Ministry of Higher Education and Scientific Research
Higher Institute of Engineering in El-Shorouk City
Department of Communications and Computer Engineering

Intelligent Driving System (IDS)

Submitted in partial fulfillment for the requirements for the degree of Bachelor's in Electronics and Communication \ Computer and Control Engineering

Project Team

Ahmed El-Qazzafi Morsi Ahmed
Ahmed Hany Sa'ad Yassin
Eslam Mohamed Hassan Hassan
Hamza Abd-Elnaser Ali Soliman
Osama Ahmed Reda Mohamed Hegazy

Supervisors

Dr. Khalid Mohamed Tawfik El-Minshawy
Higher Institute of Computer Science and Information Technology, Elshorouk Academy

Eng. Alaa Abushysha Awd
Communication and Computer Engineering Dept., Higher Institute of Engineering, Elshorouk Academy

Cairo 2024

Acknowledgement:

- We would like to express our deepest gratitude to our esteemed professors, Dr. Khalid El-Minshawy and Eng. Alaa Abushysha, for their invaluable guidance, support, and encouragement throughout our graduation project. Their insights and feedback have been instrumental in shaping this work and driving us to achieve our best.
- We are profoundly grateful to our parents and families for their endless support and encouragement. Their belief in us has been a constant source of motivation and strength. Special thanks to our friends and colleagues for their companionship and understanding during the challenging times of this project.
- We also wish to extend our appreciation to our classmates and colleagues in the Communication and Computer Engineering Department at Elshorouk Academy.
- We appreciate the financial support provided by Elshorouk Academy, which made this project possible.
- Finally, we would like to acknowledge everyone at Elshorouk Academy who has contributed to our academic journey. Thank you.

Abstract:

- Transportation networks and services are impacted by a driver's mental and physical health, and their safety as a reduced alertness and focus or a temper can risk the lives of passengers and drivers. In addition to that, Highways and travel roads need fast response to severe medical conditions. and a way to prevent hijacks and even track them. To solve this issue, we are developing a Computer Vision based Driver Monitoring system, that uses AI to analyze and respond to various driver behaviors in real-time using facial and body landmarks, alerting them when asleep or distracted, and contacting help in emergencies using cellular technology, and making use of Geolocation. Our real-time system leverages data analysis and prediction based on past data to proactively prevent alerts and dangers.
- Drowsiness detection is a critical area of research aimed at enhancing safety in various domains, particularly in transportation. This project focuses on the development of a real-time drowsiness detection system to prevent accidents caused by driver fatigue. Utilizing advanced image processing and machine learning techniques, our system continuously monitors the driver's facial expressions and eye movements to identify signs of drowsiness.
- The core of the system involves a camera that captures real-time video of the driver's face. The video feed is analyzed using facial landmark detection algorithms to track eye blink rate, eye closure duration, and yawning frequency. These parameters are fed into a machine learning model trained to recognize patterns indicative of drowsiness. Upon detecting signs of drowsiness, the system triggers an alert to wake the driver, thereby enhancing road safety.
- Our research includes the design and implementation of the detection algorithm, as well as extensive testing and validation using datasets of diverse facial expressions and driving conditions. The results demonstrate that our system achieves high accuracy and reliability in detecting drowsiness, outperforming existing solutions.
- This project was conducted within the Communication and Computer Engineering Department at Elshorouk Academy, with financial support from the academy. The outcomes of this research have significant implications for reducing road accidents and improving transportation safety.

Table of Contents: Page:

List of figures:

List of tables:

List of abbreviations:

Abbreviation	Name
IDS	Intelligent Driving System
NRSC	National Road Safety Commission of Ghana
DDAS	Drowsing Driver Alert System -
WHO	World Health Organization
DLT	Department of Land Transport
OpenCV	Open-Source Computer Vision Library
DL	Deep Learning
ML	Machine Learning
GPU	Graphics Processing Unit
SciPy	Scientific Python
TinyML	Tiny Machine Learning
API	Application Programming Interface
XGBoost	Extreme Gradient Boosting
IoT	Internet of Things
CPU	Central Processing Unit
GPIO	General Purpose Input/Output
UART	Universal Asynchronous Receiver-Transmitter
RAM	Random Access Memory
PC	Personal Computer
SD	Secure Digital -
USB	Universal Serial Bus
LAN	Local Area Network
US	United States
LPDDR4X	low Power Double Data Rate 4X
SDRAM	Single Data Rate Synchronous Dynamic Random-Access Memory
MIPI CSI/DSI	Mobile Industry Processor Interface - Camera Serial Interface/Display Serial

	Interface
HDMI	High-Definition Multimedia Interface
WI-FI	Wireless Fidelity
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GPS	Global Positioning System
MCU	Microcontroller Unit
SDK	Software Development Kit
CSI	Camera Serial Interface
CM	Centimeter
LED	Light Emitting Diode
MHz	Megahertz
IR	Infrared
DOT	Department of Transportation
AI	Artificial Intelligence
CSV	Comma-Separated Values
LiDAR	Light Detection and Ranging
ADAS	Advanced Driver-Assistance System
EEG	Electroencephalogram
NumPy	Numerical Python
MediaPipe	Media Processing Framework
PnP	Perspective-n-Point
FPS	Frames Per Second
DRX	Discontinuous Reception
3GPP	3rd Generation Partnership Project
TS	Technical Specification
Kbps	Kilobits per second
CS	Coding Scheme
PBCCH	Packet Broadcast Control Channel
SMS	Short Message Service
PDU	Protocol Data Unit
HR	Half Rate
FR	Full Rate
EFR	Enhanced Full Rate
AMR	Adaptive Multi-Rate
SIM	Subscriber Identity Module
bps	Bit per second
I2C	Inter-Integrated Circuit
SDMMC	Secure Digital Multi Media Card
ADC	Analog-to-Digital Converter
S	Seconds
°C	Degrees Celsius
g	Grams
Pi	Product Information
BDS	BeiDou Navigation Satellite System

MP	Mass Product
BD	BeiDou
SMD	Surface-Mount Device
W	Watt
V	Volt
mA	Milliampere
BCM	Broadcom
GHz	Gigahertz
ARM	Acorn RISC Machine
L2	Level 2
L3	Level 3
ES	Embedded System
SKU	Stock Keeping Unit
Gbps	Gigabits per second
PoE	Power over Ethernet
BLE	Bluetooth Low Energy
HAT	Hardware Attached on Top
RTC	Real-Time Clock
HDR	High Dynamic Range
HEVC	High-Efficiency Video Coding
SDRI	Single Data Rate Interface
UI	User Interface
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
REST	Representational State Transfer
NoSQL	Not Only SQL (Structured Query Language)
OAuth	Open Authorization

CHAPTER 1

Introduction

1.1. Introduction:

- In Ghana, vehicular accidents have become one of the growing concerns in recent times. Accidents have a tremendous effect on people, property and the environment, regardless of how minute they are. Driving is an everyday task where the driver must be conscious of the road and be aware of the dangers that may lurk. The driver must always be ready to make a split-second decision based on whatever may come his/her way. A driver must be vigilant and in the right frame of mind to effectively respond to external conditions while driving. Drowsy driving is the operation of a vehicle while being cognitively impaired due to lack of sleep.
- Many researchers believe that vehicular accidents are caused by drowsiness when driving tired, drunk or after taking some form of medication. National Road Safety Commission of Ghana has estimated that 6 people are killed every day and 2000 people die annually due to road crashes in Ghana. Among these accidents, some were caused by drowsing drivers at the steering wheel. It is against this background problem that the Drowsing Driver Alert System is being proposed.
- Driving with drowsiness is one of the main causes of traffic accidents. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident-avoidance systems. Due to the hazard that drowsiness presents on the road, methods need to be developed for counteracting its effects. In this paper, we propose a drowsy driver alert system known as Drowsing Driver Alert System (DDAS). The system is designed to keep drivers alert while driving on the road and to reduce the number of road accidents.

1.2. Motivation:

- The number of people killed in car-related traffic incidents has been steadily rising in recent years. Due to Thailand Development Research Institute data which refers from World Health Organization (WHO) in 2018, Thailand was the 9th highest rate of car accidents of the world. On average, the rate of deaths from car accidents was 22,491 persons annually, or about 32.7 persons to 100,000 persons. It means there are Thai people die from accidents 3 persons per hour. Moreover, due to

information from DLT, the third major factor that causes public transport accidents was drowsiness as it happened 43 times (10.39 % of all road accidents) and the accident time occurred the most is 04.01am.- 08.00 am. public transportation accidents are a serious issue as they can affect to a big number of people. As the problem mentioned above, we criticize that if the system is set up in all public transportation vehicles for detecting driver's drowsiness, it can reduce death rate significantly.

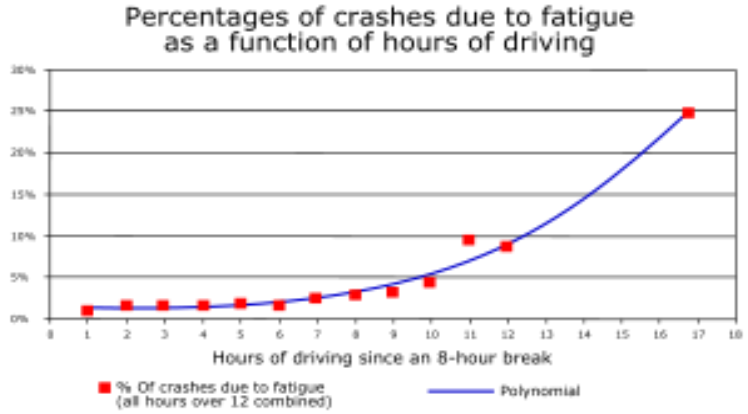

Figure 1.1 Effects of fatigue on safety

1.3. Objectives:

- Develop an algorithm for analyzing drowsiness from a driver's face using facial landmarks.
- Develop a library function program for analyzing drowsiness.
- Implement a system for real-time drowsiness detection using Nvidia Jetson Nano.
- Evaluate the efficiency of the algorithm and system
- Swiftly detecting and preventing sleep and lack of focus for casual drivers
- Providing fast response time to emergency scenarios and a security system for fleets and traveling workers with a detailed report
- Provide analysis of driver behaviors to legally aid the transportation industry
- Provide detailed analysis on accidents, and emergencies and their predicted location through a sophisticated database for the DOT

1.4. Related works:

- There were numerous cases in the world and in Egypt where aspiring creators and engineers tackled the issue of driver drowsiness causing numerous accidents and figuring out ways to detect and prevent these incidents. In this section we explore a few of these cases:
    - ➢ Worldwide: Detection of Drowsiness from Facial Images in Real-Time Video Media using Nvidia Jetson Nano
- From the previous studies, it found that face detecting with facial landmarks is the most effective method because it can detect face and separate its components to process outputs. Hence, this method was chosen to analyze data.
- Methods:
    1. Facial landmarks:
        - ➢ Facial landmarks are points on a face. Previously, OpenCV could generate face components called Haar Cascade. However, a big drawback of OpenCV is not being able to point out the location of each component. Now, there is a program called Deep Learning which indicate all components and its location by using library Dlib that shows 68 points on a face “Fig. 1.2”.

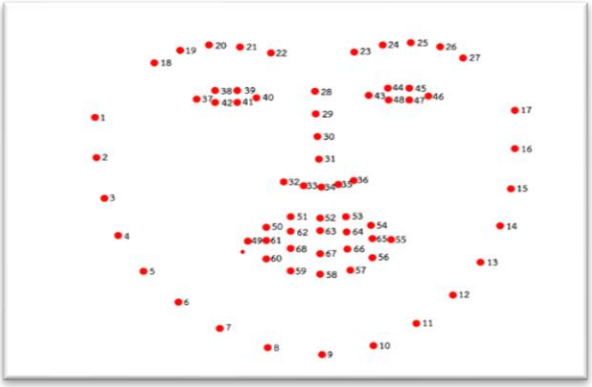

Figure 1.2 Facial landmarks

    2. Face Detection with Haar Cascade algorithm:
        - ➢ Haar Cascade Detection is one of the powerful face detection algorithms invented.

- It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

3. Landmarks drawing with mediapipe:
    - MediaPipe Face Mesh is a solution that estimates 468 3D face landmarks in real-time. It employs machine learning (ML) to infer the 3D facial surface, requiring only a single camera input without the need for a dedicated depth sensor.
    - Utilizing lightweight model architectures together with GPU acceleration throughout the pipeline, the solution delivers real-time performance critical for live experiences.
    - We used it as a feature extractor to enhance the images before modelling.

Figure 1.3 Face mesh

1.5. Implementation:

- System designing has 3 main parts which are input, process and output. The input part consists of webcam camera for detecting face structure, whereas the process part has Nvidia Jetson Nano as a main equipment to evaluate by using Ubuntu operating system and using Python as program language. Conducting OpenCV library has facial landmarks in process as shown in “Fig. 1.4”.

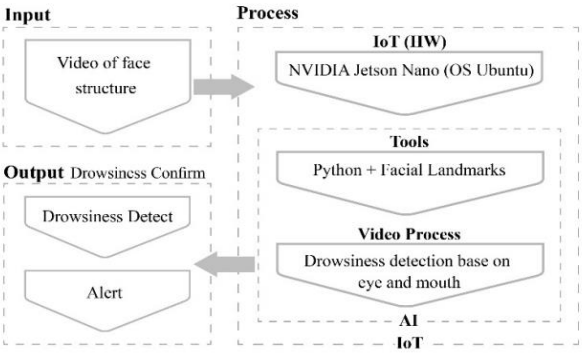

Figure 1.4 Procedure

1.6. System block diagram:

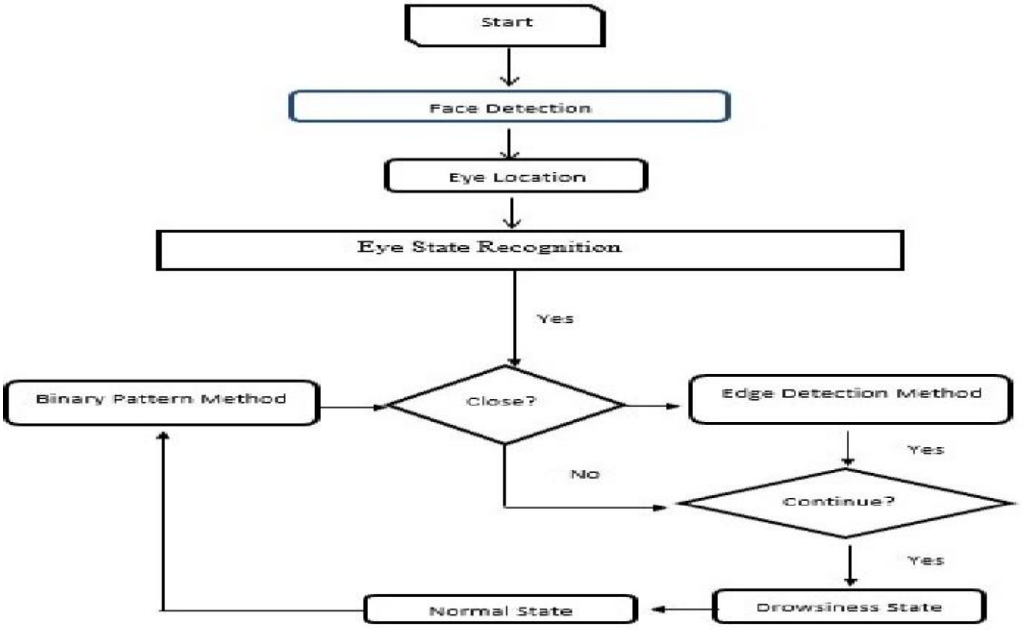

Figure 1.5 Flow chart of the system

1.7. Equipment requirements:

- Hardware:
  1. The Raspberry Pi board or Nvidia Jetson Nano for real-time image analysis.
  2. Webcam camera.
  3. Monitor
  4. Alarm
  5. Mouse
  6. Keyboard

- Software:
    1. OpenCV
    2. Python
    3. Ubuntu operating system
    4. Image processing
    5. Dlib: For tracking 68 facial landmarks.

