T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

DRIVER DROWSINESS DETECTION SYSTEM

YAKUP TALHA YOLCU

SUPERVISOR
BURCU YILMAZ

GEBZE
2023

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

DRIVER DROWSINESS DETECTION SYSTEM

YAKUP TALHA YOLCU

SUPERVISOR
BURCU YILMAZ

2023
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 15/01/2023 by the following jury.

JURY

Member
(Supervisor) : Burcu Yılmaz

Member : Habil Kalkan

ABSTRACT

Driver Drowsiness Detection System is designed to identify when a driver is at risk of falling asleep at the wheel and alert them to wake up and take a break. This system uses image processing and machine learning techniques. This system utilizes data from sensors placed in the vehicle, such as a camera and sound alarm to generate an alert when drowsiness is detected. Drowsiness detection are checked with 3 ways : eyes closed or not, head is towards right or left, head is down or not.

Keywords: driver, drowsiness, camera, alarm.

ÖZET

Sürücü Uyku Durumu Tespit Sistemi, bir sürücünün direksiyondayken uykuya dalma riski taşıdığını tespit etmek ve uyandırıp mola vermesine yardımcı olmak için tasarlandı. Bu sistemde, görüntü işleme ve makine öğrenimi teknikleri kullanıldı. Bu sistem, araç içinde yerleştirilen kamera ve alarm ile uyumu durumunda bir uyarı üretebilir. Uyuma durumu tespiti, 3 yöntemle kontrol edilir: gözler kapalı veya açık, baş sağa veya sola yatık, baş aşağıda veya değil.

Anahtar Kelimeler: sürücü, baş, alarm, kamera.

ACKNOWLEDGEMENT

My dear supervisor, who supported me in the planning, research, execution, and formation of this project, whose knowledge and experience I have benefited from, I would like to express my eternal and sincere thanks to Burcu Yılmaz.

Also, I would like to express my endless and sincere thanks to my teachers, Habil Kalkan and Başak Buluz Kömeçoğlu who are leading the way for the improvement of this project. Lastly, I would like to express my respect and love to my family, who supported me in every way during my education, and to all my teachers who set an example for me with their lives.

Yakup Talha Yolcu

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation		**Explanation**
DDDS	:	Driver Drowsiness Detection System

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Drowsy driving is a major contributor to accidents on the road, and it's a problem that affects drivers of all ages and experience levels. That's why we developed our driver drowsiness detection system, a state-of-the-art solution designed to keep drivers alert and focused while behind the wheel.

Our system is built using Python and the MediaPipe library, which allows it to accurately detect when a driver is becoming drowsy. Using a Raspberry Pi computer as its hardware platform, the system is both affordable and portable, making it easy to use in a variety of settings.

But how does it work? Our system constantly monitors the driver's facial expressions and movements, looking for signs that they may be falling asleep at the wheel. When it detects these signs, it sounds an alert to wake the driver and prevent an accident from occurring.

With its advanced algorithms and reliable hardware, our driver drowsiness detection system is a valuable tool for promoting road safety and preventing accidents. Whether you're a long-haul trucker, a delivery driver, or simply someone who wants to stay alert and focused while driving, our system is here to help you stay safe on the road.

Figure 1.1: Demonstration of project

1.1. PROJECT DEFINITION

Objective: To develop a system that uses facial recognition and other sensors to detect when a driver is becoming drowsy and alert them to prevent accidents caused by drowsy driving.

Technology: The system will be developed in Python and use the MediaPipe library for facial recognition and other sensor data processing. It will run on a Raspberry Pi 4 computer, which provides a low-cost and portable platform.

Features: The system will constantly monitor the driver's facial expressions and movements, using advanced algorithms to detect signs of drowsiness. When drowsiness is detected, the system will sound an alert to wake the driver and prevent an accident. The system will also have a user-friendly interface that allows the driver to easily start and stop monitoring, as well as customize alert settings.

Benefits: By detecting and alerting drivers to drowsiness, the system will help promote road safety and prevent accidents caused by drowsy driving. It will also provide a valuable tool for long-haul truckers and other professional drivers who may be at higher risk for drowsy driving due to the demands of their job.

