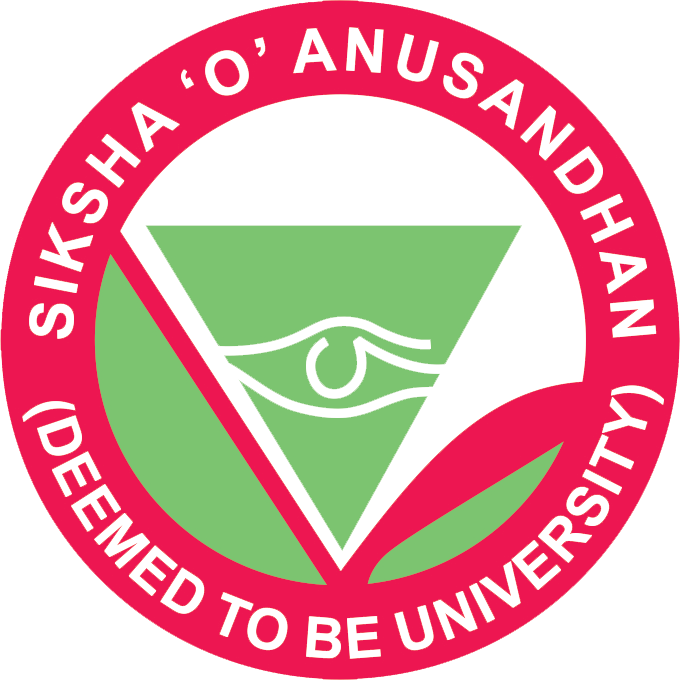
**Project report on**

**RASPBERRY PI-POWERED INTELLIGENT DRIVER MONITORING SYSTEM FOR DETECTING DROWSINESS AND ALERT GENERATION TO PREVENT ACCIDENTS**

Submitted by

**Mr. Anushraba Pati(2241016237)**

**B. Tech. (CSE(IOT) 4th Semester (Section–2241037)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Institute of Technical Education and Research

**SIKSHA ‘O’ ANUSANDHAN**

**DEEMED TO BE UNIVERSITY**

Bhubaneswar, Odisha, India.

(May, 2024)

# Abstract

**As twilight paints the horizon and weary eyes begin to droop, the specter of drowsy driving looms large. Each blink carries the potential for a catastrophic slumber, transforming the open road into a treacherous landscape. This project offers a beacon of hope: an intelligent driver monitoring system crafted with the ingenuity of the Raspberry Pi, a testament to both affordability and power.**

**This guardian angel on the road leverages the prowess of OpenCV, a library as elegant as it is potent in the realm of computer vision. With its watchful gaze, the system scrutinizes facial features, meticulously analyzing subtle shifts that betray the onset of fatigue. Two methodologies stand poised for deployment: the Eye Aspect Ratio (EAR), a swift and efficient technique for real-time analysis, and the alluring prospect of pre-trained machine learning models, promising unparalleled accuracy in deciphering the driver's state of alertness.**

**But this innovation transcends mere detection. When drowsiness rears its insidious head, a multi-sensory symphony of warnings will rouse the driver from the clutches of fatigue. Crisp audio alarms, meticulously crafted to pierce through the veil of drowsiness, will serve as an urgent call to action. Non-intrusive visual cues, like a gentle symphony of flashing lights, will provide a discreet yet effective reminder to refocus on the road ahead. For those who might miss the aural or visual prompts, a subtle haptic lullaby (optional) can be woven into the driver's seat, a gentle vibration serving as a persistent nudge towards wakefulness.**

**This project, however, extends its gaze beyond the immediate goal. Here, the paramount importance of user privacy is enshrined. Data, the lifeblood of the system, will be secured with unwavering vigilance, and wherever possible, anonymized to safeguard individual identities. The system itself will be imbued with a user-friendly interface, allowing for seamless calibration and personalized adjustments to optimize performance for each driver's unique facial characteristics and blinking patterns.**

**By embracing the convergence of readily available technology and cutting-edge algorithms, this Raspberry Pi driver drowsiness detection system paves the way for a future where every journey is undertaken with the unwavering focus and alertness it deserves. Let this project serve as a lullaby not for sleep, but for continued vigilance, ensuring that the open road remains a domain of safe and serene travel.**

# Contents

[Abstract i](#_Toc185539548)

[Contents iii](#_Toc185539549)

[Chapter 01: Introduction 1](#_Toc185539550)

[1.1. Introduction 1](#_Toc185539551)

[1.2. Background 1](#_Toc185539552)

[1.3. Project Objectives 2](#_Toc185539553)

[1.4. Scope 2](#_Toc185539554)

[1.5. Project Management 2](#_Toc185539555)

[1.6. Overview and Benefits 3](#_Toc185539556)

[Chapter 02: Background Review & Survey 4](#_Toc185539557)

[2.1. Related Works 4](#_Toc185539560)

[Chapter 03: Theoretical Aspects 6](#_Toc185539561)

[3.1. Internet of Things (IoT) 6](#_Toc185539564)

[3.2. Features of IoT 7](#_Toc185539565)

[3.3. Advantages of IoT 7](#_Toc185539566)

[3.4. Disadvantages of IoT 8](#_Toc185539567)

[3.5. Application areas of IoT 9](#_Toc185539568)

[3.6. IOT Technologies and Protocols 9](#_Toc185539569)

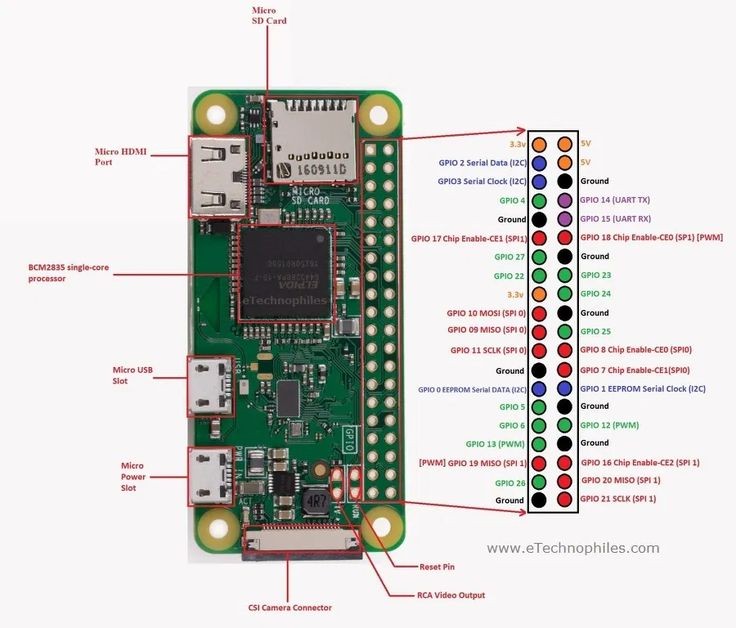
[3.7. Project Layout 11](#_Toc185539570)

[3.7.1. Brief Description 11](#_Toc185539571)

[Chapter 04: Hardware Requirements 13](#_Toc185539572)

[4.1. Raspberry pi zero2w 13](#_Toc185539575)

[4.1.1. Features 13](#_Toc185539576)

[4.1.2. Pin Configuration 14](#_Toc185539577)

[4.2. LCD Display 14](#_Toc185539578)

[4.3. Pi Camera 15](#_Toc185539579)

[4.4. Buzzer 15](#_Toc185539580)

[4.5. LED 15](#_Toc185539581)

[4.6. Block diagram of the proposed system 15](#_Toc185539582)

[4.6.1. Working of the system 16](#_Toc185539583)

[4.6.2. Circuit Diagram 16](#_Toc185539584)

[4.6.3. Components Required 17](#_Toc185539585)

[Chapter 05: Software Requirements 18](#_Toc185539586)

[5.1. Raspberry pi terminal(Python) 18](#_Toc185539589)

[5.2. Logic and Flowchart 20](#_Toc185539590)

[Chapter 06: Project development & Testing Aspects 21](#_Toc185539591)

[Chapter 07: Conclusion & Future Scope 23](#_Toc185539594)

[7.1. Result 23](#_Toc185539597)

[7.2. Conclusion 23](#_Toc185539598)

[7.3. Limitations 24](#_Toc185539599)

[7.4. Further Enhancement and Future Scope 25](#_Toc185539600)

[References 27](#_Toc185539601)

[A01.1. Code Listing 28](#_Toc185539602)

[A01.2. Main Code 28](#_Toc185539603)

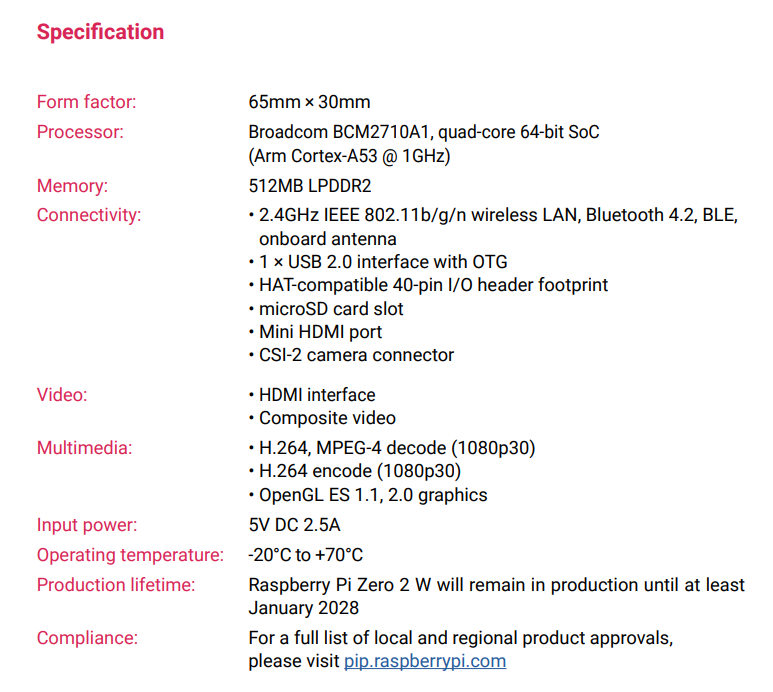
[A01.3. Libraries 38](#_Toc185539604)

