PROJECT REPORT ON

Fatigue Prevention System

SUBMITTED BY:

POPURI HEMANTH KUMAR

THOTA KOUSHIK SAI

TELLAPURI VIJAYENDRA SAI

ANCHOORI SAI VARSHITHA

GOPALSETHY SRIVALLI

BANDARI SAHASRA

Mr.KONDRAGUNTA CHAITHANYA
(Project Coordinator)
SENIOR TECHNICAL ARCHITECT
3rd FLIX

Dr. V.KUMARA SWAMY
(Project Guide)
PROFESSOR OF ECE
DEAN ALUMI RELATIONS
SNIST

Certification

This is to certify that the group project titled **”FATIGUE PREVENTION SYSTEM”** has been successfully completed and submitted by the following team members **POPURI HEMANTH KUMAR, THOTA KOUSHIK SAI, TELLAPURI VIJAYENDRA SAI, ANCHOORI SAI VARSHITHA,GOPALSETHY SRIVALLI, BANDARI SAHASRA**. The project was carried out as a part of our engineering, under the supervision of **DR.V.KUMARA SWAMY** and coordination of **KONDRAGUNTA CHAITHANYA.** The team has shown dedication and teamwork in conducting research, analysis, and development of the project.

Mr.KONDRAGUNTA CHAITHANYA
(Project Coordinator)
SENIOR TECHNICAL ARCHITECT
3rd FLIX

Dr. V.KUMARA SWAMY
(Project Guide)
PROFESSOR OF ECE
DEAN ALUMI RELATIONS
SNIST

Acknowledgement

We, the members of the project group, would like to express our profound gratitude and appreciation to all those who have contributed to the successful completion of this project titled **”FATIGUE PREVENTION SYSTEM”.**

First and foremost, we extend our deepest thanks to our project supervisor, **DR.V.KUMARA SWAMY**, for their continuous guidance, support, and constructive feedback throughout the course of this project. Their expertise and insights have been valuable in helping us understand the complexities of the subject and guiding us through various stages of the project.

We would like to acknowledge the contributions of **KONDRAGUNTA CHAITHANYA** who provided critical advice, data, or technical support during the project. We would also like to thank our peers for their constructive criticism and valuable suggestions.

Team Members:

HEMANTH KUMAR

KOUSHIK SAI

SRIVALLI

SAI VARSHITHA

SAHASRA

VIJAYENDRA SAI

Declaration

We, the undersigned, hereby declare that the project report entitled **”FATIGUE PREVENTION SYSTEM”** submitted, is a result of our original work conducted under the supervision of **DR.V.KUMARA SWAMY** and the coordination of **KONDRAGUNTA CHAITHANYA** We further declare that,This report has not been submitted to any other institution or organization for any other degree, diploma or certification. All work presented here is our own, except where specific references have been made to the work of others. We have fully acknowledged all the sources, literature, and data used in the completion of this project. This project was completed in accordance with the ethical and academic standards prescribed by our supervisior and the coordinator. We understand that any violation of this declaration may result in disciplinary action as per the guidelines of the our supervisor and coordinator.

Team Members:

HEMANTH KUMAR

KOUSHIK SAI

SRIVALLI

SAI VARSHITHA

SAHASRA

VIJAYENDRA SAI

Abstract

How do we overcome the situation of fatigue in an environment such as transportation or health, where the well beings translate into safety and performance.Fatigue is a continually increasing issue across various industries and significantly affects decision-making productivity and general health. This project proposes an integrative system for preventing fatigue and identifying tracking and mitigating fatigue-related risks in real time. We are looking to create an innovative solution that banks on technology to increase the level of awareness of fatigue so that the personal as well as automobile industrial outcomes of such are improved.

The proposed solution to address driver fatigue and drowsiness is an advanced fatigue prevention system that offers a proactive, real-time approach to improving road safety.It uses a pulse meter to continuously track the driver's heart rate.Camera monitors the driver's eye movements to detect signs of drowsiness, such as prolonged eye closure, frequent blinking .When drowsiness is detected through a combination of heart rate and eye movement data, the system sends this information to the vehicle's Engine Control Unit (ECU).The ECU then activates a multi-level warning system that alerts the driver via a visual display, sound alarm, and flashing parking lights.If the driver fails to respond, the system automatically slows the car down to a safe speed of 5 km/sec,ensuring the vehicle remains under control without abrupt or dangerous movements.The system is designed to be user-friendly, with optimized human-computer interaction (HCI).It has undergone usability testing to ensure the interface is simple, efficient, and easy to use. The software is intuitive, allowing drivers to understand and respond to alerts quickly.By combining data from both the heart rate monitor and the eye-tracking camera, the system accurately detects drowsiness in real-time and intervenes.This innovative solution effectively reduces the risk of accidents related to driver fatigue.

Contents

Chapter 1

Introduction

The drowsiness might also additionally ultimate for a couple of minutes however it's consequences can be disastrous. The main cause of sleepiness is typically exhaustion, which reduces alertness and attention, although other causes include lack of concentration, medications, sleep issues, drinking alcohol, or shift work. They are unable to predict when sleep may strike. Even though falling asleep while driving is dangerous, being tired makes it difficult to drive safely even when you are awake. One in twenty drivers is said to have fallen asleep behind the wheel. The most at risk for tired driving is truck and bus drivers with commutes of 10 to 12 hours. Driving a long distance while sleep deprived might make you drowsy, as can driving when you need to sleep.

According to National Highway Traffic Safety Administration (NHTSA), the police and hospital reports identified that 100,000 car accidents and over 1,500 deaths were caused due to drowsiness of drivers each year. Drowsy driving is thought to be responsible for approximately 1,550 fatalities, 71,000 injuries, and 12.5 billion in financial losses. A sleepy driver was a factor in 697 fatalities in 2019. NHTSA acknowledges that it is challenging to quantify the precise number of accidents, or fatalities caused by drowsy driving and that the reported figures are underestimates. Fortunately, Recognizing the complexities inherent to fatigue, the development of a comprehensive Fatigue Prevention System(FPS) emerges as a critical strategy to promote employee well-being and operational efficiency. The FPS is envisioned as a multi-faceted framework that encompasses awareness,assessment,intervention,and continuous monitoring of fatigue levels.The FPS will utilize evidence-based assessment tools to measure both subjective and objective indicators of fatigue,enabling organizations to tailor interventions.

1.1 The exsisting solutions and their limitations

1. **Fixed Rest Break Schedules** : Many workplaces implement standardized rest breaks, typically mandated by labor laws, such as scheduled lunch breaks or shorter rest periods during shifts.

Limitations: These fixed schedules do not account for individual fatigue levels or fluctuating workloads. Workers may still be fatigued after breaks or may not need a break when scheduled, leading to inefficiencies.

2. **Self-Reported Fatigue Assessments** : Workers are often asked to self-assess and report their levels of fatigue through questionnaires or verbal communication.

Limitations: This method is subjective and can be unreliable. Workers might underreport fatigue due to workplace culture, job pressure, or fear of being perceived as underperforming. Additionally, workers may not always be aware of their fatigue until it's too late, leading to accidents or errors.

3. **Manual Shift Rotation Policies** : Some industries use rotating shift schedules to ensure workers are not subjected to long, monotonous shifts. This involves rotating workers between day, night, and evening shifts to allow for some rest and recovery.

Limitations: While rotating shifts help prevent constant night work, they do not fully address the effects of circadian rhythm disruption. This can still cause fatigue, particularly for night-shift workers, as their body clocks may never fully adjust to changing schedules.

4. **Training Programs and Fatigue Awareness Campaigns** : Companies conduct training sessions and awareness programs to educate workers about the dangers of fatigue, how to recognize fatigue symptoms, and strategies to improve sleep and manage stress.

Limitations: While educational programs raise awareness, they rely heavily on workers to take personal responsibility. Without active monitoring or management tools, many workers may still suffer from fatigue despite their knowledge of its risks.

5. **Caffeine and Stimulant Use** : Workers often resort to caffeine, energy drinks, or other stimulants to stay alert during long or demanding shifts.

Limitations: While caffeine can provide a temporary boost, it does not address the underlying causes of fatigue. Overreliance on stimulants can lead to dependency, masking fatigue symptoms rather than solving the problem. Additionally, excessive caffeine use can lead to health problems like anxiety.

6. **Environmental Adjustments** : Workplaces may modify environmental conditions like lighting,

temperature, and noise levels to reduce factors that contribute to fatigue. For example, bright lighting may help improve alertness, while maintaining a comfortable temperature can prevent drowsiness.

