Swaying Attention: A Software and Hardware Approach to Students Dozing Off During Online Learning

YUSUF MORSI* and PRANAV MADDIREDDY*, University of California, San Diego, USA

After acknowledging the world-wide issue of students dozing off during online class, we utilize OpenCV, a webcam, a few Python libraries, Arduino IDE, a Sparkfun ESP32, and basic circuit hardware to not only determine when a student is dozing off, but to also vibrate a buzzer that notifies the student of their drowsiness.

Additional Key Words and Phrases: webcams, OpenCV, Python, ESP32, drowsiness, EAR, eye state detection, threshold

ACM Reference Format:
Yusuf Morsi and Pranav Maddireddy. 2021. Swaying Attention: A Software and Hardware Approach to Students Dozing Off During Online Learning. 1, 1 (March 2021), 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Swaying Attention is a project that aims to tackle the problem of student attention. Students not paying attention in class was a problem when school is in person but now that most students are doing school online as a result of quarantine the problem has gotten a lot worse. Many students, including ourselves, have a hard time paying attention in online lectures and end up missing a major portion of the content. To solve this problem we created Swaying attention, a program that uses eye detection to track students dozing off and gives them a reminder to stay focused. To achieve this we used Python, OpenCV, D-lib, Arduino-C code and a Sparkfun ESP32 board. Our prototype is able to detect drowsiness and outputs a message to the student on a display as well as making noise with a buzzer to get their attention.

2 THE PROBLEM: STUDENTS DOZING OFF DURING ONLINE CLASS

The COVID-19 pandemic has made it clear that students have been dozing off while learning online. This is because of the common one-way approach of teaching [3] and because students can easily doze off without consequences [9]. We decided that by providing a reminder that would make it more difficult for students to doze off, we would take away their ability to easily do so.. Students are put under the pretense that they can get away with dozing off, so we decided to work on a solution in which it wouldn't be so easy for students to be drowsy during class.

*Both authors contributed equally to this research.

Authors' address: Yusuf Morsi, ymorsi@ucsd.edu; Pranav Maddireddy, pmaddire@ucsd.edu, University of California, San Diego, 9500 Gilman Dr, La Jolla, California, USA, 92093.

XXXX-XXXX/2021/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Statistics

Figure 5 shows that 90% of students polled claim that they have either faced 'Zoom fatigue,' or dozed off while learning online, while out of all the students that YouthTruthSurvey.org polled, 45% claimed that they get distracted while learning virtually [1]. The main takeaway from this data is that students getting distracted and/or dozing off while learning online from home is a major issue that cannot be ignored.

3 THE DESIGN

3.1 Design Considerations

Our solution to this problem involves using OpenCV to detect eye movement. However, we did have different designs in mind that still implemented OpenCV. There were three design considerations that we eventually chose from:

Design 1: A software that uses webcams and OpenCV to track body or eye movement, and makes a loud noise with a buzzer when the user dozes off.

Design 2: A system that notifies the teacher and that buzzes the user if they do not answer an easy-to-miss mental awareness question given at a random time during their lecture.

Design 3: A service that helps teachers make their lectures more interactive and interesting to students to ensure that they pay attention.

Ultimately, we chose to use the first design, as it was dependent on knowledge that we recently started to gain experience with (OpenCV).

3.2 Design Illustration

Our design uses OpenCV and Python to detect a certain ratio attained from detecting the dimensions of the user's eyes through their webcam. When the ratio goes under a certain level that's considered to be drowsy, a buzzer connected to the computer starts to vibrate loudly, and their OLED tells them to wake up, as seen in **Figure 10**.

In order to be able to work with OpenCV, we needed to familiarize ourselves with it's features with a tutorial [4]. This provided us with the knowledge needed to detect certain features using a webcam with OpenCV and Python code.

In our design, we needed to learn how to detect blinks with OpenCV. Using the Eye Aspect Ratio (EAR) equation derived from a paper from the Czech Technical University [10], we get a number that we can use to distinguish open eyes from closed eyes.