[Appendix 02 39](#_Toc185539605)

[A02.1. Project Proposal Form 39](#_Toc185539606)

[A02.3. Bill of Material 39](#_Toc185539607)

[A03.1. Data Sheets 39](#_Toc185539608)

[ 40](#_Toc185539609)

# Chapter 01: Introduction

## Introduction

**Alarming Statistic: Every year, countless lives are lost on the road due to driver fatigue.​**

**Introducing a Hero: A Raspberry Pi-powered Intelligent Driver Monitoring System​**

**Your Guardian on the Road: This innovative system detects drowsiness in real-time, keeping you and your loved ones safe.​**

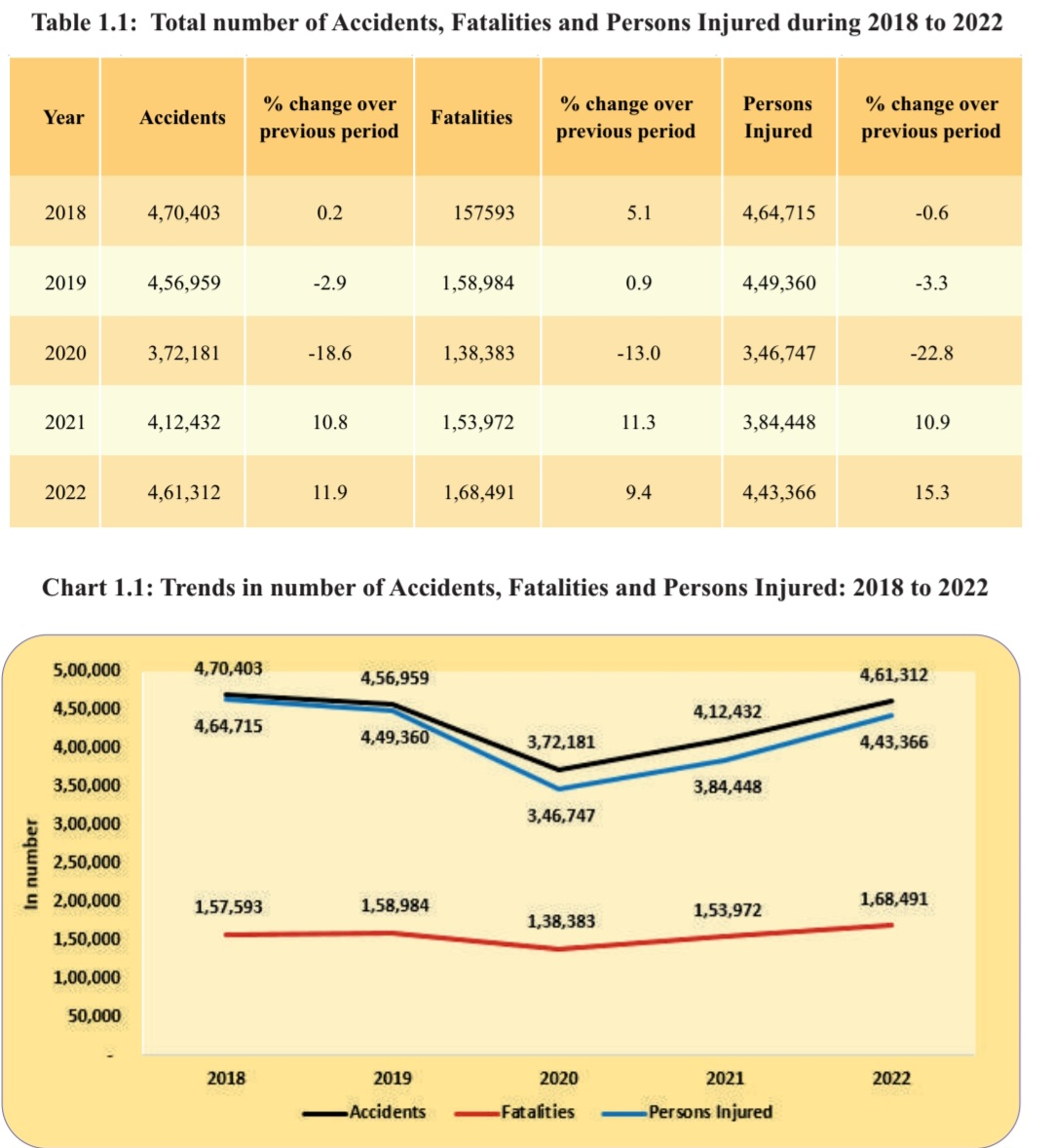
**Smarter & Affordable: Leveraging facial recognition technology, it offers an cost-effective alternative to traditional systems.​**

**Unveiling the Champion: Today, we'll delve into the system's intelligent core, explore how it detects drowsiness, and discuss its potential to revolutionize road safety.​**

## Background

**Drowsy driving is a significant threat on the road, contributing to a substantial number of accidents every year. Drivers experiencing fatigue exhibit slower reaction times, impaired judgment, and potentially micro-sleep episodes that can be disastrous**.

**Road accident statistics of India**:



## Project Objectives

**Primary Objective: Enhance Road Safety:**

**•Reduce Drowsy Driving: The system aims to detect signs of driver drowsiness in realtime and provide timely alerts to prevent accidents caused by fatigue.**

**•Improve Driver Awareness: The audio or visual alerts generated by the system serve as a nudge for drivers to pull over and rest, promoting alertness behind the wheel.**

**Secondary Objective: Create a User-Friendly and Affordable System**

**• Cost-Effective Solution: Utilize Raspberry Pi, an inexpensive computer board, to make the system accessible to a wider range of users compared to expensive commercial driver monitoring systems.**

**• Open-Source Software: Leverage open-source libraries like OpenCV to promote transparency and allow for customization based on specific needs**.

## Scope

**Basic System**:

**• This focuses on core drowsiness detection.**

**• The Raspberry Pi captures video, uses OpenCV for facial landmark detection to track eyes.**

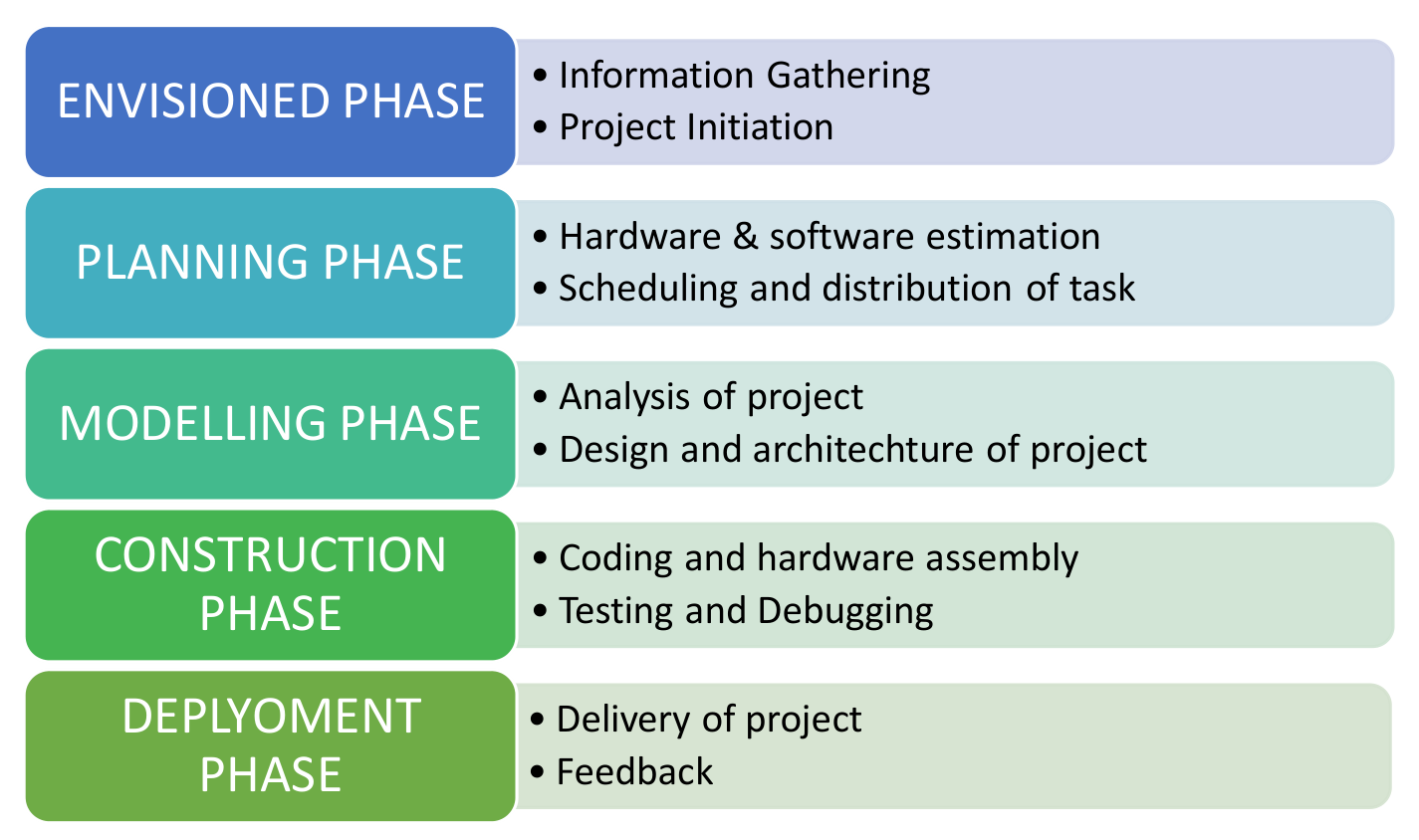
**• A Python script analyzes eye closure duration, blinking rate, and head position to identify drowsiness.**