1.2. GOAL OF THE PROJECT

The purpose of a driver drowsiness detection project is to develop a system that can detect when a driver is becoming drowsy and alert them to take a break. Drowsy driving is a major safety concern on the roads, as it can lead to accidents, injuries, and fatalities. The goal of such a system is to help prevent these types of incidents by alerting the driver to their fatigue and encouraging them to take a break before they become too tired to drive safely. This can be achieved through the use of various technologies such as cameras, sensors, and machine learning algorithms that can detect changes in the driver's behavior, such as eye movements and facial expressions, and alert them to their drowsiness.

2. LITERATURE REVIEW

This chapter is about researches, trained models and used datasets until final version of the project is reached.

2.1. Mrl Eye Dataset

Firstly I decided to use model trained with the datasets that I found on the internet. In the dataset, images are collected from 37 different person. Dataset consist of 84898 images. Each image in the image is indicated as person is wearing a glass or not. Images also have these properties : eye is closed or not, environment light is not enough or not.

Some image examples:

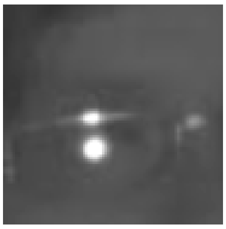

Figure 2.1: Closed eye with glasses

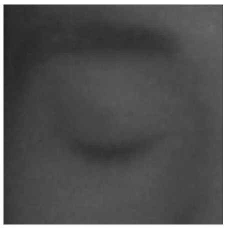

Figure 2.2: Closed eye with no glasses

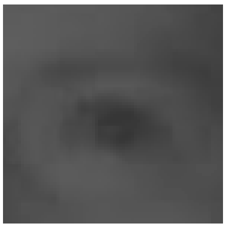

Figure 2.3: Opened eye with no glasses

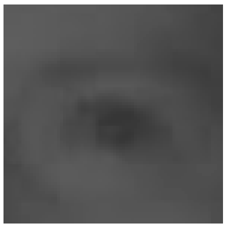

Figure 2.4: Opened eye with no glasses

2.1.1. Algorithm for closed eye detection

The Algorithm 1 shows the pseudo code of closed eye detection

Algorithm 1 Closed eye detection

```
Load the cascade classifier for eye detection
Load the image
Convert the image to grayscale
Detect eyes in the image
for each eye in detected eyes do
    Draw a rectangle around the eye
    if eye width and height is less than 30 then
        Print "The eye is closed"
    else
        Print "The eye is open"
    end if
end for
Show the image with the detected eyes
```

2.2. Yawn Eye Detection Dataset

In this dataset there are 4 classes. Closed - open eye and yawning face - non yawning face. Distribution of images into classes is given in the table

Table 2.1: Distribution of images into classes

Closed eye	617
Opened eye	617
Yawning face	616
Non-yawning face	616

Some image examples:

Figure 2.5: Closed eye

Figure 2.6: Opened eye

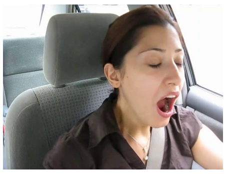

Figure 2.7: Yawning face

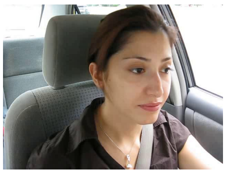

Figure 2.8: Non yawning face

There are about 100 images for each class for testing.

2.2.1. Algorithm for drowsy face detection

The Algorithm 2 shows the pseudo code of drowsy face detection

Algorithm 2 drowsy face detection

```
Load the cascade classifiers for face and eye detection
Initialize the video capture
closed_count ← 0
while True do
    Capture a frame from the webcam
    Convert the frame to grayscale
    Detect faces in the frame
    for each face in detected faces do
        Draw a rectangle around the face
        Get the grayscale image of the face
        Detect eyes in the face
        for each eye in detected eyes do
            Draw a rectangle around the eye
            if eye width and height is less than 30 then
                closed_count ← closed_count + 1
                if closed_count ≥ 5 then
                    Play an alarm sound
                end if
            else
                closed_count ← 0
            end if
        end for
    end for
    Show the frame in a window
    Wait for 'q' key press to break the loop
end while
Release the capture and close the window
```