Limitations: Environmental modifications alone cannot fully address fatigue. These adjustments may provide temporary relief, but they do not account for individual fatigue levels or physiological factors.

7.**Fitness and Wellness Programs** : Many organizations offer wellness initiatives like gym memberships, yoga classes, or health assessments to encourage workers to maintain their physical and mental health, which can help reduce fatigue over time.

Limitations: These programs are often voluntary, and participation rates may be low. While these programs can improve long-term well-being, they are not always effective in addressing acute fatigue in the workplace.

8. **Vehicle-Based Fatigue Detection Systems** : In industries like transportation, there are in-vehicle systems that monitor driver behavior, such as steering inputs, lane drifting, and braking patterns, to detect signs of driver fatigue.

Limitations: These systems are mostly limited to vehicles and cannot be applied to all types of workplaces. They also tend to detect fatigue only after its effects are noticeable, rather than preventing it from developing in the first place.

9. **Occupational Health Surveillance** : Companies may conduct regular health check-ups or employ occupational health specialists to monitor workers' health and well-being, identifying those at risk of fatigue due to underlying health conditions, stress, or sleep disorders.

Limitations: Health surveillance can be infrequent and may not catch acute instances of fatigue, particularly if workers are not undergoing regular or comprehensive assessments.

10. **Ergonomic Workplaces** : Some companies invest in ergonomic equipment like adjustable chairs, standing desks, and fatigue mats, particularly for roles that involve repetitive motions or long periods of standing.

Limitations: While these solutions can reduce physical strain, they do not account for mental fatigue or the cumulative effects of prolonged work hours.

1.2 Key challenges of existing solutions

1. **Lack of Real-Time Monitoring**: Most existing solutions do not provide real-time feedback or alerts, making it difficult to prevent fatigue before it becomes problematic. Many of these systems rely on manual reporting or static schedules.
2. **Generic Approaches**: Most fatigue prevention systems adopt a one-size-fits-all model, failing to account for individual differences in fatigue levels. Fixed break schedules or shift rotations do not consider variations in individual physiology or job demand.
3. **Post-Fatigue Interventions**: Many systems detect fatigue after it has already occurred rather than predicting and preventing it in advance. Solutions like fatigue detection cameras or self-reported assessments only react to fatigue symptoms once they are present.
4. Focus on Physical, Not Mental Fatigue: Many solutions, particularly ergonomic interventions, focus on reducing physical strain but often overlook mental fatigue, which can be just as debilitating and dangerous, especially in tasks requiring concentration and decision-making.
5. Underreporting: Workers may not always report their fatigue due to cultural or professional pressures, making self-assessment tools unreliable in many cases.

1.3 Proposed system

Our Proposed system combines a **”PULSEMETER”** and **”CAMERA”** to detect early signs of drowsiness. The pulsemeter monitors the driver’s pulse rate, while the camera tracks eyemovement.So in general person feels slepeeness then the PULSE decreases in the body.So if we keep the pulsemeter, it detects the pulse and at the same time we are using camera to detect the movement of eyes.If the eyes gets dimmer (I. E if the person gets sleep he obviously closes his eyes very frequently) and we are using Pantilt servo so that if we change the seat position the camera will be changing accordingly.Now if the pulsemeter reading and the camera’s eye movement range matches then it gives the warning and it turn’s on the parking lights after certain time the vehicle gets stopped slowly. The information of pulse Metre reading and camera reading(if both are matched) the information passes into ”ECU” then it gives warning on ”CAR’S DISPLAY ”,if the person doesn’t alert for the warning then the ECU detects information and passes to breaking system. Which controls the engine of the vehicle and stops slowly at the range of 5km/sec. By this system we can reduce the maximum accidents.

1.4 Purpose of the project

The purpose of the project is to mitigate workplace fatigue by implementing advanced technological solutions that monitor, predict, and prevent fatigue among workers. Fatigue, especially in industries with extended shifts, physically demanding tasks, or mentally exhausting work, can lead to accidents, injuries, reduced productivity, and long-term health risks.This project aims to enhance workplace safety by implementing advanced technological solutions that can detect, monitor, and predict fatigue among workers.Ultimately, the goal is to create a comprehensive and adaptable fatigue management system that ensures both the safety and efficiency of workers, reducing absenteeism, improving morale, and enhancing operational outcomes.

Enhance Workplace Safety:Detect and prevent fatigue-related incidents that can lead to accidents, injuries, and errors, especially in high-risk industries such as manufacturing, transportation, and healthcare Introduce real-time fatigue monitoring solutions using wearable sensors and environmental monitoring to provide early warnings of physical and mental fatigue, allowing supervisors and workers to intervene before critical tasks are compromised.

This project aims to leverage cutting-edge technology and data analytics to create a comprehensive, proactive approach to fatigue management that enhances workplace safety, boosts productivity, reduces long-term health risks, and promotes worker well-being. By integrating these solutions across multiple teams and departments, the project seeks to build a resilient and adaptable system that not only addresses the current challenges of fatigue but also creates a healthier, more sustainable working environment for the future.

Chapter 2

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

2.1 Product perspective

This is the latest fatigue prevention system that is invented to enhance safety in automobiles and robotics by monitoring the states of physiological conditions and the degree of alertness of people operating them. It contains several advanced components like MAX 30102, which continuously measures vital signs, such as heart rate and blood oxygen levels. The information obtained enables early signs of fatigue, thus timely alerting the driver or operator.Real-time surveillance and feedback of the environment and the user's behavior could be performed with a high-resolution OV 5640 camera. The capacitive touch sensor could add further interaction with the system for users. Moreover, an HM 10 Bluetooth module is included, allowing easy communication with a mobile application which would alert the user and download the data from the system.In addition, it employs a microcontroller known as the ESP32, which can process data and perform proper signal management towards effective operation. The circuit design targets the maximum power distribution of its components towards enhancing reliability in performance. This is a major innovation in the mechanisms of safety technology bound to minimize accidents related to operator fatigue and proper operation relating to overall activities.

2.2 User Characteristics

2.2.1 Aurdino IDE

The Arduino Integrated Development Environment (IDE) is a user-friendly platform designed for programming Arduino boards using a simplified version of C/C++. It features a code editor, libraries, and a built-in compiler, allowing users to write, compile, and upload code to their Arduino hardware seamlessly. The IDE supports a wide range of Arduino boards and includes tools for debugging, monitoring, and managing libraries, making it accessible for beginners and experienced developers alike.

[illegible]

Figure 2.1: Aurdino IDE

2.2.2 Cirkit Designer IDE

Cirkit Designer is a free tool that specializes in designing and diagramming circuit projects, with an emphasis on realistic component representations.Cirkit Design is making it easier for engineers, students, and hobbyists to design, document, and share circuits with its flagship product, Cirkit Designer.its is a software circuit design tool that caters to designing breadboard circuit layouts.

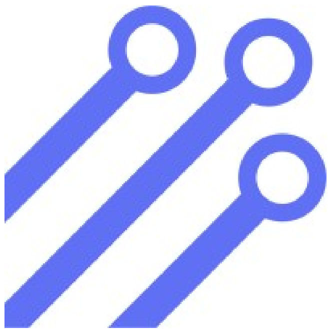

Figure 2.2: cirkit Designer IDE

2.2.3 Fusion 360

Fusion 360 is a cloud-based 3D CAD, CAM, and CAE tool developed by Autodesk that enables collaborative product design and engineering. It integrates various design processes, including parametric modeling, simulation, and manufacturing, into a single platform, making it suitable for both individual users and teams. Its cloud capabilities allow for real-time collaboration, version control, and access to design files from anywhere.

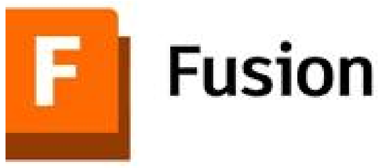

Figure 2.3: Fusion 360

Chapter 3

HARDWARE REQUIREMENTS AND SPECIFICATIONS

3.1 User Characteristics

3.1.1 Breadboard

A breadboard is a reusable platform for prototyping electronic circuits without soldering, allowing components to be easily inserted and removed. It consists of a grid of holes connected internally in rows and columns, enabling quick assembly and testing of circuit designs.A breadboard is a reusable prototyping platform that allows testing electronic circuits without soldering. It is an array of connected, usually metal, clips into which the components are easily fitted: resistors, capacitors, ICs etc., thus permitting rapid prototyping and changing of circuits. They are indispensable tools for students, hobbyists, and practicing engineers in the layout of circuits.

