table 5.1: Mitsubishi Pajero Sport 2014 specifications

First there are some necessary libraries about to install so that we can use MCP2551 module on CAN bus shield.

For vehicles with OBD2 connector, we can get the OBD2 PID code for some standard signal on Wikipedia[24]. Here, vehicle speed signal is defined as 0x0D.

0A	10	1	pressure)	0	765	kPa	$3A$
0B	11	1	Intake manifold absolute pressure	0	255	kPa	A
0C	12	2	Engine speed	0	16,383.75	rpm	$\frac{256A+B}{4}$
0D	13	1	Vehicle speed	0	255	km/h	A
0E	14	1	Timing advance	-64	63.5	° before TDC	$\frac{A}{2}-64$
0F	15	1	Intake air temperature	-40	215	°C	$A-40$
10	16	2	Mass air flow sensor (MAF) air flow rate	0	655.35	g/s	$\frac{256A+B}{100}$
11	17	1	Throttle position	0	100	%	$\frac{100}{255}A$
12	18	1	Commanded secondary air status				Bit encoded. See below
13	19	1	Oxygen sensors present (in 2 banks)				[A0..A3] == Bank 1, Sensors 1-4. [A4..A7] == Bank 2...
14	20	2	Oxygen Sensor 1 A: Voltage				

Figure 5.20: Vehicle speed PID code

Next, a request message (QUERY) must be sent to OBD2 connector via the CAN bus with the following code:

7.7 void sendPid(unsigned char __pid) {

7.8 unsigned char tmp[8] = {0x02, 0x01, 0x0D, 0, 0, 0, 0, 0};

7.9 CAN.sendMsgBuf(CAN_ID_PID, 0, 8, tmp);}

A function named “sendPid” is used to transfer a message that similar to a CAN frame mentioned in Chapter 2. After that the relevant ECU will send back a message (RESPONSE) with the vehicle speed value stored in the fourth member of the array.

There are 10 diagnostic services described in the latest OBD-II standard SAE J1979. Before 2002, J1979 referred to these services as "modes". Because we want to know the “current” value of Vehicle speed, we must check the response message.

Service / Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

Figure 5.21: OBD2 Mode

```
7.10  if (CAN_MSGAVAIL == CAN.checkReceive()) {     // check if get data
7.11          CAN.readMsgBuf(&len, buf);
7.12  if(buf[1] == 0x41) {                          //check if current data
7.13  *s = buf[3];
7.14  return 1;}
```

Finally, the fourth member of the array “buf[3]” is defined as HEX value so we must decode it to decimal value by the formula illustrated vehicle speed in the figure 5.21.