2.3. Combination of own created dataset and Yawn Eye Detection Dataset

After using two different dataset, I took pictures of myself and classified them one by one. Then I combined them and trained a model. My images has 2 classes : Yawning face and non-yawning face. Yawning face class has 35 images and non-yawning images has 35 images. Example of the images [1]

Figure 2.9: Non-yawning face

Figure 2.10: Yawning face

2.3.1. Used Algorithm

The Algorithm shows the pseudo code of test code of combination algorithm

Algorithm 3 Combination algorithm

```
Import necessary libraries
Load the model
Initialize the video capture
while cap is opened do
    Capture a frame from the webcam
    Make detections using the model on the frame
    Show the frame with detections in a window
    Wait for 'q' key press to break the loop
end while
Release the capture and close the window
```

3. METHOD AND SYSTEM ARCHITECTURE

Methods and datasets mentioned in the previous chapter is not used in the final version of the program because a better solution is found for the problem. Python MediaPipe library developed by Google. [2]

MediaPipe Face Mesh estimates 468 3D face landmarks in real-time. It employs machine learning (ML) to infer the 3D facial surface, requiring only a single camera input without the need for a dedicated depth sensor. Utilizing lightweight model architectures together with GPU acceleration throughout the pipeline, the solution delivers real-time performance critical for live experiences. ML pipeline consists of two real-time deep neural network models that work together: A detector that operates on the full image and computes face locations and a 3D face landmark model that operates on those locations and predicts the approximate 3D surface via regression.

The pipeline is implemented as a MediaPipe graph that uses a face landmark subgraph from the face landmark module, and renders using a dedicated face renderer subgraph. The face landmark subgraph internally uses a face detection subgraph from the face detection module

Here is an example of how MediaPipe works

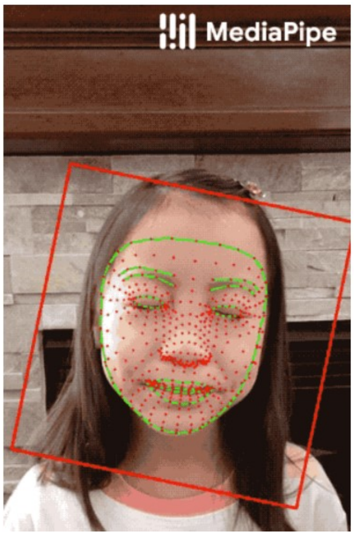

Figure 3.1: Face Mesh Example

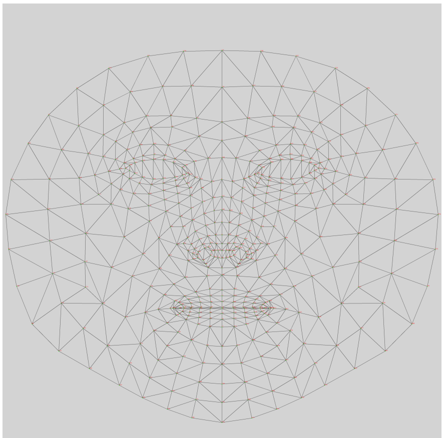

Figure 3.2: Mediapipe Face Landmarks

From now on, sections will be separated into parts that each represents program mode as closed eye detection, head right-left detection and head down detection. Exceptionally, detection of face is a common problem for all modes, it is written as a separate section.

3.1. Detection of face

As a start we have to detect face from the current image frame. When we import mediapipe as mp, we can use this code snippet to get face mesh mode of the MediaPipe library

```
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh()
```

3.2. Closed Eye Detection Mode

For closed eye detection mode, EAR formula is used.

$$EAR = \frac{||P_2 - P_6|| + ||P_3 - P_5||}{2||P_1 - P_4||}$$

The EAR formula returns a single scalar quantity that reflects the level of eye-opening.