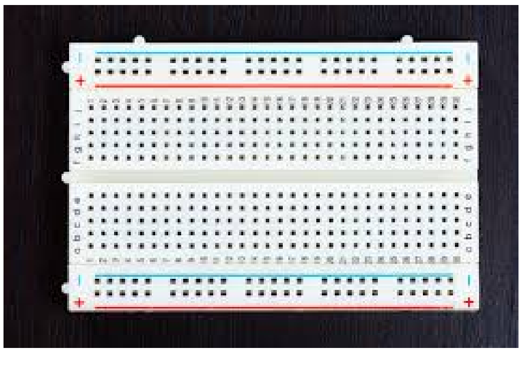

Figure 3.1: Breadboard

3.1.2 ESP32 S3

The ESP32-S3 is a highly integrated microcontroller from Espressif Systems,designed for low-power IoT applications.designed with dual-core Xtensa LX7 processors, offering advanced features like AI acceleration and enhanced I/O capabilities. It operates on a 2.4 GHz Wi-Fi and Bluetooth 5.0 dual-mode, making it suitable for IoT, wearable electronics, and smart home applications. The working principle of the ESP32-S3 revolves around its core components: Wi-Fi and Bluetooth for communication, along with GPIO (General Purpose Input/Output) pins to interact with various sensors and peripherals. It contains 45 programmable GPIO pins, which support functionalities like ADC (Analog to Digital Converter), DAC (Digital to Analog Converter), I2C, I2S, SPI, PWM, and UART communication, among others. Its low-power modes and deep sleep support make it ideal for battery-operated devices. The ESP32-S3 also integrates AI instructions and hardware acceleration for neural network inference, making it suitable for edge AI computing tasks.The reason the ESP32-S3 favored other chips was due to its stronger dual-core architecture, which offers better processing capability for complex processes. It is equipped with in-built Wi-Fi and Bluetooth functionalities that enable seamless connectivity for IoT applications. Its low power consumption combined with advanced machine learning algorithms make it suitable for use in smart devices. The ESP32-S3 also has an ample set of peripherals and substantial memory. Thus, developers can efficiently and effectively implement various applications on this board. So, the board is versatile for modern projects.

Figure 3.2: Esp32 S3

3.1.3 ESP 32

ESP 32 The ESP32 is an SoC, versatile, low-power, and of fairly low cost, which integrates both Wi-Fi and Bluetooth functionalities while being suitable for any wide range of applications for the Internet of Things, touting a dual-core Xtensa LX6 processor running at up to 240 MHz, providing plenty of computational power for multitasking and real-time processing. The ESP32 has deep sleep modes in which it draws as low a current as 10 µA. This makes it ideal for battery-powered devices such as wearables and remote sensors since it can be allowed to run longer durations without being recharged very frequently. It possesses a rich suite of peripherals with 34 GPIO pins, ADCs, DACs, PWM, and interfaces like I2C, SPI, and UART, among others. This makes the module adaptable to interface to a range of sensors, actuators, and external components. Its in-built security features such as AES encryption, secure boot, flash encryption, make it strongly protected, which makes it fit for secure IoT systems.ESP32 enjoys wide ranging support across multiple development environments, including Arduino IDE, ESP-IDF, and PlatformIO, giving the developers a great amount of flexibility in software design for projects ranging from smart home devices, industrial automation, to healthcare wearables. Compact size and wireless capabilities have also placed it as one of the most popular choices for a professional and hobbyist IoT project.There are even new dimensions of innovation when ESP32 is integrated into a fatigue prevention system to make people involved in transportation, manufacturing work, or even working remotely more secure and safe.

Figure 3.3: Esp 32

3.1.4 0v5640

The OV5640 can capture video at a range of different resolutions and frame rates, including 15 fps at maximum resolution, 30 fps at 1920 x 1080, and 60 fps at 1280 x 720 This image sensor uses a supply voltage of 3.3V for its digital components and 2.8V for the analog components, with an I/O voltage of 1.8V.It has a relatively low power usage profile, at about 294 mW when actively used and just 18 mW when set to standby. The sensor communicates through the MIPI CSI-2 interface, supporting data transmissions over up to two lanes, while communications are controlled through an I2C protocol. In terms of size, the OV5640 measures 24.5 mm x 24.5 mm x 3.45 mm and weighs less than 3 grams.It has an operating temperature range of -30°C to +70°C, and hence it is one easy solution to apply for automotive, security as well as consumer electronics fieldsFor more information about specs., please refer to datasheet from TechNexion and much more resources on the OV5640.The OV5640 image sensor will play a highly significant role in preventing the triggering of fatigue systems, especially where monitored and analyzed physical conditions are the most important. Here are some essential points of how it contributes:Real-Time Monitoring: The high-resolution imaging ability of OV5640 helps in real-time monitoring of machinery and structural integrity. A vision inspection system through this sensor helps operators detect signs of wear, cracks, or fatigue in equipment and infrastructure for enabling timely maintenance decisions to be made.Integration with Computer Vision: Once applied to a fatigue prevention system, the OV5640 can be used as part of a computer vision deployment that uses algorithms to inspect images for signs of failure.

Figure 3.4: ov7725

3.1.5 hm 10

The HM-10 module is a BLE solution popularly gained in efficiency and to be very easily integrated into various projects. Working at the 2.4 GHz ISM band, it caries the wireless communication over distances typically up to 100 meters depending on environment. Among its key features is low power consumption, which is beneficial where applications run from a battery; hence its operational life is improved with less frequent need to recharge. The HM-10 supports several modes of operation. The Master mode allows it to connect and communicate with several slave devices. The Slave mode gives it the ability to be connected to a master device, and with Transparent mode, the data transmission is simple because it passes data between devices without involving processing. This flexibility allows for a wide range of applications, from simple data logging to complex sensor networks. The HM-10 can be easily configured using simple AT commands via its UART interface, ready to be interfaced with any microcontroller, such as Arduino, Raspberry Pi, or ESP32, for example. Users can set a variety of parameters, such as the device name, pairing PIN, and baud rate, making it quite adaptable to the requirements of various projects. Additionally, the HM-10 has multiple connection capabilities and has the ability to connect up to seven slave devices simultaneously, which is beneficial for multi-sensor environments. It has a strong design and compatibility with many software platforms, hence its utilization in different projects-from smart home devices and health monitoring systems to remote control applications and wearables-is very enhanced. Generally speaking, the HM-10 module is certainly a reliable and versatile component for developers and hobbyists who aim at the implementation of Bluetooth communication in their designs.

Figure 3.5: HM-10

3.1.6 Max30102

The MAX30102 is designed as an advanced biosensor module that reads pulse oximetry, or SpO2, and heart rates. It is ideal for wearable health devices and mobile applications. It uses red LEDs and infrared, which respectively light up the skin at various points and measure the intensity of the return light reflected from blood vessels. This reflective measurement also contains oxygen saturation, as well as heart rate. It's a very small sensor size-5.6mm x 3.3mm x 1.55mm, which is ideal to integrate into small form-factor devices such as smartwatches or fitness bands. Low power consumption, less than 1mW during the measurement and as low as 0.7µA in shutdown mode-this will make constant health tracking available without much derivation on the device's battery. The MAX30102 also comes with ambient light rejection and programmable sample rates to ensure it can deliver accurate readings even in the most challenging lighting conditions or during motion. The sensor uses an I²C interface, making it easy to integrate into microcontroller-based systems. MAX30102 is used in wearable health and fitness products, portable medical devices, consumer electronics with real-time health tracking capabilities. Development can integrate any number of microcontrollers-for example, Arduino or Raspberry Pi-with the MAX30102 to build bespoke applications. Reference designs are also available to help speed prototyping along.The MAX30102 sensor is particularly useful in the application of fatigue prevention systems, especially in environments where monitoring an individual's physiological state is very important. Such environments include workplaces, long-haul driving, or industrial operations with much risk involved. These fatigue detection systems usually operate on a wide range of bio-signals for alertness determination, and heart rate variability, in combination with blood oxygen levels, plays a significant role in identifying and interpreting fatigue.