The process of our program is as follows. First, our program establishes a connection with the Arduino-C code that is connected to our SparkFun board. Then our program activates a connected webcam and starts reading from it. Then it runs a facial detection on the inputs that it is receiving from our webcam. If a face is detected, our program uses a pre-loaded model to identify facial features. Our program focuses on the left and right eyes. Once the eyes are detected, our program runs our EAR (Eye Aspect Ratio)-finding method for both eyes and finds the average EAR between them. Subsequently, our program draws two hull shapes for where it detects our eyes on the output. This is used to test the functionality of the eye detection. Then, our program compares the average EAR that it just derived to the threshold. If it is less than the threshold, the variable that stores the

number of consecutive frames of drowsiness increments by 1, and if it's greater than the threshold, this variable gets set to 0. Then the code checks if the previously defined variable is greater than or equal to our frame_check variable. If it is, our program outputs warnings of drowsiness. The connected OLED then gives a warning message (**Figure 10**), the connected buzzer vibrates, and the output video from the program displays a drowsiness alert over the video. If the variable is less than the frame_check variable, then nothing happens and our program continues to the next frame.

$$EAR = \frac{(||p_2 - p_6|| + ||p_3 - p_5||)}{(2||p_1 - p_4||)}$$

We use this number in our code to identify whether eyes are open or closed [7]. When a face is detected, the EAR ratio is calculated, and if it's lower than usual for 20 consecutive frames [7], our buzzer will start to make a noise [5]. To achieve the code for this, we used sources [7][10], which guided us when it came to importing certain libraries to get our Python code for detecting drowsiness. Subsequent to finishing our Python code implementing OpenCV, we then built upon our Arduino-C code from previous projects in ECE16 for the buzzer implementation.

4 AN ANALYSIS OF THE SOLUTION'S PERFORMANCE

In this project, we set a threshold level to count drowsiness. When the EAR (Eye Aspect Ratio) goes below the threshold our program detects it as drowsiness. The problem is how to determine the threshold for drowsiness? To find the answer we decided to collect data from multiple subjects. We collected data of their EAR when they were keeping their eyes "awake" and open as well as EAR data for their drowsy eyes. We also took a measurement of their eye dimensions to see if we could find some correlation between eye dimensions and the threshold for drowsiness.

Figure 1 shows a graph that compares the values of data attained when the eyes of a participant in our data collection, Austin, are open to when he is dozing off.

The yellow line in this graph represents the open eye EAR values of our subject Austin while the blue line represents his average Drowsy EAR (taken as an average of 5 samples). The only overlap between the graphs occurs around the one-second mark and indicated when the subject blinked during the test. We can deduce that it is a blink because of how it sharply dips to the bottom of the graph and comes right back up just as fast. Since it's a blink it is seen as an impurity in the Open eye data and is disregarded. Other eye blink data located in open eye data will be disregarded as well in the duration of this analysis. Other than the blink there is a clear difference between the EAR of Austin's open and drowsy eyes. For the purpose of this project, we decided that the highest point in the drowsiness EAR line will be used as a threshold because it will exclude all open eye data and include all drowsy data. Austin's experimental threshold value is 0.267.

Figure 1 shows that one of our data collection volunteers, Austin, has an experimental threshold value is 0.267. **Figure 2** shows that Pranav, one of the researchers on this project, has an experimental threshold value is 0.233. **Figure 3** shows that Yusuf, the other researcher on this project, has an experimental threshold value is 0.172. **Figure 4** shows that Eddie's experimental threshold value is 0.242.

Through some basic data analysis, we were able to find effective threshold values that increase the accuracy of our drowsy detection for each individual. Though this approach works it isn't very realistic if we wanted to make this into a real product. To create an individual threshold value without needing to collect lots of data and analyze graphs we decided to look for a correlation between eye dimensions and the threshold value. This way users would only have to input eye

dimensions instead of a complicated calibration process. To do this we surveyed data for the following table from our subjects.

Table 1. Table showing eye dimensions.

Subject Name	Eye Horizontal dimension (in)	Eye Vertical dimension (in)
Austin	9/8	1/4
Pranav	4/3	3/8
Yusuf	7/5	1/2
Eddie	5/4	1/2

With eye data and experimental threshold values, we can attempt to create an equation for threshold values based on horizontal and vertical eye dimensions. We decided that we would solve a system of equations created with the data of the first two subjects and test its accuracy determining the threshold value of the last subject.