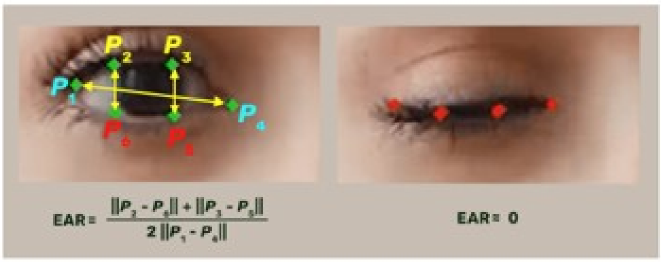

Figure 3.3: EAR formula

The EAR is mostly constant when an eye is open and gets close to zero, while closing an eye is partially person, and head pose insensitive. The aspect ratio of the open eye has a small variance among individuals. It is fully invariant to a uniform scaling of the image and in-plane rotation of the face. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.

[3]

In our case, MediaPipe has various landmarks for these points. I decided to select these indices for our problem:

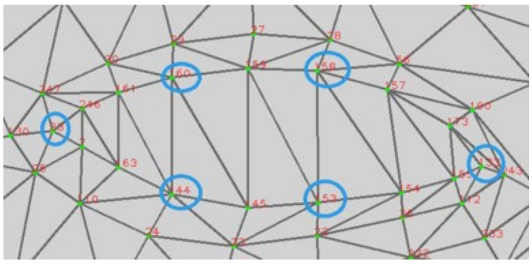

Figure 3.4: Left eye indices used

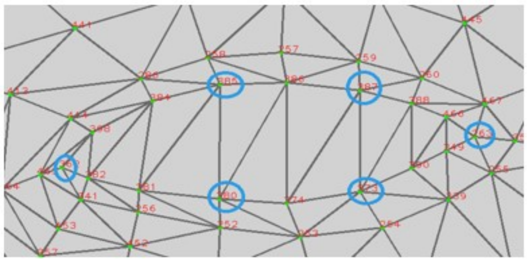

Figure 3.5: Right eye indices used

3.2.1. Initializing constants

After have a reference to face mesh, we need to initialize the necessary constants.

```
#set thresholds
EAR_THRESHOLD=0.3
WAIT_TIME=3.0
D_TIME=0

#init face mesh
mp_face_mesh=mp.solutions.face_mesh
face_mesh=mp_face_mesh.FaceMesh()

#set source video
cap=cv2.VideoCapture(0)

height=-1
width=-1

#used indices for EAR formula
l_eye_indices=[362,385,387,263,373,380]
r_eye_indices=[33,160,158,133,153,144]

#init alarm sound
mixer.init()
mixer.music.load("alarm.mp3")

isDrowsy=False
isDone=False

t1=time.time()
```

3.2.2. Infinite loop for detection

Algorithm 4 Infinite loop algorithm

```
while True do
    read image
    if i thenmage is not read correctly, break the loop
    end if
    if i thenmage's width and height are not set, set them convert image to rgb
    end if
    process image using face mesh
    if f thenacial landmarks are detected
        for each set of facial landmarks do
            calculate left and right eye aspect ratio
            if b thenoth left and right EAR are less than threshold
                calculate time passed since last check
                add to total drowsy time
                if t thenotal drowsy time exceeds wait time
                    play alarm if not already done
                    set isDrowsy and isDone to true
                end if
            else
                reset drowsy time and last check time
            end if
        end for
        for  each set of facial landmarks do
            draw circles at specific landmark points
        end for
        show image
    else
        reset drowsy time and last check time
    end if
end while
```

3.3. Head Right-Left Detection Mode

In this mode I looked the angles and slopes between right cheek and chin / left cheek and chin. If left angle is between -0.5 and 0 , then driver is drowsy. If right angle is between -0.5 and 0, then driver is drowsy Otherwise driver is not drowsy.