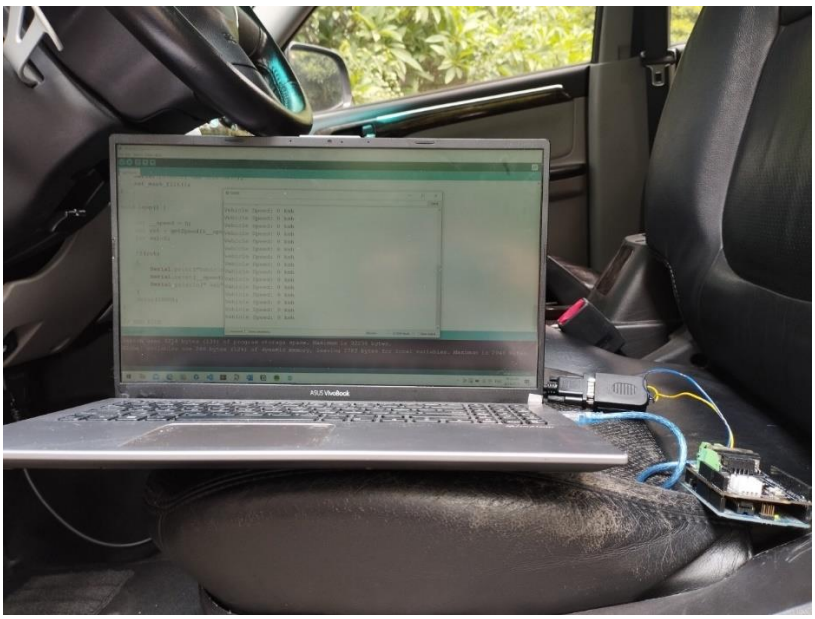

Figure 5.22: Reading data from Mitsubishi Pajero Sport 2014

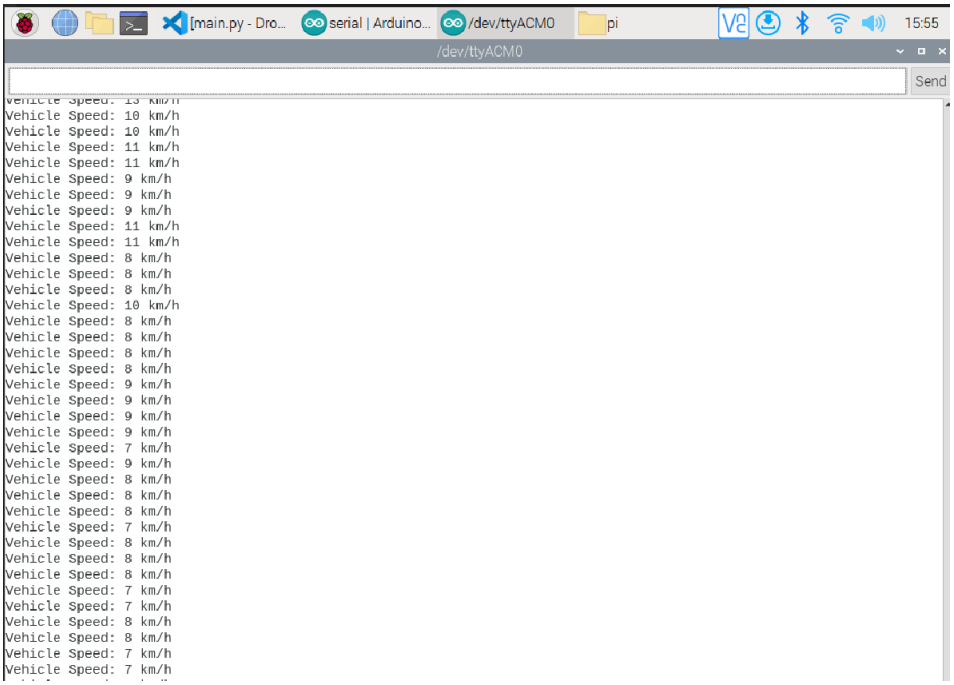

Figure 5.23 : Vehicle speed data

As described, we use vehicle speed data as an initial condition to decide whether Raspberry pi should detect the driver or not so we need to transfer this data from Arduino to it. The used protocol is USB via the available USB port in both devices.

This is a serial communication and to achieve it, we also add a library named “serial” and use command:

7.15 Serial.println()

The Raspberry Pi device will read data that is printed out from Arduino. Therefore, we print out the “speed mode”: 0 or 1 relevant to different speed range:

- Speed range <10 km/h: is defined as 0
- Speed range >10km/h: is defined as 1

And when this data is transferred to Raspberry pi via USB, we also need to decode it using the following function and store it in a integer variable:

7.16 int val = ser.readline().decode('utf-8').rstrip()

We use this signal with logic AND gate to warn the drowsy driver when the speed is bigger than 10 km/h. If we take this signal continuously, it will slow down the processing speed of the system. Therefore, the signal is obtained every 5 seconds.

CHAPTER 6: CONCLUSION AND RECOMMENDATION

6.1 Conclusion

After nearly 5 months of researching the thesis with our tremendous effort and dedicated guidance from Msc Nguyen Thien Dinh. We achieved a certain amount of success beside some limitations.

Achievements:

- Understand the background knowledge of image processing and computer vision. Understand automotive vision technology in general and especially the technology of detecting drowsy driving.
- Apply AI models to solve real-world problems, improve programming in languages like Python, C and the ability to design interfaces for a program.
- Ability to use embedded devices such as Raspberry Pi, Arduino, CAN bus shield as well as communication standards and communication protocols such as Serial Port, SPI, CAN protocol.
- Understand the operation of the CAN bus and successfully retrieve data from a specific vehicle model through OBD2 to connect to the processors.

