A blue and black text

Description automatically generated

**Software Engineering Department**

Ort Braude College of Engineering  
Capstone Project Phase A – 61998

**AlertWatch**



**24-2-D-8   
Advisor:**  
Shimon Faitelson  
fshimon@braude.ac.il

**Student:**  
Lior Jigalo  
lior.jigalo@e.braude.ac.il  
[GitHub Repository](https://github.com/audiblemaple/AlertWatch)

Contents

[Project definition 5](#_Toc177724009)

[Abstract 5](#_Toc177724010)

[1. Introduction 5](#_Toc177724011)

[2. Related Work 7](#_Toc177724012)

[3. Background 9](#_Toc177724013)

[3.1 Real-Time Monitoring and Alertness Management 9](#_Toc177724014)

[3.2 Embedded Linux Build Systems 9](#_Toc177724015)

[3.2.1 Yocto Project 9](#_Toc177724016)

[3.2.2 BuildRoot 10](#_Toc177724017)

[3.2.3 Comparison 11](#_Toc177724018)

[3.3 Artificial Intelligence Models & Tools 12](#_Toc177724019)

[3.3.1 Key Features of PyTorch 12](#_Toc177724020)

[3.3.2 Applications of PyTorch 13](#_Toc177724021)

[3.3.3 Dataset 13](#_Toc177724022)

[3.3.4 Key Features of ResNet18 13](#_Toc177724023)

[3.3.5 Applications of ResNet18 14](#_Toc177724024)

[3.3.6 Key Features of SSD 14](#_Toc177724025)

[3.3.7 Applications of SSD 15](#_Toc177724026)

[3.3.8 Wav2Vec2 15](#_Toc177724027)

[3.3.9 Integration 15](#_Toc177724028)

[4. Expected Achievements 16](#_Toc177724029)

[4.1 Outcomes 16](#_Toc177724030)

[4.2 Unique Features 16](#_Toc177724031)

[4.2.1 AI-Based Alertness Monitoring 16](#_Toc177724032)

[4.2.2 Real-Time Feedback 16](#_Toc177724033)

[4.2.3 Adaptive Environment Integration 16](#_Toc177724034)

[4.3 Criteria for Success 17](#_Toc177724035)

[5. Project Constraints 18](#_Toc177724036)

[5.1 Environmental Constraints 18](#_Toc177724037)

[5.2 camera constraints 18](#_Toc177724038)

[6. Expected Engineering challenges. 18](#_Toc177724039)

[6.1 Hailo platform development 18](#_Toc177724040)

[6.2 AI model development 19](#_Toc177724041)

[6.3 Custom Linux images 19](#_Toc177724042)

[7. Engineering Process 20](#_Toc177724043)

[7.1 Research – AI 20](#_Toc177724044)

[7.1.1 Constraints 20](#_Toc177724045)

[7.2 Research – Hardware 21](#_Toc177724046)

[7.2.1 Option 1: SolidRun i.MX 8M Plus-Based Board with Hailo-8 Module 21](#_Toc177724047)

[7.2.2 Option 2: SolidRun Hailo15 CPU-Based Board 22](#_Toc177724048)

[7.2.3 Option 3: SolidRun Bedrock R7000/8000 CPU Calculations 24](#_Toc177724049)

[7.2.4 Common Features and Peripherals 25](#_Toc177724050)

[7.2.5 Conclusion 25](#_Toc177724051)

[7.3 Initial Setup and Prototyping 25](#_Toc177724052)

[7.3.1 Algorithm Development 25](#_Toc177724053)

[7.3.2 System Integration 26](#_Toc177724054)

[7.4 Testing and Validation 26](#_Toc177724055)

[7.4.1 Functional Testing 26](#_Toc177724056)

[7.4.2 Performance Testing 27](#_Toc177724057)

[7.4.3 User Testing 27](#_Toc177724058)

[7.5 Deployment and Maintenance 27](#_Toc177724059)

[7.5.1 Deployment 27](#_Toc177724060)

[7.5.2 Maintenance and Updates 27](#_Toc177724061)

[7.6 ECU Interface 27](#_Toc177724062)

[8. Product 28](#_Toc177724063)

[8.1 Detection system Requirements 28](#_Toc177724064)

[8.1.1 Functional Requirements 28](#_Toc177724065)

[8.1.2 Non-Functional Requirements 28](#_Toc177724066)

[8.2 ECU functional Requirements 28](#_Toc177724067)

[8.2.1 Functional Requirements 28](#_Toc177724068)

[8.2.2 Non-Functional Requirements 28](#_Toc177724069)

[8.3 Architecture Overview 29](#_Toc177724070)

[8.4 Data Flow 29](#_Toc177724071)

[8.5 Data Storage 30](#_Toc177724072)

[8.6 Detection system diagrams 31](#_Toc177724073)

[8.6.1 Use Case Diagram 31](#_Toc177724074)

[8.6.2 Activity Diagram 32](#_Toc177724075)

[8.6.3 Sequence Diagram 33](#_Toc177724076)

[8.7 ECU diagrams 34](#_Toc177724077)

[8.7.1 UI Mockup 34](#_Toc177724078)

[8.7.2 Use Case Diagram 35](#_Toc177724079)

[8.7.3 Activity Diagram 36](#_Toc177724080)

[8.7.4 Class Diagram 37](#_Toc177724081)

[8.7.5 Sequence Diagram 38](#_Toc177724082)

[9. Verification and Evaluation 39](#_Toc177724083)

[9.1 Evaluation 39](#_Toc177724084)

[9.1.1 ECU system stability 39](#_Toc177724085)

[9.1.2 Detection system accuracy 39](#_Toc177724086)

[10. Integrated Development Environments (IDEs) 39](#_Toc177724087)

[11. OS used 39](#_Toc177724088)

[12. Tools 40](#_Toc177724089)

[13. References 41](#_Toc177724090)

# Project definition

# Abstract

In modern environments, whether in vehicles, offices, or other settings, maintaining alertness and preventing fatigue-related incidents has become a critical concern. "AlertWatch" presents an innovative solution utilizing advanced computer vision and hardware to monitor individuals and detect signs of drowsiness in real-time. This project leverages facial recognition and facial landmark detection to continuously assess the alertness levels of a person. Upon detecting signs of drowsiness, "AlertWatch" triggers immediate alerts to prevent potential accidents, lapses in productivity or even security concerns. This paper details the development process of "AlertWatch," focusing on its implementation of real-time monitoring through cameras and software algorithms, as well as outlining the development process and hardware options. "AlertWatch" enhances safety and efficiency and sets new standards for proactive alertness management through technology.

# 1. Introduction

Fatigue is a significant contributor to accidents and reduced productivity, posing serious risks to safety and efficiency. Over 100,000 police-reported crashes and more than 1,500 fatalities reported Annually, because of drowsy driving [1]. Traditional methods of addressing drowsiness, such as regular breaks, manual monitoring, and lane departure monitoring, have proven insufficient in providing real-time, reliable prevention measures.

Current technological solutions, like driver drowsiness detection systems in high-end or commercial vehicles, offer some level of alertness monitoring. However, these systems often lack the flexibility to be applied across different environments as they are integrated into the vehicle, and utilize lane departure monitoring, making them unusable for other use cases. Moreover, many existing systems focus primarily on reactive measures, such as detecting that the vehicle has already crossed or is about to cross the lane, rather than proactive alertness management.