1.8. Advantages:

- Effective detection of drowsiness in real-time video media.
- Potential to reduce car accidents caused by driver drowsiness.
- Utilizes facial landmarks for accurate analysis of eyes and mouth components.
- Detection accuracy can be achieved even in different scenarios (e.g., with glasses, hat, etc.).

1.9. Disadvantage:

- Dependence on specific camera hardware (infrared camera for low-light scenarios).
- Inability to detect drowsiness when eyes and mouth are completely covered.
- Lack of comparison with other drowsiness detection methods.

1.10. System requirements:

Table 1.1 System requirements

Features/Cases	[1] Seok	[2] M. Navyasree and Gitika Jain.	[3] Haar Cascade algorithm	Our proposal
CO2 sensor	O	X	X	X
Camera	O	X	X	O
Microphone	O	X	X	X
Force feedback	X	X	X	X
Video uploading	X	X	O	X
Monitor	X	O	O	X
Alarm	X	O	O	O
Buzzer	X	O	X	O

Raspberry pi	X	O	X	O
Bounding Box	X	O	X	O

- Powerful Development board or Microcontroller with proper components, with a camera with its libraries.
- SciPy: SciPy is a free and open-source Python library. It is used for scientific computing and technical computing. It includes modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing.
- Facial Recognition: In Machine Learning techniques, Face Recognition using Python is the latest trend. The most popular library for computer vision called the OpenCV provides bindings for Python. It uses machine learning algorithms to fetch faces within a picture or photograph. It is exciting to know the various ways of face detection using Python. There's also the possibility of using TensorFlow then converting it to TensorFlow Lite after the model is ready for testing to implement onto microcontrollers at a presumably higher efficacy using TinyML technology.

1.11. Libraries of Machine Learning:

- When discussing a driver drowsiness detection system, several machine learning libraries can be utilized to develop and implement the necessary algorithms. Here are some popular libraries you might consider:
    1. TensorFlow:
        - TensorFlow is an open-source library developed by Google. It provides a comprehensive ecosystem for machine learning and deep learning tasks. TensorFlow offers tools for creating and training neural networks, which can be used to build models for drowsiness detection.
    2. Keras:

- Keras is a high-level neural networks library that runs on top of TensorFlow. It provides a user-friendly API for building and training deep learning models. Keras is known for its simplicity and ease of use, making it a popular choice for implementing drowsiness detection algorithms.

3. PyTorch:
    - PyTorch is another widely used open-source library for deep learning. It offers dynamic computation graphs, which allow for more flexibility during the model development process. PyTorch provides a rich set of tools and functions that can be leveraged for building driver drowsiness detection models.
4. scikit-learn:
    - scikit-learn is a popular machine learning library in Python. It provides a wide range of algorithms and tools for various tasks, including classification, regression, and clustering. scikit-learn can be utilized to develop models for detecting drowsiness based on features extracted from driver behavior or physiological signals.
5. OpenCV:
    - OpenCV (Open-Source Computer Vision Library) is a widely used computer vision library that offers various image and video processing functions. It provides tools for face detection, eye tracking, and facial expression analysis, which can be useful in a driver drowsiness detection system.
6. Dlib:
    - Dlib is a C++ library that includes machine learning and computer vision algorithms. It offers pre-trained models for facial landmark detection, which can be used to track facial features such as the eyes and mouth. Dlib can be integrated with Python and used in combination with other libraries for drowsiness detection.
7. XGBoost:

- XGBoost is an optimized gradient boosting library that is known for its performance and efficiency. It is commonly used for classification and regression tasks. XGBoost can be employed to build models that classify driver drowsiness based on various input features.

- These are just a few examples of machine learning libraries that can be useful for developing a driver drowsiness detection system. The choice of library depends on factors such as the programming language, specific requirements of the project, and personal preferences of the developers.
- Models' comparison:

Factor	Sequential Model (TensorFlow)	Dlib	Mediapipe
Architecture	Linear stack of layers	C++ library with Python bindings	Cross-platform framework for ML pipelines
Performance	Good accuracy on various tasks	Robust performance in facial analysis tasks	Efficient real-time performance for CV tasks
Ease of Use	Beginner-friendly with high-level APIs	User-friendly Python interface	High-level APIs and pre-trained models
Flexibility	Limited architectural flexibility	Versatile library for various tasks	Support for a wide range of CV tasks
Resource Needs	Moderate computational resources	Moderate resource requirements	Efficient resource utilization

Figure 1.6 Model's comparison

1.12. Teachable machine:

- Teachable Machine is a web-based tool developed by Google's Creative Lab that allows users to create machine learning models without requiring any coding or machine learning expertise. It enables individuals to train models using their own data, such as images, audio, or gestures, and use these trained models for various applications.
- It works by doing the following processes:
    1. Data Collection: Users start by collecting and labeling their own data. For example, if you want to create a model to recognize different hand gestures, you will capture images or perform the gestures while recording video.
    2. Training: Teachable Machine provides a user-friendly interface where you can upload your data and assign labels to each class. The tool uses a technique called transfer learning, which leverages pre-trained models as a starting point. This simplifies the training process and reduces the amount of data required.
    3. Model Creation: After labeling the data, Teachable Machine trains a machine learning model using the provided data. The tool automatically selects an appropriate pre-trained model architecture based on the input data type (images, audio, or pose). It then fine-tunes the model using your labeled data.
    4. Testing and Exporting: Once the model is trained, you can test its performance using the webcam or microphone. Teachable Machine provides real-time feedback on how well the model is recognizing the different classes. If the model performs satisfactorily, you can export it for use in your applications or projects.
- Teachable Machine offers three output options for exporting the trained model:
    1. TensorFlow.js: This allows you to export the model as a JavaScript file that can run in a web browser. You can integrate the model into web applications, interactive websites, or even run it on mobile devices using frameworks like React or Angular.

2. TensorFlow Lite: This option allows you to export the model for deployment on mobile and embedded devices. It is particularly useful if you want to run the model on smartphones, tablets, or Internet of Things (IoT) devices.
3. Python: You can also export the model as a Python code snippet that uses TensorFlow, allowing you to integrate it into your Python-based applications or workflows.

- Teachable Machine is designed to make machine learning more accessible and approachable to a wider audience. It empowers users to create custom models for various tasks without needing to delve into the complexities of training models from scratch or writing code.

CHAPTER 2

Hardware

2.1. Introduction:

- Raspberry Pi, an efficient and cost-effective credit card sized computer comes under light of sun by United Kingdom-Raspberry Pi foundation with the aim to enlighten and empower computer science teaching in schools and other developing countries. Since its inception, various open-source communities have contributed tons towards open-source apps, operating systems, and various other small form factor computers like Raspberry Pi. Till date, researchers, hobbyists, and other embedded systems enthusiasts across the planet are making amazing projects using Pi which looks unbelievable and has out-of-the-box implementation. Raspberry Pi since its launch is regularly under constant development cum improvement both in terms of hardware and software which in-turn making Pi a “Full Fledged Computer” with possibility to

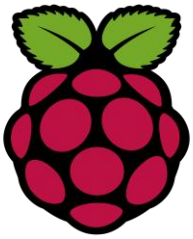

Figure 2.1 Raspberry Pi logo

be considered for almost all computing intensive tasks.

2.2. Hardware features:

- The Raspberry Pi board is equipped with essential components such as program memory (RAM), CPU, GPU, Ethernet port, GPIO pins, Xbee socket, UART, and power source connector, along with various interfaces for external devices. It relies on an SD flash memory card for mass storage, serving as its boot medium like how a PC boots from a hard disk.
- Key hardware requirements for the Raspberry Pi include an SD card loaded with the Linux operating system, a US keyboard, a monitor, a power supply, and a video cable.

Optional hardware additions encompass a USB mouse, a powered USB hub, a case, and an internet connection. For Model A or B, a USB Wi-Fi adaptor suffices for

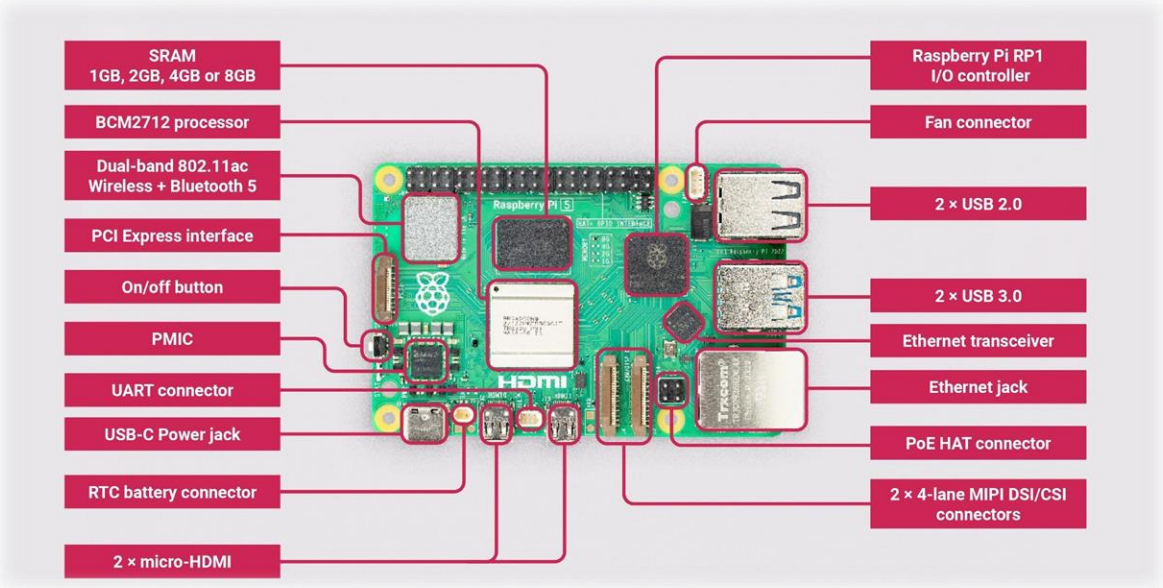

Figure 2.2 Structure of Raspberry Pi board

internet connectivity, while Model B supports LAN cable connection.

2.2.1. Memory:

- The Raspberry Pi 5 Model B (4GB) comes equipped with 4GB of LPDDR4X SDRAM. This increased memory capacity allows for smoother multitasking, running memory-intensive applications, and handling complex projects more efficiently.

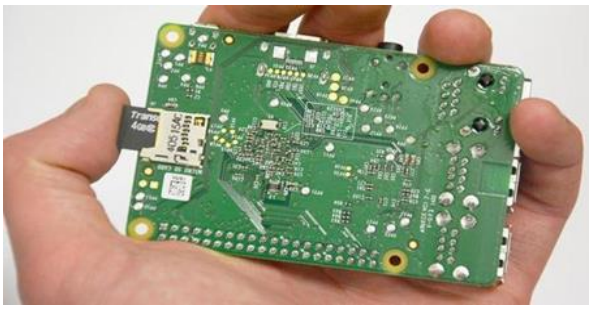

Figure 2.3 Memory card on Raspberry Pi

2.2.2. CPU (Centra Processing Unit):

- The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations. It features a Cortex A76 CPU running at 2.4GHz and a brand-new Video Core VII GPU running at 1GHz. According to the Raspberry Pi Foundation, this will result in significant performance improvement.

Figure 2.4 CPU

2.2.3. GPU (Graphics Processing Unit):

- The GPU is a specialized chip in the raspberry pi board and that is designed to speed up the operation of image calculations. This board designed with a Broadcom video core VII, and it supports OpenGL.

2.2.4. Ethernet Port:

- The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The Raspberry Pi Ethernet port is used to plug in your home router to access the internet.

Figure 2.5 Ethernet port

2.2.5. Ethernet Cable:

- The RPi 3 can be connected to the Internet using a wired.

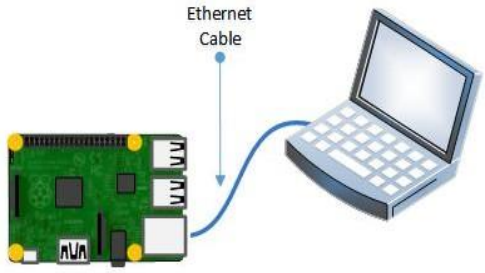

Figure 2.6 Ethernet cable

2.2.6. USB Ports:

- USB 4x2.0 ports are used to plug in a keyboard, mouse, web cam, external hubs.

Figure 2.7 USB ports

2.2.7. Storage (SD card):

- Unlike traditional computers, Raspberry Pi relies on a microSD card for its operating system and data storage. A sufficient size may be chosen depending on the project. the SD Card mounter is located at the back of the board.

Figure 2.8 SD card

2.2.8. GPIO Pins:

- The general-purpose input & output pins are used in the raspberry pi to associate with the other electronic boards. These pins can accept input & output commands based on programming raspberry pi. The raspberry pi affords digital GPIO pins.

These pins are used to connect other electronic components. For example, you can connect it to the temperature sensor to transmit digital data.

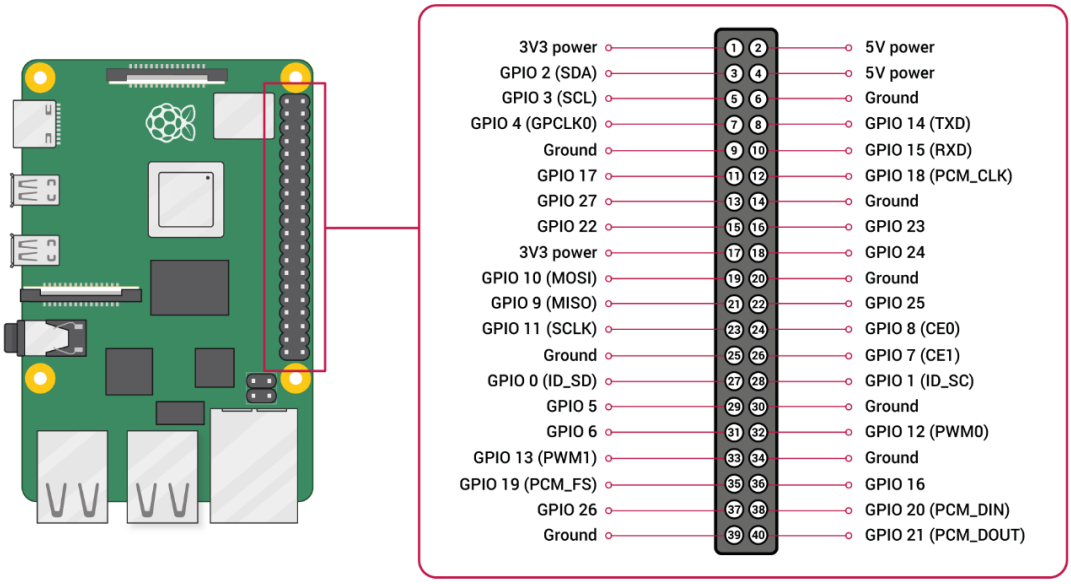

Figure 2.9 GPIO pins

2.2.9. XBee Socket:

- The XBee socket is used in raspberry pi board for wireless communication purposes.

Figure 2.10 XBee socket

2.2.10. Audio and video output:

- Can be used to plug into an external amplifier or an audio docking station.