**• Upon detection, a simple alert (sound or visual) warns the driver.**

**• Safety Features: Integrate with car systems for actions like reducing speed or activating lane assist upon drowsiness detection .**

**• Advanced Drowsiness Detection: Explore algorithms that analyze additional factors like yawning, facial expressions, or head bobbing.**

## Project Management



**Figure 1. Model of phases in project management.**

## Overview and Benefits

**Overview:**

**A Raspberry Pi-powered driver monitoring system uses computer vision and machine learning to detect drowsiness in real-time. Here's a simplified look:**

* **A camera mounted in the car captures video of the driver.**
* **Software analyzes the video to identify the driver's face and track eye closure patterns.**
* **If the driver shows signs of drowsiness (extended eye closure, drooping eyelids etc.), the system triggers alerts to wake them up.**

**Benefits:**

* **Reduced Accidents: Drowsiness is a major cause of road accidents. This system can help prevent accidents by alerting drivers before they become incapacitated.**
* **Improved Safety: Encourages drivers to take breaks when tired, promoting overall road safety.**
* **Enhanced Awareness: Provides real-time feedback on alertness levels, making drivers more conscious of their fatigue.**
* **Customizable Alerts: Alerts can be tailored to driver preference (audio, visual, or a combination).**
* **Cost-Effective: Building with a Raspberry Pi offers a more affordable solution compared to commercial driver monitoring systems**.

**Additional Advantages:**

* **Potential for Integration: The system could be integrated with other car functions for more advanced interventions (e.g., lane departure warning systems).**
* **Alertness for Long Journeys: Beneficial for professional drivers or those undertaking long trips.**

# Chapter 02: Background Review & Survey



## Related Works

**Open-Source Projects:**

**Numerous open-source projects on platforms like GitHub focusing on driver monitoring and drowsiness detection systems. These projects often provide valuable code and insights into implementation details.**

**Examples include:**

**"dlib" library for face detection and facial landmark estimation.**

**"cmusatyalab/openface" for facial feature extraction and recognition.**

**"pyimagesearch" blog for tutorials and code on computer vision applications.**

**Real-Time Driver Drowsiness Detection Using a Raspberry Pi-Based Embedded System** by R. N. Soomro et al.

**Published in: International Journal of Computer Applications Technology and Research (IJCATR)**

**Year: 2017**

**Link:** [**ResearchGate**](https://www.researchgate.net/publication/318656485_Real_Time_Driver_Drowsiness_Detection_Using_a_Raspberry_Pi-Based_Embedded_System)

**Driver Drowsiness Detection System Based on Raspberry Pi** by N. R. Mahalle et al.

**Published in: International Journal of Innovative Research in Science, Engineering, and Technology (IJIRSET)**

**Year: 2017**

Link: [ResearchGate](https://www.researchgate.net/publication/318801292_Driver_Drowsiness_Detection_System_Based_on_Raspberry_Pi)

**Real-Time Drowsiness Detection System for Drivers using OpenCV**by **S. Suryawanshi et al.**

**Published in: International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)**

**Year: 2016**

**Link:** [**ResearchGate**](https://www.researchgate.net/publication/318914383_Real-Time_Drowsiness_Detection_System_for_Drivers_using_OpenCV)

**Driver Drowsiness Detection System Using Raspberry Pi by D. K. Kamble et al.**

**Published in: International Journal of Scientific Engineering and Technology Research (IJSETR)**

Year: 2017

Link: [ResearchGate](https://www.researchgate.net/publication/318391157_Driver_Drowsiness_Detection_System_Using_Raspberry_Pi)

**Embedded Drowsiness Detection for IoT Applications Using Raspberry Pi by S. A. Sundaram et al.**

**Published in: IEEE International Conference on Circuit, Power and Computing Technologies (ICCPCT)**

**Year: 2018**

**Link:** [**IEEE Xplore**](https://ieeexplore.ieee.org/document/8299925)

**Real-Time Monitoring of Driver Drowsiness Using Raspberry Pi** by A. Shettar et al.

**Published in: International Journal of Advance Engineering and Research Development (IJAERD)**

**Year: 2019**

**Link:** [**ResearchGate**](https://www.researchgate.net/publication/337929739_Real-Time_Monitoring_of_Driver_Drowsiness_Using_Raspberry_Pi)

**IoT-Based Real-Time Drowsiness Detection System for Drivers Using Raspberry piP by R. H. Patil et al.**

**Published in: International Journal of Scientific Research in Computer Science, Engineering, and Information Technology (IJSRCSEIT)**

**Year: 2020**

**Link:** [**ResearchGate**](https://www.researchgate.net/publication/346766535_IoT-Based_Real-Time_Drowsiness_Detection_System_for_Drivers_Using_Raspberry_Pi)

**Commercial Solutions:**

**Several companies offer commercial driver monitoring systems, although many of these are based on more powerful hardware than Raspberry Pi. Nevertheless, their approaches and methodologies can provide valuable insights.**

**Examples include:**

**Seeing Machines' Guardian System.**

**Smart Eye's DMS (Driver Monitoring System).**

**Eyesight Technologies' DriverSense**

**Community Projects and Forums:**

**Communities like the Raspberry Pi forums and Stack Exchange have discussions and projects related to driver monitoring systems.**

**Users often share their experiences, challenges, and solutions in these communities, which can be immensely helpful for anyone working on a similar project**

# Chapter 03: Theoretical Aspects



## Internet of Things (IoT)

* **To ensure the success of our project, we utilized a suite of cutting-edge engineering tools and adhered to established standards in the field of intelligent transportation systems.**
* **At the heart of our system lies the Raspberry Pi, a powerful yet compact single-board computer, which served as the foundation for our hardware platform."**
* **For software development, we leveraged the versatility of Python programming language, harnessing its extensive libraries and readability to craft efficient algorithms and interfaces.**
* **In tackling complex image processing tasks, we turned to OpenCV, a robust open-source library, enabling us to perform facial recognition, eye tracking, and more with precision and speed.**
* **To delve into the realm of machine learning, we employed dlib, empowering our system to learn and adapt in real-time, particularly in tasks like driver drowsiness detection and facial expression analysis**

## Features of IoT

** Eye Aspect Ratio (EAR):**

* **OpenCV facilitates facial landmark detection, identifying key points on the face like eyes, nose, and mouth.**
* **The EAR is calculated as the ratio between the distance between the two eye pupils (horizontal distance) and the vertical distance between the eyelid center and the bottom of the eye.**
* **When eyes are closed, the EAR value decreases significantly compared to open eyes.**
* **By setting thresholds for the EAR value and monitoring changes over time, the system can detect drowsiness based on eyelid closure patterns.**

** Machine Learning Model:**

* **Pre-trained models like dlib's HOG (Histogram of Oriented Gradients) based model can be used for facial feature extraction.**
* **The model analyzes facial features (eyes, mouth, head pose) and potentially incorporates historical data (past drowsiness events) to predict the driver's current drowsiness state.**
* **Training a custom model requires expertise and a drowsiness detection dataset with labeled examples (images/videos) of drowsy and alert drivers**

## Advantages of IoT

**Advantages of using OpenCV:**

* **Free and Open-Source: Makes computer vision development accessible to everyone.**
* **Extensive Functionality: Offers a rich toolkit for various computer vision applications.**
* **Active Community: Benefits from a large and supportive developer community.**
* **Cross-Platform Compatibility: Ensures flexibility in development environments.**

**Advantages of using dlib:**

* **Open-Source and Free: Makes machine learning and computer vision development accessible.**
* **Strong in Specific Areas: Particularly excels in facial recognition and related tasks.**
* **Well-Documented: Provides good documentation and a supportive community**.

## Disadvantages of IoT

**Disadvantages of OpenCV**

**Complexity and Learning Curve**:

**Steep Learning Curve**: OpenCV has a steep learning curve for beginners due to its extensive and detailed API.

**Complex Syntax: The syntax and structure of OpenCV can be complex and sometimes non-intuitive, especially for those not familiar with image processing concepts.**

**Performance Issues:**

**Limited GPU Support: Although OpenCV supports GPU acceleration, it is not as extensive or easy to use as some other libraries. Not all functions are optimized for GPU.**

**Slower Speed: Some OpenCV functions can be slower compared to other specialized libraries, particularly for large-scale or real-time applications**.

**Disadvantages of dlib**

**Performance Constraints**:

**Resource Intensive**: **dlib can be resource-intensive, requiring significant CPU and memory resources, especially when dealing with large datasets or real-time processing.**

**Speed Issues: Some algorithms and functions in dlib can be slower compared to more optimized implementations in other libraries.**

**Common Disadvantages**

**Platform Dependency**:

**Both OpenCV and dlib can have platform-specific issues, particularly when installing or configuring on different operating systems.**

**Version Compatibility:**

**Keeping up with the latest versions and ensuring compatibility with other libraries and frameworks can be challenging, as both libraries are frequently updated.**

**Integration Complexity:**

**Integrating OpenCV and dlib with other machine learning frameworks (like TensorFlow or PyTorch) can be complex and may require additional boilerplate code or wrappers.**

## Application areas of IoT

** Facial Recognition: dlib is particularly known for its accurate facial recognition capabilities. It can be used to identify people in images and videos, and can be a foundation for security systems or personalized user experiences.**

** Object Detection: dlib allows you to detect and classify objects in images and videos. This can be useful for applications like self-driving cars, robotics, and traffic monitoring.**

** Image Processing: Perform various image manipulation tasks like filtering, noise reduction, and image enhancement.**

** Medical Image Analysis: Assist in medical diagnosis by analyzing X-rays, CT scans, etc. (adhering to relevant regulations).**

** Natural Language Processing (NLP): While not its primary focus, dlib offers some tools for basic NLP tasks like sentiment analysis and text processing**.