System of equations created from the data of Austin and Eddie:

$$(9/8)x_1 + ()x_2 = 0.267$$
$$(5/4)x_1 + (1/2)x_2 = 0.242$$
$$x_1 = 73/250x_2 = -123/500$$
$$hx_1 + vx_2 = threshold$$

Using these values we can create this general equation for threshold values where h = horizontal eye dimension and v = vertical eye dimension: (73/250)h - (123/500)v = theoretical threshold

Using this equation we got an estimated threshold of 0.266 with Pranav's data and 0.285 with Yusuf's. Though Pranav's estimate was 14%, and Yusuf's was 26% off the mark, there is clearly a correlation between eye dimensions and drowsiness threshold. Perhaps with more data we could find a more accurate equation.

Using this equation for threshold we can estimate, with our current data, that our project is about 80% accurate on average. We get this number because the error of the theoretical thresholds is decided by the values of the experimental thresholds. The experimental thresholds represent 100% of the drowsiness data for each subject so the accuracy of determining the threshold value determines the accuracy of the program as a whole. Since the estimate was off by 14% the program has an accuracy of 86%.

5 CONCLUSION

Although we haven't tested our project during a real lecture, our development of a prototype was quite successful. With our minimal data, we can conclude that if we are able to continue down this path we could create a much more accurate version of our product that can be used in the real world. However, there are many other issues that we have to solve before making the jump to a final product. Though our product is robust in some ways, works with sunglasses on (**Figure 11**, **Figure 12**), it has many other issues that would make it an unrealistic solution to the problem we discuss as it is now. For instance, the student has to be staring directly at the camera for our program to assess drowsiness effectively. When the user looks away, it may detect drowsiness and

send an alert. Also, the program can't detect the eyes of students wearing masks (**Figure 13**). This is because our program needs to detect a person's face before being able to track eyes. Finding a way to go straight into tracking eyes instead of face first or making a detection model that works for people with masks on too will be important for the future of this product. Despite the fact that this interaction was not a planned part of our product, it can act as a way to persuade students that are looking away or distracted from the lectures to pay attention! Another issue is distance. The user needs to be quite close to the webcam for accurate measurements, but it shouldn't be a major problem for students sitting at desks. However, there are a few edge cases with which our system might fail. Collecting accurate data is also a topic in this project that requires a lot of effort. The errors in our analysis not only come from our minimal amount of data, but the way we collect it as well. In this project, we had our subjects sit in front of the camera and act drowsy. This type of data isn't generated from genuine drowsiness and so the data may be skewed. For example, we needed to create a new data set for one of our subjects because he tried too hard to be drowsy and we completely missed the upper EARs that would account for his drowsiness. Finding a better strategy for data collection is necessary for moving forward with this project. Though our project isn't quite ready for production, it is a promising prototype that has the potential to become something greater with further experimentation and work.

6 FIGURES

The following is a collection of plots and figures we have collected throughout this project.

NOTE: For Figures 1 - 4, the data for opened eyes (orange) was derived from one initial round of data collection conducted with the user having opened eyes, and the data for 'drowsy' eyes was derived by taking the average of 5 trials in which the user was dozing off (except for Figure 2, in which case 10 trials were conducted). In addition, the EARs vary in the different plots, as different users have different faces, thus different EARs.