Figure 3.6: Max30102

3.1.7 Capacitive sensor

Capacitive sensors are an instrument whose changes in capacitance are primarily due to the presence or absence of a conductive object, such as a human finger. The operation is primarily based on the principle of capacitance: the ability of a system to store an electric charge. A change in the capacitance occurs when a conductive object approaches the sensor and alters the local electric field.Capacitive sensors are widely used in touch screens, proximity sensors, and in many applications of automation and robotics due to its sensitivity and responsiveness. This kind of sensor can detect non-contact gestures, and therefore is absolutely suitable for user interfaces where the interface doesn't require direct contact. Capacitive sensors are generally more reliable and durable because they have no moving parts.They can be included in different materials, such as glass and plastics. Moreover, they can be used even in high exposure environments due to the fact that they are not highly sensitive to dust and moisture content compared to the other types of sensors. In general, capacitive sensors are very versatile and efficient in modern electronic applications that contribute to device development to be intuitive and human-friendly.Enhanced Sensor Performance: Wireless power solutions will enable continuous operation of sensors used for monitoring machinery or structures in fatigue prevention systems. The idea of continuous power supply will ensure there is a record of real-time data on strain, vibratory data, and operational conditions, thus ensuring timely maintenance and reduced incidence of fatigue failure. · Safety and Reliability: Wireless power systems reduce the chances of a physical contact wearing out with time and subsequently leading to system failure.

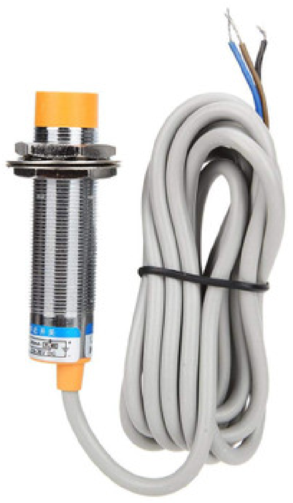

Figure 3.7: Capacitive sensor

3.1.8 Connecting Wires

Connecting wires involves joining two or more electrical wires to ensure a continuous flow of electricity. Common methods include twisting, soldering, or using connectors like wire nuts or crimp terminals. Proper insulation and secure connections are essential for safety and efficiency.

Figure 3.8: Connecting Wires

3.1.9 Potentiometer

An optical encoder is an angular or linear position transducer that translates the input into an electrical signal. It finds application in automation, robotics, and industrial machinery. In optical encoders, a rotating or moving disc interrupts light beams normally originating from LEDs with some pattern of transparent or opaque segments. As the disc rotates, light passes through or is blocked to create pulses associated with the position or speed of the moving part. These pulses are converted into digital signals by a processing unit. Optical encoders are preferred because of their accuracy, reliability, and ruggedness, so they are preferred for applications where precise position feedback is required, like CNCs, conveyor systems, and robotics. This ability of functioning in adverse conditions, along with variety configurations both incremental and absolute allows flexibility in the design and their integration in numerous systems that enlarge the scope of operation efficiency and performance.

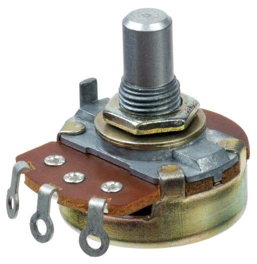

Figure 3.9: Potentiometer

3.1.10 Pantilt servo

It is the mechanism controlling panning and tilting movement in a pan-tilt servo system with a great accuracy commonly used in various fields such as robotics, surveillance cameras, and drones. There are two servos involved: one for left-right panning, and the other for up-and-down tilting to achieve the dual-axis movement. The gathering of these movements enables the system to track objects or cover a wide area without the need for adjusting the entire setup.The use of pan-tilt servos is significantly most favored in situations where flexibility and adaptability are the essentials. For instance, in security applications, a camera that can be extended to cover more ground reduces blind spots and enhances surveillance capability. In robotics, these systems allow the robot to interact more effectively with its surroundings by allowing it to point sensors or cameras toward objects or areas of interest.Such pan-tilt servo systems can be designed to provide higher accuracy for automatic operations like target tracking or object manipulation. Such advantages can be achieved by their capability of providing programmability of the movement with feed backs of systems, thus offering a dynamic solution for applications that require minute adjustments on real-time and accurate positioning-a lethal weapon in commercial and industrial applications.Pins in a pan-tilt servo system serve as crucial mechanical connectors that secure the servos to the frame, allowing for stable rotation and movement. They facilitate precise alignment of the servos, enabling smooth and accurate panning and tilting, essential for applications like robotics and surveillance systems.

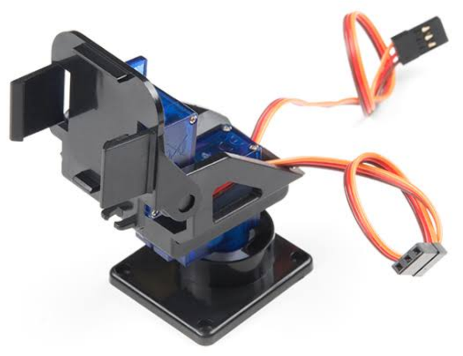

Figure 3.10: Pantiltservo

3.1.11 Optical encoder

An optical encoder is an advanced device that has accurate measurement of the angular position, speed, and direction of a rotating shaft. It usually consists of a rotatable disc having alternating transparent and opaque segments, light source, and photodetector. The disc is encoded by light that passes through the transparent ones and is not allowed to pass through the opaque ones when rotating, thus producing a stream of electrical pulses that are counted in order to determine the position and direction of the shaft rotation.Incremental encoders provide pulse counts and therefore need some external system to monitor their absolute positions; an absolute encoder provides one distinct digital code for every position and hence is always accurate even when power is lost. Optical encoders are thereby highly valued due to high resolution as well as accuracy, and they also find wide applications in industrial automation, robotics, CNC machines.Nevertheless, they were sensitive to environmental factors, like dust and vibrations, which degraded their performance. Their wide applicability and high accuracy have made these devices indispensable for several fields - starting from manufacturing to consumer electronics.Optical encoders also have a wide application in fatigue prevention systems especially in machinery and robotics where motion needs to be monitored, and also safety is always of principal importance. In the machinery and robotics applications, the optical encoders give precise feedback on their position and movement in the components. This is very important, as irregularities or excessive strain could cause failure due to fatigue. Monitoring and Feedback: The optical encoders can read the rotation and movement of machine parts in real time, thereby monitoring continuously the condition of operation. With this data analysis, the system will identify abnormal vibrations or any movement that does not normally occur. This can precede an impending failure or signs of wear and tear. Thus, maintenance or adjustment could be scheduled before a complete failure happens Safety mechanisms.

Figure 3.11: optical encoder

3.1.12 ECU (Electronic Control Unit)

An ECU is one of the most essential elements of a modern car and of many electronic systems that forms a compact computer to control particular functions or subsystems.ECUs are categorized into numerous types depending upon the various roles they perform. These include the ECU, which upgrades the performance of the engine by adjusting the fuel injection and ignition timing. It also includes the Transmission Control Unit (TCU), which controls smooth transitions of gears. Finally, there is the Body Control Module, which controls a variety of convenience features, such as lighting, door locks, and climate control.An ECU works by gathering data coming from a network of sensors scanning vehicle dynamics, environmental conditions, and user inputs and processing that information through superior algorithms to make decisions in real time, thereby enhancing performance while assuring safety.They also have diagnostics that allow them to detect problems or failures in their associated systems, typically linked to an onboard diagnostic, or OBD, systems, to return troubles to the driver or to service technicians. Network connections such as CAN allow for the multiplexed exchange of information between multiple ECUs,thus allowing coordinated control and interaction across a wide variety of vehicle systems.With ECU integration, fuel efficiency is greater, emissions decreased, and the vehicle is also added with more safety features by having anti-lock braking systems and electronic stability control.In an automobile and industries application, ECU mainly help in preventing fatigue. Some of the roles that ECU perform in this application include Real-Time Monitoring: It continuously monitors dynamics of a vehicle, the performance of the engine, besides data in sensors.

Figure 3.12: ECU

3.1.13 Airfule alliance

The AirFuel Alliance is probably the most recognized association of its kind, specifically established to further advances in wireless power technology, and through its efforts with standards for wireless charging. It was a result of the merger between the Alliance for Wireless Power, also referred to as A4WP, and the Power Matters Alliance, or PMA.This covers a wide scope of more than 195 companies within consumer electronics, automotive, and infrastructure markets while encouraging an incubative atmosphere where the innovators work on the technologies related to wireless power transfer, including inductive and resonant charging.The alliance has developed a global standard for radio frequency (RF) wireless charging. This is to support the inter-device distance-based multi-device wireless charging that needs to cater to the increasingly dominant demand of power feeding to IoT devices and brings convenience to the user. The organization avails membership access to technical resources, certification programs, and even a group for improving best practice with an ultimate goal of both enhancing product interoperability and the user experience. In integrating the world to be power wireless, in addition, wireless charging can become faster, more accessible, and more integrated in people's daily lives; in a single vision, says AirFuel Alliance, which brings together.Power Management: Ensure the system is optimized for low power consumption, especially if it's wearable.User Privacy: encrypt the data and use secure communication protocols to ensure user data safety.