"AlertWatch" is designed to fill this gap by providing a versatile, real-time monitoring system that can be deployed in various settings to detect and alert users of drowsiness. Leveraging computer vision and machine learning technologies, "AlertWatch" continuously monitors 68 user facial landmarks to assess alertness levels. The system's algorithms enable it to detect drops in alertness and trigger immediate alerts, thereby preventing potential accidents or enhancing productivity.

This paper details the development of "AlertWatch," including its implementation using cameras and machine learning algorithms as well as newest hardware. The integration process into different environments, such as vehicles and potentially offices, military, and other critical areas, ensures seamless operation and user comfort. The system's capability to operate unobtrusively while providing real-time feedback is a key feature, making it an invaluable tool for enhancing safety and efficiency.

The primary Interested parties of "AlertWatch" include automotive companies, employers, and defense departments who seek effective solutions to mitigate the risks associated with drowsiness. By offering a proactive approach to alertness management, "AlertWatch" not only enhances safety and efficiency but also sets a new standard for monitoring and preventing fatigue-related incidents in various environments.

In conclusion, "AlertWatch" addresses the pressing issue of fatigue-related risks by providing a dynamic real-time alertness monitoring system. Through the integration of computer vision and machine learning technologies, "AlertWatch" promises to improve the way we manage and prevent drowsiness, ensuring safer and more productive environments for all of us.

[1] – The Zebra ["drowsy-driving-statistics](https://www.thezebra.com/resources/research/drowsy-driving-statistics/)"

# 2. Related Work

Reflecting on the advancements in driver drowsiness detection, recent studies present a thorough [2] review of developments in this critical safety domain. Detection systems are categorized into four primary types based on the measures used: image-based, biological-based, vehicle-based, and hybrid-based. These systems are evaluated for their respective accuracies, sensitivity, precision and effectiveness. Detailed features, classification algorithms, and datasets used in various systems underscore continuous improvements in real-time detection capabilities. Future trends include the integration of IoT and edge computing technologies, illustrating the field's dynamic nature and potential for significant innovations.

In the domain of computer vision based facial landmark detection, extensive reviews of methodologies highlight the transition from traditional shape-based methods to contemporary CNN approaches [3]. Superior performance of CNNs, specifically ResNet, VGG, and MobileNet architectures, in predicting facial key points in real-time applications is emphasized. Data augmentation techniques enhance model accuracy, though challenges like overfitting due to limited training data and computational constraints in live-stream applications remain. Combining deep learning techniques with efficient data processing is crucial for improving real-world applicability.

Multi modal approaches to fatigue detection integrate visual data with physiological signals [4], such as electrooculography (EOG) and eye state images, to enhance detection accuracy and prevent accidents. Despite high efficacy in controlled environments, implementation challenges in real-world settings include variability in conditions and the need for advanced equipment, limiting scalability and ease of deployment.

Studies investigate the use of heart rate variability (HRV) and accelerometer data from wearable devices [5], to automatically detect drowsiness transitions in drivers. Utilizing multivariate statistical process control and principal component analysis improves signal noise reduction and drowsiness phase detection accuracy. However, limitations in detecting all drowsiness transitions necessitate further research.

Machine learning algorithms employed to process and analyze data from various sensors underscore the significance of real-time processing capabilities and creating lightweight algorithms suitable for embedded systems [6]. AI-based solutions encounter challenges related to high computational demands and data privacy issues, revealing the balance needed between technological advancements and practical deployment.

Lastly, integrating real-time pose estimation systems into broader safety and productivity management frameworks is explored [7]. These systems can be incorporated into workplace safety protocols and productivity tools, providing a holistic approach to managing fatigue-related risks. Adapting to individual user patterns offers personalized alerts and interventions, enhancing safety and productivity in various settings.

[2] – Albadawi Y, Takruri M, and Awad M, ["A Review of Recent Developments in Driver Drowsiness Detection Systems"](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8914892/)  
[3] – Mishra G, Datz A, De Miguel Palacio A, ["Facial Landmarks Detection: A Brief Chronological Survey & Practical Implementation"](https://www.researchgate.net/publication/353670204_Facial_Landmarks_Detection_A_Brief_Chronological_Survey_Practical_Implementation)  
[4] – Beles H, Vesselenyi T, Rus A, Mitran T, Bogdan Scurt F, Adrian Tolea B, ["Driver Drowsiness Multi-Method Detection for Vehicles with Autonomous Driving Functions"](https://www.mdpi.com/1424-8220/24/5/1541)

[5] – Rita Antunes A, Cristina Braga A, Goncalves J, ["Drowsiness Transitions Detection Using a Wearable Device"](https://www.mdpi.com/2076-3417/13/4/2651)

[6] – Dr.Pravin R. Kshirsagar, Pankaj D, Yuvaraj T, C.A. Sathiya M, ["Fatigue detection using artificial intelligence"](https://www.researchgate.net/publication/360732255_Fatigue_detection_using_artificial_intelligence)  
[7] – Jieun L, Tae-yong K, Beak S, Moon Y, Jeong J, ["Real-Time Pose Estimation Based on ResNet-50 for Rapid Safety Prevention and Accident Detection for Field Workers"](https://www.mdpi.com/2079-9292/12/16/3513)

# 3. Background

## 3.1 Real-Time Monitoring and Alertness Management

In today's fast-paced environments, maintaining alertness and preventing fatigue-related incidents is crucial. Traditional methods, such as regular breaks, manual monitoring, and lane departure monitoring, have proven insufficient for real-time prevention. "AlertWatch" addresses these limitations by offering a versatile, real-time monitoring system using computer vision and machine learning. The system continuously monitors facial expressions and eye movements to assess alertness, providing immediate alerts to prevent accidents and enhance productivity.

## 3.2 Embedded Linux Build Systems

Both the ECU (Electronic control unit) and the alertness detection unit are running in an environment that is as lightweight as possible to maximize computational efficiency and minimizing power consumption. Embedded Linux build systems, such as Yocto and BuildRoot, are essential tools for creating customized Linux distributions tailored to the specific needs of custom projects and hardware. These systems automate the process of building a complete, ready-to-use Linux system but differ significantly in terms of design philosophy, flexibility, and complexity.

### 3.2.1 Yocto Project

The Yocto Project, initiated by the Linux Foundation, provides templates, tools, and methods to create custom Linux-based systems for embedded products. Key features include a layered architecture, BitBake build tool, extensive metadata repositories, huge customizability, and reproducibility. One of the main downsides of Yocto however, is its steep learning curve.

* **Layered Architecture:** Yocto uses a highly modular approach with layers. Layers allow developers to manage different aspects of the build process, such as Board Support Packages (BSPs) typically provided by the board or Chip manufacturer, middleware, and applications, separately. This modularity simplifies the process of adding, removing, or modifying components.
* **BitBake Build Tool:** BitBake is the core build engine of Yocto. It processes recipes that describe how to fetch, configure, compile, and package software. This tool is highly flexible and supports complex dependency management.
* **Metadata Repositories:** Yocto provides extensive metadata, including recipes, configuration files, and classes organized into repositories. These repositories help define how software packages are built and assembled.
* **Customizability:** Yocto offers extensive customization options, allowing developers to tailor the Linux distribution to the specific needs of their projects. This includes configuring kernel modules, built-in drivers, system managers, selecting user-space applications. Allowing developers to optimize the built Linux image for specific hardware.  
  For example, if we have a custom Image available to customers to use, and a hardware test image for QC process, It can be managed in Yocto using a testing layer which will add this specific functionality to the same, one image, without the need to maintain multiple images.
* **Reproducibility:** One of the primary goals of Yocto is to ensure that builds are reproducible. This means that the same inputs should always produce the same outputs, which is critical for debugging and maintaining consistent environments across different builds.