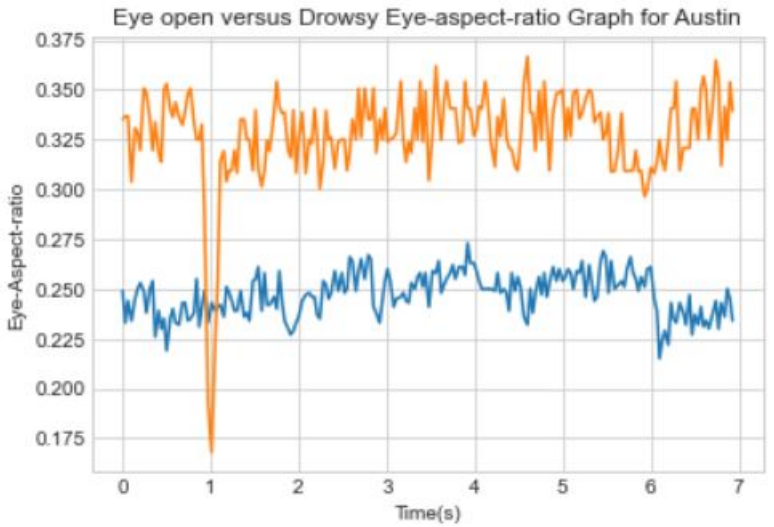

Fig. 1. Plot Comparing Open Eyes and 'Drowsy' Eyes- Austin).

Figure 1: This is a plot comparing open eyes (orange) and 'drowsy' eyes (blue) for Austin, a participant in our data collection. In this plot, x is the time measured in seconds, while y is the Eye Aspect Ratio. The opened eyes show higher values for the EAR, while the 'drowsy' eyes show lower values of around 0.25.

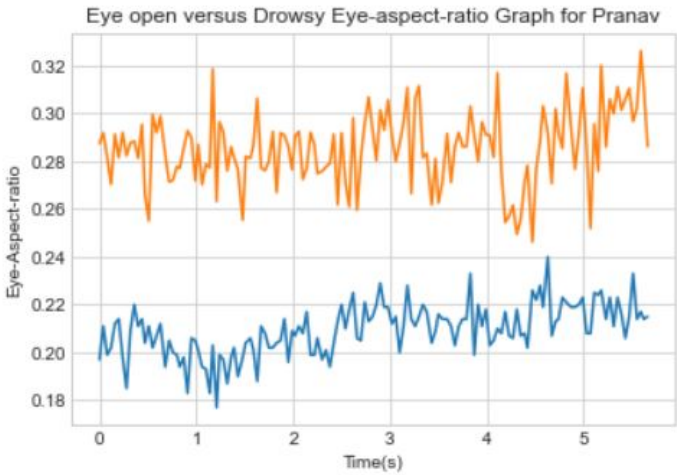

Fig. 2. Plot Comparing Open Eyes and 'Drowsy' Eyes- Pranav).

Figure 2: This plot compares open eyes (orange) and 'drowsy' eyes (blue) for Pranav, one of the researchers in this project. It can be inferred from the figure that the variable x is the time measured in seconds, while y is the EAR. The opened eyes show higher values for the EAR, while the 'drowsy' eyes show lower values of around 0.22.

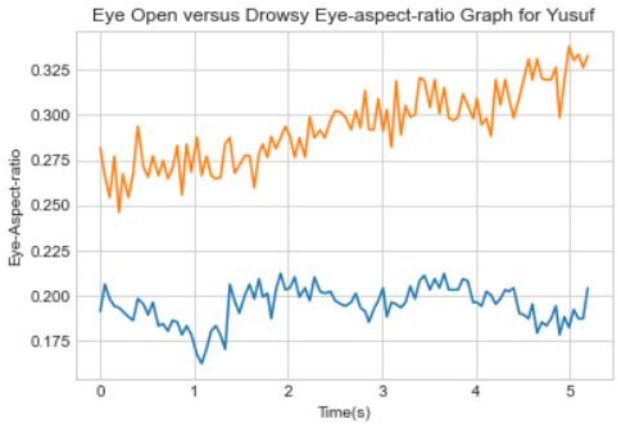

Fig. 3. Plot Comparing Open Eyes and 'Drowsy' Eyes- Yusuf).

Figure 3: This plot compares open eyes (orange) and 'drowsy' eyes (blue) for Yusuf, one of the researchers in this project. It can be inferred from the figure that the variable x is the time measured in seconds, while y is the EAR. The opened eyes show higher values for the EAR, while the 'drowsy'

eyes show lower values of around 0.3. The large dip in the data is a blink, ask the EAR momentarily goes to a low value when one blinks.