Used indices are Right cheek index = 93 Left cheek index = 323 Chin index = 152

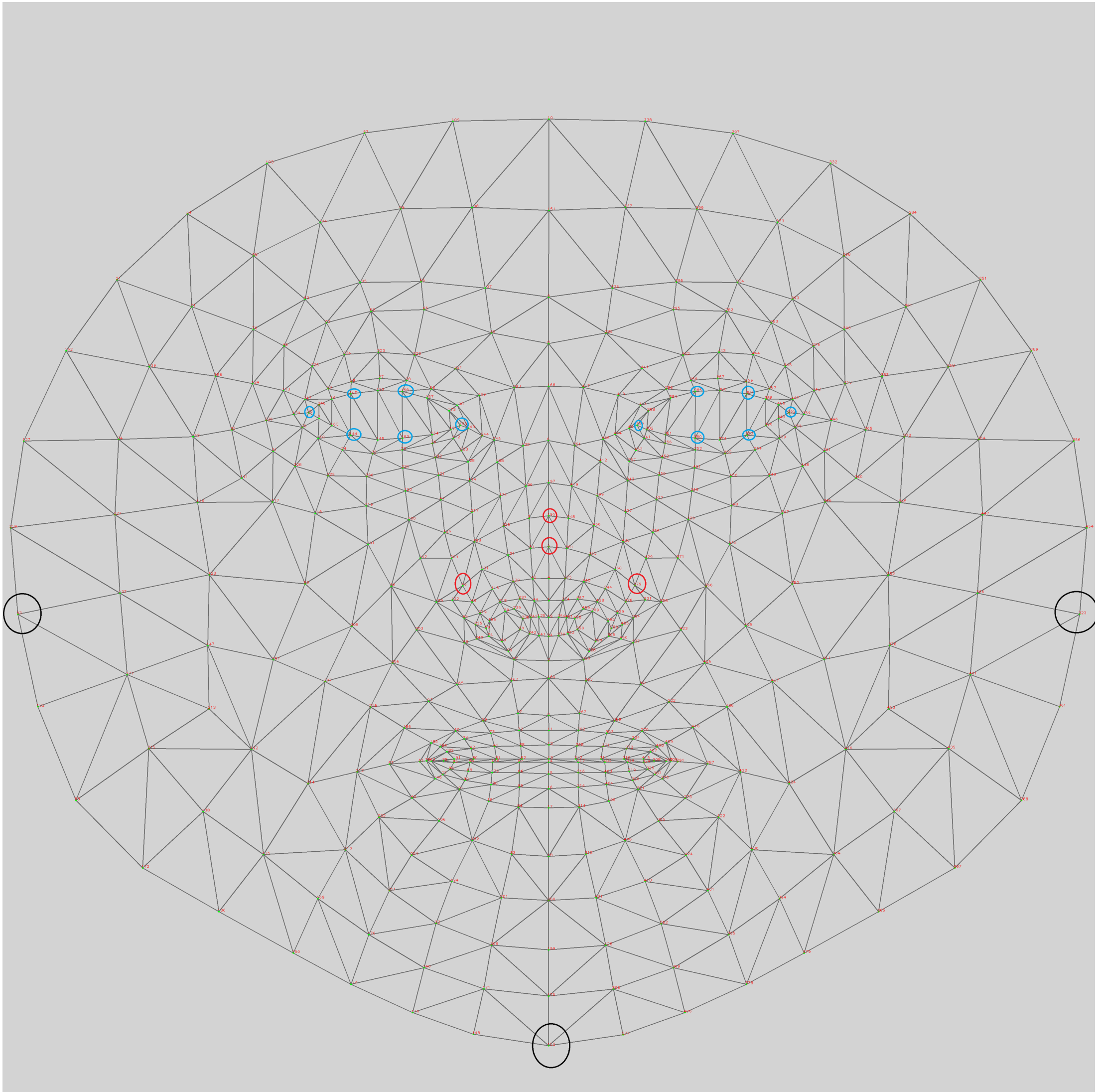

Figure 3.6: All landmarks with used indices

3.3.1. Infinite loop for detection

Algorithm 5 Pseudocode for detecting drowsiness using facial landmarks

```
while True do
    Read image
    if unable to read image then
        break
    end if
    if width and height have not been set then
        get them from the image
    end if
    Convert image to RGB
    Process image
    if multiple face landmarks are detected then
        for all facial landmarks do
            Get points for right cheek, left cheek, and chin
            Calculate slopes of lines connecting right cheek, left cheek, and chin
            Calculate angles of lines connecting right cheek, left cheek, and chin
            if angle of left cheek is between -0.5 and 0 then
                Update D_TIME with time since last check
                if D_TIME is greater than or equal to WAIT_TIME and alarm has not already been played then
                    Play alarm
                    Set isDrowsy to True and isDone
                end if
            end if
        end for
    end if
end while
```

3.4. Head Down Detection Mode

In this mode, I took 2 points on the nose edge and nose line, I also took 2 points that right and left sides of the nose. I extracted 2 average points between these 4 points. Then I compared their y indices, if first ones y coordinate is greater than the second one, then driver is drowsy.