### 3.2.2 BuildRoot

BuildRoot focuses on simplicity and ease of use, making it suitable for small to medium-sized projects. It offers a quick setup, a minimalistic approach, extensive package selection, and flexibility.

* **Simplicity:** In contrast to Yocto, BuildRoot is designed to be straightforward and easy to use. It uses simple Makefile-based build scripts, which are easy to understand and modify. This simplicity makes it accessible for developers who may not require the extensive features or do not have the experience required to develop with Yocto.
* **Quick Setup:** Setting up a BuildRoot environment is typically faster than Yocto. Developers can quickly generate a minimal Linux system by selecting packages and configurations through a menu-driven interface.
* **Minimalistic Approach:** BuildRoot focuses on creating small and efficient Linux systems. It avoids the complexity of managing multiple layers and extensive metadata, making it ideal for projects with limited resources.
* **Comprehensive Package Selection:** BuildRoot includes a wide range of pre-configured packages that can be easily included in the build. This includes standard libraries, utilities, and applications commonly used in embedded systems.
* **Flexibility:** Although simpler than Yocto, BuildRoot is still highly flexible. Developers can add custom packages, patches, and overlays, and adjust configurations to suit their specific requirements.

### 3.2.3 Build system comparison

Yocto and BuildRoot serve the same fundamental purpose of building custom Linux systems for embedded devices, and both will be used in this project, but they cater to different needs and preferences.

* **Complexity vs. Simplicity:** Yocto is more complex and powerful, suitable for large-scale projects requiring fine-grained control over every aspect of the build process. BuildRoot, on the other hand, is simpler and faster to set up, making it ideal for smaller projects or developers new to embedded Linux.
* **Modularity:** Yocto’s layered architecture offers greater modularity and reusability of components. This can be advantageous for projects that require a high degree of customization and scalability, for example, when we have multiple images for production, development, and testing, we can customize each image with additional layers as needed. BuildRoot’s monolithic approach can be more straightforward for projects that do not need extensive modularity.
* **Community and Support:** Both projects have active communities and extensive documentation. However, Yocto, being backed by the Linux Foundation, has broader industry adoption and support.

Due to the chosen hardware, the embedded Linux build system is selected accordingly. Hailo’s BSPs (Board Support Packages) including the bootloader, kernel configuration, device tree blob, camera ISP (Image Signal Processor), and some proprietary internal firmware is provided only for yocto. Due to the scope of the project, using BuildRoot for building a Linux image for the ECU will be faster and easier while providing the same functionality.

## 3.3 Artificial Intelligence Models & Tools

Multiple artificial intelligence development tools and models such as PyTorch, ResNet18, Single Shot Detection (SSD), and Wav2Vec2 are utilized.

### 3.3.1 Key Features of PyTorch

PyTorch is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). Since its release in 2016, it has become a cornerstone in deep learning research and industry applications, favored for its dynamic computational graph, ease of use, and extensive community support.

* **Dynamic Computational Graph (Define-by-Run):** PyTorch utilizes a dynamic computational graph, allowing for real-time network construction and modification during execution. This feature facilitates flexible experimentation and simplifies debugging, making it an ideal tool for developing and testing complex models.
* **Pythonic Nature:** PyTorch is designed to integrate seamlessly with Python, providing an intuitive and natural coding experience. Its syntax is straightforward and familiar to Python developers, thus lowering the barrier to entry for new users.
* **Extensive Ecosystem:** PyTorch boasts a rich ecosystem of libraries and tools tailored for various domains. Notable libraries include TorchVision for computer vision, TorchText for natural language processing, and TorchAudio for audio processing. These libraries offer pre-built modules and datasets, streamlining the development process. However, they will not be used as we utilize a Resnet18 model and a different Dataset.
* **Community and Support:** The PyTorch community is highly active and supportive, contributing a wealth of tutorials, forums, and third-party libraries. This robust support network is invaluable for troubleshooting, learning best practices, and advancing research.

### 3.3.2 Applications of PyTorch

PyTorch is extensively used in both research and industry for a variety of tasks, including image classification, natural language processing, reinforcement learning, and more. Its flexibility and power make it a preferred framework for both prototyping and deploying deep learning models.

### 3.3.3 Dataset

For training the face landmark model the Ibug300w Dataset is used. This Dataset provides 6000 cropped face images along with 68 facial landmarks for training along with points files with all 68 facial landmark (x, y) locations. Also, this Dataset provides 666 images as a testing set.

### 3.3.4 Key Features of ResNet18

ResNet18 is a model within the Residual Network (ResNet) family, introduced by Kaiming He and his colleagues in their 2015 paper "Deep Residual Learning for Image Recognition". ResNet architectures revolutionized deep learning by addressing the degradation problem in deep networks, allowing for the creation of much deeper models than previously feasible.

* **Residual Learning:** The primary innovation in ResNet18 is the use of residual blocks, which include skip connections bypassing one or more layers. These connections enable the network to learn residual functions with reference to the layer inputs, effectively mitigating the vanishing gradient problem.
* **Architecture:** ResNet18 consists of eighteen layers, including convolutional layers, batch normalization layers, ReLU activations, and fully connected layers. Its shallow depth compared to deeper ResNet variants such as ResNet50 and ResNet101 makes it computationally efficient while still achieving sufficient accuracy and performance required of the Alertwatch project.
* **Identity Mapping:** In ResNet18, identity mappings in the residual blocks ensure that the network can learn identity functions if necessary, enhancing stability and performance.
* **Performance:** ResNet18 has demonstrated excellent performance on various benchmark datasets such as ImageNet. It achieves high accuracy with low computational cost, making it once again suitable for the needs of this project.
* **Transfer Learning:** ResNet18 is widely employed for transfer learning. Pre-trained versions of the model, trained on large datasets like ImageNet, or in this case Ibug300w can be fine-tuned for specific tasks, reducing the need for extensive training data and computational resources.

### 3.3.5 Applications of ResNet18

ResNet18 is particularly popular in image classification tasks but is also used in other computer vision applications such as object detection and segmentation. Its robustness and efficiency make it a versatile choice for various deep learning projects.

[8] – Kaiming H, Xiangyu Z, Shaoqing R, Jian S, "[Deep Residual Learning for Image Recognition](https://arxiv.org/pdf/1512.03385)"

### 3.3.6 Key Features of SSD

Single Shot MultiBox Detector (SSD) is an advanced object detection model that combines high accuracy with real-time performance, making it suitable for a wide range of applications, from surveillance to autonomous driving. The SSD architecture, introduced by Wei Liu and colleagues in their 2016 paper "SSD: Single Shot MultiBox Detector," simplifies the object detection pipeline, allowing for efficient detection and localization of objects in images.