Figure 2.11 Audio and video output

2.2.11. Camera/Display ports x2:

- The Raspberry Pi 5 introduces a significant change in its I/O configuration compared to previous models. It utilizes two four-lane MIPI CSI/DSI (Mobile Industry Processor Interface - Camera Serial Interface/Display Serial Interface) transceivers. This replaces the dedicated camera and display ports present in earlier models. It allows plugging either the camera or a display providing the new option of adding 2 cameras or 2 displays.

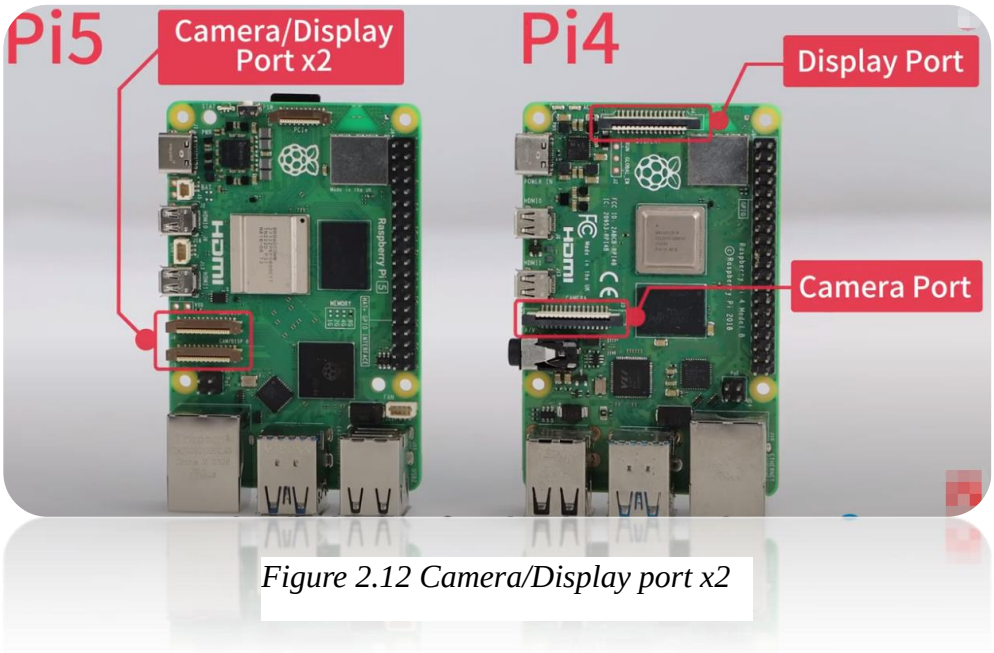

Figure 2.12 Camera/Display port x2

2.2.12. Power Source Connector:

- The power source cable is a small switch, which is placed on the side of the shield. The main purpose of the power source connector is to enable an external power source.

Figure 2.13 Power source connector

2.2.13. UART:

- The Universal Asynchronous Receiver/ Transmitter is a serial input & output port. That can be used to transfer serial data in the form of text, and it is useful for converting the debugging code.

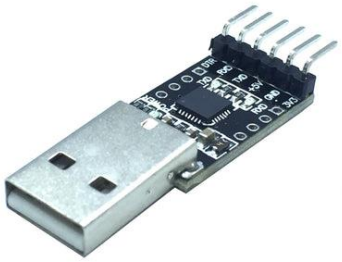

Figure 2.14 UART

2.2.14. HDMI port:

- The HDMI port is the main video (and audio) output of the RPi, allowing you to display videos on the desktop at a resolution of up to 1080p. TVs that support it will also pick up the audio automatically through it.

Figure 2.15 HDMI port

2.2.15. Dual-band Wi-Fi and Bluetooth 5.0:

- These wireless connectivity options enable you to connect to the internet and pair with compatible Bluetooth devices.

2.3. Raspberry Pi models:

- There have been many generations of the Raspberry Pi line.

Table 2.1 Raspberry Pi models

models	*USD*	*CPU*	*RAM*	*Wireless*	*I/O Ports*
Raspberry Pi Zero	$5	1-GHz, 1-core Broadcom BCM2835 (ARM1176JZF-S)	512MB	N/A	1x micro-USB, 1x mini-HDMI
Raspberry Pi Zero WH	$17	1-GHz, 1-core Broadcom BCM2835 (ARM1176JZF-S)	512MB	802.11n / Bluetooth 4.1	1x micro-USB, 1x mini-HDMI
Raspberry Pi Zero W	$10	1-GHz, 1-core Broadcom BCM2835 (ARM1176JZF-S)	512MB	802.11n / Bluetooth 4.1	1x micro-USB, 1x mini-HDMI
Raspberry Pi 3 B+	$35	1.4-GHz, 4-core Broadcom BCM2837B0 (Cortex-A53)	1GB	802.11ac, Bluetooth 4.2, Ethernet	4 x USB 2.0, HDMI, 3.5mm audio
Raspberry Pi 4 B (2GB)	$35	1.5-GHz, 4-core Broadcom BCM2711 (Cortex-A72)	2GB	802.11ac / Bluetooth 5.0	2x USB 3.0, 2x USB 2.0, 1x Gigabit Ethernet, 2x micro-HDMI

Raspberry PI 4 B (4GB)	$55	1.5-GHz, 4-core Broadcom BCM2711 (Cortex-A72)	4GB	802.11ac / Bluetooth 5.0	2x USB 3.0, 2x USB 2.0, 1x Gigabit Ethernet, 2x micro-HDMI
Raspberry PI 5 B (4GB)	$60	2.4-GHz, 4-core Broadcom BCM2712 (Cortex-A76)	4GB	802.11ac / Bluetooth 5.0	2x USB 3.0, 2x USB 2.0, 1x Gigabit Ethernet, 2x micro-HDMI

- There are even more variations with up to 8GBs of RAM known currently. Even though choosing a powerful board is key to running high end AI models, a balance must be made to account for power consumption and external peripherals.

2.4. Raspberry Pi software:

2.4.1. Introduction:

- Your Raspberry Pi needs an operating system to work. Raspberry Pi OS (previously called Raspbian) is our official supported operating system.

2.4.2. Operating System Options:

- Raspberry Pi OS (formerly Raspbian): This free and open-source operating system, based on Debian Linux, is the official choice from the Raspberry Pi Foundation. It comes pre-loaded with essential applications and tools, making it beginner-friendly and ideal for various projects.
- Other Linux Distributions: Several Linux distributions have been optimized for the Raspberry Pi architecture. Popular options include Ubuntu MATE, Fedora, and Arch Linux, each offering a unique user experience and catering to different skill levels.

2.4.3. Installation considerations:

- Installing an operating system on Raspberry Pi differs from traditional computers. The OS resides on a microSD card, which acts as the Pi's primary

storage. The Raspberry Pi Foundation provides detailed instructions and downloadable OS images for easy installation on the official website https://www.raspberrypi.com/software/.

2.4.4. Programming Languages:

- The Raspberry Pi excels as a learning platform for programming. Here are some popular programming languages used with Raspberry Pi 5:
    - Python: A versatile and beginner-friendly language widely used for various purposes, from web development to scientific computing. Its clear syntax and vast libraries make it ideal for Raspberry Pi projects.
    - Scratch: A visual programming language designed for beginners, especially children. It allows creating programs by dragging and dropping blocks, eliminating the need to write complex code.
    - C/C++: These powerful languages offer greater control over hardware and are used for performance-critical applications or interfacing with electronic components.
    - Java: A widely used language for enterprise applications. While not the most resource-efficient choice for Raspberry Pi, some projects can benefit from its vast ecosystem of libraries.

2.5. AI-Thinker A9G:

2.5.1. Introduction:

- The AI-Thinker A9G Development Board stands as a versatile solution for GPRS, GSM, and GPS functionalities. It offers a compact, yet powerful platform tailored for applications demanding robust communication capabilities. Developed by AI-Thinker, this board integrates GPRS (General Packet Radio Service), GSM (Global System for Mobile Communications), and GPS (Global Positioning System) functionalities, catering to a diverse range of IoT (Internet of Things) and tracking applications. With its compact design and rich feature set, the AI-Thinker A9G Development Board emerges

as a pivotal tool for developers seeking seamless integration of wireless communication and location tracking in their projects.

Figure 2.16 AI-Thinker logo

2.5.2. Hardware features:

- A9G is a complete quad-band GSM/GPRS+GPS module based on the RDA8955 chip. The cost reduction of the core chip provides users with a cost-effective IoT solution. Integrating protocol stacks such as GSM/GPRS inside, this module supports the basic phone voice call/SMS, serial to GPRS and GPS data transmission functions, which can be used in a wide range of applications, such as IoT, vehicle-mounted equipment, remote localization, electric power environment monitoring.
- Only one mobile phone card or Internet of Things card is needed to enable the device to have information transmission functions such as GPRS/GPS/ SMS/voice call, A9G could also work with other devices by AT command. Besides, the module comes with 29 GPIOs and integrated SDK that could greatly facilitate private redevelopment.
- **Technical features:**
    - Lithium battery Charge Management: including providing percentage and values.
    - Operating Voltage: 3.3V – 4.2V
    - Quad Band Frequencies: 850/900/1800/1900 MHz
    - Standby Current: < 2mA
    - Data Service: Supports GPRS data service, with a maximum download data rate of 85.6 Kbps and upload data rate of 42.8 Kbps.
    - Working voltage 3.8V-4.2V.

- Low power consumption average current is below 2mA.
- Support GSM/GPRS four frequency bands, including 850, 900, 1800, 1900MHZ.
- GPRS Class 10.
- Support voice calls.
- Support SMS.
- GPIO level is at 2.8V.
- Support GPRS data service, maximum data rate, download 85.6Kbps, upload 42.8Kbps.
- Support standard GSM07.07,07.05 AT commands and Ai-Thinker extended commands.
- A9G supports 3 serial ports, one download serial port and one GPS serial port.
- AT commands support standard AT and TCP/IP command ports.
- Support digital audio and analog audio, support HR, FR, EFR, AMR voice coding.
- Support GPS+AGPS.

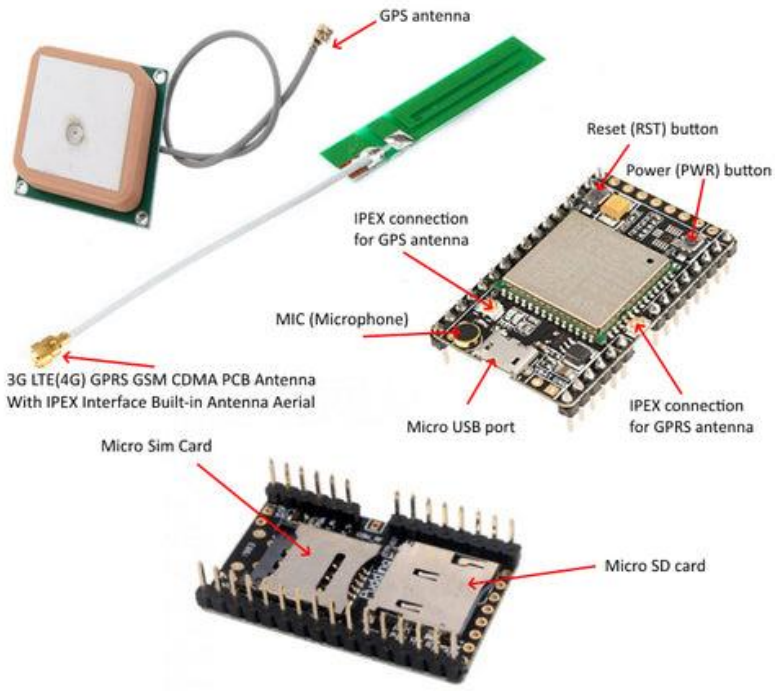

Figure 2.17 Technical features of AI-Thinker A9G

- Pinout overview:

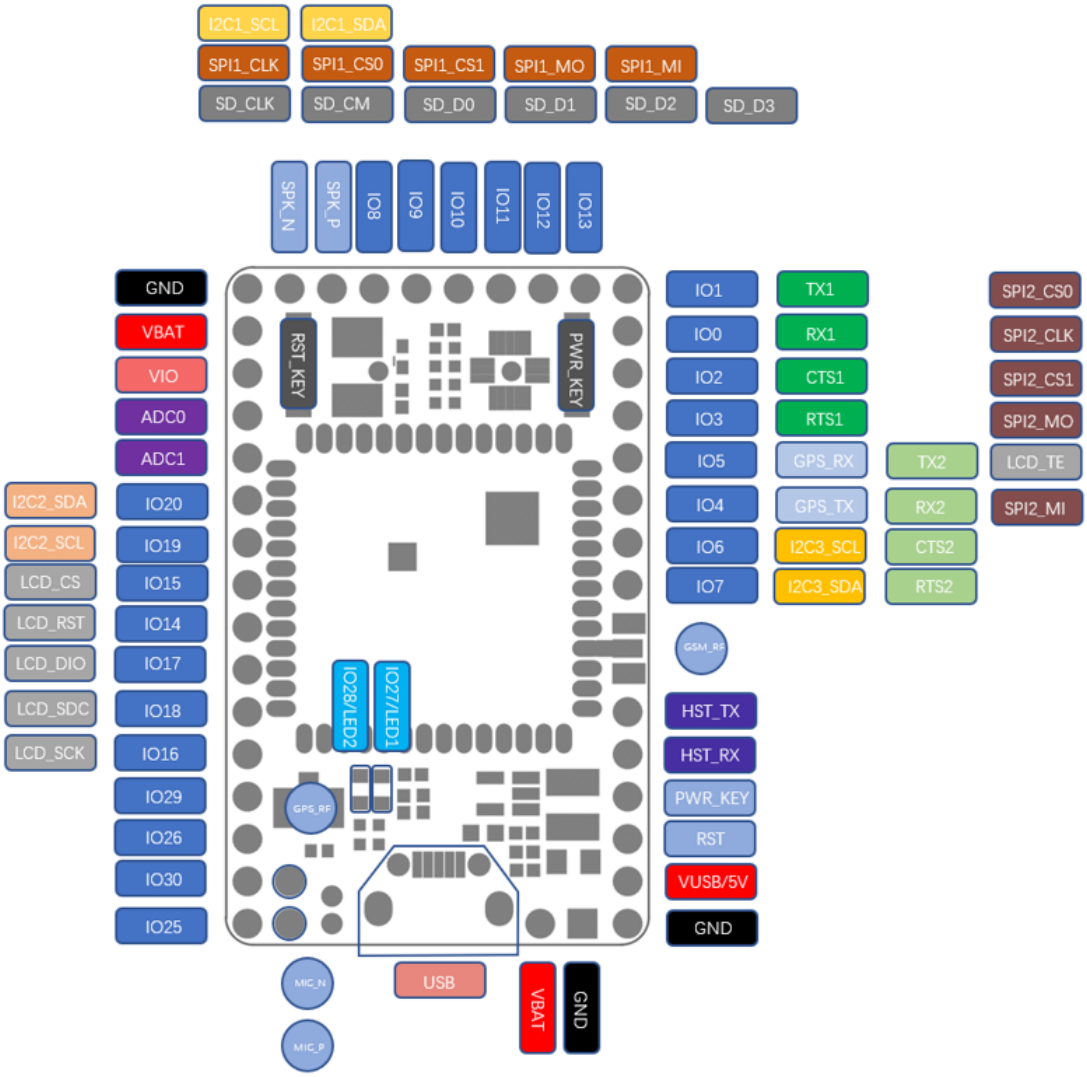

Figure 2.18 Pinout overview of hardware of AI-Thinker A9G

- Equipped with external power capability and multiple serial ports, this board serves as an ideal IoT slave companion for the master Raspberry Pi 5, unlocking the combined power of GPS and GSM functionalities.

2.5.3. A9G Software:

- This development Board doesn't have a storage interface so like any MCU, Firmware needs to be burned first onto it before it can be used in any

application and for that there are 3 options each with their own merits and programming languages.