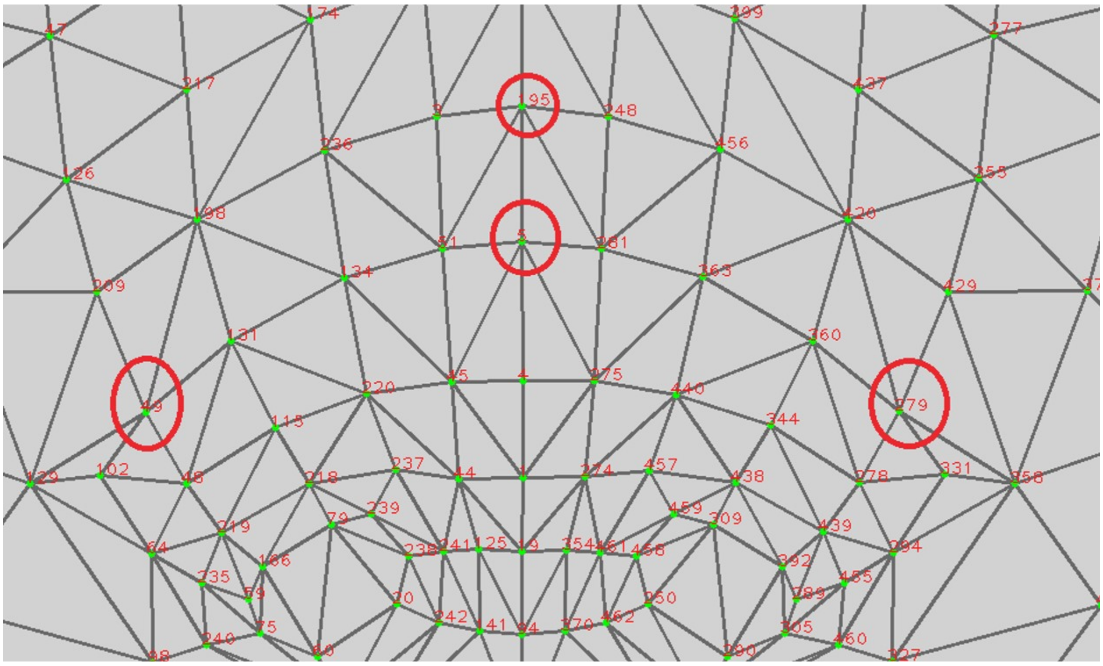

Figure 3.7: Head down mode indices

3.4.1. Infinite loop for detection

Algorithm 6 Drowsy Detection

Initialize video capture, face mesh, and audio mixer
Set drowsy time, wait time, and timer variables to 0
Set indices for facial landmarks to track
while True **do**
    Read image from video source
    if unable to read image **then**
        break out of loop
    end if
    if width and height have not been set **then**
        get them from the image
    end if
    Convert image to RGB
    Process image using MediaPipe face mesh
    if multiple face landmarks are detected **then**
        for each facial landmark **do**
            Draw circles on the image at the points corresponding to certain facial landmarks on the nose
        end for
        Get points for right and left sides of nose, and upper part of nose
        Calculate middle point of nose based on y-coordinates of left and right nose points
        if middle point is below upper part of nose **then**
            Update timer
        end if
        if drowsy time exceeds wait time **then**
            if alarm has not already been triggered **then**
                Trigger alarm
            end if
        end if
        Display image with facial landmarks
    else
        Reset drowsy time and timer
    end if
end while