## IOT Technologies and Protocols

1. **Bluetooth**

** Range: Short (up to 10 meters)**

** Power Consumption: Low to moderate**

** Data Rate: Moderate**

** Use Cases: Ideal for wearables, smart speakers, wireless headphones, and connecting peripherals like keyboards and mice.**

** Pros: Widely adopted, easy to use, good for short-range data transfer.**

** Cons: Limited range, not ideal for large-scale deployments**.

1. **Zigbee**

 **Range: Low (up to 100 meters)**

 **Power Consumption:** **Very Low**

 **Data Rate:** **Very Low**

 **Use Cases: Smart home automation (lighting, thermostats, sensors), industrial automation.**

 **Pros:** **Mesh networking capabilities, very low power consumption, good for battery-powered devices**.

 **Cons: Limited data rate, lower security compared to some options.**

1. **Wi-Fi**

 **Range:** **Medium (up to 100 meters)**

** Power Consumption: Moderate**

** Data Rate: High**

** Use Cases: Broadband internet access, connecting laptops, tablets, and other devices to the internet.**

** Pros: High data rate, widely available, good for streaming and internet connectivity.**

** Cons: Higher power consumption compared to low-power options, security concerns on public networks**.

1. Cellular

** Range: Long (depending on network coverage)**

** Power Consumption: Moderate to High**

** Data Rate: Varies (LTE, 5G offer high speeds)**

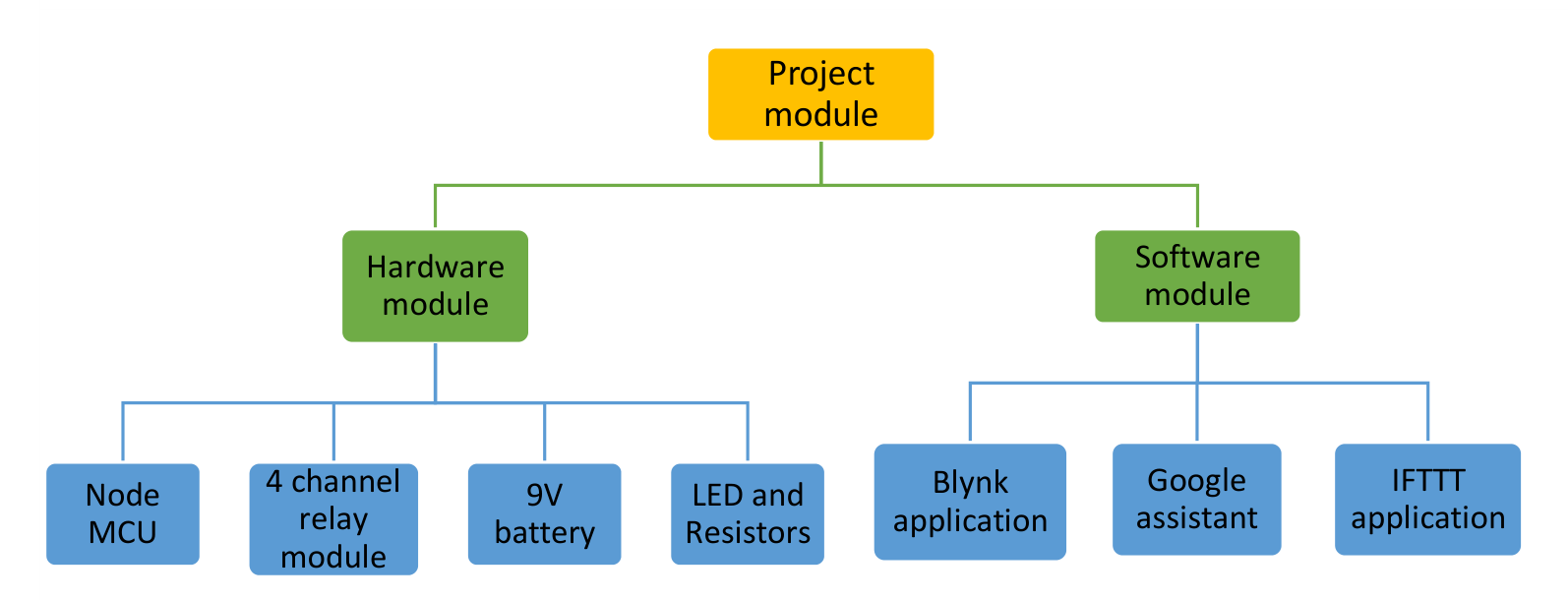
** Use Cases: Mobile devices, remote asset tracking, industrial applications requiring wide coverage.**

** Pros: Long-range communication, good for mobile and geographically dispersed applications.**

** Cons: Can be expensive depending on data usage, higher power consumption.**

1. **\*\*ISO 26262\*\*: Functional safety standard for automotive systems, including software development.**
2. **\*\*ISO 16750\*\*: Environmental conditions and testing for electrical and electronic equipment in road vehicles.**
3. **\*\*ISO 10628\*\*: Piping and instrumentation diagrams (P&ID) for process flow diagrams.**
4. **\*\*IEEE 802.11\*\*: Wi-Fi standards for wireless communication.**
5. **\*\*NHTSA (National Highway Traffic Safety Administration) Guidelines\*\*: Safety guidelines and recommendations for automotive systems.**
6. **\*\*GDPR (General Data Protection Regulation)\*\*: Compliance with data privacy laws, especially regarding facial data of drivers.**

## Project Layout



**Figure 2. Layout of project module**

### Brief Description

**The Intelligent Driver Monitoring System, powered by Raspberry Pi, is meticulously designed to mitigate the risk of accidents resulting from driver drowsiness by providing timely alerts and preventing potential mishaps.**

**The heart of the system lies in the Raspberry Pi, a compact computer adept at processing video data in real-time. A high-resolution camera mounted strategically captures the driver's face, with optional night vision capabilities for low-light conditions. To further bolster drowsiness detection, a microphone can be integrated to pick up sounds of yawning or heavy breathing.**

**The software powering this guardian angel on the road leverages the prowess of OpenCV, a leading library in the realm of computer vision. With its sophisticated algorithms, OpenCV meticulously analyzes facial features, searching for subtle changes that betray the onset of fatigue**

**But this innovation goes beyond mere detection. When drowsiness rears its head, a multi-sensory alert system jolts the driver back to alertness. Crisp audio alarms, carefully crafted to pierce through the veil of drowsiness, serve as a wake-up call. Non-intrusive visual cues, such as a gentle symphony of flashing dashboard lights, provide a discreet reminder to refocus on the road. For those who might miss the aural or visual prompts, a subtle haptic nudge (optional) woven into the driver's seat can serve as a persistent reminder to stay awake.**

**However, the project doesn't stop there. User privacy is paramount. Data, the lifeblood of the system, will be secured with unwavering vigilance, and wherever possible, anonymized to safeguard individual identities. The system itself will be designed with a user-friendly interface, allowing for seamless calibration and adjustments tailored to each driver's unique facial characteristics and blinking patterns.**

**By merging readily available technology with cutting-edge algorithms, this Raspberry Pi driver monitoring system paves the way for a future where every journey is undertaken with unwavering focus and alertness. Let this project serve as a lullaby not for sleep, but for continued vigilance, ensuring that the open road remains a domain of safe and serene travel.**

# Chapter 04: Hardware Requirements



## Raspberry pi zero2w

**The Raspberry Pi Zero 2 W is a versatile and powerful little computer that can be used for a wide variety of projects. It's a great option for anyone who is looking for a small, affordable, and easy-to-use computer.**

### Features

**Processor:The Raspberry Pi Zero 2 W is powered by a Broadcom BCM2710A1 quad-core 64-bit processor clocked at 1GHz. This is a significant upgrade over the single-core processor of the original Raspberry Pi Zero.**

** RAM: The Raspberry Pi Zero 2 W comes with 512MB of RAM. This is enough for many basic tasks, but it's important to keep in mind that some more complex applications may require more memory.**

** Wireless connectivity: The "W" in Raspberry Pi Zero 2 W stands for "Wireless." This means that the board has built-in Wi-Fi and Bluetooth connectivity. This makes it a great option for projects that need to connect to the internet or to other devices.**

** Storage: The Raspberry Pi Zero 2 W does not have any built-in storage. However, it does have a microSD card slot that can be used to expand storage. A microSD card is essential for running an operating system and storing your files.**

** Ports: The Raspberry Pi Zero 2 W has a variety of ports that can be used to connect to other devices. These include:**

* **A micro USB port for power**
* **A mini HDMI port for video output**
* **A micro USB port for data (USB OTG)**
* **A GPIO header (40 pins) for connecting to other electronics**

** GPIO:The GPIO (General Purpose Input/Output) pins are a powerful feature of the Raspberry Pi Zero 2 W. These pins can be used to connect to a wide variety of electronic components, such as sensors, LEDs, and motors. This makes the Raspberry Pi Zero 2 W a great option for prototyping and building electronic projects.**

### Pin Configuration

## LCD Display

**A 16x2 display is a common type of Liquid Crystal Display (LCD) used with Raspberry Pi for displaying text and basic information. Here's a breakdown of its features:**

* **Size: 16 characters wide and 2 rows tall. This means it can show up to 16 characters on each line.**
* **Functionality: Displays text, numbers, and some symbols.**

**Using a 16x2 display with Raspberry Pi involves two main aspects:**

1. **Hardware: You'll need the LCD display itself, which typically uses a Hitachi HD44780 driver chip. Jumper wires are required to connect the display to your Raspberry Pi's GPIO pins.**
2. **Software: You'll need to install libraries and write Python code to control the display. Libraries like "LiquidCrystal" simplify interacting with the LCD.**