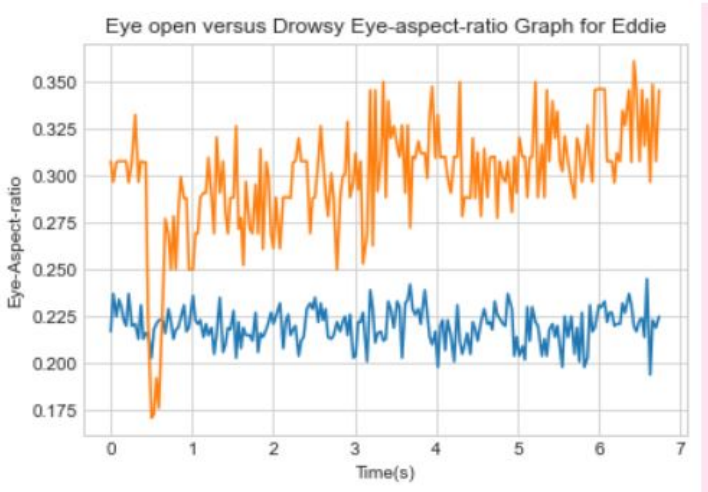

Fig. 4. Plot Comparing Open Eyes and 'Drowsy' Eyes- Eddie

Figure 4: This is a plot comparing open eyes (orange) and 'drowsy' eyes (blue) for Eddie, a participant in our data collection. In this plot, x is the time measured in seconds, while y is the Eye Aspect Ratio. The opened eyes show higher values for the EAR, while the 'drowsy' eyes show lower values of around 0.225. The large dip in the data is a blink, ask the EAR momentarily goes to a low value when one blinks.

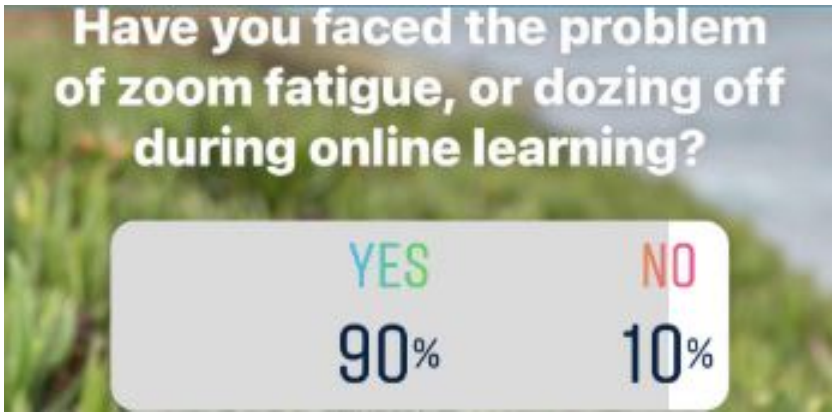

Fig. 5. Poll Showing Percentage of Students That Doze Off While Learning Online).

Figure 5: This is a poll that we conducted through Instagram that shows that 90% of the students that replied face 'zoom fatigue,' or in other words. doze off while learning online. About 85 students participated in this poll.