3.5. Flowchart of the general structure

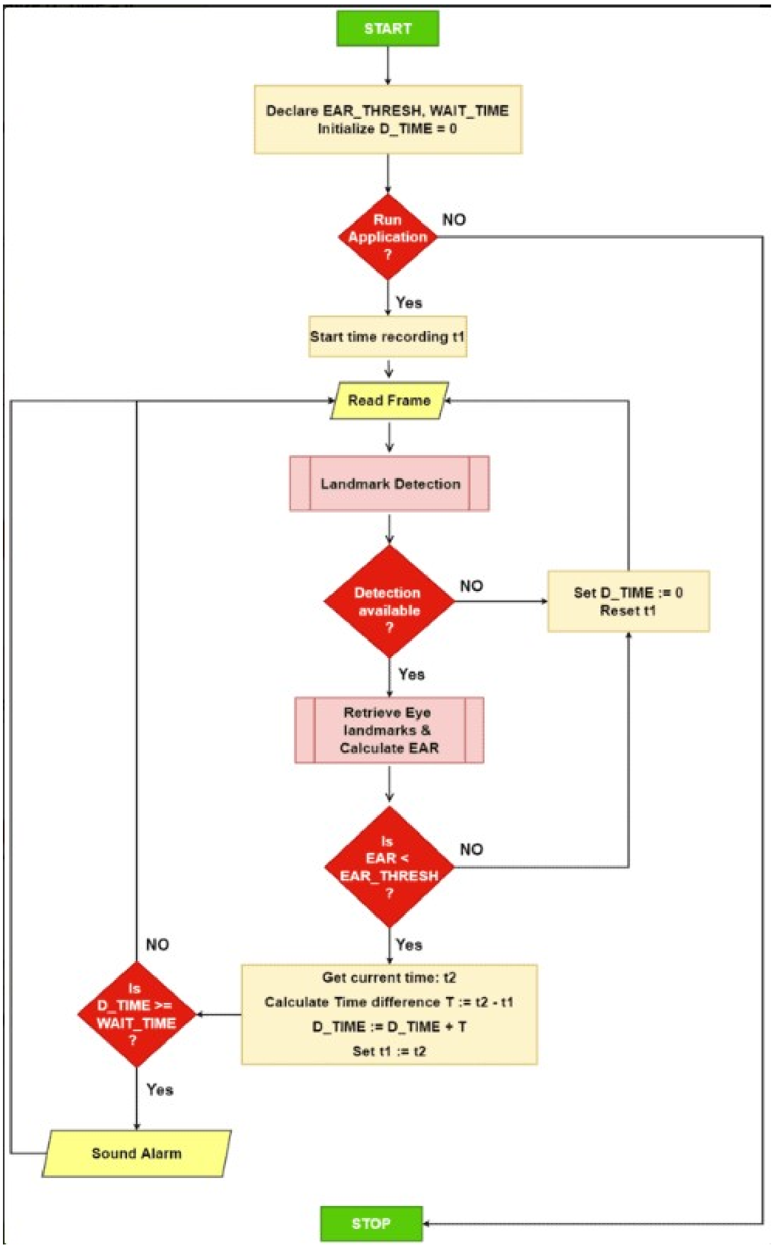

Figure 3.8: Flowchart of the program

4. EXPERIMENTS

Here are some tests and results

Figure 4.1: No drowsniness situation

Figure 4.2: Closed eye situation

Figure 4.3: Head is down situation

Figure 4.4: Head is right situation

Figure 4.5: Head is left situation

5. CONCLUSIONS

The driver drowsiness detection project uses a Raspberry Pi, a camera module, and a buzzer to detect drowsiness in a driver. The system checks for drowsiness in three ways: if the driver closes their eyes for an extended period of time, if they move their head to the left or right for an extended period of time, and if they move their head down for an extended period of time. The system uses image processing techniques to detect these actions and triggers an alarm sound using a buzzer when drowsiness is detected. This project serves as an effective solution to prevent accidents caused by drowsy driving and can be integrated into vehicles to ensure the safety of both the driver and other individuals on the road.

BIBLIOGRAPHY

[1] Radovan Fusek. “Mrl eye dataset.” (2021), [Online]. Available: `http://mrl.cs.vsb.cz/eyedataset` (visited on 01/17/2023).

[2] Google. “Mediapipe library.” (2012), [Online]. Available: `https://google.github.io/mediapipe/solutions/face_mesh.html` (visited on 01/17/2023).

[3] Tereza Soukupova and Jan ´ Cech. “Real-time eye blink detection using facial landmarks.” (2016), [Online]. Available: `http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf` (visited on 01/17/2023).

CV

APPENDICES