- **AT commands debugging:**
  - Description: AT commands debugging involves interacting with the A9G module through AT commands, which are simple textual commands used to control the module and retrieve information from it. This method is often used for quick testing and debugging purposes.
  - Merits: AT commands provide a simple interface for controlling the module without needing to write any code. It's suitable for users who prefer a command-line interface or need to quickly test functionality without diving into programming.
  - Programming Language: No specific programming language is required. AT commands are typically sent over a serial interface using a terminal program or a microcontroller.
- **C SDK Development:**
  - Description: C SDK development involves writing firmware for the A9G module using the C programming language. This option provides the most flexibility and control over the module's functionality, allowing developers to create custom applications and interact with hardware peripherals directly.
  - Merits: C SDK development offers high performance and low-level access to the hardware, making it suitable for resource-constrained environments and applications requiring real-time responsiveness. It allows developers to optimize code for efficiency and customize functionality as needed.
  - Programming Language: C programming language is used for developing firmware using the A9G module's Software Development Kit (SDK).
- **Micropython Development:**
  - Description: Micropython development involves writing firmware for the A9G module using the Micropython programming language, which

is a subset of Python optimized for microcontrollers. This option provides a higher-level programming interface compared to C SDK development, making it easier for beginners and rapid prototyping.

- Merits: Micropython development offers a more intuitive and beginner-friendly programming experience, allowing developers to leverage Python's simplicity and readability. It simplifies tasks such as string manipulation, data parsing, and networking, making it suitable for IoT applications and quick development cycles.
- Programming Language: Micropython, a subset of Python, is used for developing firmware for the A9G module. Micropython code can be written and uploaded to the module using a compatible development environment or interactive prompt.

- In summary, while AT commands are quick for testing and debugging and Micropython offers simplicity and ample resources, C SDK development provides the best performance and control, albeit with a steeper learning curve.

2.6. Hardware peripherals:

2.6.1. PIR Sensor:

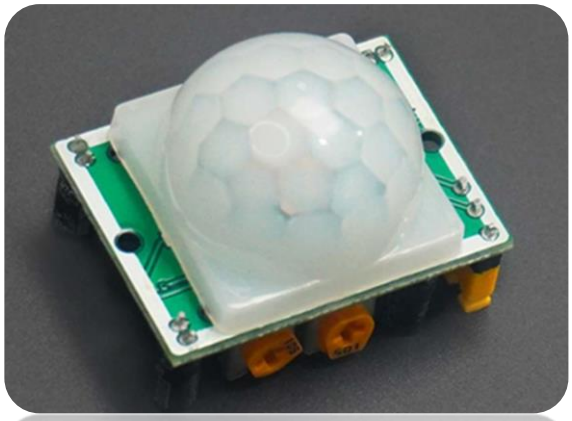

Figure 2.19 RIP Sensor

- **Description:**
  - The PIR (Passive Infrared) sensor detects changes in infrared radiation within its field of view, commonly used for motion detection.

- **Features:**
  - Voltage Range: Operates within a voltage range of 4.8 V to 20 V.
  - Low Power Consumption: Consumes less than 50 µA in idle mode, conserving energy in battery-powered applications.
  - Logic Output: Provides a 3.3 V signal when motion is detected and 0 V otherwise.
  - Delay Time: Adjustable delay time from 0.5 seconds to 200 seconds, with custom settings possible up to 10 minutes.
  - Lock Time: Default lock time of 2.5 seconds after detecting motion.
  - Trigger Options: Trigger can be configured to repeat or disable.
  - Sensing Range: Detects motion within a wide range of up to 120°, within up to 7 meters.
  - Temperature Range: Operates within a temperature range of -15°C to +70°C.
  - Dimension: Compact size of 32 x 24 mm with a lens diameter of 23 mm.
  - Mounting: Screw-screw 28 mm, M2 compatible for easy installation.
- **Connection:**
  - Wiring: Typically connected to a microcontroller or development board via GPIO pins.
  - Interface: Often provides an analog output signal proportional to detected motion intensity but can also offer digital output for motion detection.
  - Power: Requires power supply within the specified voltage range.
  - Signal Output: Can provide either analog voltage or digital signal depending on the sensor model and configuration.
  - Software Integration: Analog output may require analog-to-digital conversion for processing, while digital output can be directly interfaced with microcontroller GPIO pins.

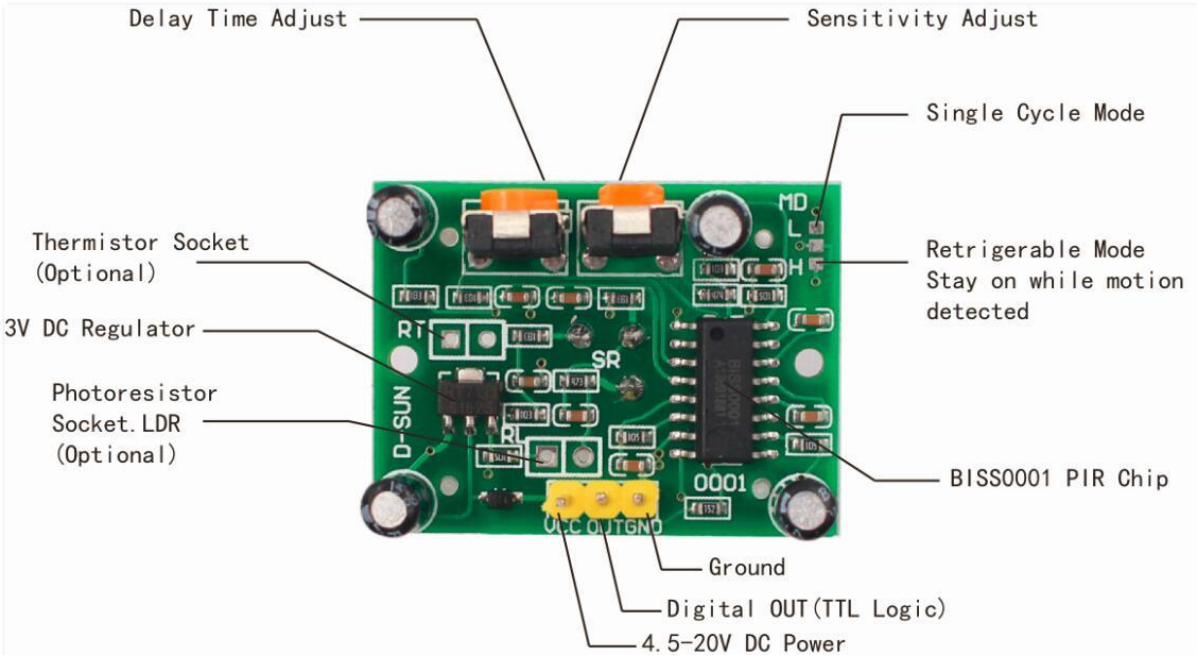

Figure 2.20 Connections of PIR sensor

2.6.2. PI Camera:

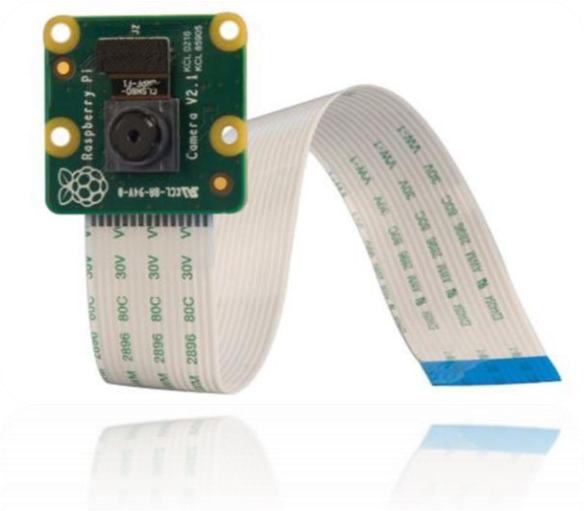

Figure 2.21 PI Camera

- **Description:**
  - The camera is a high-resolution imaging device capable of capturing still images and video footage, connected to the Raspberry Pi.
- **Features:**
  - Resolution: Available in various resolutions, including 5 MP, 8 MP, and higher, depending on the camera model.
  - Frame Rate: Capable of capturing video at up to 1080p resolution with adjustable frame rates.
  - Lens Options: Supports interchangeable lenses for different focal lengths and viewing angles.
  - Interface: Connects to the Raspberry Pi via a ribbon cable, utilizing the Camera Serial Interface (CSI) port.
  - Image Sensor: Utilizes a high-quality image sensor for accurate color reproduction and low-light performance.
  - Software Support: Supported by Raspberry Pi OS with dedicated camera libraries and APIs for image capture and processing.
  - Mounting: Compatible with various mounting options, including official Raspberry Pi camera cases and accessories.
  - Operating Temperature: Designed to operate within a wide temperature range suitable for diverse environments.
- **Connection:**
  - Physical Connection: Attached to the Raspberry Pi via the dedicated Camera Serial Interface (CSI) port.
  - Interface: Utilizes the CSI interface for high-speed data transfer.
  - Software Support: Supported by Raspberry Pi OS with dedicated camera libraries and APIs for image capture and processing.
  - Power: Draws power from the Raspberry Pi through the ribbon cable.
  - Software Integration: Utilizes Raspberry Pi OS camera libraries and APIs for capturing and processing images or video.
  - Ports: Connects specifically to the CSI port on the Raspberry Pi board.

2.7. Appendices (datasheet of hardware):

2.7.1. Specification:

- **Processor:**
  - Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with Cryptographic Extension, 512KB per-core L2 caches, and a 2MB shared L3 cache.
- **Features:**
  - Videocore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2
  - Dual 4Kp60 HDMI® display output with HDR support
  - 4Kp60 HEVC decoder
  - LPDDR4X-4267 SDRAM (4GB and 8GB SKUs available at launch)
  - Dual-band 802.11ac Wi-Fi'
  - Bluetooth 5.0/ Bluetooth Low Energy (BLE)
  - microSD card slot, with support for high-speed SDRI 04 mode
  - 2 x USB 3.0 ports, supporting simultaneous 5Gbps operation
  - 2 x USB 2.0 ports
  - Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT)
  - 2 x 4-lane MIPI camera/display transceivers
  - PCIe 2.0 xl interface for fast peripherals (requires separate M.2 HAT or another adapter)
  - 5V/5A DC power via USB-C, with Power Delivery support
  - Raspberry Pi standard 40-pin header
  - Real-time clock (RTC), powered from external battery
  - Power button
- **Production lifetime:**
  - Raspberry Pi 5 will remain in production until at least January 2036.
- **Compliance:**
  - For a full list of local and regional product approvals, please visit pip.raspberrypi.com.

2.7.2. Physical specification:

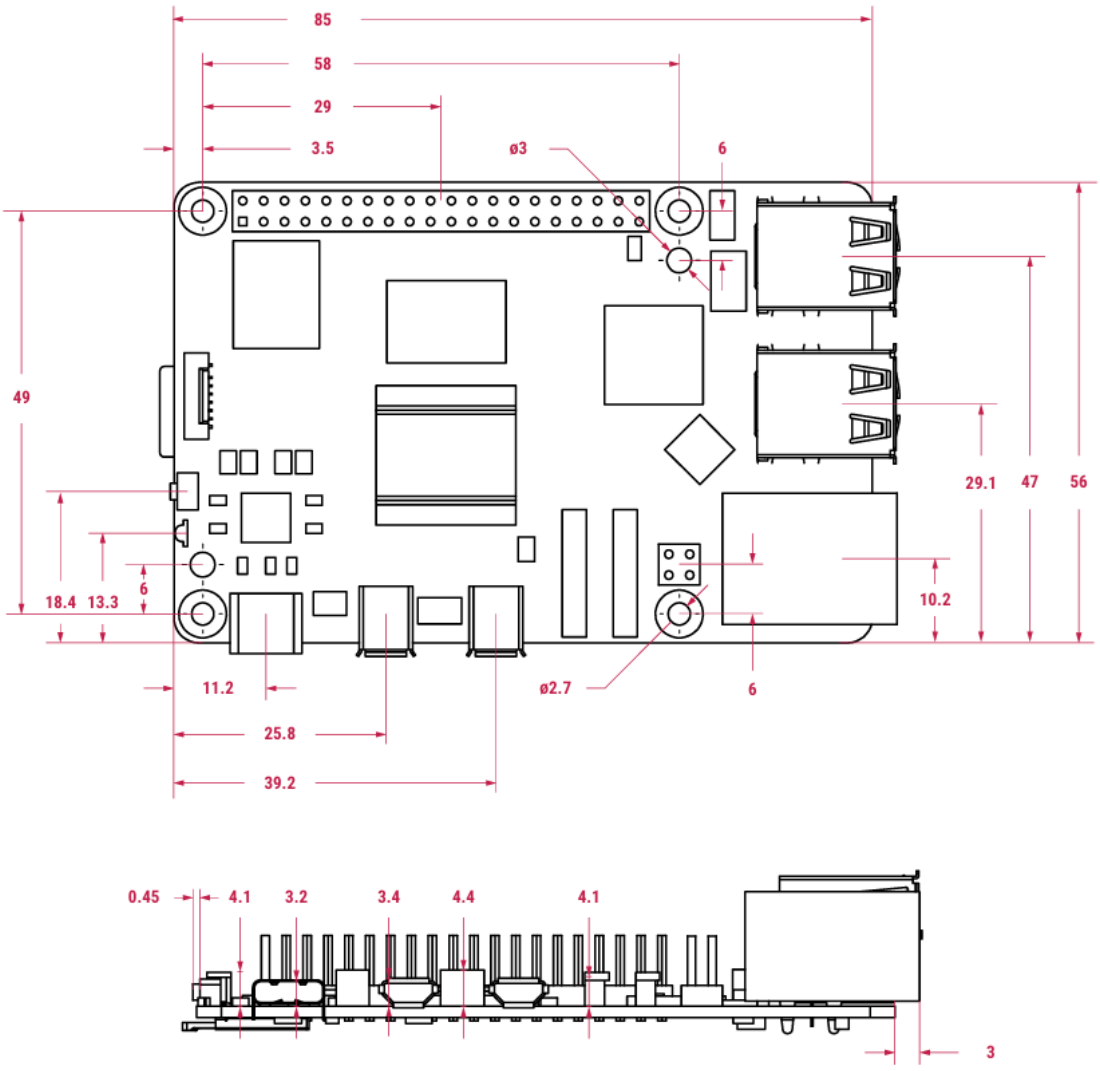

Figure 2.22 Physical specification

- **Notes:**
  - All dimensions in mm.
  - All dimensions are approximate and for reference purposes only. The dimensions shown should not be used for producing production data.
  - The dimensions are subject to part and manufacturing tolerances.
  - Not all the board components are shown. Please reference a physical board for representation of componentry.
  - Dimensions may be subject to change.

- **Warnings:**
  - This product should be operated in a well-ventilated environment, and if used inside a case, the case should not be covered-
  - While in use. This product should be firmly secured or should be placed on a stable. flat. non-conductive surface and should not be contacted by conductive items.
  - The connection of incompatible devices to Raspberry Pi 5 may affect compliance. result in damage to the unit and invalidate the warranty.
  - All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.
- **Safety instructions:**
  - Do not expose yourself to water or moisture or place it on a conductive surface while in operation.
  - Do not be exposed to heat from any source; Raspberry Pi 5 is designed for reliable operation at normal ambient temperatures.
  - Store in a cool, dry location.
  - Take care while handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
  - While it is powered, avoid handling the printed circuit board, or handle it only by the edges, to minimize the risk of electrostatic discharge damage.