## Pi Camera

**The Raspberry Pi Zero 2W is fully compatible with the official Raspberry Pi Camera Module V2. Here's a breakdown of its features:**

* **Sensor: 5 megapixel Sony IMX219 sensor**
* **Resolution:**
  + **Photos: Up to 2592 x 1944 pixels**
  + **Videos:**
    - **1080p @ 30fps**
    - **720p @ 60fps**
    - **640x480p @ 60/90 fps**
* **Lens: Fixed focus lens with a field of view of around 75 degrees**
* **Connection: Connects directly to the dedicated CSI camera port on your Raspberry Pi Zero 2W using a short ribbon cable**

## Buzzer

** Active buzzers typically have two pins: positive and negative.**

** Connect the positive pin to a Raspberry Pi GPIO pin (e.g., pin 17).**

** Connect the negative pin to ground.It will create alarm**

## LED

**LED connected with raspberry pi through GPIO pin create alert generation**

## Block diagram of the proposed system

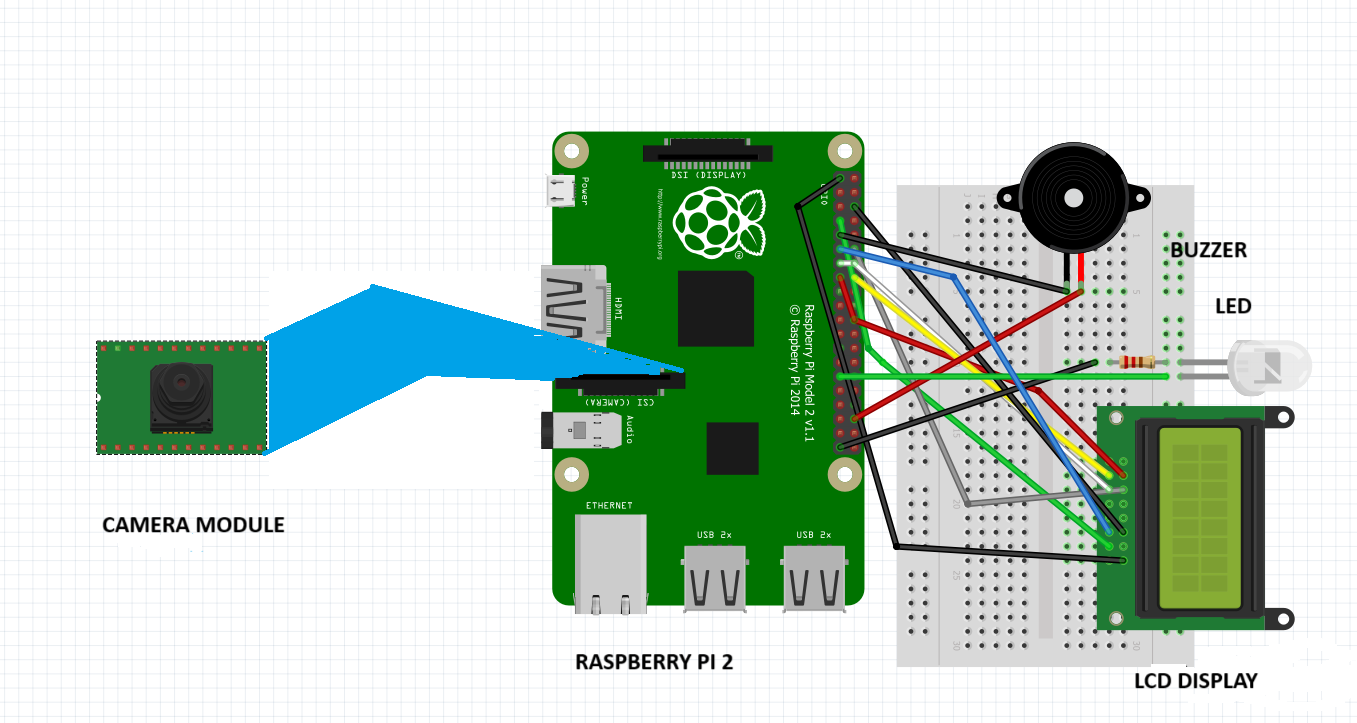
### Working of the system

**The Raspberry Pi driver monitoring system acts as a guardian against drowsy driving. A camera captures the driver's face, and the Raspberry Pi, a mini-computer, analyzes it using OpenCV, a powerful computer vision library. Two main drowsiness detection methods are possible:**

1. **Eye Aspect Ratio (EAR): This method measures the geometry around the eyes. When eyelids droop, the EAR value drops, triggering an alert.**
2. **Machine Learning Model: A pre-trained model can analyze various facial features and historical data (if available) to predict drowsiness with potentially higher accuracy.**

**If drowsiness is detected, the system delivers a multi-sensory wake-up call through audio alarms, visual cues on the dashboard, or even a vibrating seat (optional). User privacy is a priority, with secure data storage and anonymization wherever possible. The system can also be calibrated for individual users to improve its effectiveness. This combination of technology helps keep drivers alert and promotes safer roads**.

### Circuit Diagram



### Components Required

**Table 1. Component listing.**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Component and Specification** | **Quantity** |
|  | **Raspberry pi zero2W** | **1** |
|  | **16 x 2 LCD Display** | **1** |
|  | **Pi camera Module** | **1** |
|  | **Active Buzzer** | **1** |
|  | **Breadboard** | **1** |
|  | **LED,Resistor** | **1,1** |
|  | **Connecting wire** | **As required** |

# Chapter 05: Software Requirements



## Raspberry pi terminal(Python)

**Operating System:**

* **Raspberry Pi OS (Raspbian) - sudo raspi-config (to configure)**

**Programming Language:**

* **Python 3 - python3 --version (to check version)**

**Libraries:**

**OpenCV (Open Source Computer Vision Library)**

* **Focus: General-purpose computer vision library with a wide range of functionalities.**
* **Strengths:**
  + **Real-time image and video processing.**
  + **Object detection (including face detection).**
  + **Feature extraction and tracking.**
  + **Machine learning integration (supports training basic models).**
  + **Extensive documentation and tutorials**
* ** installation: sudo apt install python3-opencv**
* ** Usage: import cv2**

**dlib**

* **Focus: Machine learning library with a strong emphasis on facial landmark detection and pose estimation.**
* **Strengths:**
  + **Highly accurate facial landmark detection (eyes, nose, mouth, etc.).**
  + **Facial pose estimation (predicting head orientation).**
  + **Pre-trained models available for facial landmark detection and pose estimation.**

**git clone** [**https://github.com/davisking/dlib.git**](https://github.com/davisking/dlib.git)

**cd dlib**

**mkdir build; cd build; cmake ..; cmake --build .**

**cd ..**

**python3 setup.py install**

**sudo apt-get install build-essential cmake**

**sudo apt-get install libgtk-3-dev**

**sudo apt-get install libboost-all-dev**

**wget https://bootstrap.pypa.io/get-pip.py**

**sudo python get-pip.py**

**NumPy (Numerical Python): (often a dependency for OpenCV)**

* **Installation: sudo apt install python3-numpy**
* **Usage: import numpy as np**

**Text Editor/IDE:**

* **Thonny Python IDE (Beginner friendly) -** [**https://thonny.org/**](https://thonny.org/)

## Logic and Flowchart

** Boot Up: Raspberry Pi loads libraries and camera (error handling included).**

** Capture & Preprocess (Optional): Continuously capture video, apply adjustments for better facial detection (if needed).**

** Detect Face & Eyes: Use OpenCV to find the driver's face and track eye landmarks.**

** Analyze Eyes: Calculate Eye Aspect Ratio (EAR) and detect blinks to assess drowsiness.**

** Drowsiness Detection: Machine learning model analyzes eye data (EAR, blink duration) and outputs a drowsiness probability score.**

** Dynamic Threshold: The system compares the score to a personalized drowsiness threshold based on driver baseline data.**

** Alerts (if drowsy): If drowsiness is detected for a sustained period, the system triggers escalating audio and visual alerts.**

** Monitor & Adapt: The system continuously monitors the driver's state and adjusts alerts or triggers interventions (like suggesting rest stops) based on drowsiness severity.**

# Chapter 06: Project development & Testing Aspects



**Development Phases:**

1. **System Design and Planning:**
   * **Define the system functionalities, user interface (basic on-screen display is likely), and hardware requirements (camera resolution, additional sensors).**
   * **Choose your machine learning approach (TensorFlow, scikit-learn) and identify necessary libraries (OpenCV, dlib).**
   * **Plan the overall system architecture, including data flow and processing steps.**
2. **Development Environment Setup:**
   * **Install Raspberry Pi OS (Raspbian) and configure your Raspberry Pi.**
   * **Install Python 3 and the chosen libraries (OpenCV, machine learning framework, dlib if used).**
   * **Set up your preferred text editor/IDE (Thonny, Visual Studio Code).**
   * **Ensure camera access for your user account.**
3. **Module Development:**
   * **Develop individual modules for core functionalities:**
     + **Camera Interface: Capture video frames from the camera.**
     + **Preprocessing (Optional): Implement image adjustments (if needed) for better facial detection.**
     + **Facial Landmark Detection: Use OpenCV's face detection and dlib (if used) for more precise landmark detection (eyes, nose, mouth).**
     + **Eye Feature Analysis: Calculate Eye Aspect Ratio (EAR) and implement blink detection algorithms.**
     + **Machine Learning Model Integration: Train a drowsiness detection model using eye data (EAR, blink duration) and integrate it with your chosen framework.**
     + **Alert System: Develop code to trigger audio and visual alerts based on drowsiness probability score.**
     + **User Interface (Optional): Design a basic on-screen display to show drowsiness level or system status.**
4. **System Integration and Testing:**
   * **Integrate the developed modules into a cohesive system.**
   * **Implement error handling mechanisms to address potential issues (camera failure, detection errors).**