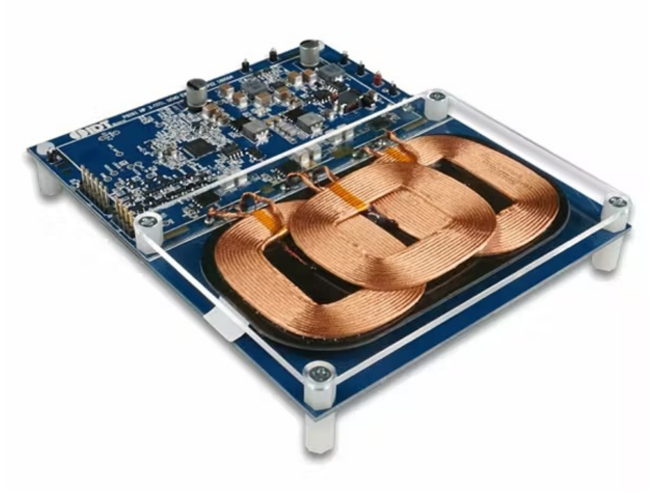

Figure 3.13: Airfuel alliance

3.1.14 Cad Model

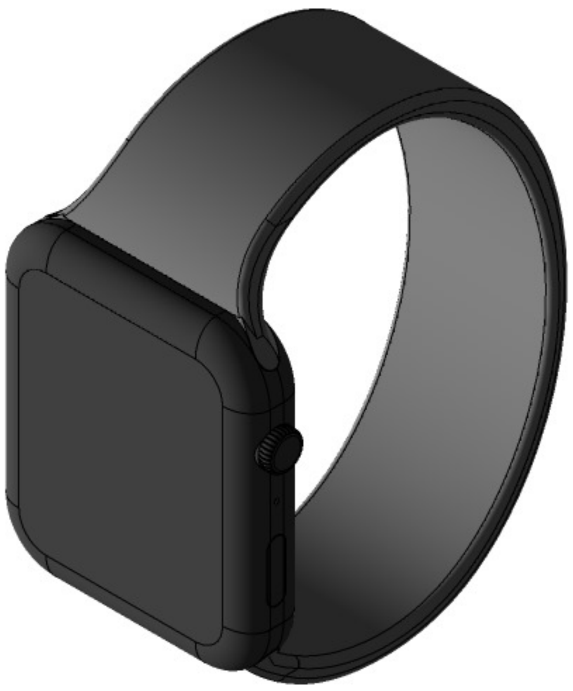

Figure 3.14: Cad

The elaborate description of the simple 3D model of the image of a wrist containing a watch strap and watch face makes it unlikely to have a pulsemeter without more information. Its requirement includes an incorporated light source such as a green or infra-red LED associated with a photo detector that measures heart rate by evaluating changes in blood volume in the capillaries just underneath the skin. These critical components would allow the device to measure the amount of light absorbed by blood during one cycle of heartbeats, and it would then be possible to calculate the heart rate with great accuracy. Without these essential components, it would be challenging to provide an approximate measurement of heart rate. Actually, in order to be applicable for heart rate monitoring, devices should be equipped with special sensors and algorithms for processing the information obtained using reflected light. Thence, unless such 3D model contains required optical elements and is made as intended for the simulation of a device with a heart rate-measuring ability, it can be concluded that such a 3D model doesn't contain a pulsemeter. If the purpose is to add this functionality to the model, then the consideration must be on the corresponding hardware and design components involved in this process to carry out effective heart rate monitoring.

Chapter 4

Implementation and working

4.1 Circuit Diagram

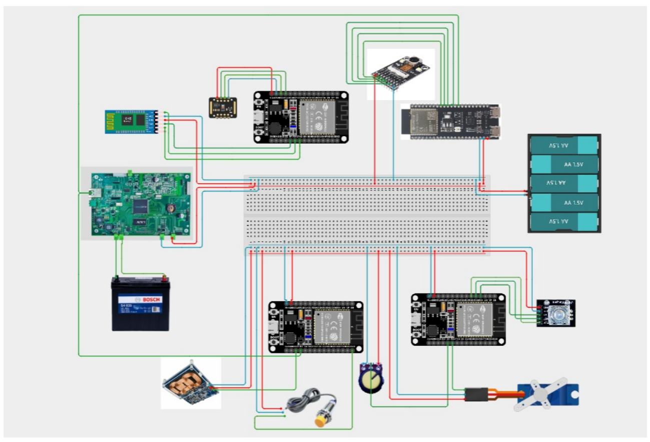

Figure 4.1: Circuit Diagram

The circuit diagram illustrates a complex system with many interconnects, color-coded for ease of identification. The green wires supply all the major components with power from the battery, along with the ECU and several ESP32 microcontrollers. Red wires highlight the connections which indicate interaction with each ESP32 unit and sensor, such as the MAX 30102 used for monitoring physiological parameters and the OV 5640 camera for capturing video data. Connects blue-coloured wires to HM 10 Bluetooth Module that allows for wireless communication with external devices. In addition, green-colored connect capacitive sensor for touch inputs and potentiometer for variable resistive inputs are connected through green-color wires. Red wires connect a servo motor that is powered by one of the units to control its position to send feedback to an optical encoder.

4.2 Block Diagram

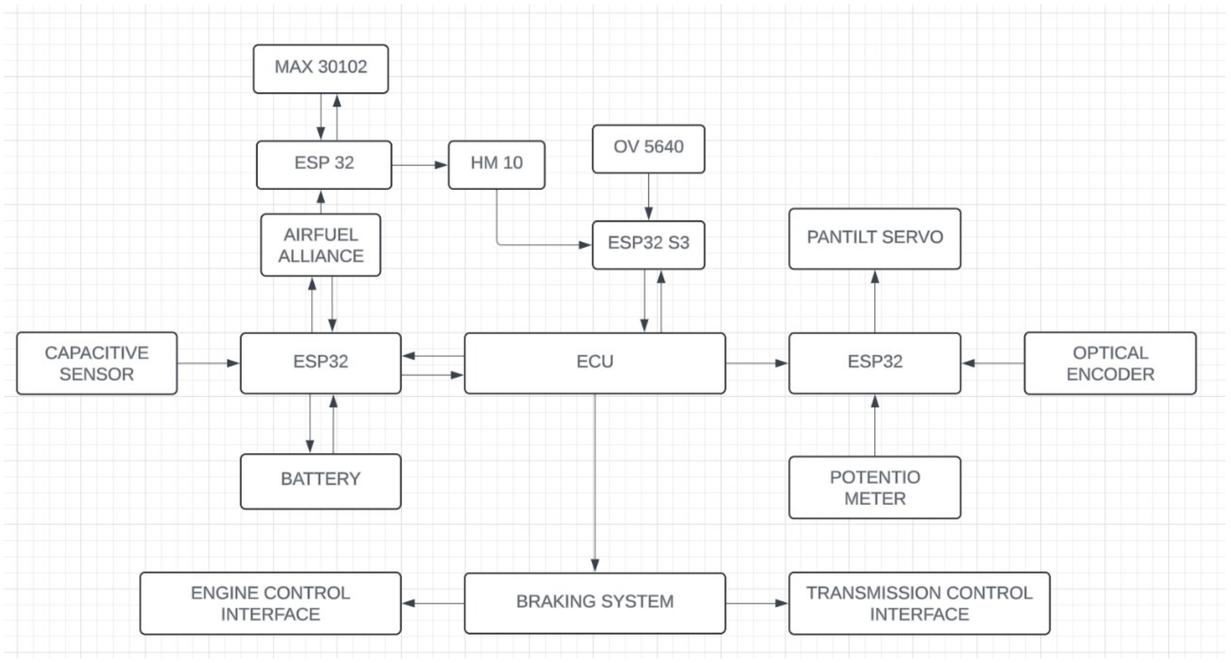

Figure 4.2: Block Diagram

The block diagram presents an efficient system that is applicable for automotive or robotics use, primarily operated by an ECU that takes in data from several sensors and feeds the outputs to an actuator. There are multiple data streams produced by the ESP32 microcontroller, one of which would be for the integration of MAX30102 sensor to monitor physiological signals such as heart rate and blood oxygen levels, monitoring vital signs of health. There is a connection to an OV5640 camera module via an ESP32 S3 bridge to intake visual data for recognition or surveillance. A wireless power transfer module is added - called Airfuel Alliance. This increases the general mobility and versatility of this system. HM 10: It can connect to external devices and even apps on the phone via Bluetooth. The Cap sensor will be able to allow touch or proximity interaction. An independent battery block addresses power needs for mobility. Again, another ESP32 drives a servo with an optical encoder for precise movement, which is very useful for dynamic applications such as camera positioning or robotic arms. A potentiometer allows for variable resistance, hence the user input for dynamic settings. The engine control unit interfaces with the braking system for the real-time monitoring and control of vehicle performance and related transmission functions-that is, sophisticated integration of technology in advanced monitoring and automation.