2.7.3. A9G GPRS/GSM+GPS/BDS Module:

- **Features:**
  - Complete quad-band GSM / GPRS module, 800/900 / 1800 / 1900MHz
  - SMD package for easy MP & testing
  - Low power mode, average current 2mA or less
  - Support GPS, BD.
  - Supports digital audio and analog audio, supports HR, FR, EFR, and AMR voice coding.

- Support voice calls and SMS messages.
- Embedded network service protocol stack.
- Support standard GSM07.07,07.05AT command and Anxin expandable command set.
- Support PBCCH.
- Supports firmware upgrade via serial port

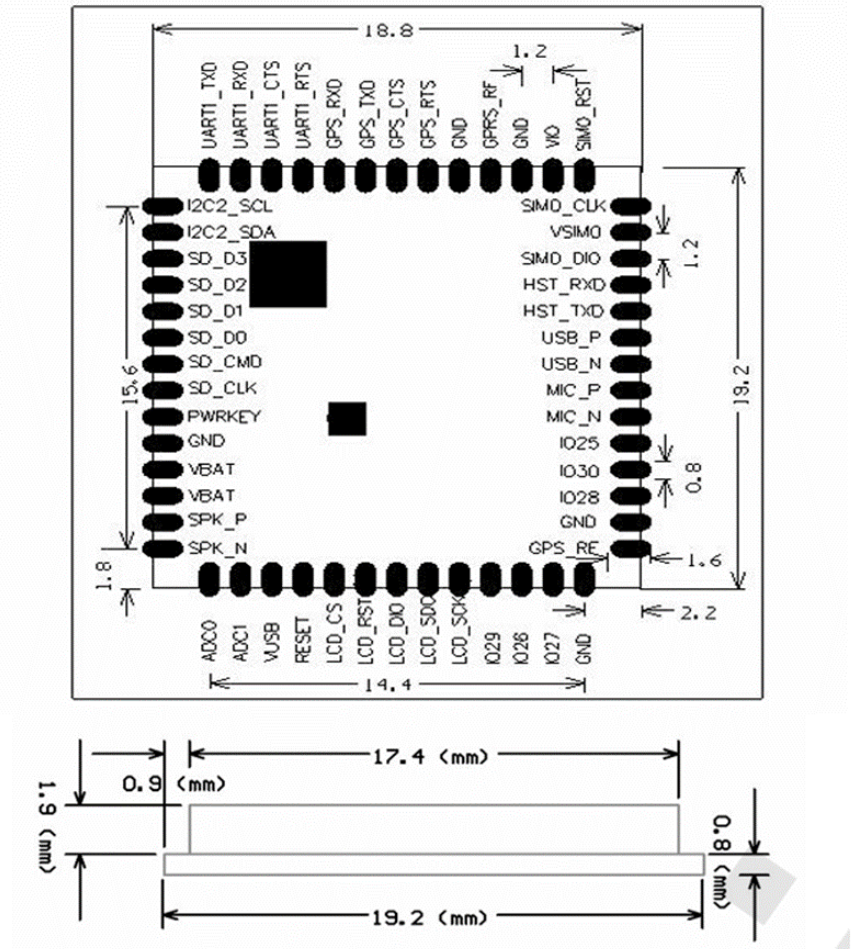

Figure 2.23 A9G GPRS/GSM+GPS/BDS Module

- **<u>Product specifications:</u>**

Table 2.2 Product specifications

Model Name	A9G
Package	SMD54
Size	19.2*18.8*2.7 (±0.2) mm
Frequency	850/900/1800/1900MHz
GPRS Mufti-slot	Class 12
GPRS Mobile Station	Class B
Compatible with GSM Phase 2/2+	Class 4 (2W @ 850/900MHz) Class 1 (1W @ 1800/1900MHz)
Power supply	3.5 ~ 4.2V typical value 4.0V
Current	1.14mA @ DRX=5 1.03mA @ DRX=9
AT command	3GPP TS 27.007, 27.005
GPRS Class 12	Max 85.6kbps (up & down)
Coding scheme	CS 1,2,3,4
PBCCH	Support
Text	Point to point SMS send and receive Cellular broadcast message, Text/PDU mode
Voice coding mode	Half rate (HR), Full rate (FR), Enhanced Full rate (EFR), Adaptive Multi-Rate (AMR)
Audio processing mechanism	Echo cancellation, Echo suppression, Noise suppression
SIM Card	1.8V/3V
UART	3 pcs (including firmware upgrade serial port), baud rate support, 2400~183200bps, default 115200bps
Antenna	Pad (include GSM, GPS)
Communication Interface GPS Sensitivity	I2C, USB, UART, SDMMC, GPIO, ADC
GPS boot time	Cold start < 27.5s H start < 1s Recapture < 1s

GPS accuracy	Horizontal positioning accuracy: 2.5m High positioning accuracy: 3.5m
Working temperature	-20℃ ~ +75℃
Weight	3.0g

CHAPTER 3

System Analysis

3.1. Requirements Gathering:

- **The primary goal:**
  - To develop a system capable of detecting signs of drowsiness in real-time.
- **Environmental context:**
  - The system will be deployed primarily in vehicles (e.g., cars, trucks) and possibly in other contexts such as monitoring operators in control rooms or heavy machinery operators.
- **Sensors and inputs:**
  - Utilize video feed from cameras mounted in the vehicle, as well as physiological signals such as heart rate or eye movements, which may be captured through specialized sensors.
- **Accuracy and reliability:**
  - Aim for high accuracy in detecting drowsiness to prevent accidents, with a low false positive rate to avoid unnecessary alerts.

3.2. Stakeholder Analysis:

- **End-users:**
  - Drivers, operators, or workers who will benefit from the drowsiness detection system by being alerted when they are becoming drowsy.
- **System administrators: (Project Team)**
  - Responsible for managing and maintaining the system, including updating software and ensuring proper functioning.
- **Regulatory bodies:**
  - Compliance with safety regulations and standards related to drowsiness detection in vehicles or workplaces.

3.3. Functionalities:

- **Real-time monitoring:**
  - Continuously analyze data from sensors to detect signs of drowsiness as soon as they occur.
- **Alert generation:**
  - Trigger alerts (e.g., auditory) when drowsiness is detected to prompt the driver or operator to take corrective action.
- **Data logging: (Black-box)**
  - Record relevant data (e.g., timestamps, sensor readings) for post-analysis and evaluation.

3.4. Architecture Design:

- **Hardware components:**
  - Raspberry Pi 5 Model B (4GB) including:
    - Memory
    - CPU (Central Processing Unit)
    - GPU (Graphics Processing Unit)
    - Ethernet port
    - Ethernet cable
    - USB port
    - Storage (SD card)
    - GPIO (General Purpose Input & Output)
    - XBee socket
    - Audio and Video output
    - Camera/Display port x2
    - Power source connector
    - UART
    - HDMI port
    - Dual-band Wi-Fi and Bluetooth 5.0
- **Software modules:**
  - Data preprocessing, drowsiness detection algorithms, alert generation, and user interface.

- **Communication protocols:**
  - Ensure seamless communication between hardware components and software modules.
- **Mobile Application:**
  - It will be a remote interface for safety and security systems.
  - It allows users to monitor the condition of the vehicle and receive alerts and notifications.

3.5. Data Management

- **Data sources:**
  - Video streams from cameras, physiological signals from sensors.
- **Data preprocessing:**
  - Remove noise, extract relevant features, and normalize data for analysis.
- **Data storage:**
  - Store historical data for analysis and model training in a secure and accessible manner.

3.6. User Interface Design:

- **Intuitive interfaces (easy to use):**
  - Design user-friendly interfaces for both drivers/operators and system administrators.
- **Real-time alerts:**
  - Provide clear and actionable alerts to prompt immediate response.
- **Customizable settings:**
  - Allow users to adjust alert thresholds and preferences based on their individual needs.

3.7. Testing and Evaluation:

- **Test cases:**

- Develop scenarios to test the system's performance under various conditions (e.g., different levels of drowsiness, lighting conditions).

- **Usability testing:**
  - Gather feedback from users to assess the system's ease of use and identify areas for improvement.
- **Performance evaluation:**
  - Measure the accuracy and reliability of the drowsiness detection algorithms using metrics such as sensitivity and specificity.

3.8. Deployment and Maintenance:

- **Deployment strategy:**
  - Plan the installation and integration of the system into vehicles or workplaces, ensuring minimal disruption to operations.
- **Training and documentation:**
  - Provide comprehensive training materials and documentation for end-users and system administrators.
- **Continued maintenance:**
  - Establish procedures for regular software updates, hardware maintenance, and performance monitoring to ensure the system remains effective over time.

3.9. Constrains:

- **Technical constraints:**
  - Ensure the system can operate effectively within the computational and memory constraints of the target hardware platform (e.g., onboard vehicle computer).
  - Unavailability of a reliable Raspberry Pi 4 for a long time
  - Unavailability of Sime7600-H 4G hat
  - Unavailability of Cord display adapters for Raspberry Pi 5

- Absence in adapter cord causing incompatibility with infrared night vision cameras, increasing development cost, drops in overall project performance, and lack of necessary features
- Unavailability of a reliable outlet power supply for the project
- GPRS board having inconsistent performance caused by cellular services in the country
- Trained CV models being very resource intensive
- Mediapipe library having inaccurate tracking capabilities
- Currently owned personal devices being too weak and slow, limiting development efforts
- Faulty products and delayed failing of various hardware beyond the return viable date

- **Regulatory constraints:**
  - Comply with privacy regulations regarding the collection and storage of personal data, especially in the case of physiological signals.
  - The recent change for online services and APIs like Google Maps, Firebase, Flutterflow, Mapbox, etc., requiring a non-prepaid foreign currency credit card which severely limited development and wasted time
  - Reductions in available services provided by platforms like Google Cloud to backend services like Cloud Functions with predatory marketing
- **Environmental constraints:**
  - Account for varying lighting conditions and noise levels that may affect the performance of sensors.
  - Relentless power and internet outages throughout the year.
- **Financial/Times constraints:**
  - Due to increased prices and unavailability of the primary equipment within Egypt and inexperience of dealers of such equipment, we have

had difficulty finding the necessary equipment for the project and caused time constraints.

- Time constraints and lack of sufficient prepared datasets to convert to a real-time statistical model for improved performance and accuracy
- Skyrocketing prices for the second half of 2023 due to limitations in imports from other countries
- Heavily condensed curriculums limiting development time throughout the year

CHAPTER 4

Implementation and Integration

4.2. Implementation of IDS:

4.1.1. Introduction:

- In this chapter, we embark on an in-depth exploration of the development and implementation of a sophisticated Driver Attentiveness Monitoring System. This advanced system is meticulously designed to observe and analyze the driver's facial features, enabling the detection of signs of drowsiness and inattention. Utilizing state-of-the-art real-time video processing and machine learning techniques, the system ensures highly accurate and reliable monitoring.
- The following sections provide a comprehensive and detailed explanation of the system's architecture and functioning. We begin with the initialization phase, where the system's components are set up and configured to begin monitoring. Next, we delve into the real-time processing capabilities, explaining how the system continuously analyzes video input to detect any signs of driver inattentiveness or drowsiness.
- Additionally, we cover the data logging aspect, where the system records relevant data for further analysis and review. This component is crucial for understanding long-term patterns and behaviors. Finally, we examine the analytical components that process the logged data to provide insights and actionable feedback.
- Through this detailed breakdown, readers will gain a thorough understanding of how the Driver Attentiveness Monitoring System operates, from initialization to real-time processing and data analysis, highlighting its significance in enhancing road safety.

4.1.2. System Initialization:

- The first step in our implementation involves importing the necessary libraries and initializing the system's components. We use several Python libraries, such as:
    - OpenCV for image processing.

- NumPy for numerical operations.
- MediaPipe for facial landmark detection.
- Pygame for audio alerts.

```
import cv2 as cv  # OpenCV for image processing
import numpy as np  # NumPy for numerical operations
import mediapipe as mp  # MediaPipe for facial landmark detection
import pygame  # Pygame for playing audio alerts
from pygame import mixer  # Mixer module for audio playback
import datetime  # For timestamping data
import csv  # For logging data into CSV files
import os  # For file operations
```

Figure 4.1 System initialization

- Explanation:
  - OpenCV (cv2): Used for capturing and processing video frames from the webcam.
  - NumPy (np): Facilitates array operations for numerical computations.
  - MediaPipe (mp): Provides a high-fidelity solution for detecting facial landmarks.
  - Pygame (pygame, mixer): Used to play audio alerts based on detected driver states.
  - Datetime (datetime): Helps in timestamping events for logging purposes.
  - CSV (csv): Allows data to be stored in a CSV file format for further analysis.
  - OS (os): Handles file existence checks and other OS-level operations.

4.1.3. CSV file initialization:

- We need a CSV file to log the data collected during the monitoring process. This file stores timestamps, blink counts, yawn counts, drowsiness incidents, instances when the driver is out of frame, and the driver's fatigue status.

```python
csv_filename = 'attentiveness_data.csv'
def initialize_csv_file():
    if not os.path.exists(csv_filename):
        with open(csv_filename, 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(["Time", "Blink Count", "Yawn Count", "Drowsiness Count", "Out of Frame", "Fatigue Status"])

def append_data_to_csv(time, blink_count, yawn_count, drowsiness_count, outof_frame, fatigue_status):
    with open(csv_filename, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([time, blink_count, yawn_count, drowsiness_count, outof_frame, fatigue_status])

initialize_csv_file()
```

Figure 4.2 CSV file initialization

- Explanation:
  - CSV Filename: Specifies the name of the CSV file where data will be logged.
  - initialize_csv_file(): Checks if the CSV file exists. If not, it creates a new CSV file and writes the header row.
  - os.path.exists(): Used to check if the file already exists to prevent overwriting.
  - csv.writer(): Writes data to the CSV file in a structured format.

4.1.4. Audio alerts initialization:

- We use Pygame's mixer module to load and play audio alerts that notify the driver of different states, such as drowsiness, blinks, and yawns.

```python
mixer.init()
watch_out = mixer.Sound(r"C:\Users\LOVE\Desktop\Grad project\media pipe\watch_out.wav")
eyes_blink = mixer.Sound(r"C:\Users\LOVE\Desktop\Grad project\media pipe\wake_up_alarm.mp3")
yawn = mixer.Sound(r"C:\Users\LOVE\Desktop\Grad project\media pipe\coffe.wav")
welcome_sound = mixer.Sound(r"C:\Users\LOVE\Desktop\Grad project\media pipe\welcome.wav")
```

Figure 4.3 Audio alerts initialization

- Explanation:
  - mixer.init(): Initializes the mixer module to handle audio playback.
  - mixer.Sound(): Loads audio files for different alert types.
  - watch_out: Alert sound for when the driver looks away from the road.
  - eyes_blink: Alert sound for detecting prolonged eye closure indicating drowsiness.
  - yawn: Alert sound for detecting yawning, a sign of fatigue.
  - welcome_sound: Plays a welcome sound when the system starts.

4.1.5. Utility functions:

- We define several utility functions to handle specific tasks, such as calculating the openness of the eyes and mouth.

```
def open_len(arr):
    y_arr = [y for _, y in arr]
    min_y = min(y_arr)
    max_y = max(y_arr)
    return max_y - min_y
```

Figure 4.4 Utility functions

- Explanation:
  - open_len(): Calculates the vertical distance between the highest and lowest points of a set of facial landmarks (e.g., eyes or mouth), which helps in determining whether they are open or closed.