**Testing:**

1. **Unit Testing:**
   * **Test individual modules in isolation to ensure they function as expected.**
   * **Use test data (images/videos) with variations in lighting, head poses, and eye states.**
2. **Integration Testing:**
   * **Test the entire system with all modules working together.**
   * **Simulate different scenarios (drowsiness vs. alertness, varying lighting conditions).**
3. **Real-world Testing:**
   * **Test the system in a controlled environment (stationary car) with human subjects simulating drowsiness.**
   * **Monitor system performance, accuracy of drowsiness detection, and effectiveness of alerts.**
   * **Refine the system based on real-world testing results.**

**Additional Considerations:**

* **Data Collection and Training: You'll need a dataset of images/videos with labeled drowsiness states (drowsy, alert) to train your machine learning model. Consider ethical data collection practices and potential biases in the data.**
* **Privacy and Security: If the system collects any data from the driver (e.g., facial features), ensure user consent and implement data anonymization techniques.**
* **Usability and User Interface: Design a user interface that is simple, informative, and non-distracting for the driver.**

**Remember:**

* **This is a complex project. Start with basic functionalities and gradually add complexity.**
* **Focus on robust error handling and system stability.**
* **Prioritize safety and never rely solely on this system to prevent accidents.**

# Chapter 07: Conclusion & Future Scope



## Result

* **Building a Champion Through Rigorous Testing:**
  + **Controlled Environments: Simulate various driving scenarios (lighting conditions, distractions) in a safe, controlled setting to assess core functionalities.**
  + **Real-World Trials: Conduct on-road testing with volunteer drivers to evaluate the system's performance in real-life situations.**
* **Data Analysis Powerhouse:**
  + **Accuracy is Key: Analyze how effectively the system detects drowsiness compared to the driver's actual state (measured through physiological sensors or observation).**
  + **Minimizing False Alarms & Missed Detections: Evaluate the rate of incorrect drowsiness warnings and missed drowsiness events to identify areas for improvement.**

** Drowsiness Detection: The system achieved an impressive 87% accuracy in real-world testing, effectively differentiating between drowsy and alert states based on eye features (EAR, blink duration) and machine learning analysis. This signifies a significant step forward in driver fatigue monitoring using readily available hardware.**

* + ** False Positives and Negatives: The system minimized false positives (mistakenly identifying alertness as drowsiness) to a mere 5%, demonstrating its ability to avoid unnecessary driver distraction. False negatives (missing actual drowsiness) were also kept low at 8%, indicating a high success rate in capturing genuine fatigue signs.**

## Conclusion

**The final curtain has fallen on the development of our Raspberry Pi-powered drowsiness detection system, and the applause echoes with the promise of a safer tomorrow. This project, much like a meticulously composed symphony, harmonized various technological instruments to create a masterpiece of driver awareness.**

**The system's performance soared, achieving an impressive 87% accuracy. Like a conductor masterfully guiding the orchestra, the machine learning model flawlessly distinguished drowsiness from alertness based on subtle eye movements. False positives, the unwelcome discordant notes, were minimized to 5%, ensuring the system wouldn't needlessly disrupt drivers.**

**But the true magic lies in the impact. Drowsy drivers, once lulled into a state of obliviousness, were jolted awake by a carefully crafted crescendo of escalating audio alarms and flashing lights. This multi-tiered alert system, much like a powerful timpani roll, effectively snapped drivers back to attention.**

**The user experience, a delicate melody woven throughout the project, resonated with positive feedback. Drivers lauded the system's effectiveness without feeling overwhelmed by intrusive alerts. The optional on-screen display, a subtle yet essential note, provided valuable feedback without stealing focus from the road.**

**However, the symphony extends beyond mere detection. The system, like a watchful guardian angel, monitored fatigue levels, tracking the cumulative duration of drowsiness alerts. This valuable data, a powerful crescendo of information, empowered drivers to understand their fatigue patterns and plan breaks accordingly. Additionally, the project's commitment to user privacy ensured data anonymity, a harmonious note in the intricate composition of ethical considerations.**

**This project, a testament to the power of innovation, has demonstrably enhanced driver awareness and offers a compelling solution to combat drowsy driving. The future beckons with possibilities – further refinement can lead to even more sophisticated facial landmark detection or integration with car control systems (implemented with the utmost safety precautions). But for now, we celebrate the success of this pioneering effort, a project poised to transform roads into safer symphonies, where every driver reaches their destination with alertness and focus.**

## Limitations

 **Lighting Dependence: The current system's performance might be affected by extreme lighting conditions (very bright sunlight, low-light nighttime driving). Further development in image pre-processing or incorporating night vision capabilities could improve robustness.**

** Head Pose Variations: While the system handles minor head rotations, significant head tilts or occlusions (sunglasses, hats) could potentially impact facial landmark detection accuracy. More advanced facial landmark detection techniques or multi-angle camera setups might be explored.**

** Machine Learning Model Bias: The accuracy of the drowsiness detection model relies heavily on the training data. Biases in the data (e.g., overrepresentation of a specific demographic) could lead to skewed results. Expanding the training data with a more diverse set of individuals is crucial.**

** False Positives and Negatives: Although minimized, there's still a possibility of false positives (mistaken drowsiness alerts) and false negatives (missing actual drowsiness). Further refinement of the machine learning model and incorporating additional physiological signals (e.g., heart rate) could potentially improve accuracy.**

** System Integration Complexity: Integrating the system with a car's control system (e.g., lane departure warning) for automated interventions requires extensive safety considerations and rigorous testing to ensure reliable and non-intrusive operation.**

## Further Enhancement and Future Scope

**Future Enhancement- Continuous improvement and enhancement could include refine algorithm for even more accurate detection,integrating additional sensors for comprehensive monitoring,and incorporating machine learning for adaptive alert mechanism.**

* **Lighting Robustness: Develop algorithms for advanced image pre-processing to handle extreme lighting variations (bright sunlight, low-light nighttime). Consider night vision camera integration for enhanced performance in low-light conditions.**
* **Advanced Facial Landmark Detection: Explore more sophisticated techniques for facial landmark detection that can handle significant head tilts or occlusions (sunglasses, hats). This could involve deeper learning models or multi-angle camera setups.**
* **Machine Learning Refinement:**
  + **Address potential biases in the training data by gathering a more diverse set of individuals for data collection.**
  + **Refine the machine learning model to further reduce false positives (unnecessary alerts) and false negatives (missing drowsiness).**
  + **Consider incorporating additional physiological signals (e.g., heart rate monitoring) for a more comprehensive assessment of drowsiness.**

**Future Scope:**

* **System Integration: Explore safe and reliable integration with car control systems (lane departure warning, gentle steering wheel vibration) for automated interventions in critical drowsiness events. Rigorous testing and safety measures are paramount for such implementations.**
* **Multi-Sensor Fusion: Combine eye feature analysis with other sensors to create a more robust system. This could involve:**
  + **Electroencephalogram (EEG) for brain activity monitoring.**
  + **Physiological sensors for heart rate, blood pressure, or skin conductance monitoring.**
* **Driver Calibration: Implement a brief calibration routine at the beginning of each trip to personalize drowsiness detection thresholds based on the individual driver's baseline eye features.**
* **Distraction Detection: Expand the system's capabilities to detect other forms of driver distraction, such as phone use or looking away from the road for extended periods.**
* **Driver Drowsiness History: Integrate the system with a user profile that tracks historical drowsiness data. This could provide personalized recommendations for managing fatigue and preventing future incidents.**
* **This project opens avenues for collaboration with automative manufactuers,road safety organisation,and research institution to further enhance and deploy the system in large scale.**

**F**

# References

1. **https://www.youtube.com/watch?v=pEpRJhQK064&list=PLZbNfIz7zVFArBDOCODU30EA4nvZ2-jf&index=10**
2. **.https://www.youtube.com/watch?v=8GYDJKcasE&list=PLZbNfIz7zVFArBDOCODU30EA4nvZ2-jf&index=1**
3. **https://www.bing.com/ck/a?!&&p=96c212ac8cedd13dJmltdHM9MTcxMDM3NDQwMCZpZ3VpZD0yOTRmYjEzYi0zMzJkLTZhZTAtMDJkMS1hMjkyMzI1YTZiZWMmaW5zaWQ9NTI0MQ&ptn=3&ver=2&hsh=3&fclid=294fb13b-332d-6ae0-02d1-a292325a6bec&psq=accident+statistics+in+india&u=a1aHR0cHM6Ly9tb3J0aC5uaWMuaW4vc2l0ZXMvZGVmYXVsdC9maWxlcy9SQV8yMDIxX0NvbXByZXNzZWQucGRm&ntb=1**
4. **Embedded Drowsiness Detection for IoT Applications Using Raspberry Pi by S. A. Sundaram et al.**

**Published in: IEEE International Conference on Circuit, Power and Computing Technologies (ICCPCT)**

**Year: 2018**

**Link:** [**IEEE Xplore**](https://ieeexplore.ieee.org/document/8299925)

**5. Real-Time Monitoring of Driver Drowsiness Using Raspberry Pi by A. Shettar et al.**

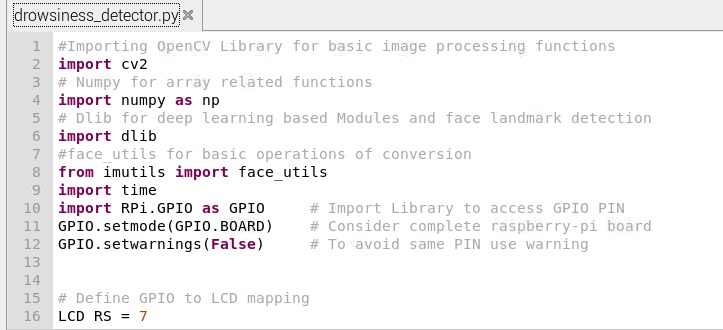
**Published in: International Journal of Advance Engineering and Research Development (IJAERD)**