Chapter 5

Results and Conclusions

5.1 Results

The fatigue prevention system was evaluated based on several performance and user well-being metrics. The system consistently demonstrated its ability to accurately monitor and detect early signs of fatigue using a combination of eye movement tracking, and skin conductance. These physiological signals were analyzed in real-time, and the system's algorithm successfully identified fatigue thresholds before users themselves were aware of their declining alertness.When fatigue thresholds were reached, the system prompted users to take micro-breaks, adjust their workload, or engage in guided relaxation techniques.

Over the testing period, users who adhered to these interventions showed a substantial improvement in focus and performance consistency. Specifically, task completion times improved by 15

Another key finding was the system's impact on safety. Fatigue-related incidents, such as errors in precision tasks or delayed response times, were reduced in environments where the system was employed. These results underscore the potential of the fatigue prevention system in high-stakes industries, where sustained attention is critical to safety.

5.2 Future Enhancements

This feature enhancement proposal aims to integrate advanced sensory technologies into modern automobiles for improved safety, comfort, and driver assistance:

Brain Wave Sensors: Monitor drivers' cognitive states (e.g., fatigue, distraction) in real-time, alerting them to take breaks or switch to autonomous driving mode if needed.

Heart Rate Sensor: Continuously tracks the driver's heart rate to detect irregular patterns that may signal stress or medical emergencies, automatically notifying emergency services if necessary.

Infrared Camera: Enhances night vision and visibility in adverse weather conditions by detecting heat signatures, improving safety for both pedestrians and drivers.

Advanced Driver Assistance Systems (ADAS): are technologies in cars designed to enhance vehicle safety and improve the driving experience by automating and augmenting certain tasks. These systems rely on sensors, cameras, radar, and software to monitor the environment around the vehicle and assist the driver in various ways.

5.3 Conclusion

In conclusion, the Fatigue Prevention System represents a pivotal advancement in vehicle safety technology, serving as a vital tool in preventing injuries and fatalities resulting from drowsy driving incidents. Early detection and timely alerts are essential in averting potential accidents and saving lives on the road. The proposed system utilizes image processing techniques, detection of the pluse by plusemeter, to gauge the level of driver drowsiness. By establishing a camera and plusemeter the value indicative of drowsiness, the system can effectively identify and alert drivers to mitigate the risk of accidents. Currently, the detection system demonstrates consistent performance in identifying driver drowsiness with minimal limitations. The Fatigue Prevention System functions reliably, providing timely alerts to drivers, thereby reducing the incidents of accidents caused by drowsy driving.

In summary, the ongoing evolution of drowsiness detection technology holds immense promise in enhancing road safety and reducing the toll of accidents attributed to driver fatigue. By leveraging advanced image processing techniques, detection of pluse and incorporating adaptive alarm systems, future iterations of the driver drowsiness detection system will further bolster vehicle safety standards, ultimately saving lives and preserving well-being on our roads.

5.4 References

[1]”National Highway Traffic Safety Administration,” 2021. [Online]. Available: https://www.nhtsa.gov/risky-driving/drowsy-driving...

[2] Jadhav, Gopal, and Shailendra Aswale. ”Drowsiness Detection System for Preventing Road Accidents.” Procedia Computer Science 78 (2016): 611-616.

[3] Zhang, Xingyu, et al. ”A robust driver drowsiness detection system using convolutional neural networks.” IEEE Transactions on Intelligent Transportation Systems 22.6 (2021): 3452-3464.

[4]Hsu W, Baumgartner C, Deserno TM; Section Editors of the IMIA Yearbook Section on Sensors, Signals, and Imaging Informatics. Notable Papers and New Directions in Sensors, Signals, and Imaging Informatics. Yearb Med Inform 2021 Aug;30(1):150-8. [PMC free article] [PubMed]

[5] Sharma, Avinash, and Ritesh Kumar. ”Image Processing based Eye Blink Detection System.” Procedia Computer Science 93 (2016): 103-109.

[6] Yadav, Atul, et al. ”A Review on Vision Based Drowsiness Detection Systems.” 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT). IEEE, 2018.

[7] Wang, Qiaolan, et al. ”Driver drowsiness detection based on convolutional neural networks and adaptive particle swarm optimization.” IET Intelligent Transport Systems 14.7 (2020): 515-525.) [9] Wu, Jianan, et al. ”Real-time drowsiness detection using EEG signals: A systematic review.” Journal of Neuroscience Methods 347 (2021): 108961.

[8] Liu, Yu, et al. ”Driver Drowsiness Detection Using Deep Learning: A Comprehensive Review.” IEEE Access 9 (2021): 59394-59408.

[9] Singh, Saurabh, et al. ”A Comparative Study of Drowsiness Detection Techniques for Automobile Safety.” International Journal of Advanced Computer Science and Applications 11.1 (2020): 131-138.

[10] Kim, Youngjun, et al. ”A Review of Eye Movement-Based Drowsiness Detection Systems.” IEEE Transactions on Intelligent Transportation Systems (2022).

[11] Patel, Ruchika, et al. ”A Review on Drowsiness Detection Techniques Using Image Processing.” International Journal of Computer Applications 183.2 (2018): 21-27

Chapter 6

APPENDIX

6.1 Source code

6.1.1 code to transfer from esp32 to esp32 s3

```
#include <SoftwareSerial.h>

#include <ArduinoJson.h>

SoftwareSerial BTSerial(16, 17);  // RX, TX for HM-10

void setup() {

  Serial.begin(115200);  // Start serial communication

  BTSerial.begin(9600);  // Initialize Bluetooth communication

  Serial.println("Waiting for data from ESP32...");

}

void loop() {

  if (BTSerial.available()) {

    String jsonData = BTSerial.readStringUntil('\n');

    processSensorData(jsonData);  // Process received JSON data

  }

  // Example: You can send START and STOP commands manually

  if (Serial.available()) {

    String command = Serial.readStringUntil('\n');

    command.trim();
    if (command == "START") {
      Serial.println("Sending START command to ESP32...");
      BTSerial.println("START");
    } else if (command == "STOP") {
      Serial.println("Sending STOP command to ESP32...");
      BTSerial.println("STOP");
    }
  }
}
void processSensorData(String jsonData) {
  StaticJsonDocument<200> jsonDoc;
  DeserializationError error = deserializeJson(jsonDoc, jsonData);
  if (error) {
    Serial.print("Failed to parse JSON data: ");
    Serial.println(error.c_str());
    return;
  }
  float heartRate = jsonDoc["heartRate"];
  float SpO2 = jsonDoc["SpO2"];
  Serial.print("Heart Rate: ");
  Serial.print(heartRate);
  Serial.print(" bpm, SpO2: ");
  Serial.print(SpO2);
  Serial.println(" %");
}
```