4.1.6. Facial landmark detection initialization:

- We initialize MediaPipe's FaceMesh solution for detecting facial landmarks.

```
mp_face_mesh = mp.solutions.face_mesh  # Initialize FaceMesh module
face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence=0.5, min_tracking_confidence=0.5)  # Configure FaceMesh
mp_drawing = mp.solutions.drawing_utils  # Utility for drawing facial landmarks
drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)  # Drawing specifications
```

Figure 4.5 Facial landmark detection initialization

- Explanation:
  - mp_face_mesh: Reference to the FaceMesh module in MediaPipe.
  - face_mesh: Configures the FaceMesh with minimum detection and tracking confidence thresholds.
  - mp_drawing: Provides utilities for drawing facial landmarks on the frame.
  - drawing_spec: Specifies the appearance (thickness, circle radius) of the drawn landmarks.

4.1.7. Defining facial landmarks:

- We define arrays of indices representing different facial features, such as the right and left eyes, and the upper and lower lips.

```
RIGHT_EYE = [362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385, 384, 398]
LEFT_EYE = [33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161, 246]
UPPER_LIP = [61, 185, 40, 39, 37, 0, 267, 269, 270, 409, 291]
LOWER_LIP = [146, 91, 181, 84, 17, 314, 405, 321, 375, 291, 308, 324, 318, 402, 317, 14]
```

Figure 4.6 Defining facial landmarks

- Explanation:

- RIGHT_EYE, LEFT_EYE: Lists of indices corresponding to the landmarks that outline the right and left eyes.
- UPPER_LIP, LOWER_LIP: Lists of indices corresponding to the landmarks that outline the upper and lower lips.

4.1.8. Capturing video from the webcam:

- We capture video input from the webcam using OpenCV.

```
cap = cv.VideoCapture(0)  # Capture video from the default webcam
welcome_sound.play()  # Play welcome sound
```

Figure 4.7 Capturing video from the webcam

- Explanation:
  - cv.VideoCapture(0): Initializes video capture from the default webcam (device 0).
  - welcome_sound.play(): Plays a welcome sound to indicate that the system has started.

4.1.9. Main processing loop:

- The main loop captures each video frame, processes it, and displays the results.

```
start_time = datetime.datetime.now()  # Start time for logging
# Initialize counters and states

with mp_face_mesh.FaceMesh(
        max_num_faces=1,
        refine_landmarks=True,
        min_detection_confidence=0.5,
        min_tracking_confidence=0.5
) as face_mesh:
    drowsy_frames = 0
    max_left = 0
    max_right = 0
    blink_count = 0
    eye_closed = False
    eye_open_frames = []
    yawn_frames = 0
    yawn_count = 0
    drowsiness_count = 0
    fatigue_status = "Normal"
    left_frames = 0
    right_frames = 0
    down_frames = 0
    threshold_frames = 60
    outof_frame=0
    screenshot_counter = 0
```

Figure 4.8 Main processing loop

- Explanation:
  - start_time: Records the start time for logging purposes.
  - drowsy_frames: Counter for frames where the eyes appear closed.
  - max_left, max_right: Track the maximum openness of the left and right eyes.
  - blink_count: Counts the number of blinks detected.
  - eye_closed: Boolean flag indicating if the eyes are currently closed.
  - eye_open_frames: List to store the average openness of the eyes over recent frames.
  - yawn_frames: Counter for frames where the mouth appears open.
  - yawn_count: Counts the number of yawns detected.
  - drowsiness_count: Counts incidents of detected drowsiness.
  - fatigue_status: String indicating the current fatigue status of the driver.
  - left_frames, right_frames, down_frames: Counters for frames where the head is oriented left, right, or down.
  - threshold_frames: Number of frames to trigger an alert if the head orientation is off.
  - outof_frame: Counter for instances when the driver is out of the frame.
  - screenshot_counter: Counter for naming screenshot files.

4.1.10. Real-Time video processing:

- We process each frame captured from the webcam to detect facial landmarks and calculate eye and mouth openness.

```
while cap.isOpened():
    ret, frame = cap.read()  # Read a frame from the webcam
    if not ret:
        break   # Exit the loop if no frame is captured

    frame = cv.flip(frame, 1)
    rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB) # Convert frame to RGB color space
    img_h, img_w = frame.shape[:2]
    cv.putText(frame, "Hi I'm HIMOQ! your AI Attentive assistant.", (50, img_h - 50), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
    results = face_mesh.process(rgb_frame)
```

Figure 4.9 Real-Time video processing

- Explanation:
  - while cap.isOpened(): Checks if the video capture is open.
  - ret, frame = cap.read():Captures a frame from the webcam.
  - frame = cv.flip(frame, 1): Flips the frame horizontally, creating a mirror effect.
  - cv.COLOR_BGR2RGB): Converts the frame from BGR to RGB color space
  - img_h, img_w = frame.shape[:2]: Gets the height and width of the frame.
  - cv.putText(frame, "Hi I'm HIMOQ! your AI Attentive assistant.", (50, img_h - 50), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2): Adds the text "Hi I'm HIMOQ! your AI Attentive assistant." to the frame near the bottom.
  - results = face_mesh.process(rgb_frame): Processes the RGB frame to detect facial landmarks using a face mesh model.

4.1.11. Detecting and drawing facial landmarks and calculating openness:

- If facial landmarks are detected, they are drawn on the frame.

```
if results.multi_face_landmarks:
    for face_landmarks in results.multi_face_landmarks:
        all_landmarks = np.array(
            [np.multiply([p.x, p.y], [img_w, img_h]).astype(int) for p in face_landmarks.landmark])

        right_eye = all_landmarks[RIGHT_EYE]
        left_eye = all_landmarks[LEFT_EYE]
        upper_lip = all_landmarks[UPPER_LIP]
        lower_lip = all_landmarks[LOWER_LIP]
```

Figure 4.10 Detecting and drawing facial landmarks and calculating openness (1)

- Explanation:
  - Detection Check (if results.multi_face_landmarks:): Checks if multiple face landmarks (multi_face_landmarks) are detected in the frame (results).
  - Coordinate Conversion: Converts each normalized landmark (p.x, p.y) to pixel coordinates scaled to the frame's dimensions (img_w, img_h).
  - Feature Extraction: Collects pixel coordinates for specific facial features like right_eye, left_eye, upper_lip, and lower_lip.

```
cv.polylines(frame, [left_eye], True, (0, 255, 0), 1, cv.LINE_AA)
cv.polylines(frame, [right_eye], True, (0, 255, 0), 1, cv.LINE_AA)
cv.polylines(frame, [upper_lip], True, (255, 0, 0), 1, cv.LINE_AA)
cv.polylines(frame, [lower_lip], True, (255, 0, 0), 1, cv.LINE_AA)

len_left = open_len(right_eye)
len_right = open_len(left_eye)
mouth_openness = open_len(upper_lip) + open_len(lower_lip)


current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
cv.putText(frame, current_time, (400, 30), cv.FONT_HERSHEY_SIMPLEX, .5, (255, 255, 255), 2)
```

Figure 4.11 Detecting and drawing facial landmarks and calculating openness (2)

- Explanation:
  - Visualizing Facial Landmarks: Draws polylines on frame to visualize specific facial landmarks like left_eye, right_eye, upper_lip, and lower_lip using cv.polylines.
  - Calculating Eye and Mouth Openness: len_left and len_right likely represent the openness of the eyes, calculated using a function open_len applied to right_eye and left_eye. mouth_openness calculates the combined openness of the upper and lower lips, also presumably using open_len.
  - Displaying Current Time: Retrieves the current date and time (current_time) formatted as %Y-%m-%d %H:%M:%S (e.g., "2024-06-28 15:30:00").

```
eye_open_frames.append((len_left + len_right) / 2)
if len(eye_open_frames) > 5:
    eye_open_frames.pop(0)
avg_eye_openness = sum(eye_open_frames) / len(eye_open_frames)
if avg_eye_openness <= (max_left + max_right) / 8 + 1:
    if not eye_closed:
        blink_count += 1
        eye_closed = True
else:
    eye_closed = False
```

Figure 4.12 Detecting and drawing facial landmarks and calculating openness (3)

- Explanation:
  - eye_open_frames.append((len_left + len_right) / 2): Calculates and adds the average openness of both eyes to a list (eye_open_frames).
  - if len(eye_open_frames) > 5:: Maintains a sliding window of the last 5 frames by removing the oldest entry when the list exceeds 5 items (eye_open_frames.pop(0)).
  - avg_eye_openness = sum(eye_open_frames) / len(eye_open_frames): Computes the average eye openness from the values in eye_open_frames.
  - The code checks if the average eye openness is below a calculated threshold ((max_left + max_right) / 8 + 1). If so, it increments blink_count if eyes were previously open (if not eye_closed). It sets eye_closed to True to track closed eyes; otherwise, it sets eye_closed to False when eyes are open.

4.1.12. Drowsiness and yawn detection:

```
if len_left > max_left:
    max_left = len_left
if len_right > max_right:
    max_right = len_right

if (len_left <= int(max_left / 2) + 1 and len_right <= int(max_right / 2) + 1):
    drowsy_frames += 1
else:
    drowsy_frames = 0

if (drowsy_frames > 50):
    drowsiness_count += 1
    drowsy_frames = 0
    eyes_blink.play()
    screenshot_filename = f"drowsiness_screenshot_{datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}_{screenshot_counter}.jpg"
    cv.imwrite(screenshot_filename, frame)
    screenshot_counter += 1

if mouth_openness > 45:
    yawn_frames += 1
    if yawn_frames > 40:
        yawn_count += 1
        yawn_frames = 0
        yawn.play()
else:
    yawn_frames = 0
```

Figure 4.13 Drowsiness and yawn detection

- Explanation:
  - Updating Maximum Eye Openness: max_left and max_right are updated to store the maximum observed values of len_left and len_right, respectively.
  - Tracking Drowsiness:
    1. drowsy_frames increments if both eyes are mostly closed (len_left <= int(max_left / 2) + 1 and len_right <= int(max_right / 2) + 1).
    2. drowsiness_count increases if drowsy_frames surpasses 50, indicating prolonged eye closure suggestive of drowsiness.
    3. Upon detection, it plays an alert sound (eyes_blink.play()), captures a screenshot (cv.imwrite(screenshot_filename, frame)), and increments screenshot_counter.
  - Tracking Yawning:

1. yawn_frames counts frames where mouth_openness exceeds 45 (indicative of a yawn).
2. yawn_count increments if yawn_frames exceeds 40, triggering a yawn alert sound (yawn.play()).

4.1.13. Fatigue assessment:

```
if drowsiness_count < 5 and yawn_count < 5:
    fatigue_status = "Normal"
elif 5 <= drowsiness_count <= 10 or 5 <= yawn_count <= 10:
    fatigue_status = "Tired"
else:
    fatigue_status = "Very Tired"
```

Figure 4.14 Fatigue assessment

- Explanation:
  - fatigue_status is determined based on the counts of drowsiness_count and yawn_count.
  - If both counts are less than 5, fatigue_status is set to "Normal".
  - If either count is between 5 and 10 (inclusive), fatigue_status is set to "Tired".
  - If either count exceeds 10, or both counts exceed 10, fatigue_status is set to "Very Tired".

4.1.14. Head pose detection:

```
face_2d = []
face_3d = []
for idx, lm in enumerate(face_landmarks.landmark):
    if idx in [33, 263, 1, 61, 291, 199]:
        x, y = int(lm.x * img_w), int(lm.y * img_h)
        face_2d.append([x, y])
        if idx == 1:
            nose_2d = (x, y)
            nose_3d = (x, y, lm.z * 3000)
        face_3d.append([x, y, lm.z])

face_2d = np.array(face_2d, dtype=np.float64)
face_3d = np.array(face_3d, dtype=np.float64)

# Camera matrix
focal_length = 1 * img_w
cam_matrix = np.array([[focal_length, 0, img_h / 2],
                       [0, focal_length, img_w / 2],
                       [0, 0, 1]])
distortion_matrix = np.zeros((4, 1), dtype=np.float64)

# Solve PnP
success, rotation_vec, translation_vec = cv.solvePnP(face_3d, face_2d, cam_matrix, distortion_matrix)
rmat, jac = cv.Rodrigues(rotation_vec)
angles, mtxR, mtxQ, Qx, Qy, Qz = cv.RQDecomp3x3(rmat)

# Convert to degrees
x = angles[0] * 360
y = angles[1] * 360
z = angles[2] * 360
```

Figure 4.15 Head pose detection (1)

- Explanation:
  - Extracting Facial Landmarks:
    1. It iterates through specific facial landmarks (face_landmarks.landmark) identified by their indices.
    2. For each landmark, it computes and stores its 2D coordinates (face_2d) scaled to the image dimensions (img_w, img_h).
    3. For the nose tip (index 1), it calculates both its 2D (nose_2d) and 3D (nose_3d) coordinates, adjusting depth for the 3D position.
  - Camera Matrix Setup: Defines a camera matrix (cam_matrix) using the image dimensions (img_w, img_h) and a focal length.
  - PnP (Perspective-n-Point) Solver for Head Pose Estimation: Utilizes cv.solvePnP to estimate the rotation and translation vectors (rotation_vec, translation_vec) of the face in 3D space (face_3d) based

on its 3D and 2D coordinates (face_2d), using the camera matrix and distortion coefficients (distortion_matrix).

- Rotation Decomposition and Conversion to Degrees:
    1. Applies cv.Rodrigues to convert rotation_vec to a rotation matrix (rmat).
    2. Uses cv.RQDecomp3x3 to decompose rmat into Euler angles (angles), representing the rotation around the X, Y, and Z axes.
    3. Converts these angles from radians to degrees (x, y, z) for easier interpretation.

```
if y >= -10:
    left_frames = 0
if y <= 10:
    right_frames = 0
if x >= -10:
    down_frames = 0
if y < -10:
    left_frames += 1
    text = "Looking Left!!!"
    if left_frames > threshold_frames:
        watch_out.play()
        left_frames = 0
        outof_frame +=1
elif y > 10:
    right_frames += 1
    text = "Looking Right!!!"
    if right_frames > threshold_frames:
        watch_out.play()
        right_frames = 0
        outof_frame +=1
elif x < -10:
    down_frames += 1
    text = "Looking Down!!!"
    if down_frames > threshold_frames:
        watch_out.play()
        down_frames = 0
        outof_frame +=1
else:
    text = "Forward!!!"

cv.putText(frame, text, (40,300), cv.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 2)
```

Figure 4.16 Head pose detection (2)

- Explanation:
  - Monitoring Gaze Direction: Resets left_frames, right_frames, and down_frames to zero based on head pose angles (x and y).
  - Alert for Prolonged Gaze Deviation:
    1. If the head pose indicates looking left (y < -10), increments left_frames and triggers an alert (watch_out.play()) if left_frames exceeds threshold_frames.
    2. Similarly, for looking right (y > 10) and looking down (x < -10), increments right_frames and down_frames, respectively, triggering alerts as conditions are met.
  - Displaying Gaze Direction Alert: Displays the detected gaze direction (text) on the video frame (frame) at coordinates (40, 300) using OpenCV's cv.putText.