Limitations:

- System design is mostly tested in a static environment, so it is not possible to evaluate its good performance in many different environments.
- The devices have not been packaged into a complete unit, to increase flexibility in use.
- The ability to handle in the dark is limited without any additional light source.

6.2 Practical significance

Social and economic contribution: Drowsiness warning devices are mostly only equipped in expensive cars. Economically, this project can commercialize the product if it is well invested. version and tested in practice. Thereby helping this device be accessible to more drivers and reduce traffic accidents.

Contribution to research: The topic contributes to a reliable foundation for students to practice creating, researching and extending other AI systems to improve the driver's car experience. Besides creating a foundation for students, in this project, many programming

languages such as Arduino and Python have been used as well as communication standards, especially CAN protocol, which is very useful in the current automotive industry.From there, it can serve the teaching process so that students can go deeper into programming embedded applications in cars. That improves the quality of students and contributes to the development of Vietnam's automobile industry.

6.3 Proposing directions for research and development

- According to the results obtained and in the process of researching the topic, there are still many points that need to be overcome, need to improve, and develop many factors of software and hardware so that the product has the highest optimization but still ensures the highest quality. Due to some difficulties in financial conditions as well as knowledge in this thesis, we have not been able to take advantage of the best equipment to build the model. Therefore, we propose some improvements as follows:
- First about the module's hardware. In this topic using Raspberry Pi because there is no GPU graphics processor, which limits the image processing speed as well as storage memory. Therefore, we recommend upgrading to an embedded computer with a higher configuration such as the Jason Nano. The Jason Nano configuration can completely think of applying AI to computer vision with its extremely powerful GPU processor.
- About the acquisition of images through the group camera towards the simultaneous use of 2 cameras in different conditions. In addition to the current camera, we are using that works well in a well-lit environment, it is possible to apply it more when it detects dark, it will run an infrared camera instead. This improves the ability to identify more flexibly in low light conditions.
- In addition to the speed signal, we read from the CAN BUS signal of the vehicle communication system, the application of additional steering angle sensor signal will help increase the information source for the system to improve accuracy.

About the software we want to add many features such as:

- When the driver falls asleep, the warning system still cannot wake up. At this time, it is necessary to affect the vehicle's control system, we will use an additional GPS module to find the nearest parking space and combined with the car's self-driving system to get there. This will significantly contribute to reducing traffic accidents.
- In addition, another feature is that if the vehicle has a traffic collision, based on the vehicle speed to detect it, then send an emergency signal, take a screenshot of the scene and information at the time of the accident problem.

REFERENCE

[1] Yen Chi. " Báo động gia tăng TNGT do tài xế ngủ gật, mệt mỏi". atgt.baogiaothong.vn. https://atgt.baogiaothong.vn/bao-dong-gia-tang-tngt-do-tai-xe-ngu-gat-met-moi-d560157.html

[2] New Vehicle General Safety Regulation. "New rules to improve road safety and enable fully driverless vehicles in the EU". https://ec.europa.eu. https://ec.europa.eu/commission/presscorner/detail/en/ip_22_4312

[3] Đỗ Văn Linh, "Nghiên cứu hệ thống cảnh báo ngủ gật và mất tập trung cho người lái xe", Graduation Thesis. Vehicle and Energy engineering Department, HCMC University of Technology and Education.

[4] Thái Thị Hoà Vân, "Nghiên cứu tình trạng buồn ngủ của người lái xe dựa trên nhận dạng cử chỉ khuôn mặt". Master Thesis , Da Nang University

[5] Nguyen Dinh Quan " Ứng dụng sóng não phát hiện dấu hiệu buồn ngủ và đưa ra tín hiệu cảnh báo đối với người lái xe", Master Thesis, Vehicle and Energy engineering Department, HCMC University of Technology and Education.