6.1.2 pantilt servo adjustment

```
#include <Servo.h>
// Optical Encoder pins
#define ENCODER_PIN_A 15  // Pin connected to encoder output A
```

```
#define ENCODER_PIN_B 4   // Pin connected to encoder output B
// Linear potentiometer pin
#define POT_PIN 34        // Pin connected to linear potentiometer (ADC1 pin)
// Servo motor pins
#define PAN_SERVO_PIN 18  // Pin for controlling pan servo
#define TILT_SERVO_PIN 19 // Pin for controlling tilt servo
// Servo objects
Servo panServo;
Servo tiltServo;
// Encoder variables
volatile int encoderPosition = 0;
int lastEncoded = 0;
int encoderValue = 0;
int lastEncoderValue = 0;
unsigned long lastEncoderChange = 0;  // Debounce time
// Potentiometer variables
int potValue = 0;  // Store linear potentiometer reading
int smoothedPotValue = 0;
const int numPotReadings = 10;
int potReadings[numPotReadings];  // Array to store potentiometer readings for avera
int potReadIndex = 0;
int totalPot = 0;  // Sum of all readings
// Servo position variables
int panAngle = 90;  // Initial pan servo angle (centered)
int tiltAngle = 90; // Initial tilt servo angle (centered)
// Constants for debounce and timing
const unsigned long debounceDelay = 50;  // Minimum time between encoder changes
const int minPanAngle = 0;    // Minimum angle for pan servo
const int maxPanAngle = 180;  // Maximum angle for pan servo
const int minTiltAngle = 0;   // Minimum angle for tilt servo
const int maxTiltAngle = 180; // Maximum angle for tilt servo
// Interrupt service routine for encoder
void IRAM_ATTR encoderISR() {
  unsigned long currentTime = millis();
  // Debounce encoder signal
  if (currentTime - lastEncoderChange > debounceDelay) {
    int MSB = digitalRead(ENCODER_PIN_A);  // Most significant bit
    int LSB = digitalRead(ENCODER_PIN_B);  // Least significant bit
    int encoded = (MSB << 1) | LSB;  // Combine MSB and LSB
    int sum = (lastEncoded << 2) | encoded;  // Determine the rotation direction
    if (sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) {
      encoderPosition++;  // Clockwise rotation
    } else if (sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) {
      encoderPosition--;  // Counterclockwise rotation
   }
    lastEncoded = encoded;  // Update last encoded state
    lastEncoderChange = currentTime;
  }
}
void setup() {
  Serial.begin(115200);
  // Initialize encoder pins
  pinMode(ENCODER_PIN_A, INPUT);
  pinMode(ENCODER_PIN_B, INPUT);
  attachInterrupt(digitalPinToInterrupt(ENCODER_PIN_A), encoderISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(ENCODER_PIN_B), encoderISR, CHANGE);
  // Initialize potentiometer pin
  pinMode(POT_PIN, INPUT);
  // Initialize servos
  panServo.attach(PAN_SERVO_PIN);
  tiltServo.attach(TILT_SERVO_PIN);
  // Set servos to initial positions
  panServo.write(panAngle);  // Initial pan position
  tiltServo.write(tiltAngle);  // Initial tilt position
  // Initialize the potentiometer readings array
  for (int i = 0; i < numPotReadings; i++) {
    potReadings[i] = 0;
  }
}
void loop() {
  // Handle the potentiometer with averaging
  handlePotentiometer();
  // Handle the encoder with optimized input reading
  handleEncoder();
  delay(10);  // Small delay to smooth the loop
}
// Potentiometer input handling with smoothing (averaging)
void handlePotentiometer() {
  // Remove the last reading from the total
  totalPot = totalPot - potReadings[potReadIndex];
  // Read new potentiometer value
  potValue = analogRead(POT_PIN);
  // Add the new reading to the total
  potReadings[potReadIndex] = potValue;
  totalPot = totalPot + potReadings[potReadIndex];
  // Advance to the next position in the array
  potReadIndex = potReadIndex + 1;
  // If we're at the end of the array, wrap around to the beginning
  if (potReadIndex >= numPotReadings) {
    potReadIndex = 0;
  }
  // Calculate the average of the readings
  smoothedPotValue = totalPot / numPotReadings;
  // Map the smoothed potentiometer value to tilt angle
  tiltAngle = map(smoothedPotValue, 0, 4095, minTiltAngle, maxTiltAngle);
  // Write the tilt angle to the servo
  tiltServo.write(tiltAngle);
  // Debug output
  Serial.print("Potentiometer Value (Smoothed): ");
  Serial.print(smoothedPotValue);
  Serial.print("\tTilt Angle: ");
  Serial.println(tiltAngle);
}
// Encoder input handling with debounce and accuracy
void handleEncoder() {
  // Map the encoder position to the pan angle
  encoderValue = encoderPosition;  // Get current encoder position
  // Apply smoothing or constraints as needed
  if (encoderValue != lastEncoderValue) {
    panAngle = map(encoderValue, -100, 100, minPanAngle, maxPanAngle);  // Map encoc
    panServo.write(panAngle);  // Adjust pan based on seat rotation
    lastEncoderValue = encoderValue;
    // Debug output
    Serial.print("Encoder Value: ");
    Serial.print(encoderValue);
    Serial.print("\tPan Angle: ");
    Serial.println(panAngle);
  }
}
```

6.1.3 image processing code

```
#include "esp_camera.h"
#include <WiFi.h>
#include <FS.h>
#include <SPIFFS.h>
#define CAMERA_MODEL_OV7725 // Define the camera model
#include "camera_pins.h"
 // This header will include pin definitions for your OV7725 camera model
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const int captureInterval = 1000; // Capture every 1 second
const unsigned long deleteInterval = 180000; // 3 minutes in milliseconds
unsigned long lastCaptureTime = 0;
unsigned long lastDeleteTime = 0;
// Function to check if eyes are closed
bool areEyesClosed(camera_fb_t *fb) {
// Placeholder function: Replace with actual eye detection logic
// For example, analyze pixel data or apply machine learning model
return false; // Change this according to your eye detection logic
}
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  // Initialize SPIFFS
  if (!SPIFFS.begin(true)) {
```

```
  Serial.println("An error occurred while mounting SPIFFS");
  return;
}
// Configure the camera
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL;
config.ledc_timer = LEDC_TIMER;
config.pin_d0 = Y2_GPIO_NUM; // Data pin D0
config.pin_d1 = Y3_GPIO_NUM; // Data pin D1
config.pin_d2 = Y4_GPIO_NUM; // Data pin D2
config.pin_d3 = Y5_GPIO_NUM; // Data pin D3
config.pin_d4 = Y6_GPIO_NUM; // Data pin D4
config.pin_d5 = Y7_GPIO_NUM; // Data pin D5
config.pin_d6 = Y8_GPIO_NUM; // Data pin D6
config.pin_d7 = Y9_GPIO_NUM; // Data pin D7
config.pin_xclk = XCLK_GPIO_NUM; // XCLK pin
config.pin_pclk = PCLK_GPIO_NUM; // PCLK pin
config.pin_vsync = VSYNC_GPIO_NUM; // VSYNC pin
config.pin_href = HREF_GPIO_NUM; // HREF pin
config.pin_sscb_sda = SIOD_GPIO_NUM; // SDA pin
config.pin_sscb_scl = SIOC_GPIO_NUM; // SCL pin
config.pin_pwdn = PWDN_GPIO_NUM; // PWDN pin
config.pin_reset = RESET_GPIO_NUM; // RESET pin
config.xclk_freq_hz = 20000000; // Set the XCLK frequency
config.pixel_format = PIXFORMAT_JPEG; // Use JPEG format for easy storage
// Initialize the camera
if (esp_camera_init(&config) != ESP_OK) {
  Serial.println("Camera Init Failed");
  return;
}
```

```
}

void loop() {

  unsigned long currentTime = millis();

  // Capture image every second

  if (currentTime - lastCaptureTime >= captureInterval) {

    lastCaptureTime = currentTime;

    camera_fb_t *fb = esp_camera_fb_get();  // Capture frame

    if (!fb) {

      Serial.println("Camera capture failed");

      return;

    }

    // Check if eyes are closed

    if (areEyesClosed(fb)) {

      // Save image to SPIFFS

      String path = "/image_" + String(currentTime) + ".jpg";

      File file = SPIFFS.open(path.c_str(), FILE_WRITE);

      if (!file) {

        Serial.println("Failed to open file in writing mode");

      } else {

        file.write(fb->buf, fb->len); // Save the image

        Serial.printf("Saved image to: %s\n", path.c_str());

        file.close();

      }

    }

    esp_camera_fb_return(fb);  // Free the frame buffer

  }

  // Delete old images every 3 minutes

  if (currentTime - lastDeleteTime >= deleteInterval) {

    lastDeleteTime = currentTime;

    Serial.println("Deleting old images...");
    Dir dir = SPIFFS.openDir("/");
    while (dir.next()) {
    String fileName = dir.fileName();
    SPIFFS.remove(fileName); // Remove the file
    Serial.printf("Deleted: %s\n", fileName.c_str());
    }
  }
  delay(100); // Small delay to avoid flooding
}
```