4.1.15. Displaying metrics and alerts:

- Metrics Display: Shows various metrics such as blink count, yawn count, drowsiness count, fatigue status, and out-of-frame count on the frame.

```
cv.putText(frame, f'Blinks: {blink_count}', (50, 50), cv.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
cv.putText(frame, f'Yawns: {yawn_count}', (50, 100), cv.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
cv.putText(frame, f'Drowsiness: {drowsiness_count}', (50, 150), cv.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
cv.putText(frame, f'Fatigue : {fatigue_status}', (50, 200), cv.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
cv.putText(frame, f'out of frame : {outof_frame}', (50, 250), cv.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
```

Figure 4.17 Metrics Display

- Timing Logic**:** Records metrics to a CSV file (append_data_to_csv) periodically based on a time threshold (start_time).

```
current_time = datetime.datetime.now()
if (current_time - start_time).total_seconds() >= 20:
    append_data_to_csv(current_time.strftime("%Y-%m-%d %H:%M:%S"), blink_count, yawn_count, drowsiness_count, outof_frame, fatigue_status)
    start_time = current_time  # Reset start time
```

Figure 4.18 Timing Logic

4.1.16. Data conversion, analysis, and visualization:

```python
def read_and_aggregate_data(csv_filename):
    aggregated_data = defaultdict(lambda: {'blink_count': 0, 'yawn_count': 0, 'drowsiness_count': 0
                                           , 'out_of_frame': 0, 'fatigue_status_counts': defaultdict(int)})

    with open(csv_filename, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            date = datetime.strptime(row['Time'].split(' ')[0], '%Y-%m-%d').date()
            aggregated_data[date]['blink_count'] += int(row['Blink Count'])
            aggregated_data[date]['yawn_count'] += int(row['Yawn Count'])
            aggregated_data[date]['drowsiness_count'] += int(row['Drowsiness Count'])
            aggregated_data[date]['out_of_frame'] += int(row['Out of Frame'])
            aggregated_data[date]['fatigue_status_counts'][row['Fatigue Status']] += 1

    for date, data in aggregated_data.items():
        data['fatigue_status'] = max(data['fatigue_status_counts'], key=data['fatigue_status_counts'].get)
        del data['fatigue_status_counts']

    return aggregated_data
```

Figure 4.19 Data conversion, analysis, and visualization (1)

- Explanation:
  - Purpose: Reads driver attentiveness metrics from a CSV file and aggregates them by date.
  - Tools Used: csv.DictReader for reading CSV, defaultdict for structured data storage, datetime.strptime for date handling.
  - Process: Iterates through CSV rows, aggregates metrics (blink_count, yawn_count, etc.), and determines the most frequent fatigue_status for each date.
  - Return Value: Returns a dictionary (aggregated_data) where each date maps to aggregated metrics and dominant fatigue_status.

4.1.17. Generating insights from aggregated data:

```python
def generate_attentiveness_insights(date, data):
    print(f"Generating insights for {date}:")
    print(f"daily_summary: On {date}, you blinked {data['blink_count']} times, yawned {data['yawn_count']} times,
          had {data['drowsiness_count']} drowsiness incidents, and were out of frame {data['out_of_frame']} times.")
    if data['yawn_count'] > data['blink_count']:
        print("trend_insight: You tend to yawn more frequently than blink, which might indicate tiredness.")
    else:
        print("trend_insight: Your blinking rate is higher than yawn count, indicating general alertness.")
    if data['drowsiness_count'] > 5:
        print("recommendation: Consider taking more frequent breaks or adjusting your driving schedule to avoid drowsiness.")
    else:
        print("recommendation: Keep up the good work! Your attentiveness levels are commendable.")
    if data['fatigue_status'] == "Very Tired":
        print("fatigue_status_overview: You were very tired on this day. Ensure you get some rest before your next drive.")
    elif data['fatigue_status'] == "Tired":
        print("fatigue_status_overview: You showed signs of tiredness. Stay hydrated and take short breaks.")
    else:
        print("fatigue_status_overview: Your fatigue status is normal. Continue driving safely.")
    print("\n")
```

Figure 4.20 Generating insights from aggregated data

- Explanation:
  - Purpose: This function generates insights and recommendations based on aggregated data for a specific date.
  - Parameters:
    1. date: Date for which insights are being generated.
    2. data: Aggregated metrics for that date (blink_count, yawn_count, drowsiness_count, out_of_frame, fatigue_status).
  - Insights Generated:
    1. Daily Summary: Provides a summary of activities based on the aggregated data.
    2. Trend Insight: Compares yawn count to blink count to infer potential tiredness or alertness.
    3. Recommendation: Suggests actions based on drowsiness count (e.g., taking breaks).
    4. Fatigue Status Overview: Provides an overview of the fatigue status recorded for the day.
  - Output: Prints insights and recommendations formatted for easy understanding and action

4.1.18. Data conversion, analysis, and visualization:

```
def convert_to_dataframe(aggregated_data):
    df = pd.DataFrame.from_dict(aggregated_data, orient='index')
    df.index = pd.to_datetime(df.index)
    df['day_of_week'] = df.index.day_name()
    df['month'] = df.index.month
    return df

def perform_correlation_analysis(df):
    correlation_matrix = df[['blink_count', 'yawn_count', 'drowsiness_count']].corr()
    sns.heatmap(correlation_matrix, annot=True)
    plt.show()

def train_predictive_model(df):
    X = df[['blink_count', 'yawn_count']]
    y = df['drowsiness_count']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestRegressor(n_estimators=100)
    model.fit(X_train, y_train)
    return model

def detect_anomalies(df):
    clf = IsolationForest(random_state=42)
    df['anomaly'] = clf.fit_predict(df[['drowsiness_count']])
    anomalies = df[df['anomaly'] == -1]
    return anomalies

def visualize_data(df):
    plt.figure(figsize=(10, 7))
    plt.scatter(df.index, df['blink_count'], color='blue', label='Blink Count')
    plt.scatter(df.index, df['yawn_count'], color='red', label='Yawn Count')
    plt.scatter(df.index, df['drowsiness_count'], color='green', label='Drowsiness Count')
    plt.gca().xaxis.set_major_locator(mdates.DayLocator())
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.legend()
    plt.xlabel('Date')
    plt.ylabel('Count')
    plt.title('Driver Behavior Metrics Over Time')
    plt.gcf().autofmt_xdate()
    plt.tight_layout()
    plt.show()
```

Figure 4.21 Data conversion, analysis, and visualization (2)

- Explanation:
  - Data Conversion (convert_to_dataframe):
    1. Converts the aggregated data (aggregated_data) into a pandas DataFrame (df).
    2. Sets the index of df to datetime format and adds columns for day of the week (day_of_week) and month (month).
  - Correlation Analysis (perform_correlation_analysis):
    1. Computes and visualizes correlation matrix between blink_count, yawn_count, and drowsiness_count.

2. Uses seaborn to generate a heatmap for easy interpretation.

- Predictive Modeling (train_predictive_model):
  1. Trains a Random Forest regressor model (RandomForestRegressor) to predict drowsiness_count based on blink_count and yawn_count.
  2. Splits data into training and testing sets using train_test_split.
- Anomaly Detection (detect_anomalies):
  1. Uses Isolation Forest (IsolationForest) for anomaly detection in drowsiness_count.
  2. Identifies and returns anomalies where drowsiness_count significantly deviates from normal patterns.
- Data Visualization (visualize_data):
  1. Generates a scatter plot to visualize trends in blink_count, yawn_count, and drowsiness_count over time (df.index).
  2. Configures x-axis to display dates (mdates.DayLocator, mdates.DateFormatter) for clarity.

4.1.19. Data loading and visualization setup:

```
import dash
from dash import html, dcc
import plotly.express as px
import pandas as pd

df = pd.read_csv(r"C:\Users\LOVE\Desktop\Grad project\media pipe\.ipynb_checkpoints\attentiveness_data-checkpoint.csv")
df['Time'] = pd.to_datetime(df['Time'])

fig = px.scatter(df, x='Time', y=['Blink Count', 'Yawn Count', 'Drowsiness Count'], title="Attentiveness Metrics Over Time")
```

Figure 4.22 Data loading and visualization setup

- Explanation:
  - Purpose: Loads attentiveness metrics data from a CSV file and visualizes it using Plotly Express in a Dash web application.
  - Tools Used: pd.read_csv to load CSV data into a pandas DataFrame, px.scatter to create a scatter plot.
  - Process: Converts the 'Time' column to datetime format, then plots 'Blink Count', 'Yawn Count', and 'Drowsiness Count' over time (Time).
  - Visualization: Generates a scatter plot (fig) showing how metrics change over time.

4.1.20. Analysis and recommendations:

```
average_blink_count = df['Blink Count'].mean()
blink_recommendation = "Normal" if average_blink_count < 20 else "Consider taking more breaks."

latest_yawn_count = df.iloc[-1]['Yawn Count']
yawn_trend = "Increasing" if df.iloc[-1]['Yawn Count'] > df.iloc[-2]['Yawn Count'] else "Stable/Decreasing"
yawn_recommendation = "Yawning is increasing, ensure you are well-rested." if yawn_trend == "Increasing" else "Yawn count is stable or decreasing."
```

Figure 4.23 Analysis and recommendations

- Explanation:
  - Purpose: Computes average blink count and assesses yawn count trends to provide personalized recommendations.
  - Metrics: Calculates average_blink_count and determines if it's within normal range (< 20) or suggests breaks if higher.
  - Yawn Trend: Analyzes the latest yawn count trend (yawn_trend) compared to the previous data point to recommend rest or stability.
  - Recommendations: Displays recommendations based on computed metrics and trends for better attentiveness.

4.2. Mobile applications:

4.2.1. Integration of Flutter and FlutterFlow for Rapid Application Development:

- Flutter, a powerful UI toolkit from Google, is ideal for cross-platform app development with its single codebase approach, offering native performance and expressive UI components. However, due to economic constraints limiting access to Google services, alternative methods are necessary. Its main purpose for us is to test and simulate mobile functions fast instead of waiting for apk building for every single test on slow devices.

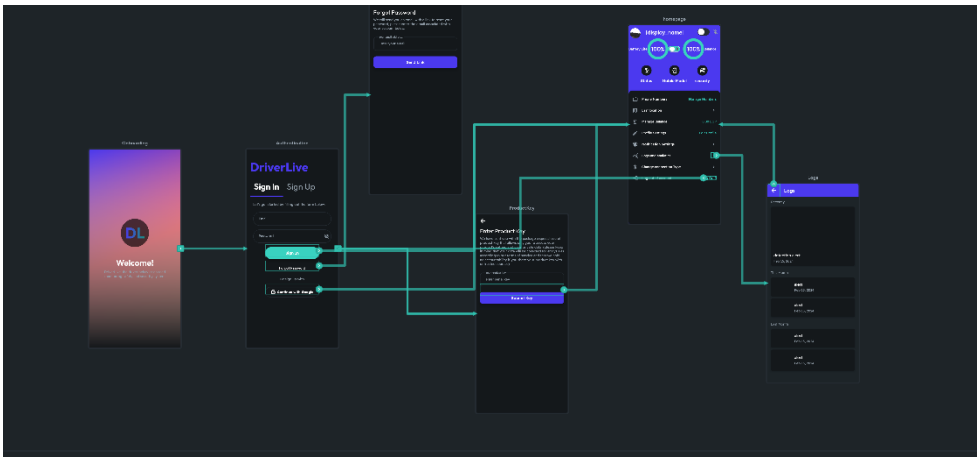

Figure 4.24 Integration of Flutter and FlutterFlow for Rapid Application Development

- Utilizing FlutterFlow:
  - ➢ FlutterFlow enhances development efficiency by combining visual design capabilities with the flexibility of custom code implementation:
    1. Visual Interface Design: Developers can use FlutterFlow's intuitive drag-and-drop interface to design app layouts visually. This includes arranging widgets, defining interactions, and previewing the app's appearance across different devices and screen sizes.
    2. Custom Code Integration: Beyond visual design, FlutterFlow supports the integration of custom code. Developers can write Flutter code directly within the platform, allowing for advanced functionality and tailored business logic implementation.
    3. Source Code Modification: FlutterFlow facilitates direct access to the app's source code. Developers can modify generated

Flutter code or add new functionalities seamlessly, maintaining full control over the app's behavior and appearance.

- Key features and benefits:
  - Visual Simulation and Iteration: The platform provides real-time visual simulation, enabling developers to see how their design translates into the actual app interface. This iterative process speeds up development and ensures design fidelity.
  - Flexibility and Extensibility: By supporting custom code, FlutterFlow accommodates complex app requirements that go beyond standard visual design. This includes integrating APIs, managing state, and implementing platform-specific features.
- Advantages:
  - Enhanced Collaboration: Visual tools in FlutterFlow foster collaboration among team members, including designers and developers, by providing a common platform for design and implementation.
  - Efficient Development Cycles: Rapid prototyping and iterative design capabilities reduce development time, allowing teams to deliver feature-rich applications faster and with fewer errors.
- Disadvantages:
  - Need to export project to implement native functions.
  - Updates and bugs can affect stability.
  - Like Google cloud recently, it started requiring a specific type of credit card for free services.

4.2.2. Utilizing Google Maps URL Modification for Location Services:

- Google Maps provides extensive global coverage and real-time updates, serving over 1 billion users across 200+ countries. However, economic restrictions have barred access to Google Maps API and related services due to bans on prepaid cards and non-foreign currency transactions. To address these limitations, our project employs Google Maps URLs for location-based functionalities.

- Key features and implementation:
  - Universal URL Structure: The URLs follow a uniform structure (https://www.google.com/maps/...) with parameters (api=1, query, origin, destination, etc.) defining specific actions like search queries or route directions.
  - Cross-Platform Compatibility: Regardless of the user's device (Android, iOS, or web), the same URL can launch Google Maps either in the app or browser, ensuring consistent functionality.
  - Technical Considerations: URL parameters must be properly encoded for compatibility across platforms and adhere to a 2,048-character limit per request. This ensures reliability and avoids truncation issues.
  - Here's an example of how using Coordinates in the URL works: https://www.google.com/maps/search/?api=1&query=47.5951518%2C-122.3316393

