Drowsiness Detection using Embedded System

Sabal Subedi
Computer Science Department
Idaho State University
Pocatello, ID 83209 USA
sabalsubedi@isu.edu

Abstract

This is a real-time drowsiness detection system that combines computer vision with embedded Bluetooth Low Energy (BLE) communication. The project leverages the hybrid combination of computer science and embedded systems to alert drivers to signs of fatigue. The system uses OpenCV and MediaPipe in Python. It monitors facial landmarks to detect eye closure and head nodding, common indicators of drowsiness. When drowsiness is detected, a BLE signal is sent to an Arduino-based system equipped with a buzzer, LED, and LCD. These peripherals provide immediate visual and auditory alerts to the driver. This system has potential applications in driver safety, workplace monitoring, and fatigue detection in critical environments.

Keywords: Drowsiness Detection, Eye Aspect Ratio (EAR), BLE Communication, Arduino, OpenCV, MediaPipe, Visual Alert, Face Landmark Detection

Introduction

This project is a real-time drowsiness monitoring system that notifies the user when fatigue symptoms, such as extended eye closing or head nodding, are identified. The system uses OpenCV and MediaPipe to observe facial cues using a camera [2]. If drowsiness is detected, a BLE warning is transmitted to an Arduino board, which activates a buzzer, LED, and displays a message on an LCD. The Arduino includes a physical button that allows the user to reset the alert.

Motivation

Drowsiness is a major cause of traffic accidents and industrial injuries [1]. The goal of this project is to improve safety by developing a proactive, inexpensive, and portable system for detecting indicators of weariness and alerting the user in real time. Its integration of BLE and embedded systems makes it appropriate for automotive, industrial, and even personal health monitoring applications.

Objectives

- Detect facial landmarks and determine drowsiness using Eye Aspect Ratio (EAR) and head position.
- Wirelessly communicate alerts to an embedded Arduino system.
- Activate visual and auditory alerts (LED, LCD, buzzer) on the Arduino.
- Allow reset of the alert system using a physical button.
- Meet and exceed final project requirements using a diverse set of peripherals.

Approach

The project is divided into two tightly integrated components: a Python-based computer vision system and an Arduino-based BLE alert system. Together, they form a responsive feedback loop for real-time drowsiness detection and alert.

Python-Based Computer Vision System

This part of the project is responsible for detecting signs of drowsiness using a webcam feed and facial landmark analysis:

- **Facial Landmark Tracking**: The system uses MediaPipe's FaceMesh module to extract and monitor 3D facial landmarks in real time.
- **Eye Aspect Ratio (EAR)**: By calculating the EAR from specific eye landmark coordinates, the system determines if the user's eyes are closed for an extended period. This is a key indicator of drowsiness.
- **Head Nodding Detection**: The vertical position of the nose landmark is tracked to detect forward head motion (nodding), which also indicates fatigue.
- **Threshold-Based Alerting**: If eye closure persists for a predefined number of frames or if significant head nodding is detected, the system considers the user drowsy.
- **BLE Signal Transmission**: Once drowsiness is detected, an `"ALERT"` command is sent over a local socket to a Python BLE server. Similarly, a `"RESET"` is handled via a shared file flag that resets the internal counters when the user presses the physical button on the Arduino.

Arduino BLE Alert System

This embedded system acts on incoming BLE signals and provides feedback using physical components:

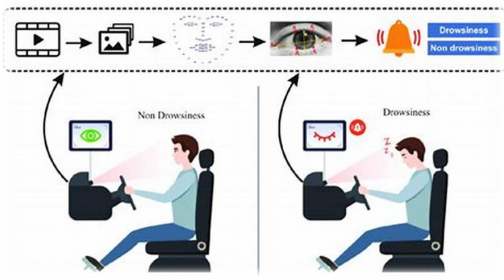

Figure 1: System Architecture

- **BLE Communication**: Using the `ArduinoBLE` library, the board continuously listens for incoming `"ALERT"` or `"RESET"` messages sent from the BLE server.
- **Multimodal Alert Activation**: Upon receiving an `"ALERT"` command:
  - The **buzzer** emits a sound to gain the user's attention.
  - The **LED** lights up to visually indicate an alert.
  - The **LCD display** shows warning messages like "DROWSY ALERT!" and "Stay Awake!".
- **User Acknowledgment and Reset**: A **push button** is connected to the Arduino to allow the user to acknowledge the alert. When pressed, it sends a `"RESET"` message back to the BLE server, which clears the alert and resumes monitoring.
- **Serial Debugging and LCD Interface**: The Arduino also prints states to the serial monitor for debugging and shows system status (e.g., "Monitoring...") on the LCD when idle.