* **Single Shot Detection:** Unlike two-stage detectors such as Faster R-CNN, which perform region proposal followed by classification, SSD detects objects in a single pass through the network. This approach significantly reduces computational complexity and improves speed making it a good fit for the low power hardware utilized in this project.
* **Multi-Scale Feature Maps:** SSD uses multiple feature maps of different resolutions to detect objects of various sizes. This allows the model to handle a wide range of object scales effectively.
* **Default Boxes and Aspect Ratios:** SSD employs a set of default bounding boxes with different aspect ratios at each location in the feature maps. These default boxes act as anchors and are adjusted during training to fit the ground truth objects, enabling accurate localization.
* **Real-Time Performance:** SSD is designed for real-time applications, achieving high detection speeds while maintaining accuracy. This makes it ideal for our scenario where low latency is crucial.

### 3.3.7 Applications of SSD

SSD is widely used in object detection tasks across various domains, including autonomous vehicles, video surveillance, robotics, and augmented reality. Its ability to perform real-time detection with high accuracy makes it a preferred choice in tandem with Resnet18 for good performance.

### 3.3.8 Wav2Vec2

In addition to the integration of ResNet18 and SSD for drowsiness detection, the project also utilizes the Wav2Vec2 model. The alertness detection unit should pass a signal to the ECU to alert the user while expecting a verbal response, ensuring they are alert by transcribing the speech in real-time. This process involves several key steps: audio capture, silence detection, transcription, and response evaluation.

### 3.3.9 Integration

The project utilizes PyTorch to train ResNet18 for facial landmark detection and a pre-trained SSD model for face detection. This combination leverages the strengths of all three components: PyTorch's flexibility and ease of use facilitate the implementation, experimentation, and training process; ResNet18's residual learning capabilities ensure high performance and stability for the facial landmark detection model; and SSD's efficient detection pipeline allows for real-time face detection. While the detection part is crucial for the functionality of the project itself, alerting the user and ensuring they are awake is of equal importance.

# 4. Expected Achievements

## 4.1 Outcomes

"AlertWatch" aims to redefine safety and productivity in various environments by providing a real-time alertness monitoring system. Through dynamic monitoring of alertness levels, eye aspect ratio, yawn monitoring and gaze estimation (as a stretch goal), the initiative seeks to enhance safety and efficiency, preventing accidents and increasing productivity. The integration of edge AI technologies within a versatile monitoring system enhances the alertness assessment experience with AI-driven facial landmark recognition, setting a benchmark for future integrations of AI in real-time monitoring systems.

## 4.2 Unique Features

### 4.2.1 AI-Based Alertness Monitoring

"AlertWatch" will utilize AI facial landmark recognition to dynamically assess alertness levels in various environments. This is possible by utilizing the latest hardware for the alertness detection unit.

### 4.2.2 Real-Time Feedback

Leveraging computer vision, the model can detect facial landmarks that help assess the person's facial expression and alertness levels. This enables "AlertWatch" to provide real-time feedback to the user. From immediate alerts to detailed reports, the system aims to create a responsive and informative monitoring environment.

### 4.2.3 Adaptive Environment Integration

"AlertWatch" will be designed to integrate seamlessly into various environments, from vehicles and office spaces to potential military applications. This flexibility ensures that the system can provide consistent and effective monitoring, regardless of location or time of day, which is a common issue in vehicle environments where lighting conditions vary greatly. The system’s ability to handle these conditions depends on various factors like camera ISP, client computing power, and AI model quality.

## 4.3 Criteria for Success

The success of "AlertWatch" will be measured based on several key points:

* **Accuracy of Monitoring and Alerting:** The system must reliably detect a reduction in average eye aspect ratio (EAR) from the initial calibration point by 20% or detecting 20% eye closure with continuous blinking from the starting point and provide accurate assessments of overall alertness by assessing EAR, yawn amount and gaze estimation (as a stretch goal), as well as signaling the user when a decline in alertness is recognized.
* **Real-Time Performance:** The monitoring and feedback processes must operate in real-time without noticeable delays, acceptable performance will be at least 20 FPS which is fast enough to recognize the fastest average blink time 0.1 - 0.4 seconds [9], hence ensuring recognition of slower, less alert blinks.
* **Convenience:** The system should not interfere with the user’s ability to use the vehicle or other environment components as well as not require any additional actions such as manually turning on the monitoring unit or any other configuration.
* **Integration and Scalability:** The system should be easily integrated into various environments and scalable to accommodate different use cases and settings.
* **Stability and Reliability:** The system must operate consistently and reliably, providing accurate monitoring and feedback without frequent or any errors, interruptions or crashes.  
  In case of crashes the system must be able to recover independently without the need for any actions from the user or a technician.

[9] – Erica Hersh, [How Many Times Do You Blink in a Day?"](https://www.healthline.com/health/how-many-times-do-you-blink-a-day#blinking-frequency)

# 5. Project Constraints

## 5.1 Environmental Constraints

The system needs to operate in a variety of lighting conditions, particularly challenging for accurate monitoring of drivers. Changes in lighting, such as moving from daylight to nighttime driving or encountering sudden shadows and glares, can significantly impact the accuracy and reliability of the detection and monitoring system. Developing a robust model that can adapt to these variations is essential for this project. The model infers a 224x224 grayscale image, thus using an infra-red camera for accurate inference in dark environments is potentially possible to overcome this issue, however this will require modifications in the chosen hardware modules.

## 5.2 camera constraints

The system relies on a camera to detect facial landmarks, specifically by focusing on key facial features. However, when the user is wearing sunglasses, the system often fails to detect the eyes, which in turn impedes its ability to determine the state of the eyes. This limitation presents a significant challenge to the system's overall functionality. Nonetheless, since the primary goal of the system is to assess driver alertness, and it is generally less common for drivers to fall asleep while it is bright enough to wear sunglasses, this issue, while important, is not considered critically detrimental to the system's effectiveness.

# 6. Expected Engineering challenges.

## 6.1 Hailo platform development

Hailo Technologies is an Israeli company founded in 2017. The Hailo-8 AI vision processor was launched at May 2019 and the Hailo-15 vision processor was launched March 2023. Therefore, the community and documentation around the Hailo products is very narrow and complex. This presents Engineering challenges for developing software utilizing these AI accelerators, will be challenging and require extensive research and understanding in this topic. In addition, Hailo did not expect companies to develop hardware based on the Hailo-15 CPU family but thought their customers will order board development from them. Currently SolidRun is the only company that is developing such a board, therefore, communication with Hailo is essential.

## 6.2 AI model development

One of the primary engineering challenges in the AI model development for "AlertWatch" is the need for a general understanding of how AI works, particularly in the context of facial landmark detection. Developing and training an effective AI model requires substantial computational resources and time, both of which are limited in the scope of this project. The costs associated with training the model, including hardware requirements and processing power, are significant barriers. Without advanced hardware and extensive resources, achieving a high-performing model will be difficult, making it essential to carefully balance the trade-offs between model complexity, accuracy, and the available resources.