[6] Robert Chen-Hao Chang, Chia-Yu Wang, Wei-Ting Chen and Cheng-Di Chiu, "Drowsiness Detection System Based on PERCLOS and Facial Physiological Signal", IEEE International Conference on Consumer Electronics, Taiwan (IEEE 2021 ICCE-TW), Penghu, Taiwan, 15–17 September 2021.

[7] Anjith George, " Design and implementation of real-time algorithms for eye tracking and PERCLOS measurement for on board estimation of alertness of drivers", Master thesis, Indian Institute of Technology, Kharagpur, April 2012.

[8] Tayaba Azim, M.Arfan Jaffar, Anwar M. Mirza, "Fully automated real time fatigue detection of drivers through Fuzzy Expert Systems", Applied Soft Computing 18 in 2014.

[9] Jyotsna Gabhane, Dhanashri Dixit, Pranali Mankar, Ruchika Kamble, Sayantani Gupta, “Drowsiness Detection and Alert System: A Review”, Volume 6 Issue IV International Journal for Research in Applied Science & Engineering Technology (IJRASET).

[10] Smart Eye, “Driver monitoring system”, https://smarteye.se/solutions/automotive/driver-monitoring-system

[11] BOSCH, “Interior Monitoring System”,https://www.bosch-mobility.com/en/solutions/interior/interior-monitoring-systems

[12] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (pp. 1097-1105).

[13] Burger, W., & Burge, M. J. (2016). Digital Image Processing: An Algorithmic Introduction using Java. Springer International Publishing.

[14] Kazemi, V., & Sullivan, J. (2014). One Millisecond Face Alignment with an Ensemble of Regression Trees. In 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1867-1874). IEEE.

[15] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788). IEEE.

[16] Zhang, Z., Luo, P., Loy, C. C., & Tang, X. (2016). Facial Landmark Detection by Deep Multi-task Learning. In European Conference on Computer Vision (ECCV) (pp. 94-108). Springer.

[17] “CAN in Automation (CiA),” CAN in Automation (CiA): CAN data link layers in some detail. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-data-link-layers/.

[18] R. Bosch, Bosch Automotive Handbook, 7th ed. Plochingen, DE: Wiley, 2007, pp. 1086-1103.

[19] A. Winning, “Number of Automotive ECUs Continues to Rise,” eeNews Automotive, 16-May-2019.

https://www.eenewsautomotive.com/news/number-automotive-ecus-continues-ris

[20]Martin Falch, “CAN bus the ultimate guide”, April 2022, https://www.csselectronics.com/pages/can-bus-ultimate-guide

[21] Azim Eskandarian, “Handbook of Intelligent Vehicles”, Section 7: Drowsy and Fatigued Driver Detection, Monitoring, Warning.

[22] Tereza Soukupova and Jan Cech, “Real-Time Eye Blink Detection using Facial Landmarks”, 21st Computer Vision Winter Workshop (February 3–5, 2016).

[23] “Source sound” ,https://pixabay.com/sound-effects/search/alert/

[24] OBD2 PID, https://en.wikipedia.org/wiki/OBD-II_PIDs.