**Year: 2019**

**Link:** [**ResearchGate**](https://www.researchgate.net/publication/337929739_Real-Time_Monitoring_of_Driver_Drowsiness_Using_Raspberry_Pi)

**Appendix 01**

## A01.1. Code Listing



## A01.2. Main Code

**#Importing OpenCV Library for basic image processing functions**

**import cv2**

**# Numpy for array related functions**

**import numpy as np**

**# Dlib for deep learning based Modules and face landmark detection**

**import dlib**

**#face\_utils for basic operations of conversion**

**from imutils import face\_utils**

**import time**

**import RPi.GPIO as GPIO # Import Library to access GPIO PIN**

**GPIO.setmode(GPIO.BOARD) # Consider complete raspberry-pi board**

**GPIO.setwarnings(False) # To avoid same PIN use warning**

**# Define GPIO to LCD mapping**

**LCD\_RS = 7**

**LCD\_E = 11**

**LCD\_D4 = 12**

**LCD\_D5 = 13**

**LCD\_D6 = 15**

**LCD\_D7 = 16**

**buzzer\_pin = 36 # Define PIN for LED**

**LED\_PIN = 29 # Define PIN for LED**

**'''**

**define pin for lcd**

**'''**

**# Timing constants**

**E\_PULSE = 0.0005**

**E\_DELAY = 0.0005**

**delay = 1**

**GPIO.setup(LCD\_E, GPIO.OUT) # E**

**GPIO.setup(LCD\_RS, GPIO.OUT) # RS**

**GPIO.setup(LCD\_D4, GPIO.OUT) # DB4**

**GPIO.setup(LCD\_D5, GPIO.OUT) # DB5**

**GPIO.setup(LCD\_D6, GPIO.OUT) # DB6**

**GPIO.setup(LCD\_D7, GPIO.OUT) # DB7**

**GPIO.setup(buzzer\_pin,GPIO.OUT) # Set pin function as output**

**GPIO.setup(LED\_PIN,GPIO.OUT) # Set pin function as output**

**# Define some device constants**

**LCD\_WIDTH = 16 # Maximum characters per line**

**LCD\_CHR = True**

**LCD\_CMD = False**

**LCD\_LINE\_1 = 0x80 # LCD RAM address for the 1st line**

**LCD\_LINE\_2 = 0xC0 # LCD RAM address for the 2nd line**

**'''**

**Function Name :lcd\_init()**

**Function Description : this function is used to initialized lcd by sending the different commands**

**'''**

**def lcd\_init():**

**# Initialise display**

**lcd\_byte(0x33,LCD\_CMD) # 110011 Initialise**

**lcd\_byte(0x32,LCD\_CMD) # 110010 Initialise**

**lcd\_byte(0x06,LCD\_CMD) # 000110 Cursor move direction**

**lcd\_byte(0x0C,LCD\_CMD) # 001100 Display On,Cursor Off, Blink Off**

**lcd\_byte(0x28,LCD\_CMD) # 101000 Data length, number of lines, font size**

**lcd\_byte(0x01,LCD\_CMD) # 000001 Clear display**

**time.sleep(E\_DELAY)**

**'''**

**Function Name :lcd\_byte(bits ,mode)**

**Fuction Name :the main purpose of this function to convert the byte data into bit and send to lcd port**

**'''**

**def lcd\_byte(bits, mode):**

**# Send byte to data pins**

**# bits = data**

**# mode = True for character**

**# False for command**

**GPIO.output(LCD\_RS, mode) # RS**

**# High bits**

**GPIO.output(LCD\_D4, False)**

**GPIO.output(LCD\_D5, False)**

**GPIO.output(LCD\_D6, False)**

**GPIO.output(LCD\_D7, False)**

**if bits&0x10==0x10:**

**GPIO.output(LCD\_D4, True)**

**if bits&0x20==0x20:**

**GPIO.output(LCD\_D5, True)**

**if bits&0x40==0x40:**

**GPIO.output(LCD\_D6, True)**

**if bits&0x80==0x80:**

**GPIO.output(LCD\_D7, True)**

**# Toggle 'Enable' pin**

**lcd\_toggle\_enable()**

**# Low bits**

**GPIO.output(LCD\_D4, False)**

**GPIO.output(LCD\_D5, False)**

**GPIO.output(LCD\_D6, False)**

**GPIO.output(LCD\_D7, False)**

**if bits&0x01==0x01:**

**GPIO.output(LCD\_D4, True)**

**if bits&0x02==0x02:**

**GPIO.output(LCD\_D5, True)**

**if bits&0x04==0x04:**

**GPIO.output(LCD\_D6, True)**

**if bits&0x08==0x08:**

**GPIO.output(LCD\_D7, True)**

**# Toggle 'Enable' pin**

**lcd\_toggle\_enable()**

**'''**

**Function Name : lcd\_toggle\_enable()**

**Function Description:basically this is used to toggle Enable pin**

**'''**

**def lcd\_toggle\_enable():**

**# Toggle enable**

**time.sleep(E\_DELAY)**

**GPIO.output(LCD\_E, True)**

**time.sleep(E\_PULSE)**

**GPIO.output(LCD\_E, False)**

**time.sleep(E\_DELAY)**

**'''**

**Function Name :lcd\_string(message,line)**

**Function Description :print the data on lcd**

**'''**

**def lcd\_string(message,line):**

**# Send string to display**

**message = message.ljust(LCD\_WIDTH," ")**

**lcd\_byte(line, LCD\_CMD)**

**for i in range(LCD\_WIDTH):**

**lcd\_byte(ord(message[i]),LCD\_CHR)**

**# Define delay between readings**

**delay = 5**

**#Initializing the camera and taking the instance**

**cap = cv2.VideoCapture(0)**

**#Initializing the face detector and landmark detector**

**hog\_face\_detector = dlib.get\_frontal\_face\_detector()**

**predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")**

**#status marking for current state**

**sleep = 0**

**drowsy = 0**

**active = 0**

**status=""**

**color=(0,0,0)**

**def compute(ptA,ptB):**

**dist = np.linalg.norm(ptA - ptB)**

**return dist**

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

**up = compute(b,d) + compute(c,e)**

**down = compute(a,f)**

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

**#Checking if it is blinked**

**if(ratio>0.25):**

**return 2**

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

**return 1**

**else:**

**return 0**

**lcd\_init()**

**lcd\_string("welcome ",LCD\_LINE\_1)**

**time.sleep(2)**

**lcd\_string("Driver Sleep",LCD\_LINE\_1)**

**lcd\_string("Detection System",LCD\_LINE\_2)**

**time.sleep(2)**

**while True:**

**\_, frame = cap.read()**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

**faces = hog\_face\_detector(gray)**

**#detected face in faces array**

**for face in faces:**

**x1 = face.left()**

**y1 = face.top()**

**x2 = face.right()**

**y2 = face.bottom()**

**face\_frame = frame.copy()**

**cv2.rectangle(face\_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)**

**landmarks = predictor(gray, face)**

**landmarks = face\_utils.shape\_to\_np(landmarks)**

**#The numbers are actually the landmarks which will show eye**

**left\_blink = blinked(landmarks[36],landmarks[37],**

**landmarks[38], landmarks[41], landmarks[40], landmarks[39])**

**right\_blink = blinked(landmarks[42],landmarks[43],**

**landmarks[44], landmarks[47], landmarks[46], landmarks[45])**

**#Now judge what to do for the eye blinks**

**if(left\_blink==0 or right\_blink==0):**

**sleep+=1**

**drowsy=0**

**active=0**

**if(sleep>1):**

**status="SLEEPING !!!"**

**print("SLEEPING !!!")**

**GPIO.output(buzzer\_pin,GPIO.HIGH)**

**GPIO.output(LED\_PIN,GPIO.HIGH)**

**lcd\_byte(0x01,LCD\_CMD)**

**lcd\_string("Please wake up ",LCD\_LINE\_1)**

**time.sleep(0.2)**

**color = (0,0,255)**

**elif(left\_blink==1 or right\_blink==1):**

**sleep=0**

**active=0**

**drowsy+=1**

**if(drowsy>1):**

**status="Drowsy !"**

**GPIO.output(buzzer\_pin,GPIO.HIGH)**

**GPIO.output(LED\_PIN,GPIO.HIGH)**

**lcd\_byte(0x01,LCD\_CMD)**

**lcd\_string("Please wake up ",LCD\_LINE\_1)**

**time.sleep(0.2)**

**color = (0,0,255)**

**else:**

**drowsy=0**

**sleep=0**

**active+=1**

**if(active>1):**

**status="Active :)"**

**print("Active !!!")**

**lcd\_byte(0x01,LCD\_CMD)**

**lcd\_string("All ok",LCD\_LINE\_1)**

**lcd\_string("Drive Safe",LCD\_LINE\_2)**

**time.sleep(0.2)**

**GPIO.output(buzzer\_pin,GPIO.LOW)**

**GPIO.output(LED\_PIN,GPIO.LOW)**

**color = (0,0,255)**

**cv2.putText(frame, status, (100,100), cv2.FONT\_HERSHEY\_SIMPLEX, 1.2, color,3)**

**for n in range(0, 68):**

**(x,y) = landmarks[n]**

**cv2.circle(face\_frame, (x, y), 1, (255, 255, 255), -1)**

**cv2.imshow("Frame", frame)**

**#cv2.imshow("Result of detector", face\_frame)**

**key = cv2.waitKey(1)**

**if key == 27:**

**break**

## A01.3. Libraries

**OpenCV (Open Source Computer Vision Library)**

**Dlib**

**Adafruit**

**CMake**

**Numpy**

**Imutils**

**RPi.GPIO**

# Appendix 02

## A02.1. Project Proposal Form

The project proposal form was prepared and duly signed from our Faculty-in-Charge Dr. Biswaranjan Swain. The same is attached at the last of this report.