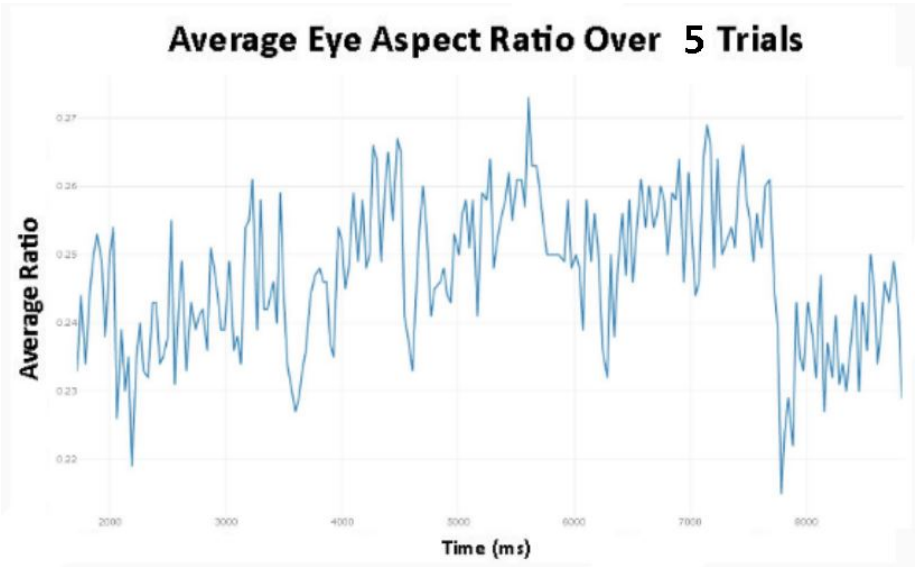

Fig. 6. Austin's Avg EAR

Figure 6: This is a plot showing the average EAR of one of our volunteers, Austin, over 5 trials. The x-axis was time in milliseconds, while the y-axis is the Average ratio, which was calculated in the code. In all of these trials, Austin had his eyes in a drowsy state, so these values can be used to determine his threshold.

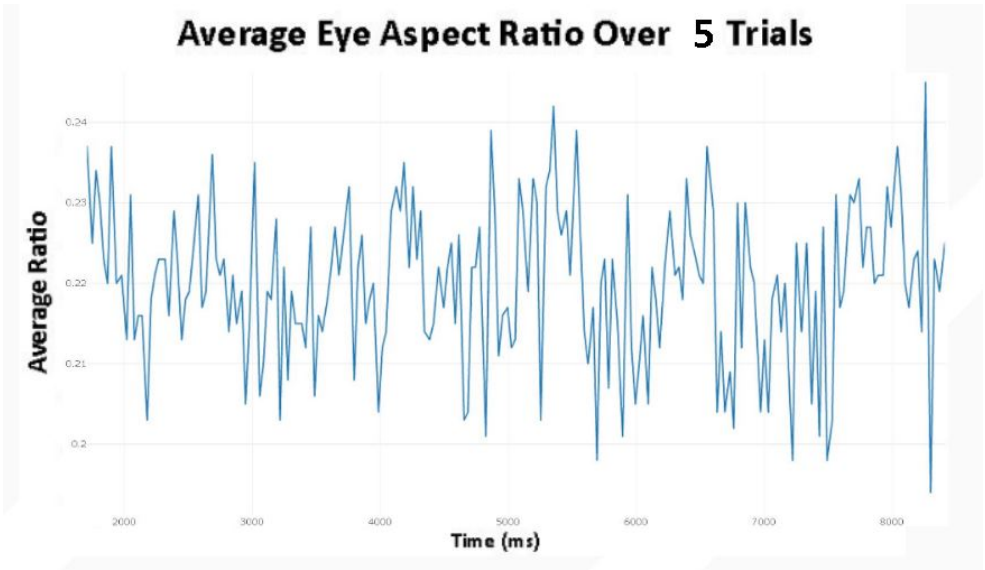

Fig. 7. Eddie's Avg EAR.

Figure 7: This is a plot showing the average EAR of one of our volunteers, Eddie, over 5 trials. The x-axis was time in milliseconds, while the y-axis is the Average ratio, which was calculated in

the code. In all of these trials, Eddie had his eyes in a drowsy state, so these values can be used to determine his threshold.

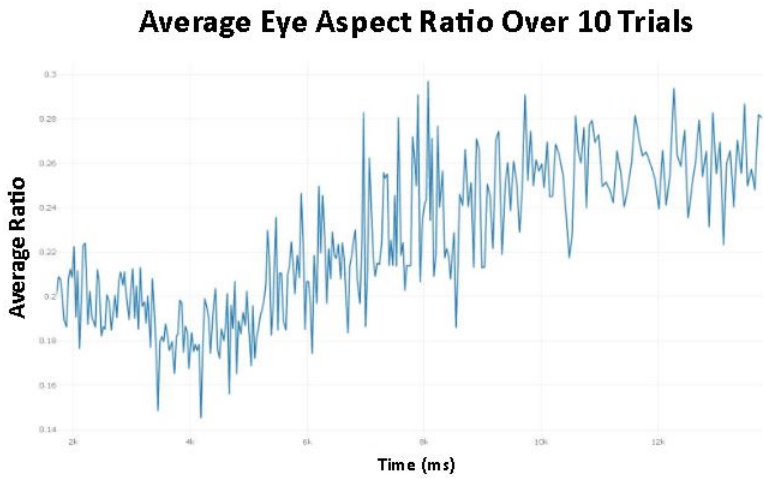

Figure 8: Demonstration of Final Product In Action

Fig. 8. Pranav's Avg EAR.