APPENDIX

Appendix 1: Main program code Main.py

```
import dlib
from imutils.video import VideoStream
from sub import Functions
from tkinter import *
from PIL import ImageTk, Image
import os
import time
from threading import Thread,Event
from preferredsoundplayer import playsound
import serial

def shutdown():
    os.system("sudo shutdown -h now")

def status_active():
    label_status.config(image=img_active)
def status_drowsy():
    label_status.config(image=img_drowsy)
def status_sleep():
    label_status.config(image=img_sleep)

def update_frame():
    global RCV ,LCV ,speed
    if (ser.in_waiting>0):
        speed = int(ser.readline().decode())
    frame = video.read()
    face_frame = video.read()
```

```
    check                                                                    =
method.calculate(frame,face_detect,landmark_detect,face_frame,RCV,LCV,speed)
    if (check == 2):
        status_active()
    elif (check == 1):
        status_drowsy()
    elif (check == 0):
        status_sleep()
    window.after(15 ,update_frame)

def timer(hours,countdown_label):
    seconds = hours * 3600
    start_time = time.time()

    while  not stop_event.is_set():
        if (speed == 1):
            elapsed_time = int(time.time() - start_time)
            hours_elapsed = elapsed_time // 3600
            minutes_elapsed = (elapsed_time % 3600) // 60
            seconds_elapsed = (elapsed_time % 3600) % 60
            time_str = f"{hours_elapsed:02d}:{minutes_elapsed:02d}:{seconds_elapsed:02d}"
            countdown_label.config(text=time_str)
            time.sleep(1)
            if elapsed_time >= seconds:
                    break
    countdown_label.config(text="Time's up!")
    playsound(r"/home/pi/Documents/Drowsy/Audio/time_4.mp3")
    if stop_event.is_set():
        countdown_label.config(text="Timer stopped!")
```

```
def on_close():
    global stop_event
    stop_event.set()
    window.destroy()

if __name__ == '__main__':
    speed = 0
    #************************CAN************************
    ser = serial.Serial('/dev/ttyACM0',115200, timeout=1.0)
    time.sleep(3)
    ser.reset_input_buffer()
    #========================= Load Method ==========================
    method = Functions()
    #========================= Interface ==========================
    window = Tk()
    window.title("Sleep Detection Program")
    #========================= Background ==========================
    background_path = Image.open(r"/home/pi/Documents/Drowsy/Photo/BSC37.jpeg")
    background= ImageTk.PhotoImage(background_path)
    window.geometry('1080x1920')
    label1 = Label(window,image=background)
    label1.place(x=0,y=0)
    #========================= Button ==========================
    img_button = Image.open(r"/home/pi/Documents/Drowsy/Photo/button1.png")
    img_button = ImageTk.PhotoImage(img_button)
    button = Button(label1,image=img_button,command=shutdown)
    button.place(x=500,y=1600)
    #========================= Status ==========================
    img_active                                                    =
ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/Active.png"))
```

```
    img_drowsy = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/Drowsy.png"))
    img_sleep = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/Sleep.png"))
    img_none = ImageTk.PhotoImage(Image.open(r"/home/pi/Documents/Drowsy/Photo/None.png"))
    label_status = Label(window,image=img_none)
    label_status.place(x=110,y=1325)
    #======================== Canvas ========================
    RCV = Canvas(window,width=450,height=450)
    RCV.place(relx=0.57, rely=0.5, anchor=W)
    LCV = Canvas(window,width=450,height=450)
    LCV.place(relx=0.43, rely=0.5, anchor=E)

    countdown_label = Label(window,text='',font=("Arial",50),fg="white",background="black")
    countdown_label.place(x=700,y=1350)
    #Initializing the face detector and landmark detector
    face_detect = dlib.get_frontal_face_detector()
    landmark_detect = dlib.shape_predictor(r"/home/pi/Documents/Drowsy/Models/shape_predictor_68_face_landmarks.dat")
    #Initializing the camera and taking the instance
    video = VideoStream(src=0).start()
    #======================== Countdown ========================
    stop_event = Event()
    window.protocol("WM_DELETE_WINDOW",on_close)
    time_thread = Thread(target=timer , args=(4,countdown_label))
    time_thread.start()

    update_frame()
```

```
    window.mainloop()
```