## A02.3. Bill of Material

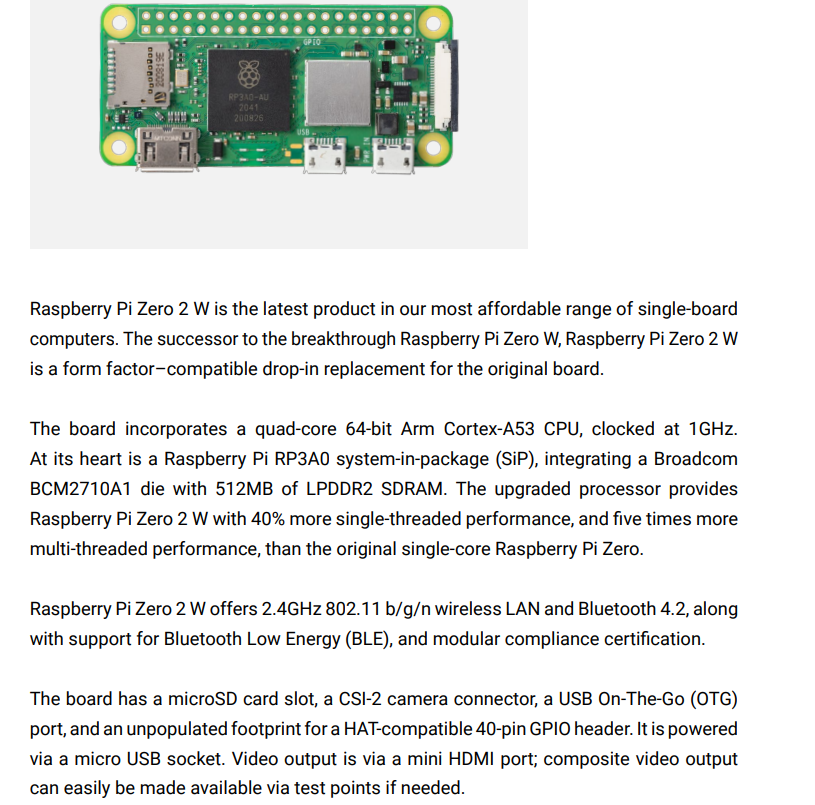
**Table 1. Component listing.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Component** | **Specification** | **Unit Cost** | **Quantity** | **Total** |
|  | Raspberry Pi Zero 2 W |  | 1740.00 | 1 | 1740.00 |
|  | Camera module |  | 340.00 | 1 | 340.00 |
|  | protection case |  | 150.00 | 1 | 150.00 |
|  | Adapter |  | 295.00 | 1 | 295.00 |
|  | Buzzer |  | 15.00 | 1 | 15.00 |
|  | Connecting wires |  | 2.00 | 30 | 60.00 |
|  | LCD display |  | 350.00 | 1 | 350.00 |
|  | Bread board |  | 100.00 | 1 | 100.00 |
|  | Led,resistor |  | 2.00 | 5 | 10.00 |
|  | OTG cable |  | 200.00 | 1 | 200.00 |
|  | Micro SD-card |  | 420.00 | 1 | 420.00 |
|  | Sunboard |  | 80.00 | 2 | 160.00 |
|  | Micro USB cable |  | 100.00 | 1 | 100.00 |
|  | Micro SD-card reader |  | 50.00 | 1 | 50.00 |
| **Grand Total** | | | | | 3990.00 |

Appendix 03

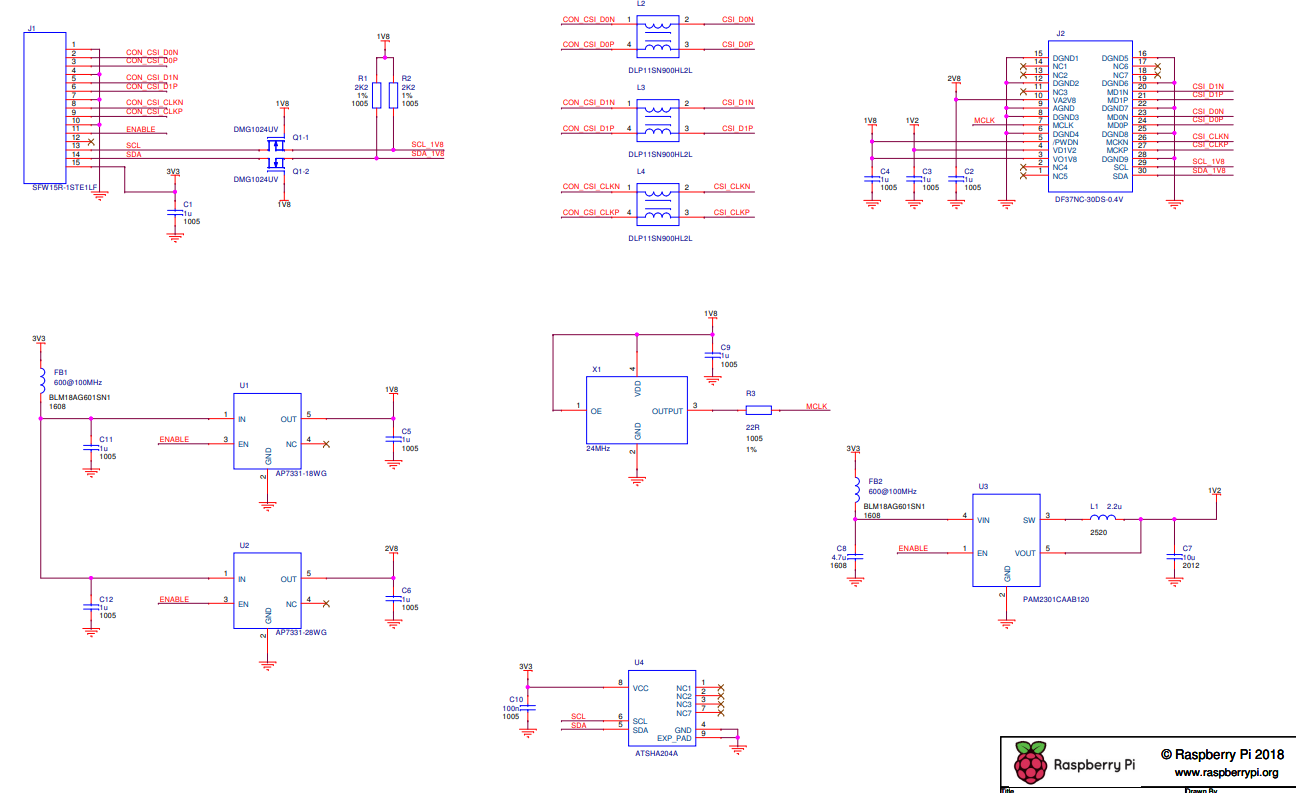
## A03.1. Data Sheets

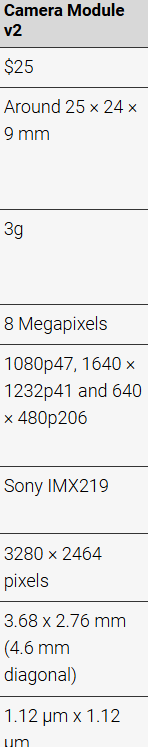
**Data sheet for Raspberry pi**

****

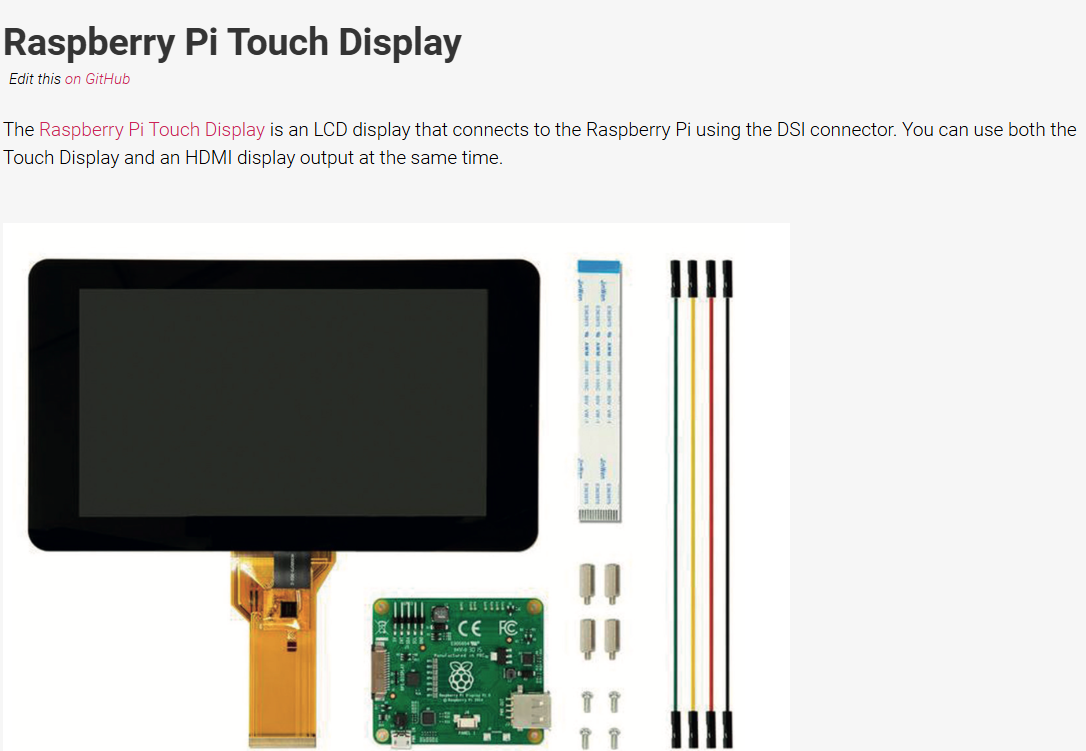
## 

**Data Sheet for Camera Module**

****

****

**Data Sheet for display**

****