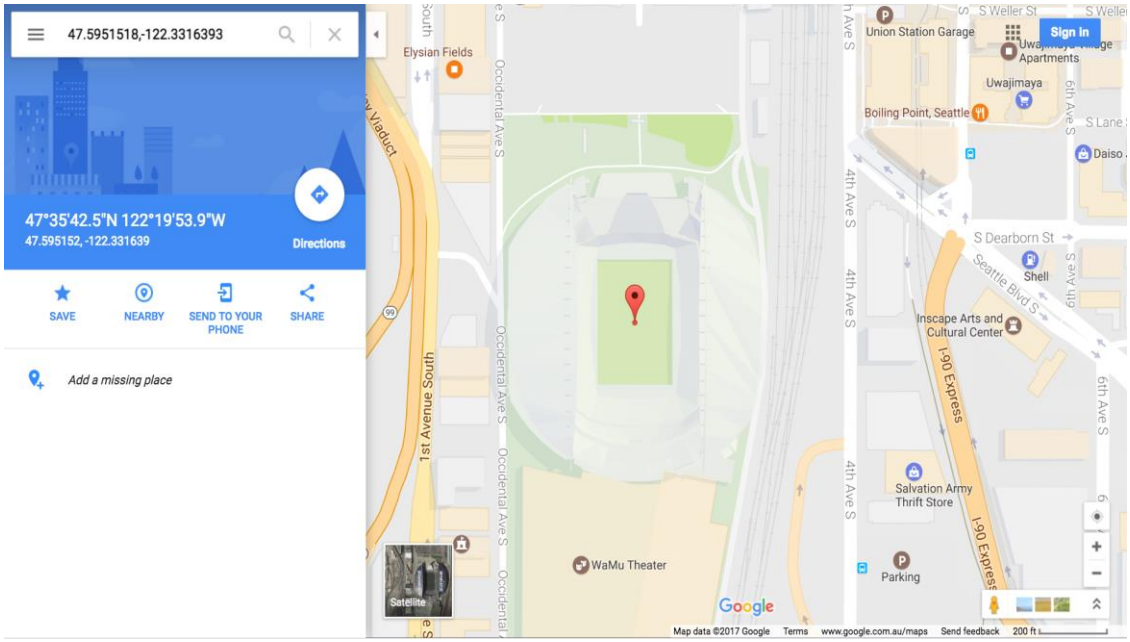

Figure 4.25 Example of how using Coordinates in the URL works

4.2.3. Firestore Database and REST API:

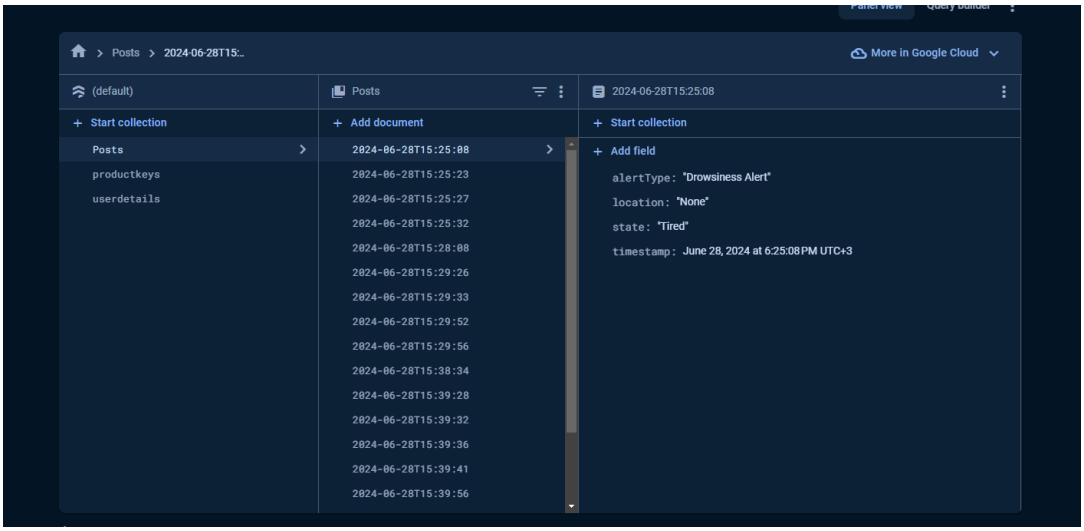

Figure 4.26 Firestore Database and REST API

- Firestore Database overview:
  - Firestore is a flexible, scalable NoSQL cloud database provided by Google, designed for building rich, serverless applications. It allows for real-time synchronization and offers both web and mobile SDKs to simplify development. Key Features:
    1. Real-time Updates: Changes made to the database are instantly propagated to all connected clients, ensuring a seamless user experience with up-to-date information.
    2. Offline Support: Firestore allows devices to store and synchronize data locally even when offline. When connectivity is restored, data is synchronized automatically, maintaining application functionality regardless of network status.
    3. Security Rules: Firestore provides robust security rules that allow developers to define who has access to which parts of the database. This ensures data integrity and protects against unauthorized access.

4. Scalability: The database scales automatically to handle application growth, managing increasing data volumes efficiently without requiring manual intervention.
5. Cost-Effective: Firestore offers a generous free tier, making it accessible for startups and projects with moderate data needs. Pricing scales with usage, ensuring cost-effectiveness as applications grow.

- Firestore REST API overview:
  - Firestore offers a REST API that allows developers to interact with the database over HTTP. Here are the key operations supported by the REST API:
    1. Authentication and Database URL:
       - Before interacting with Firestore via REST API, you'll need:
         - **Firebase Project ID:** This uniquely identifies your Firebase project.
         - **Database URL:** The base URL for your Firestore instance, typically in the format https://firestore.googleapis.com/v1/projects/[PROJECT_ID]/databases/(default)/documents.
         - **Authentication:** Firestore allows authentication through various methods, such as using an OAuth 2.0 token in your requests. This ensures that only authorized users can read from or write to your database, adhering to your security rules.
         1. GET - Reading Data: Retrieve data from a specific document in the database using an HTTP GET request.
         2. PUT - Writing Data: Write or overwrite data at a specific document location using a PUT

3. POST - Creating Documents: Add new data to a collection using a POST request.
4. PATCH - Updating Data: Update specific fields of data at a document location without overwriting other fields using a PATCH request.
5. DELETE - Removing Data: Delete a document at a specific location using a DELETE request.

2. Additional Capabilities:
    - Authentication: Secure REST requests using OAuth 2.0 tokens, ensuring access control as per Firestore Security Rules.
    - Query Parameters: Enhance data retrieval with parameters like mask, orderBy, and others, allowing customization of API responses based on specific needs.
    - Streaming Support: The Firestore REST API supports streaming changes to data, enabling real-time updates in client applications.

CHAPTER 5

Future work and Conclusion

5.1. Our long-term vision:

- Our model is designed for detection of drowsy state of eye and give and alert signal or warning may be in the form of audio or any other means. But the response of the driver after being warned may not be sufficient to stop causing the accident meaning that if the driver is slow in responding towards the warning signal, then accident may occur. Hence to avoid this we can design and fit a motor driven system and synchronize it with the warning signal so that the vehicle will slow down after getting the warning signal automatically. Also, we can avoid the use of Raspberry Pi which is not so fast enough for video processing by choosing our own mobile phone as the hardware. This can be done by developing a proper mobile application which will perform the same work as Raspberry Pi and response will be faster and effective and, we can continue working to develop a fully automatic vehicle parking system using digital image processing techniques. This would take full control of parking the vehicle. At the nearest possible place as soon as driver's drowsiness is detected once forestalling drowsy driver from driving.

5.2. Forced Feedback:

5.2.1. Introduction:

- Nowadays, racing wheels have forced feedback. It allows you to feel vibrations via the steering wheel, which makes it feel like you're driving on the track. The 3 types of force feedback work in different ways and have different intensities.

5.2.2. Special vibration effect:

- When you race in games, you'll find all kinds of obstacles on the track. If you take a sharp turn, you may end up on rough terrain. How cool would it be if you could feel the g-force in a curve or the resistance of the surface? A racing wheel with force feedback technology creates vibrations and mimics gravity. This way, it feels like you're behind the wheel of a racecar. How advanced the vibration effects differ per racing wheel.

5.3. Lane detections:

- Incorporating lane detection into our drowsiness detection system can enhance the overall safety of the vehicle. Lane detection algorithms use computer vision techniques to identify lane markings on the road, providing critical information about the vehicle's position relative to the lane boundaries. Future work could focus on integrating real-time lane detection systems that alert the driver if the vehicle starts to drift out of its lane, a common occurrence when the driver is drowsy. Advanced machine learning algorithms and image processing techniques can be employed to improve the accuracy and reliability of lane detection, even in challenging conditions such as poor lighting or adverse weather.

5.4. Obstacles detection:

- Obstacle detection is another crucial aspect that can be integrated into the drowsiness detection system. This involves using sensors such as LiDAR, radar, and cameras to identify and classify obstacles on the road, including other vehicles, pedestrians, and debris. By combining obstacle detection with drowsiness detection, the system can provide comprehensive safety alerts to the driver. Future work could explore the development of sophisticated algorithms that not only detect obstacles but also predict potential collisions and suggest evasive actions. This integration can significantly reduce the risk of accidents caused by drowsy driving.

5.5. Advanced behavioral metrics:

- To further enhance the drowsiness detection system, incorporating advanced behavioral metrics can provide a more accurate assessment of the driver's state. These metrics could include analyzing the driver's eye movement patterns, head position, and facial expressions using advanced computer vision and machine learning techniques. Additionally, physiological signals such as heart rate variability and electroencephalogram (EEG) readings can be used to monitor the driver's alertness levels. Future research could focus on developing multimodal systems that combine these various metrics to create a robust and reliable drowsiness detection mechanism.

5.6. Driving emergency assistance:

- Incorporating driving emergency assistance into the drowsiness detection system can provide critical support during emergency situations. This feature could include automatic braking, steering assistance, and adaptive cruise control to help the driver maintain control of the vehicle when drowsiness is detected. Future work could explore the integration of these advanced driver-assistance systems (ADAS) with real-time drowsiness detection to provide seamless and immediate responses to potential hazards. Developing communication protocols between the drowsiness detection system and the vehicle's control systems will be essential to ensure timely and effective intervention.

5.7. Conclusion:

- In conclusion, the development of a drowsiness detection system represents a significant step forward in enhancing road safety. Our project has demonstrated the feasibility and effectiveness of using computer vision and machine learning techniques to monitor driver alertness and provide timely warnings. By continuously analyzing the driver's eye movements and other behavioral cues, the system can detect signs of drowsiness and prevent potential accidents.
- Future work should focus on integrating additional safety features such as lane detection, obstacle detection, advanced behavioral metrics, and driving emergency assistance. These enhancements will not only improve the accuracy of drowsiness detection but also provide comprehensive safety solutions to address various driving hazards. By leveraging cutting-edge technologies and interdisciplinary research, we can continue to advance the field of driver safety and reduce the incidence of accidents caused by drowsy driving.
- The successful implementation of this project highlights the importance of innovative approaches to solving real-world problems. As we move forward, continued collaboration between researchers, engineers, and industry professionals will be crucial in developing and deploying advanced safety systems that protect drivers and passengers alike.

References:

[1] Seok-Woo Jang and Byeongtae Ahn, Implementation of Detection System for Drowsy Driving Prevention Using Image Recognition and IoT,Received: 26 February 2020; Accepted: 7 April 2020; Published: 10 April 2020.

[2] M. Navyasree and Gitika Jain. Satyabhama institute of science and technology (deemed to be university) accredited with grade "a" by NAAC JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119. December2020 to April 2021.

[3] https://sist.sathyabama.ac.in/sist_naac/documents/1.3.4/b.e-ece-batchno-128.pdf

[4] Detection of Drowsiness from Facial Images in Real-Time Video Media using Nvidia Jetson Nano | IEEE Conference Publication | IEEE Xplore.

[5] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlett, and J. Movellan, "Drowsy driver detection through facial movement analysis," International Workshop on Human-Computer Interaction, vol. 4796, 2007.

[6] R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, "Driver drowsiness detection model using convolutional neural networks techniques for android application," in 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), IEEE, 2020.

[7] G. Turan and S. Gupta, "Road accidents prevention system using drivers drowsiness detection," International Journal of Advanced Research in Computer Engineering Technology, 2013.

[8] Aly, M. (2008). Real time detection of lane markers in urban streets. IEEE Intelligent Vehicles Symposium. DOI: 10.1109/IVS.2008.4621256

[9] He, Y., & Kise, M. (2012). A lane detection method for lane departure warning system. Journal of Transportation Technologies, 2(4), 210-214. DOI: 10.4236/jtts.2012.24023

[10] Santos, P., Lourenço, A., & Ferreira, J. C. (2016). Real-time obstacle detection system for autonomous vehicles. Sensors, 16(11), 1916. DOI: 10.3390/s16111916

[11] Huang, S., Liu, G., & Ma, L. (2019). Obstacle detection and recognition for intelligent vehicles based on deep learning. IEEE Access, 7, 170732-170749. DOI: 10.1109/ACCESS.2019.2954920.

[12] Kircher, K., Uddman, M., & Sandin, J. (2002). Vehicle control and drowsiness. Technical Report No. 1042A, Swedish National Road and Transport Research Institute (VTI).

[13] Deng, Z., Zhu, C., Cheng, D., Zeng, W., & Zhan, F. (2020). Driver drowsiness detection using multi-channel second order blind identifications. IEEE Access, 8, 50131-50140. DOI: 10.1109/ACCESS.2020.2979453.

[14] Petit, J., & Shladover, S. E. (2014). Potential cyberattacks on automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 16(2), 546-556. DOI: 10.1109/TITS.2014.2342271

[15] Asadi, B., & Vahidi, A. (2011). Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. IEEE Transactions on Control Systems Technology, 19(3), 707-714. DOI: 10.1109/TCST.2010.2047860

[16] Dong, Y., Hu, Z., Uchimura, K., & Murayama, N. (2011). Driver inattention monitoring system for intelligent vehicles: A review. IEEE Transactions on Intelligent Transportation Systems, 12(2), 596-614. DOI: 10.1109/TITS.2010.2092770

[17] Tiwari, A., & Gupta, N. (2019). Driver drowsiness detection system using machine learning. Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC). DOI: 10.1109/ICMLC.2019.8886661.

ملخص:

- تتأثر شبكات النقل والخدمات بشكل كبير بالصحة العقلية والجسدية للسائق وسلامته ، حيث يؤدي انخفاض اليقظة والتركيز أو الانفعال إلى المخاطرة بحياة الركاب والسائقين .بالإضافة إلى ذلك ، تحتاج الطرق السريعة وطرق السفر إلى استجابة سريعة للحالات الطبية الحرجة .وطريقة لمنع عمليات السرقة وحتى تعقبها .لحل هذه المشكلة، نقوم بتطوير نظام مراقبة السائق بناءً على الرؤية الحاسوبية، والذي يستخدم الذكاء الاصطناعي لتحليل ومراقبة سلوكيات السائق المختلفة في الوقت الفعلي باستخدام معالم الوجه والجسم، وينبههم عند النوم أو تشتت انتباههم، كما يتصل بالمساعدة في حالات الطوارئ باستخدام تقنيات الشبكات الخلوية، و تحديد الموقع الجغرافي .يستفيد نظامنا الذي يعمل في الوقت الفعلي من تحليل البيانات والتنبؤ بناءً على البيانات السابقة للوقاية من التنبيهات والمخاطر بشكل استباقي.
- الكشف عن النعاس هو مجال بحثي حيوي يهدف إلى تعزيز السلامة في مختلف المجالات، ولا سيما في مجال النقل. يركز هذا المشروع على تطوير نظام للكشف عن النعاس في الوقت الفعلي لمنع الحوادث الناتجة عن تعب السائقين. باستخدام تقنيات معالجة الصور المتقدمة وتقنيات التعلم الآلي، يقوم نظامنا بمراقبة تعابير وجه السائق وحركات عينه بشكل مستمر لتحديد علامات النعاس.
- يتضمن جوهر النظام كاميرا تلتقط فيديو في الوقت الفعلي لوجه السائق. يتم تحليل الفيديو باستخدام خوارزميات اكتشاف المعالم الوجهية لتتبع معدل رمش العين، ومدة إغلاق العين، وتكرار التثاؤب. تُغذى هذه المعايير في نموذج تعلم آلي مدرب على التعرف على الأنماط التي تدل على النعاس. عند اكتشاف علامات النعاس، يقوم النظام بإطلاق تنبيه لإيقاظ السائق، مما يعزز سلامة الطرق.
- يتضمن بحثنا تصميم وتنفيذ خوارزمية الكشف، بالإضافة إلى إجراء اختبارات شاملة والتحقق من صحتها باستخدام مجموعات بيانات تحتوي على تعابير وجه متنوعة وظروف قيادة مختلفة. تظهر النتائج أن نظامنا يحقق دقة وموثوقية عالية في الكشف عن النعاس، متفوقًا على الحلول الحالية.
- تم تنفيذ هذا المشروع داخل قسم هندسة الاتصالات والحاسبات بأكاديمية الشروق، مع الدعم المالي من الأكاديمية. تسهم نتائج هذا البحث بشكل كبير في تقليل حوادث الطرق وتحسين سلامة النقل.

وزارة التعليم العالي والبحث العلمي
المعهد العالي للهندسة بمدينة الشروق
قسم هندسة الإتصالات والحاسبات

نظام القيادة الذكى (IDS)

مقدمة كجزء من متطلبات الحصول علي درجة البكالوريوس في هندسة الإلكترونيات والإتصالات / الحاسبات والتحكم

فريق المشروع

أحمد القذافي مرسي أحمد
أحمد هاني سعد ياسين
اسلام محمد حسن حسن
اسامة احمد رضا محمد حجازي
حمزة عبد الناصر علي سليمان

المشرفون

د. خالد محمد توفيق المنشاوي

المعهد العالي للحاسبات وتكنولوجيا المعلومات – أكاديمية الشروق

م. آلاء ابوشعيشع عوض

قسم هندسة الإتصالات والحاسبات – المعهد العالي للهندسة – أكاديمية الشروق

القاهرة 2024