Appendix 2: Code of signal processing program sub.py

```
import cv2
import numpy as np
from imutils import face_utils
from preferredsoundplayer import playsound
from threading import Thread
import datetime
from tkinter import *
import PIL.Image , PIL.ImageTk
import time

class Functions():
    def __init__(self):
        #Status marking for current state
        self.sleep  = 0
        self.drowsy = 0
        self.active = 0
        self.status = ""
        self.color  = None
        self.check  = None
        self.audio  = r"/home/pi/Documents/Drowsy/Audio/Wakeup.mp3"
        self.audio_no_face = r"/home/pi/Documents/Drowsy/Audio/no_face.mp3"
        self.last_alert = None
        self.alert_each = 10
        self.RCVPhoto = None
        self.LCVPhoto = None
        self.start_time = None
```

```
        self.no_face_duration = 0
        self.pre_condition = False

    def compute(self,ptA,ptB):
        return np.linalg.norm(ptA - ptB)

    def blinked(self,a,b,c,d,e,f):
        up = self.compute(b,d) + self.compute(c,e)
        down = self.compute(a,f)
        ratio = up/(2.0*down)
        #Checking if it is blicked
        if (ratio > 0.25):
            return 2
        elif (ratio > 0.21 and ratio <= 0.25):
            return 1
        else:
            return 0

    def subalert(self):
        playsound(self.audio)

    def no_Face_Alert(self):
        playsound(self.audio_no_face)

    def alert(self,frame,speed):
        if (self.check == 0):
            self.status = "SLEEPING !!!"
            self.color = (0,0,255)
            if (self.last_alert is None) or ((datetime.datetime.utcnow() -
self.last_alert).total_seconds() > self.alert_each):
```

```
            self.last_alert = datetime.datetime.utcnow()
            if (speed == 1):
                t1 = Thread(target=self.subalert)
                t1.start()
        elif (self.check == 1):
            self.status = "DROWSY !!"
            self.color = (255,0,0)
        else :
            self.status="ACTIVE"
            self.color = (0,255,0)
            self.pre_condition = True

cv2.putText(frame,self.status,(0,50),cv2.FONT_HERSHEY_SIMPLEX,1.2,self.color,2)

    def calculate(self,frame,face_detect,landmark_detect,face_frame,RCV,LCV,speed):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_detect(gray)
        if (self.pre_condition == True):
            if len(faces) > 0:
                self.start_time = None
                self.no_face_duration = 0
            elif self.start_time is None:
                self.start_time = time.time()
            else:
                self.no_face_duration = time.time() - self.start_time
            if self.no_face_duration >= 5 and speed == 1:
                if (self.last_alert is None) or ((datetime.datetime.utcnow() -
self.last_alert).total_seconds() > self.alert_each):
                    self.last_alert = datetime.datetime.utcnow()
                    t1 = Thread(target=self.no_Face_Alert)
```

```
            t1.start()

    #Detected face in faces array
    for face in faces:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()
        face_frame = frame.copy()
        cv2.rectangle(face_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        landmarks = landmark_detect(gray, face)
        landmarks = face_utils.shape_to_np(landmarks)
        #The numbers are actually the landmarks which will show eye
        left_blink = self.blinked(landmarks[36],landmarks[37],
            landmarks[38], landmarks[41], landmarks[40], landmarks[39])
        right_blink = self.blinked(landmarks[42],landmarks[43],
            landmarks[44], landmarks[47], landmarks[46], landmarks[45])

        if(left_blink== 0 or right_blink==0):
            self.sleep+=1
            self.drowsy=0
            self.active=0
            if(self.sleep>10):
                self.check = 0
        elif(left_blink==1 or right_blink==1):
            self.sleep=0
            self.active=0
            self.drowsy+=1
            if(self.drowsy>6):
                self.check = 1
```

```
            else:
                self.drowsy=0
                self.sleep=0
                self.active+=1
                if(self.active>6):
                    self.check = 2
            self.alert(frame,speed)

            for n in range(0,68):
                (x,y) = landmarks[n]
                cv2.circle(face_frame,(x,y),1,(255,255,255),-1)
        #Opencv is BGR but tkinter is RGB
        frame = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        face_frame = cv2.cvtColor(face_frame,cv2.COLOR_BGR2RGB)
        # Put the frame in the photo, convert the readable image into an image to display it
        self.RCVPhoto = PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(frame))
        self.LCVPhoto                                                    =
PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(face_frame))
        #Show
        RCV.create_image(0,0,image = self.RCVPhoto,anchor=NW)
        LCV.create_image(0,0,image = self.LCVPhoto,anchor=NW)

        return self.check
```