Figure 8: This is a plot showing the average EAR of one of the researchers on this project, Pranav, over 10 trials. The x-axis was time in milliseconds, while the y-axis is the Average ratio, which was calculated in the code. Pranav was in a drowsy state while all of these trials were conducted, so these values can be used to determine his threshold.

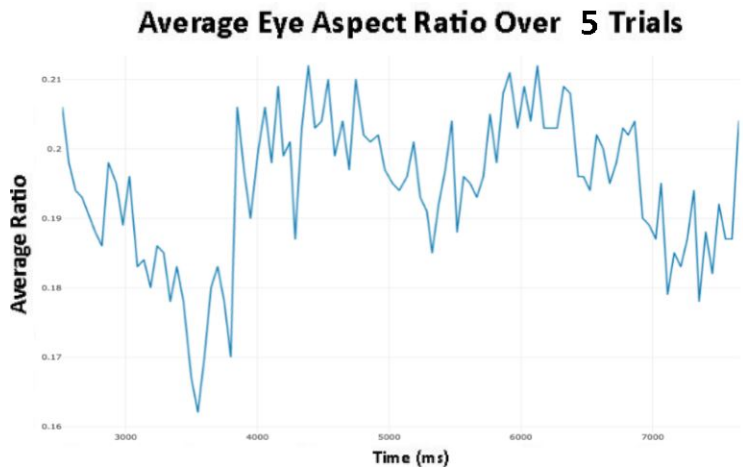

Fig. 9. Yusuf's Avg EAR.

Figure 9: This is a plot showing the average EAR of one of the researchers on this project, Yusuf, over 5 trials. The x-axis was time in milliseconds, while the y-axis is the Average ratio, which was

calculated in the code. Yusuf was in a drowsy state while all of these trials were conducted, so these values can be used to determine his threshold.

Fig. 10. Message on OLED.

Figure 10: This is a picture showing what message is displayed on the user's OLED when then become drowsy. As mentioned previously, the user's drowsiness is detected based on their Eye Aspect Ratio. In addition to this OLED displaying a message, a buzzer vibrates when the user is drowsy.

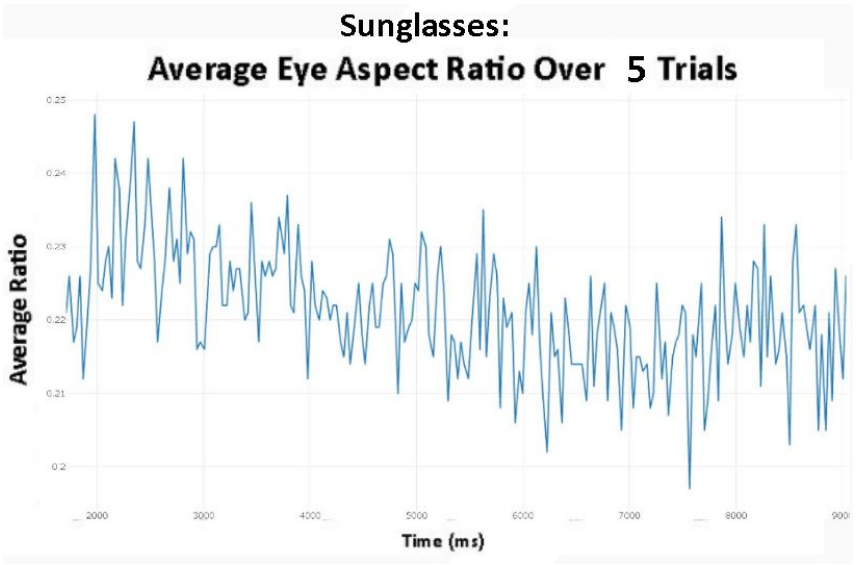

Fig. 11. Avg EAR with Sunglasses.