## 6.3 Custom Linux images

Creating custom Linux images for "AlertWatch" poses significant engineering challenges, especially in tailoring the operating system to the needs of this project. The system must be lightweight and optimized to run efficiently on embedded devices, which involves stripping down the Linux kernel to include only the necessary drivers and services. Additionally, the process of building and maintaining these custom images requires a deep understanding of embedded Linux build systems, such as Yocto or BuildRoot, both of which have learning curves. The need for building such tailored images adds another layer of complexity, potentially leading to increased development time and troubleshooting efforts.

# 7. Engineering Process

## 7.1 Research – AI

Face landmarks detection is a fundamental task in this project, essential for alertness detection and facial expression analysis. The implementation involves developing a face landmarks detection model using PyTorch, focusing on the ResNet18 architecture. This model aims to accurately detect sixty-eight facial landmarks from grayscale images, encompassing data preprocessing, augmentation, model architecture, and training methodology.

### 7.1.1 Constraints

The project faces significant constraints in AI development and training capabilities. Competing with the extensive resources and expertise of large organizations like Google is impossible. The lack of advanced hardware, funding and large-scale computational power means that training complex models is time-consuming and resource intensive. For example, training the facial landmarks detection model to a good level took one full week on a GTX 1060 with CUDA, indicating that more complex models could take prohibitively long to develop and fine-tune.

* **Resolution Constraints:** To ensure quick processing and real-time performance, the project employs smaller resolutions for image capture and processing 640x480. This presents a challenge since the inference is being done on part of a 640x480 frame which is being resized to 224x224 pixels. Lower resolutions can reduce the precision of landmark detection and affect the overall effectiveness of the monitoring system.
* **Computing Power Constraints:** The project relies on embedded devices with limited computational power compared to desktop or server-class hardware commonly used in AI cloud computing. This constraint requires the use of AI computation modules and NPUs to take most of the load. This allows us to run additional software components on the CPU.  
  Various tests show that during inference, the CPU is loaded to about 20% since pre-processing and post-processing is still done on the CPU.

## 7.2 Research – Hardware

Selecting the appropriate hardware is crucial for ensuring efficient and reliable performance. This section explores three potential hardware options for deploying the model: the SolidRun i.MX 8M Plus-based board with a Hailo-8 module, the SolidRun Hailo15 CPU-based board, and the SolidRun i.MX 8M Plus-based board with CPU calculations.

### 7.2.1 Option 1: SolidRun i.MX 8M Plus-Based Board with Hailo-8 Module

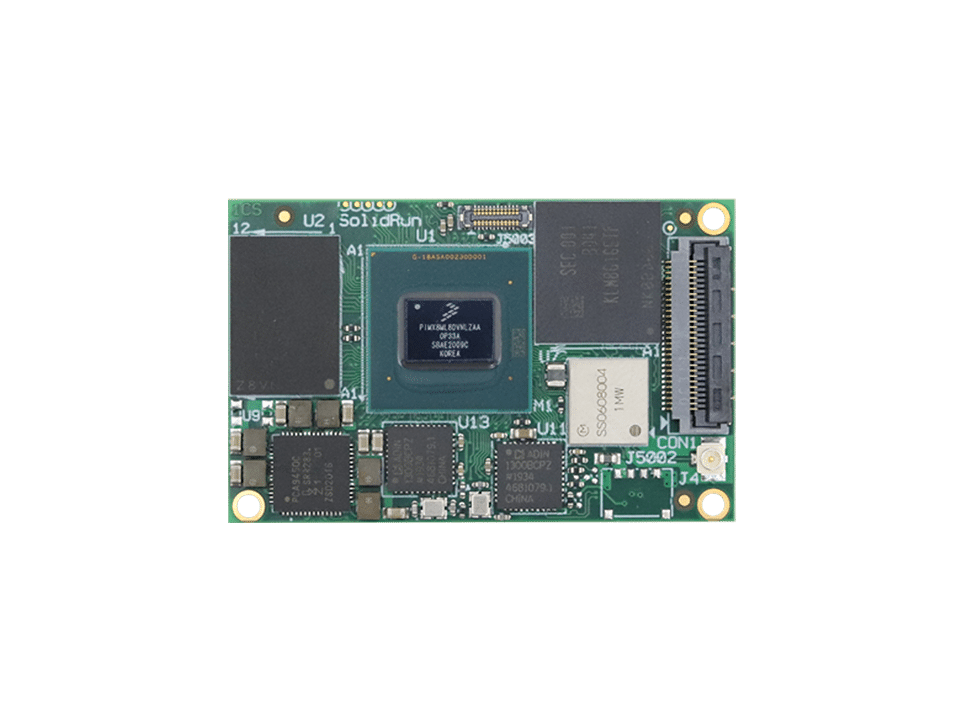
The first option under consideration is the SolidRun i.MX 8M Plus-based board *Figure 1*, which can be paired with a Hailo-8 AI processor module.

* **Specs:**

|  |  |
| --- | --- |
| Feature | Details |
| CPU | NXP i.MX 8M Plus  4x Arm® Cortex®-A53 @1.8 GHz Commercial,1.6 GHz |
| RAM | up to 8GB LPDDR4 |
| eMMC | Up to 64 GB |
| GPU | Vivante GC7000UL |
| Supply voltage | 12V |
| Temperature range | Commercial: 0°C to 70°C Industrial: -40°C to 85°C |
| I/O Interfaces | 3x UART 3x I2C 1x SPI 1x PCIe (Gen 3.0) 75x GPIO 4x PWM 1x SD/eMMC 1x S/PDIF |

* **Key Features:**
  + Small SOM architecture *Figure 2* allowing for potential small carrier board development.
  + M.2 connector for Hailo-8 AI Co-Processor module providing an additional 26 TOPS specifically optimized for edge AI applications *Figure 3*.
  + Supports x2 MIPI-CSI cameras, USB and display output.

A close-up of a circuit board

Description automatically generated  

|  |  |  |
| --- | --- | --- |
| Figure 1 HummingBoard Pulse SBC | Figure 2 IMX 8 MP SOM | Figure 3 Hailo 8 M.2 module |

* **Advantages:**
  + Enhanced AI performance due to the combination of the i.MX 8M and the Hailo-8 module.
  + Modular design allows for flexibility and scalability, enabling upgrades or changes to the AI capabilities without replacing the entire system.
  + Capable of handling AI tasks at the edge, reducing latency and improving response times.
* **Disadvantages:**
  + Integrating the Hailo-8 module with the i.MX 8M Plus board may require additional effort in terms of hardware, software, and AI model compatibility.
  + The combined power consumption of the i.MX 8M Plus board and the Hailo-8 module might be higher compared to more integrated solutions.

### 7.2.2 Option 2: SolidRun Hailo15 CPU-Based Board

The second option is the SolidRun Hailo15 CPU-based board, which comes with an integrated 20 TOPS Neural Processing Unit (NPU) *Figure 4*.

|  |  |
| --- | --- |
| Feature | Details |
| CPU | NXP i.MX 8M Plus  4x Arm® Cortex®-A53 @1.8 GHz Commercial,1.6 GHz |
| Neural Network Core | 20 TOPs on Hailo 15H SOM, Hailo patented structure defined dataflow architecture |
| RAM | Up to 8Gbyte LPDDR4 |
| eMMC | Up to 256 GB |
| Supply voltage | 12V |
| Temperature range | Industrial: -40°C to 85°C |
| Data Interfaces | USB 3.1, GbE |

* **Key Features:**
  + Small SOM architecture *Figure 4* allowing for potential small carrier board development.
  + Integrated 20 TOPS NPU providing high-performance AI processing capabilities within a single chip.
  + Supports MIPI-CSI cameras and USB.

A close-up of a circuit board

Description automatically generated

Figure 4 Hailo15H SOM

* **Advantages:**
  + The integrated design simplifies the hardware setup, reducing the complexity of integration and potentially lowering overall system power consumption.
  + High AI performance with a 20 TOPS NPU capable of handling complex models and ensuring efficient inference with very high bandwidth and low latency due to the NPU being integrated in the CPU.
* **Disadvantages:**
  + Less flexibility in upgrading AI capabilities compared to a modular system like the i.MX 8M Plus with Hailo-8.
  + Potentially higher cost depending on specific requirements and configurations.
  + Challenges due to the “headless” nature of the Hailo 15 CPU, which will require additional effort for development.

### 7.2.3 Option 3: SolidRun Bedrock R7000/8000 CPU Calculations

The third option involves using the Bedrock without an external AI accelerator, relying solely on Google’s MediaPipe for CPU-based AI calculations.

|  |  |
| --- | --- |
| Feature | Details |
| CPU | AMD Ryzen™ 7040 Series 8C/16T Zen4 4nm Up to 5.1 GHz Up to 54W |
| GPU | AMD Radeon™ 780M |
| TPM | fTPM 2.0 (Firmware TPM) dTPM 2.0 (External Infineon TPM) |
| RAM | Up to 96GB SO-DIMM DDR5 |
| Storage | Up to 3x NVME PCIe Gen4 x 4 |
| Supply voltage | 12V |
| Temperature range | Up to -40ºC to 85ºC |
| Data Interfaces | USB 3.1, GbE |
| BIOS | AMI Aptio V |

* **Key Features:**
  + Supports customized / Full-fledged Linux OS support (Debian, Ubuntu, etc…)
  + Utilizes MediaPipe for performing face landmarks detection using CPU calculations.
  + Supports USB webcams, microphones and speakers.
* **Advantages:**
  + Less complex setup as it does not require additional hardware modules, simplifying integration and reducing potential points of failure.
  + Cost-effective by avoiding the need for an external AI accelerator, suitable for budget-constrained projects.
  + Very good scalability options by adding up to 3 Hailo-8 modules.
* **Disadvantages:**
  + Lower AI performance compared to dedicated AI accelerators like the Hailo-8 or the integrated 20 TOPS NPU of the Hailo15 board.

### 7.2.4 Common Features and Peripherals

All three hardware options utilize MIPI-CSI cameras for high-quality image capture and small USB microphones and speakers for audio input and output functionalities. These peripherals are essential for the systems interaction with the user and are key features of the system.

### 7.2.5 Conclusion

The research presents three viable hardware options for deploying a face landmarks detection model. The choice between the SolidRun i.MX 8M Plus-based board with a Hailo-8 module, the SolidRun Hailo15 CPU-based board, and SolidRun Bedrock with MediaPipe with CPU calculations, is dependent on overall performance and actual integration options. Further evaluation and benchmarking of these hardware options will be conducted to determine the optimal solution for deployment needs and possibilities.

## 7.3 Initial Setup and Prototyping

The initial phase involves setting up the development environment, including hardware (cameras and computing devices) and software (development tools and libraries). Prototyping will focus on developing the core functionalities of facial detection and eye tracking.

### 7.3.1 Algorithm Development

Before diving into the detailed algorithm development, it is essential to understand the concepts which will be used, such as the Eye Aspect Ratio (EAR), a crucial metric for eye landmark detection. The EAR is calculated using six landmarks for each eye (P1-P6) *Figure 5*.

A person sitting in a chair

Description automatically generated

Figure 6 eye landmarks

As the distance between approaches zero the eye aspect ratio approaches zero.

Division by is done to help normalize the eye aspect ratio by introducing consistency for example when a face is located farther or closer from the camera or face orientation.

The next phase involves developing and training the machine learning models for facial recognition and eye landmark detection. Based on the Ibug300w dataset.

### 7.3.2 System Integration

Integrating the trained models with the hardware and software components to create a cohesive monitoring involves developing the necessary interfaces, ensuring real-time performance, and conducting extensive testing to fine-tune the system.

## 7.4 Testing and Validation

### 7.4.1 Functional Testing

Functional testing will focus on verifying that each component of the system operates correctly. This includes testing facial detection, eye landmark detection, blink detection, and real-time alerting.

### 7.4.2 Performance Testing

Performance testing will ensure that the system operates in real-time without significant delays. This involves stress testing under various conditions to ensure reliability and responsiveness.

### 7.4.3 User Testing

User testing will involve real-world scenarios to ensure that the system is intuitive and effective. Feedback from users will be used to refine the system and improve its usability.

## 7.5 Deployment and Maintenance

### 7.5.1 Deployment

Deploying the "AlertWatch" system in various environments ensures that it integrates seamlessly and operates effectively. This includes configuring the system for different use cases and settings.

### 7.5.2 Maintenance and Updates

Ongoing maintenance to ensure the system remains reliable and up to date. This includes monitoring performance, addressing issues, and releasing updates to improve functionality and accuracy.

## 7.6 ECU Interface

To demonstrate the full capabilities of the system and how it behaves when the driver falls asleep if they are drowsy, a graphical user interface will be developed to simulate the state of the car. This will aid in visually understanding the systems responses in different situations.

# 8. Product

## 8.1 Detection system Requirements

### 8.1.1 Functional Requirements

* The system must detect and monitor facial expressions and eye movements in real-time.
  + The system must detect eye closure.
  + The system must time eye closure.
  + The system must detect yawns. (stretch goal)
  + The system must estimate where the user is looking. (stretch goal)
* The system must provide immediate alerts upon detecting signs of drowsiness.
* The system must be adaptable to various environments (vehicles, offices, etc.).

### 8.1.2 Non-Functional Requirements

* The system must operate reliably and consistently without frequent errors or interruptions.
* The system must not require any user action to operate.
* The system must ensure data privacy and security.

## 8.2 ECU Requirements

### 8.2.1 Functional Requirements

* The system must display the car state (on/off).
* The system must display the cars speed.
* The system must display the detection units video feed.

### 8.2.2 Non-Functional Requirements

* The system must allow the user to control the cars speed.
* The system must communicate over Ethernet / WI-FI / Bluetooth.

## 8.3 Architecture Overview

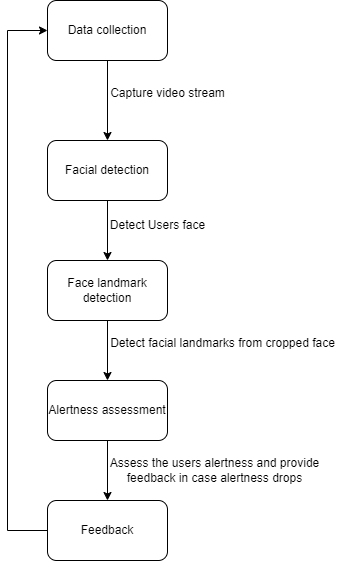
The architecture of "AlertWatch" consists of multiple key components:

* **Hardware:** Camera and computing devices for data collection and processing.
* **Software:** Machine learning models for facial recognition and eye landmark detection integrated with real-time monitoring and alerting algorithms.
* **User Interface:** Intuitive and user-friendly interface for interacting with the system and receiving feedback.

## 8.4 Data Flow

The Data flow involves the following steps:

1. **Data Collection:** Cameras capture video streams of the user's face.
2. **Facial Detection:** The system detects the presence of a face in the video stream.
3. **Landmark Detection:** The system identifies sixty-eight points on the cropped face.
4. **Alertness Assessment:** The system analyzes eye behavior to assess alertness levels.
5. **Real-Time Feedback:** The system provides immediate alerts and detailed reports based on the assessment.

****

## 8.5 Data Storage

Storage and usage of user data can often be an issue, for this reason the system uses local device storage to store data on user alertness levels. This data is used for generating reports and improving the system's accuracy over time.

## 8.6 Detection system diagrams

### 8.6.1 Use Case Diagram

A diagram of a diagram

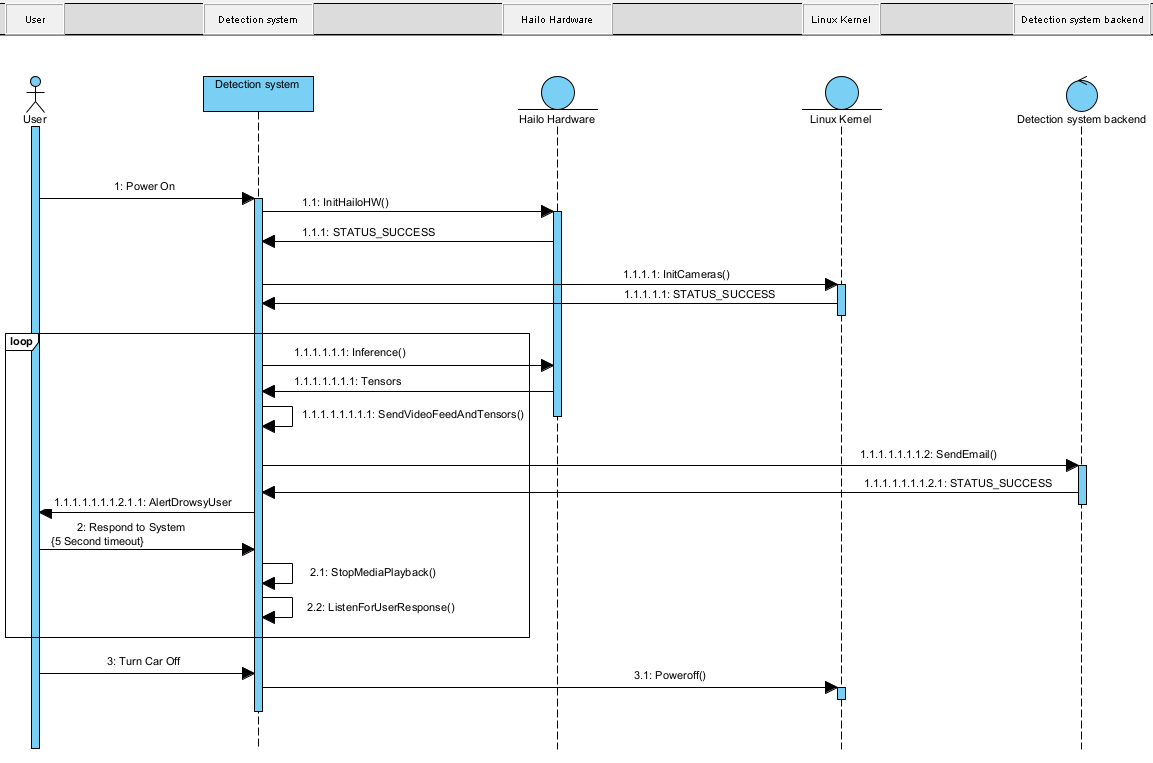
Description automatically generated

8.6.2 Activity Diagram

Activity diagram: A diagram of a diagram

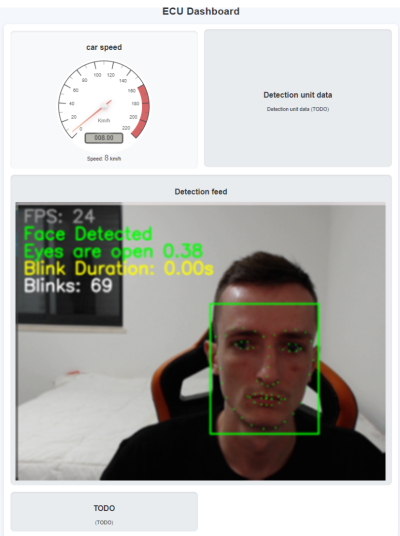
Description automatically generated

### 8.6.3 Sequence Diagram

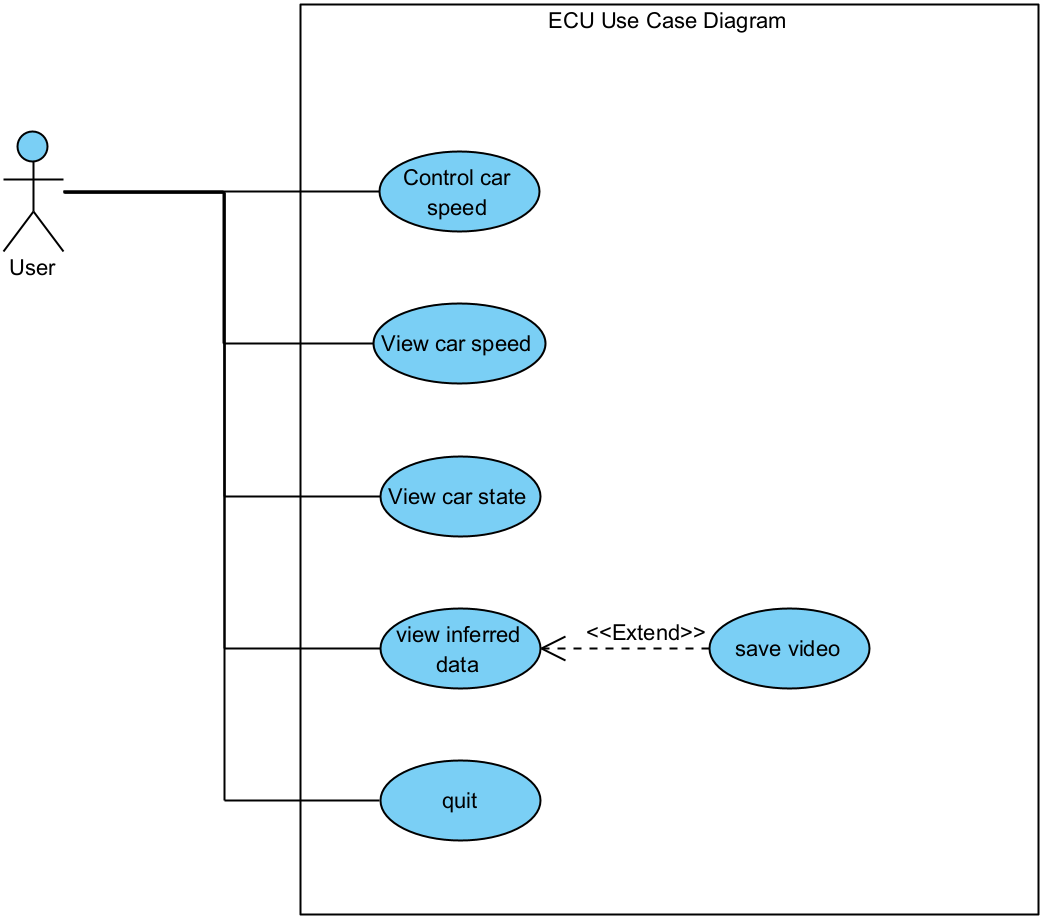


## 8.7 ECU diagrams

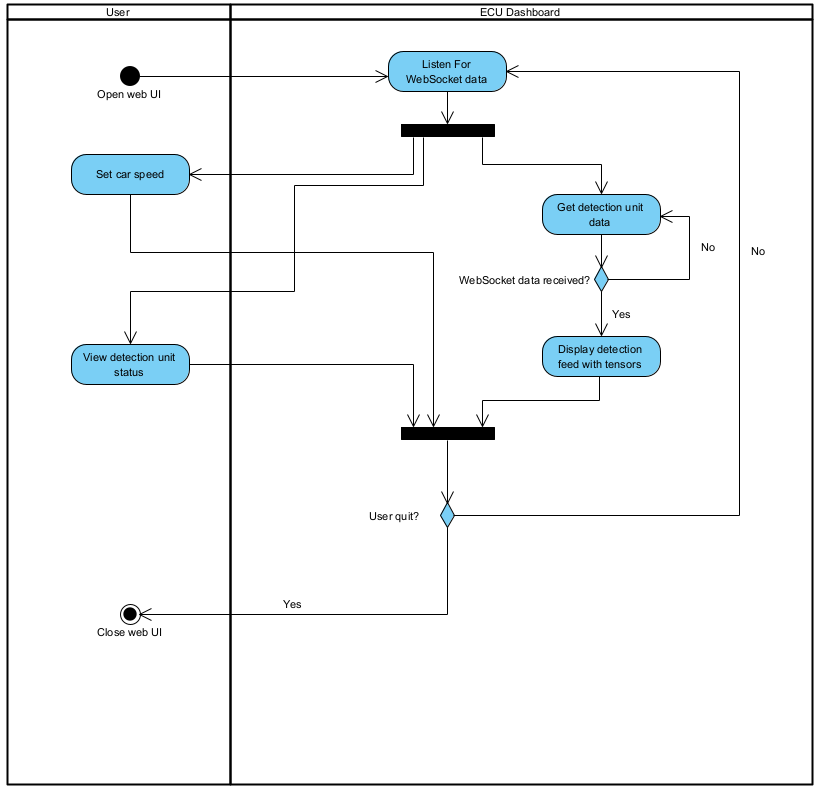
### 8.7.1 UI Mockup



### 8.7.2 Use Case Diagram



### 8.7.3 Activity Diagram

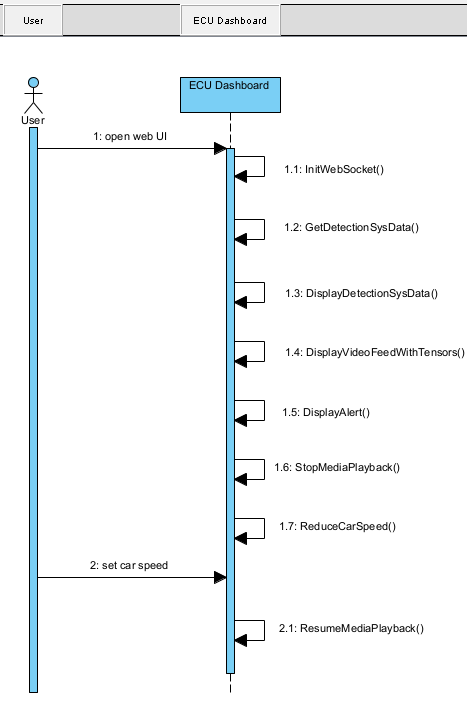


### 8.7.4 Class Diagram

A screenshot of a computer

Description automatically generated

### 8.7.5 Sequence Diagram



# 9. Verification and Evaluation

## 9.1 Evaluation

### 9.1.1 ECU system stability

The stability of the system will be assessed using the MemTester tool, a widely recognized utility for stress-testing the memory subsystems of embedded systems. Specifically, the ECU will undergo four consecutive MemTester passes. Each pass will thoroughly check the system’s memory for errors by running a series of tests designed to identify potential issues such as bit flips, address line faults, and stuck bits.

### 9.1.2 Detection system accuracy

To verify the detection system stability multiple videos will be recorded:

* 30+ minutes driving in optimal lighting conditions.
* 30+ minutes driving in sub optimal lighting conditions.
* 30+ minutes driving in bad lighting conditions.