Appendix 3: Getting speed signal on Arduino serial.ino

```
#include <SPI.h>
#include "mcp_can.h"
#define SPI_CS_PIN  10
```

```
MCP_CAN CAN(SPI_CS_PIN);                                    // Set CS pin

#define PID_VEHICLE_SPEED   0x0D
#define CAN_ID_PID          0x7DF

void set_mask_filt()
{
   // set mask, set both the mask to 0x3ff
   CAN.init_Mask(0, 0, 0x7FC);
   CAN.init_Mask(1, 0, 0x7FC);
   CAN.init_Filt(0, 0, 0x7E8);
   CAN.init_Filt(1, 0, 0x7E8);
   CAN.init_Filt(2, 0, 0x7E8);
   CAN.init_Filt(3, 0, 0x7E8);
   CAN.init_Filt(4, 0, 0x7E8);
   CAN.init_Filt(5, 0, 0x7E8);
}

void sendPid(unsigned char __pid) {
   unsigned char tmp[8] = {0x02, 0x01, __pid, 0, 0, 0, 0, 0};
   CAN.sendMsgBuf(CAN_ID_PID, 0, 8, tmp);
}
bool getSpeed(int *s)
{
   sendPid(PID_VEHICLE_SPEED);
   unsigned long __timeout = millis();

   while(millis()-__timeout < 1000)      // 1s time out
   {
      unsigned char len = 0;
```

```
        unsigned char buf[8];

        if (CAN_MSGAVAIL == CAN.checkReceive()) {            // check if get data
            CAN.readMsgBuf(&len, buf);

            if(buf[1] == 0x41)
            {
                *s = buf[3];
                return 1;
            }
        }
    }

    return 0;
}

void setup() {
    Serial.begin(115200);
    while(!Serial);
    while (CAN_OK != CAN.begin(CAN_1000KBPS)) {            // init can bus : baudrate
= 500k
        Serial.println("CAN init fail, retry...");
        delay(100);
    }
    Serial.println("CAN init ok!");
    set_mask_filt();
}

void loop() {
```

```
    int __speed = 0;
    int ret = getSpeed(&__speed);
    int val=0;


    if(ret)
    {
        if (__speed <= 3)
        {
            val=0;
        }else val=1;
        Serial.println(val);
    }
    //if(ret) { Serial.print("Vehicle Speed: "); Serial.print(__speed); Serial.println(" km/h");
}
    delay(3000);
}
// END FILE
```

Appendix 4: System operation video

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

Ho Chi Minh City, July, 2023

ADVISOR'S EVALUATION SHEET

Student name: Phan Quốc Bảo — Student ID: 19145136

Student name: Mai Đức Tùng — Student ID: 19145187

Major: Automotive Engineering Technology

Project title: Research and design a driver drowsiness detection system.

Advisor: Nguyễn Thiện Dinh, MS.

EVALUATION

1. Content and workload of the project

- Students have detected the user's face by image processing method and tested with different lighting conditions.
- Students have built a display interface for ADS system on the monitor
- This system uses the vehicle speed value from CAN network to activate system,...

2. Strengths:

- Students used image processing method to perform functions.
- Beside that, this system is connected to CAN network, which is a new method and adaptive to many kind of modern vehicle easily.

3. Weaknesses:

- The delay time of this system needs to be improved.
- Warning only, this system should take some action.

4. Approval for oral defense? *(Approved or denied)*

Approved

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

Excellent

6. Mark: 10 (*in words*: Ten)

Ho Chi Minh City, July , 2023

ADVISOR

(Sign with full name)

Nguyễn Thiện Dinh

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom– Happiness

Ho Chi Minh City, July , 2023

PRE-DEFENSE EVALUATION SHEET

Student name: Phan Quốc Bảo Student ID: 19145136

Student name: Mai Đức Tùng Student ID: 19145187

Major: Automotive Engineering Technology

Project title: Research and design a driver drowsiness detection system.

Name of Reviewer: Assoc. Prof. Ly Vinh Dat

EVALUATION

1. Content and workload of the project

The project has built a driver drowsiness detection system that detects drowsiness driver in driving. Content & workload meets a capstone project requirements.

2. Strengths:

Students have used detection algorithm for recognizing drowsy drivers and display in interface of monitor in different conditions.

3. Weaknesses:

The system need test many times for conclusion. & The experiment must to perform in various drivers

4. Approval for oral defense? *(Approved or denied)*

Approval

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

Good

6. Mark: 9./10 (*in words*: Nine for baths)

Ho Chi Minh City, July 8th, 2023

REVIEWER

(Sign with full name)

Ly Vinh Dat

iii