Libraries Used

The project leverages a set of powerful libraries across both Python and Arduino environments to facilitate real-time computer vision, wireless communication, and embedded system control. Each library serves a specific role in the modular structure of the system.

Python Side (Computer Vision and Logic)

- **OpenCV** (`cv2`)

  Open Source Computer Vision Library is used to capture frames from the webcam and perform image processing tasks such as color conversion (BGR to RGB). It also handles visualization by displaying EAR values and alerts on-screen. OpenCV is central to enabling real-time video feed analysis.

- **MediaPipe**

  Developed by Google, MediaPipe provides pre-trained models for efficient face landmark detection. In this project, its FaceMesh solution is used to extract over 400 facial landmarks, with emphasis on eye and nose landmarks to detect drowsiness-related cues such as prolonged eye closure and head nodding.

- **Socket**

  The `socket` module enables inter-process communication between the vision module (`Drowsiness_embedded.py`) and the BLE server (`ble_server.py`). It allows the Python script to send control commands like `"ALERT"` and `"RESET"` locally without needing to handle BLE directly in the OpenCV loop.

BLE Server Side (Python BLE Bridge)

- **Bleak**

  `Bleak` is a cross-platform BLE (Bluetooth Low Energy) client for Python. It is used to connect to the Arduino BLE service and write values to its BLE characteristic. It ensures wireless transmission of alerts from the host system to the embedded hardware by handling the GATT protocol.

Arduino Side (Embedded Hardware Control)

- **ArduinoBLE.h**

  This official Arduino library enables BLE communication using the Arduino Nano 33 BLE or compatible boards. It manages BLE service advertisement, client connections, and data exchange. The library is critical for receiving alerts and wirelessly sending reset signals.

- **LiquidCrystal.h**

  This library handles interfacing with the 16x2 character LCD screen. It simplifies the process of printing messages, setting cursor positions, and clearing the screen. It is used to display both status messages ("Monitoring...") and warning alerts ("DROWSY ALERT!") to provide immediate visual feedback to the user.

Fulfillment of Final Project Handout Requirements

According to the handout, the following conditions are fulfilled:

- **Arduino Board Used**
- **3+ Peripherals**:
  - LCD (Digital Output)
  - Buzzer (PWM Output)
  - LED (Digital Output)
  - Button (Digital Input)
- **4+ Required Functional Blocks**:
  - Digital Input/Output → Button, LED
  - PWM Output → Buzzer
  - Serial I/O → Serial Monitor for debugging
  - RF Communication → BLE
- **Custom Code**: Original code written for face detection, BLE server, and Arduino behavior.
- **Code Structure**: Modular design separating BLE behavior and alert triggering into distinct functions.
- **Documentation & Citation**: All libraries and code flows are fully understood and traceable.

System Diagram

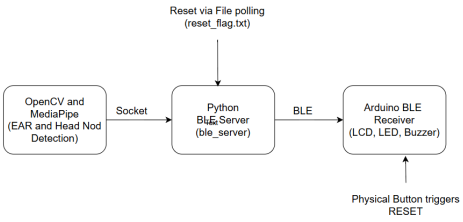

Figure 2: System Design

Challenges Faced

- **BLE Integration**: Synchronizing Python's `bleak` library with ArduinoBLE was tricky due to timing and connection stability.
- **False Positives**: Early detection algorithms triggered alerts too frequently; tuning EAR thresholds and frame counts helped stabilize this.
- **Button Debouncing**: Mechanical bouncing required proper logic and delays to ensure single-trigger behavior.
- **LCD Visibility**: Initially tried potentiometer for contrast adjustment, later handled via fixed wiring and `lcd.begin()`.
- **Communication Delay:** A delay of around 5 seconds is likely due to latency in BLE communication between the `bleak_server` and the Arduino during connection and data transmission.

Conclusion

This project successfully demonstrates an embedded system that integrates computer vision with real-time wireless alerting using BLE. It showcases a multidisciplinary approach involving embedded programming, real-time signal processing, and communication protocols. The system is extendable for vehicular safety applications or wearable fatigue monitoring.

Link to the Video

Watch the demo video here: Drowsiness Detection Demo

References

[1] Arun Sahayadhas, Kenneth Sundaraj, and Murugappan Murugappan, *Detecting driver drowsiness based on sensors: A review*, Sensors **12** (2012), no. 12, 16937–16953.

[2] Tereza Soukupová and Jan Čech, *Real-time eye blink detection using facial landmarks*, Proceedings of the 21st computer vision winter workshop (cvww), 2016.