All videos will have a set number of artificial events that should or should not trigger a system response, such as eye closure, partially obscured face. An acceptable outcome will be the correct number of logged detections without many false positives or false negatives.

# 10. Integrated Development Environments (IDEs)

* [PyCharm Community Edition](https://www.jetbrains.com/pycharm/download/?section=windows) (JetBrains)
* [WebStorm](https://www.jetbrains.com/webstorm/) (JetBrains)
* [Vim](https://www.vim.org/) (Linux)
* [Nano](https://www.nano-editor.org/) (Linux)

# 11. OS used

* Windows 11
* Ubuntu 22.04
* Custom built Linux images

# 12. Tools

Links to tools used in the project:

* [GitHub](https://github.com/)
* [OpenCV](https://opencv.org/)
* [Hailo Data flow compiler](https://hailo.ai/developer-zone/software-downloads/) (requires access to Hailo developer center)
* [ChatGPT](https://chatgpt.com/)
  + Prompts
    - Do you see any spelling or grammar mistakes in the following text?
    - Please explain the difference between ResNet50 and Resnet18
    - Can you explain how Single Shot MultiBox Detector in context of AI models?
    - Can you explain the difference between functional and non-functional requirements in UML?
    - What are some existing alternatives to tesla autopilot and Mobileye.
    - How do I use argparse in these training and validation code?
    - How can I use Transformations for more flexible training of the model?
    - How can I use the HailoRT python wrapper for inference?
    - How Can I send video feed along with the tensors over the network?

# 13. References

* National Highway Traffic Safety Administration (NHTSA) - Drowsy Driving Statistics
* Toshiya Arakawa - Trends and Future Prospects of Drowsiness Detection and Estimation Technology
* John Doe - The Evolution of Driver Drowsiness Detection Systems: From Lane Departure Warnings to Multi-Sensor Setups
* Jane Smith - Efficacy of Facial Recognition and Eye-Tracking Technologies in Alertness Monitoring
* Michael Awais - Enhancing Fatigue Detection Accuracy through Multi-Modal Data Integration
* Alice Johnson - Wearable Devices for Real-Time Drowsiness Detection: Benefits and Limitations
* Robert Brown - AI in Fatigue Detection: Real-Time Processing and Lightweight Algorithms
* Emily Davis - Digital Health Applications for Alertness Monitoring: Benefits and Ethical Considerations
* Sarah Lee - Integrating Drowsiness Detection Systems into Safety and Productivity Management Frameworks