6.1.4 pulse band detection

```
#include <Arduino.h>
#include <EEPROM.h>
#include <time.h>
#define PULSEBAND_PIN 34          // Capacitive sensor input pin
#define WARNING_PIN 25             // Pin for warning display (LED)
#define ECUPIN 26                  // ECU control pin (for motor control)
#define DISTANCE_TRAVELLED_PIN 27  // Pin to measure distance
#define CHARGING_PIN 32            // Pin to control charging circuit
#define EEPROM_ADDR 0              // EEPROM address for storing state
bool isPulsebandPresent = false;
bool isCarStarted = false;
unsigned long lastChargeTime = 0;
unsigned long chargeDuration = 300000; // 5 minutes charge duration
float distanceTravelled = 0;
bool charging = false;
// EEPROM States
struct State {
    bool isCarStarted;
    bool isPulsebandPresent;
    unsigned long lastChargeTime;
    bool charging;
};
// Initialize state
State currentState;
void setup() {
    Serial.begin(115200);
    EEPROM.begin(sizeof(State)); // Initialize EEPROM
    readState(); // Load state from EEPROM
    pinMode(PULSEBAND_PIN, INPUT);
    pinMode(WARNING_PIN, OUTPUT);
    pinMode(ECUPIN, OUTPUT);
    pinMode(CHARGING_PIN, OUTPUT);
    pinMode(DISTANCE_TRAVELLED_PIN, INPUT);

    // Initialize timekeeping
    configTime(0, 0, "pool.ntp.org"); // Set timezone if needed
}
void loop() {
    // Update current states
    isPulsebandPresent = digitalRead(PULSEBAND_PIN);
    isCarStarted = checkCarStarted();
    currentState.isCarStarted = isCarStarted;
    currentState.isPulsebandPresent = isPulsebandPresent;
    // Get current time
    time_t now = time(nullptr);
    struct tm* timeinfo = localtime(&now);
    int currentHour = timeinfo->tm_hour;
    // Handle warnings based on conditions
    if (isCarStarted) {
        handleWarnings(currentHour);
        // Measure distance travelled
        distanceTravelled = measureDistance();
        Serial.print("Distance travelled: ");
        Serial.println(distanceTravelled);

        // If warnings are ignored, stop the car
        if (!isPulsebandPresent) {
            stopCar();
        }

        // Charging process
        if (isPulsebandPresent) {
            chargePulseband();
        } else {
            autoOffCharging();
        }
    }

    // Save state to EEPROM
    saveState();
    delay(1000); // Adjust as necessary
}
bool checkCarStarted() {
    // Logic to determine if the car has started
    // This could be based on a specific ignition signal or another input
    return true; // Placeholder
}
```

```
void handleWarnings(int currentHour) {
    if (currentHour >= 18 || currentHour < 4) { // 6 PM to 4 AM
        displayWarning("Warning: Pulseband Required");
    }
    if (distanceTravelled > 75) {
        displayWarning("Wear Pulseband");
    }
void displayWarning(const char* message) {
    Serial.println(message);
    digitalWrite(WARNING_PIN, HIGH);
    delay(5000); // Display warning for 5 seconds
    digitalWrite(WARNING_PIN, LOW);
}
void stopCar() {
    Serial.println("Car stopping...");
    // Implement gradual stop logic here (pseudo logic)
    for (int speed = 100; speed > 0; speed -= 10) {
        analogWrite(ECUPIN, speed); // Gradually reduce speed
        delay(1000); // Wait before the next speed reduction
    }
    analogWrite(ECUPIN, 0); // Ensure the car is stopped}
void chargePulseband() {
    if (!charging) {
        Serial.println("Starting pulseband charging...");
        digitalWrite(CHARGING_PIN, HIGH);
        charging = true;
        currentState.charging = true; // Update state
        lastChargeTime = millis(); // Start charge time
    }
    // Check if charging duration is exceeded
    if (millis() - lastChargeTime >= chargeDuration) {
        Serial.println("Pulseband fully charged.");
        autoOffCharging();
    }
void autoOffCharging() {
    if (charging) {
        Serial.println("Stopping charging...");
        digitalWrite(CHARGING_PIN, LOW);
        charging = false;
        currentState.charging = false; // Update state
    }}
float measureDistance() {
    // Simulate distance measurement logic (replace with actual logic)
    // Placeholder logic for distance traveled
    // Here you would implement actual distance measuring logic
    return random(0, 100); // This is just a placeholder
}
void saveState() {
    EEPROM.put(EEPROM_ADDR, currentState);
    EEPROM.commit();
}
void readState() {
    EEPROM.get(EEPROM_ADDR, currentState);
    Serial.print("Restored State: ");
    Serial.print("Car Started: ");
    Serial.print(currentState.isCarStarted);
    Serial.print(", Pulseband Present: ");
    Serial.print(currentState.isPulsebandPresent);
    Serial.print(", Charging: ");
    Serial.println(currentState.charging);
}
```

6.1.5 Fatigue detection

```
#include <Arduino.h>
#include <EEPROM.h>
#include <time.h>
#define PULSEBAND_PIN 34          // Capacitive sensor input pin
#define WARNING_PIN 25             // Pin for warning display (LED)
#define ECUPIN 26                  // ECU control pin
#define PARKING_LIGHTS_PIN 27      // Pin for parking lights
#define MAX_ALARMS 2               // Max number of alarms before stopping
#define ALARM_DELAY 30000          // 30 seconds delay for multiple alarms
struct FatigueState {
    bool isFatigueDetected;
    int alarmCount;
    unsigned long lastAlarmTime;
};
// Initialize fatigue state
FatigueState fatigueState;
void setup() {
    Serial.begin(115200);
    pinMode(PULSEBAND_PIN, INPUT);
    pinMode(WARNING_PIN, OUTPUT);
    pinMode(ECUPIN, OUTPUT);
    pinMode(PARKING_LIGHTS_PIN, OUTPUT);
    // Load initial state from EEPROM
    readFatigueState();
}
void loop() {
    // Read pulseband values (you'll need to implement this)
    float pulseValue = readPulseband();

    // Process image for eye detection (you’ll need to implement this)

    bool eyesClosing = detectEyesClosing();

    // Check for fatigue condition

    if (pulseValue < THRESHOLD_PULSE && eyesClosing) {

        handleFatigueDetection();

    }

    // Delay for next iteration

    delay(1000);

}

void handleFatigueDetection() {

    unsigned long currentMillis = millis();

    // Check if enough time has passed since the last alarm

    if (currentMillis - fatigueState.lastAlarmTime >= ALARM_DELAY) {

        // Reset the alarm count if enough time has passed

        fatigueState.alarmCount = 0;

    }

    if (fatigueState.alarmCount < MAX_ALARMS) {

        // Send notification to ECU

        sendFatigueNotification();

        // Turn on parking lights

        digitalWrite(PARKING_LIGHTS_PIN, HIGH);

        // Increment alarm count and update last alarm time

        fatigueState.alarmCount++;

        fatigueState.lastAlarmTime = currentMillis;

        // Play alarm sound on car display or notify driver (implement your notifica

        Serial.println("Fatigue detected! Alarm activated.");

        // Wait for the driver’s response (simulate driver interaction)

        if (waitForDriverResponse()) {

            // Driver responded, stop the alarm
            stopAlarm();
            fatigueState.alarmCount = 0; // Reset alarm count
            return; // Exit without further actions
        }
    }
    // If alarms exceeded, stop the car
    if (fatigueState.alarmCount >= MAX_ALARMS) {
        stopCar();
    }
    // Save state to EEPROM
    saveFatigueState();
}
void sendFatigueNotification() {
    // Implement sending notification to ECU
    Serial.println("Notification sent to ECU: Fatigue Detected!");
}
bool waitForDriverResponse() {
    // Implement a method to wait for the driver to respond (e.g., button press)
    // Return true if the driver stops the alarm, otherwise return false
    // This is a placeholder; replace with actual implementation
    delay(10000); // Simulate waiting for 10 seconds
    return false; // Change this based on driver response
}
void stopAlarm() {
    Serial.println("Driver responded, stopping alarm.");
    digitalWrite(WARNING_PIN, LOW); // Stop warning display
    digitalWrite(PARKING_LIGHTS_PIN, LOW); // Turn off parking lights
}
void stopCar() {
    Serial.println("Car stopping due to fatigue detection...");
    for (int speed = 100; speed > 0; speed -= 10) {
        analogWrite(ECUPIN, speed); // Gradually reduce speed
        delay(1000); // Wait before the next speed reduction
    }
    analogWrite(ECUPIN, 0); // Ensure the car is stopped
}
float readPulseband() {
    // Replace this with your actual pulseband reading logic
    return random(50, 100); // Simulating pulse value
}
bool detectEyesClosing() {
    // Replace this with your actual image processing logic for eye detection
    return random(0, 2) == 0; // Simulating eyes closing with random value
}
void saveFatigueState() {
    EEPROM.put(0, fatigueState);
    EEPROM.commit();
}
void readFatigueState() {
    EEPROM.get(0, fatigueState);
    Serial.print("Restored Fatigue State: ");
    Serial.print("Fatigue Detected: ");
    Serial.print(fatigueState.isFatigueDetected);
    Serial.print(", Alarm Count: ");
    Serial.println(fatigueState.alarmCount);
}}
```

***** **THE END** *****