Figure 11: This is a plot showing the average EAR of one of the researchers on this projects, Yusuf, while wearing sunglasses indoors. The x-axis is time (in milliseconds), and the y-axis is the

average Eye Aspect Ratio. The values fluctuated and reflected upon the user's drowsiness, but very inaccurately compared to eyes without sunglasses. This shows how our solution can detect eyes, even when sunglasses are worn indoors.

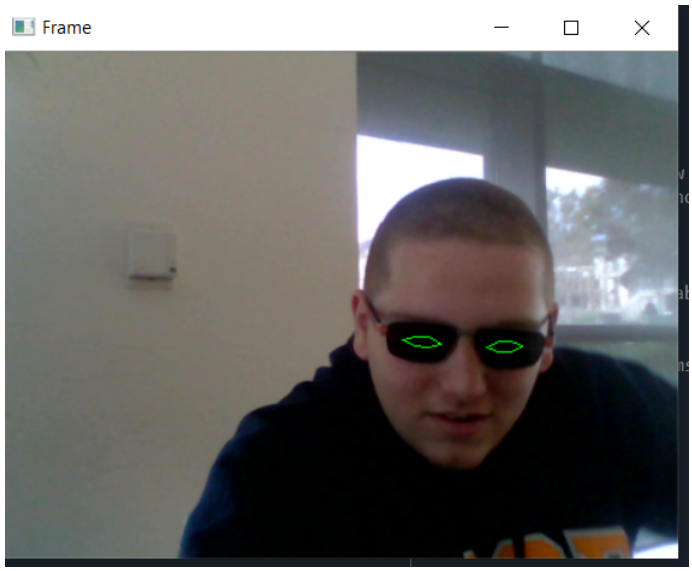

Fig. 12. Eyes Being Detected Through Sunglasses (Yusuf).

Figure 12: This is a picture showing how eyes are detected even when sunglasses are worn indoors. The design was able to detect the eyes based on the other facial measurements, but was not very accurate in detecting drowsiness.

Fig. 13. The Result of a User (Pranav) Attempting to use the Design With a Mask on.

Figure 13: This is a picture of what is detected when the user tries to use the design with a mask on. Because both the mouth and nose are both concealed, the design is unable to detect the dimensions of the face, which was part of the process of detecting eyes. When the mask is taken off, green ovals can be seen over the eyes, showing that they are detected.

ACKNOWLEDGMENTS

To Professor Wang, for an amazing and educational quarter; and Adrian Rosebrock [6] [5] [8] [7] and Akshay Bahadur [2] for the assistance that their research was to us in this paper.

REFERENCES

[1] [n.d.]. *Students Weigh In: Learning Well-Being During COVID-19.* Retrieved March 14, 2021 from https://youthtruthsurvey.org/student-weigh-in/

[2] Akshay Bahadur. 2017. *Drowsiness Detection.* Retrieved March 13, 2021 from https://github.com/akshaybahadur21/Drowsiness_Detection

[3] Robert Frost. 2014. *Here's Why Students Fall Asleep During Lectures.* Retrieved March 13, 2021 from https://www.businessinsider.com/why-students-fall-asleep-during-lectures-2014-7

[4] Harrison Kinsley. 2015. *OpenCV with Python Intro and loading Images tutorial.* Retrieved March 05, 2021 from https://pythonprogramming.net/loading-images-python-opencv-tutorial/

[5] Adrian Rosebrock. 2017. *Drowsiness detection with OpenCV.* Retrieved March 14, 2021 from https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

[6] Adrian Rosebrock. 2017. *Eye blink detection with OpenCV, Python, and dlib.* Retrieved March 14, 2021 from https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

[7] Adrian Rosebrock. 2017. *Facial landmarks with dlib, OpenCV, and Python.* Retrieved March 13, 2021 from https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[8] Adrian Rosebrock. 2017. *How to install dlib.* Retrieved March 13, 2021 from https://www.pyimagesearch.com/2017/03/27/how-to-install-dlib/

[9] Ken Shore. [n.d.]. *Sleeping In Class.* Retrieved March 13, 2021 from https://www.educationworld.com/a_curr/shore/shore005.shtml

[10] Tereza Soukupova and Jan Cech. 2016. *Real-Time Eye Blink Detection using Facial Landmarks.* Retrieved March 13, 2021 from http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf