Eyestrain-Alleviating Monitor Adjustment

ECE 445 Design Document

Bryan Huang, Peter Siborutorop, Raghav Verma
Group 69
TA: Bonhyun Ku

3/4/21

1. Introduction

1.1 Background

Digital eye strain is a common condition that affects people who spend time using display devices. It has been shown to have multiple negative effects on vision, and most people are potentially affected by it. The primary strategy for preventing digital eye strain is prevention - adjusting the viewing environment to minimize the potential for eye strain [1].

Two symptoms associated with digital eye strain are reduced blink rate and blink completeness. These symptoms can be detected through computer vision, and serve as a sign that the current viewing environment is not healthy for the user's eyes [1]. There exist some software-only solutions that aim to detect and alleviate eye strain using this strategy, but their functionality is limited and require the user to actively work against their eye strain [2].

1.2 Objective

Our project aims to relieve eye strain from viewing devices without inconvenience to the user. We plan on taking a vision-based approach to detect eye strain through blink data, and automatically adjust the user's display. We will use a camera, ambient light photoreceptor and a microcontroller to detect and communicate potential eye strain to software on a target device, which will adjust the target device's display to accommodate the user.

1.3 High-Level Requirements

1. The software application should be able to detect the user's blink rate accurately within ±20% of the actual rate.
2. The software application should be able to make a noticeable change to the user's device display settings based on the inputs.
3. The blink detection system should be able to run continuously without stalling or crashing.

2. Design

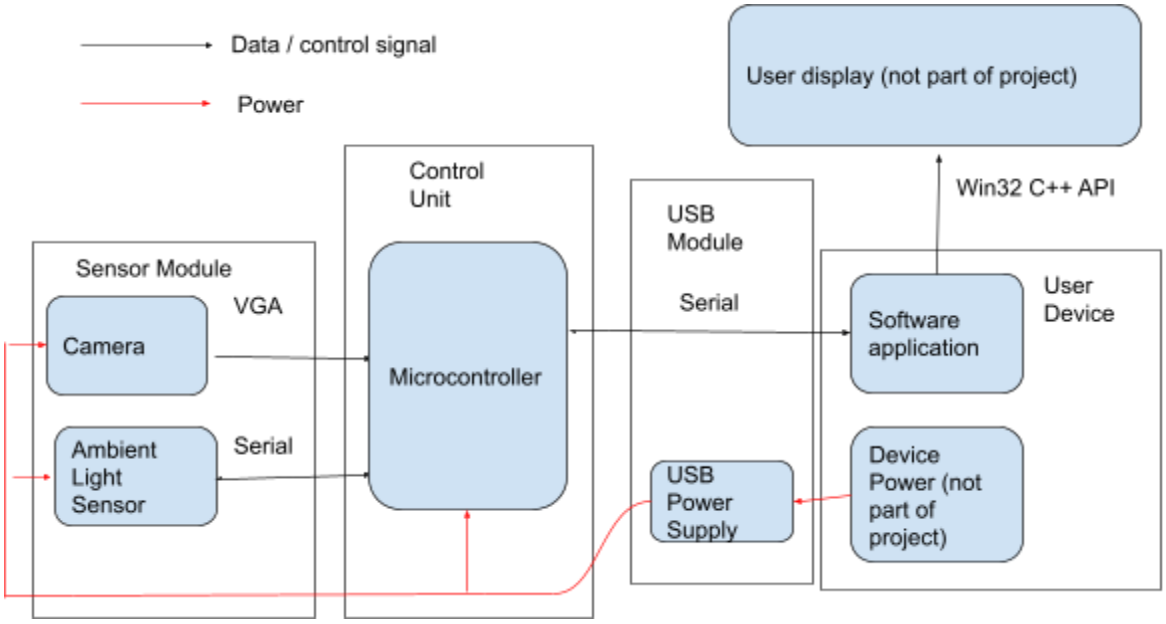


Figure 1. Block Diagram

2.1 Physical Design

Because our device has few moving parts and does not require a particular structure, a particular physical design is not needed. However, it should still be noted that a mounting bracket for the OV7670 camera module exists, and that this bracket can be used with some standard camera mounts. The camera module and device can be mounted this way to allow for adjusting the camera's viewing angle.

2.2 Sensor Unit

2.2.1 Camera

The OV7670 camera module provides 8-bit VGA pin output to the microcontroller at 30 frames per second, as well as the necessary clocks to read properly. It must be driven by an external clock. This part was chosen for its availability and compatibility with our microcontroller.

Requirement	Verification
1. The camera should be able to capture images and deliver them to the microcontroller at a rate of at least 10 frames per second.	1. Display a timer or test pattern on another device, then have the camera capture images at the maximum capture rate and intended resolution

	for 5 seconds. At least 50 distinct images are captured. a. The microcontroller will be needed to transfer and confirm the captured images.

2.2.2 Ambient Light Sensor

The TI OPT3001 ambient light sensor can communicate with the microcontroller through serial output [7]. It outputs luminosity as a digital value in the serial output, when initialized by the microcontroller.

Requirement	Verification
1. The photodetector delivers a signal with levels that can be read in software on the microcontroller. It should be able to detect the actual ambient light level within 10 lux.	1. Use a dimmer switch or other adjustable light to change the ambient light level of the environment. Use the microcontroller to check the outputs of the photodetector, and compare the results with a digital light meter in the same environment.

2.3 Control Unit

2.3.1 Microcontroller

The ATMEGA32U4 microcontroller has support for USB communication as well as compatibility with our sensor modules. It performs pre-processing on raw image data from the camera and transmits the data to the software for blink detection.

Requirement	Verification
1. The microcontroller must be able to pre-process each camera image before requesting the next image at a rate of at least 10 frames per second.	1. Have the microcontroller pre-process the worst-case image data for our chosen algorithm multiple times in a row, and mark a timestamp on a connected console when each image is finished processing. The time between all timestamps should be less than 100ms. a. Unprocessed image data should be pre-loaded on-board for the purpose of this verification.

2.4 User-side Software

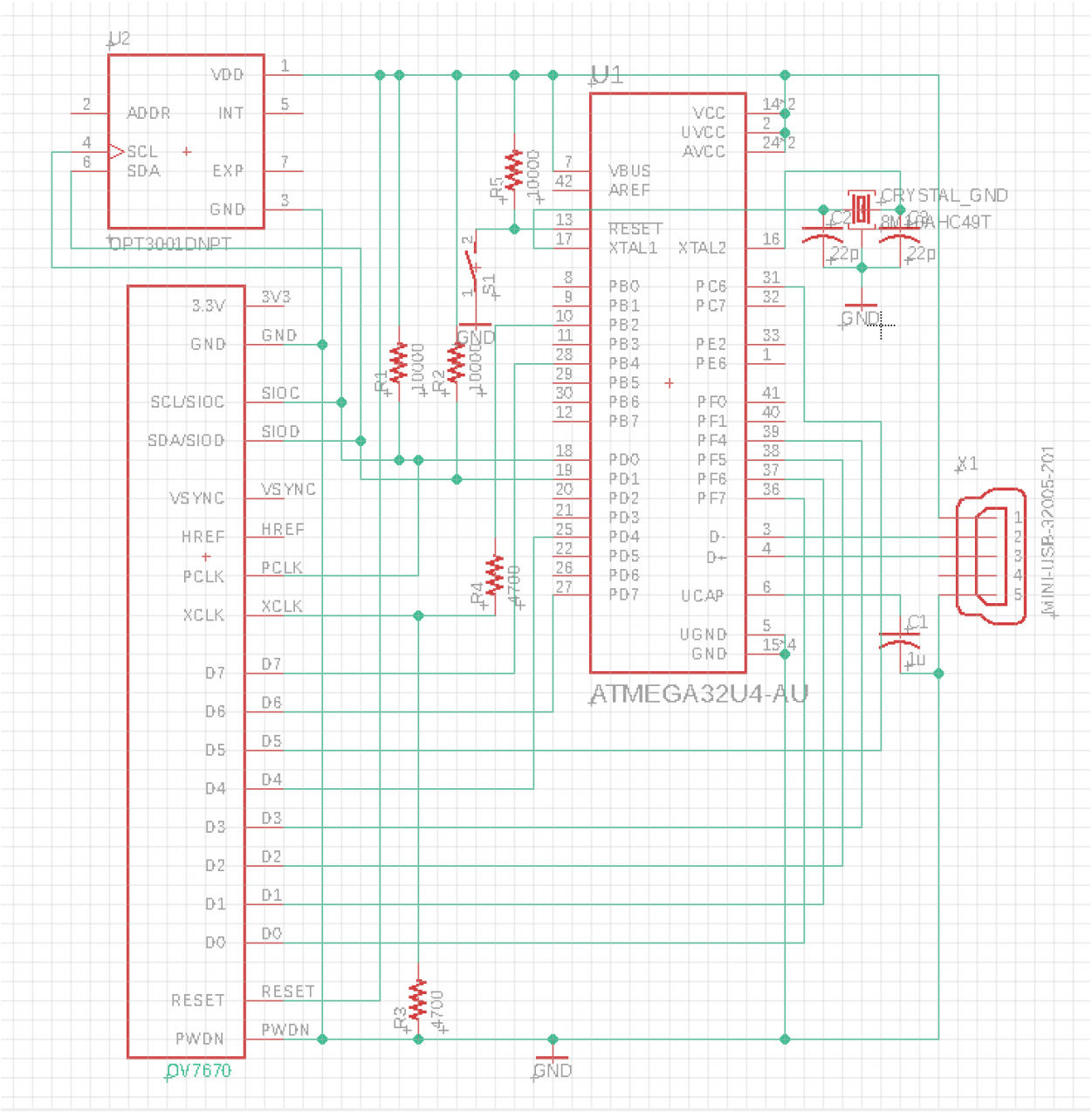
This software will be a C++ application running on the device whose displays are to be adjusted.

Given the preprocessed images, the software will perform a Haar cascade on the image to determine the presence of a face, and then given that cascade, determine the presence of open or closed eyes.

Using the Win32 C++ API, it will then take the blink rates and adjust display brightness, contrast, display area size, and/or color temperature accordingly.

Requirement	**Verification**
1. The blink detector must be able to recognize if there is an open or closed eye in the frame at least 75% of the time if the user is looking at the monitor. 2. The blink detector must be able to run detection on the images at a rate of 10 frames per second.	1. Create a set of images where the contents are known, with the resolution of our camera, and run it through the detection. At least 75% of the images should be correctly categorized as having an open eye, closed eye, or no eyes. 2. In the same way as testing the microcontroller's pre-processing, continuously run the program and timestamp when each frame is finished processing. The time between each timestamp should be less than 100ms.

2.5 Circuit Schematics



2.6 Tolerance Analysis

We will need to perform tolerance analysis on the built-in image settings of the camera. The subsampling rate, color balance, color mode, and cropping of the camera directly impact the data transfer rate between all the modules, and the effectiveness of the eye detection.

Extracting the eyes as face features and detecting their state can be achieved in multiple ways. The technique we will consider is the use of Haar cascades in face detection and eye detection.

At its default 640 x 480 resolution, OVM7690 camera provides 30fps VGA output [6]. Haar cascades will need the image to be converted to grayscale, and then an integral image created. We will have two 30fps frames' worth of time between each frame in our target frame rate of 10fps. Assuming a constant capture time derived from the 30fps baseline, this gives us 66.6 ms between each frame to perform our operations, send the image to the USB, and retrieve the next frame.

With respect to the efficiency of the actual detection - Computing a two-feature classifier amounts to 60 microprocessor instructions, given that an integral image is computed [4]. OpenCV provides pre-trained Haar cascades for eyes and faces. While we may need to train our own to fit our camera better, these are a useful benchmark. Their frontal face cascade has ~2000 features, while their eye cascade has ~3000 features. These features are arranged as a tree for efficiency, but in the worst case we should assume we might need to compute results for all of these features, ending in about 300,000 instructions. Assuming the Arduino's 16MHz speed, this takes ~19 ms to execute for one image. If we are running the detection on the target device rather than the microcontroller, we can ignore this execution time entirely, but it is well within reason to execute these functions entirely on the microcontroller.

Conversion to grayscale can be achieved trivially in very few instructions by adding the R, G, B signals and normalizing the result. There exists a study on efficiently computing the integral image. For a resolution of 720 x 576, they find a serial FPGA implementation takes ~3ms to compute the result [6].

3. Cost Analysis

3.1 Labor Costs

For the labor costs, we researched what a computer engineering/ electrical engineering undergrad makes out of college. These are the most recent results available:

- Computer Engineer: $96,992 / year
- Electrical Engineer: $79,714 / year

The average for these majors is $88,353 / year which converts to an estimated hourly rate of $44.18 assuming 40 hrs/week and 50 weeks/year (to account for holidays). We estimate for this project, each of us will work 150 hours to complete it.

Thus, our total labor cost is estimated to be 3 people * 150 hours each * $44.18/hour * 2.5 = $49,702.

3.2 Component Costs

Part	Cost	Quantity	Total Cost	Source
Microcontroller (Arduino ATMEGA32U4)	$22	3	$66	
PCB (Custom Design)	$5	2	$10	
Camera Sensor (OV7670 - subject to change, available for checkout from ECE445 inventory)	$9.06	2	$18.12	[8]
Ambient Light Sensor (OPT3001)	$2.93	2	$5.86	[9]
USB Power Supply (Already owned and interchangeable)	$0	1	$0	
I/O Cables,	$15	1	$15	

Resistors, Wires, and Other Accessories				
Total	$53.99	N/A	$144.98	

3.3 Total Cost

Adding up our labor costs with the component costs yields an estimated total cost for this project to be as such:

$49,702 + $111.82 = **$49,846.98**

4. Schedule

Week	Bryan	Raghav	Peter
3/8 (Revisions to machine shop due)	Design review Choose and order parts	Design review Choose and order parts	Design review Choose and order parts
3/15 (First round of PCBWay orders)	Program basic functionality for detection and display adjustment	Research into different eye detection techniques and script/test them for accuracy	Initial design and order PCB design for detection module
3/22 (Second round of PCBWay orders)	Program display adjustment logic and interface	Program best detection technique into software pipeline and perform verification	Conduct testing for initial PCB and order design revision for second order
3/29	Reconfigure detection software to fit hardware	Program the microcontroller	Conduct testing for revised PCB
4/5 (Third round of PCBWay orders)	Verification on software requirements	Verification on microcontroller requirements	Final revision for PCB design
4/12 (Mock demo)	Mock demo, debug	Mock demo, debug	Mock demo, debug
4/26 (Demonstration week)	Demo	Demo	Demo
5/3 (Presentation week)	Presentation	Presentation	Presentation

5. Ethics and Safety

One consideration to be made is the security of the eye tracking information we collect. Given Principle 1.6 of the ACM Code of Ethics, we should respect the privacy of the user, and keep this data limited in its scope [3]. Because we are gathering image data from the user, we need to make sure that the data is only used for the purpose of checking the user's eye-strain condition and the surrounding environment. This data should be processed immediately, exclusively for our project's purpose, and immediately disposed of when no longer needed.

Another concern is adjusting the brightness of the system too rapidly. As we are adjusting the monitor settings according to the data from the sensors, there is a possibility that the brightness setting could be changed in a harmful manner. In accordance with the IEEE Code of Ethics #9, we must avoid injuring others [4]. Thus we need to ensure that the system is not too reactive, as doing so risks changing the brightness too quickly, potentially damaging the system and posing an epilepsy risk. Thus we need to limit the rate of brightness change for the system itself. We plan to limit the adjustment interval to 5-10 minutes, and limit the amount to about a 10-20% change in brightness.

6. Citations

[1] Coles-Brennan in Clinical and Experimental Optometry, 'Management of digital eye strain', 2019. [Online].
Available:
https://onlinelibrary.wiley.com/doi/full/10.1111/cxo.12798. [Accessed: 17-Feb-2021]
[2] A. Jain, 'Eye Strain Detection', 2020. [Online].
Available: https://github.com/Anukriti2512/Eye-Strain-Detection [Accessed: 17-Feb-2021]
[3] ACM, 'ACM Code of Ethics and Professional Conduct', 2018. [Online].
Available: https://www.acm.org/code-of-ethics [Accessed 18-Feb-2021]
[4] IEEE, 'IEEE Code of Ethics', 2021. [Online] Available:
https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 18-Feb-2021]
[5] P. Viola, M. Jones, 'Rapid Object Detection Using a Boosted Cascade of Simple Features,' Mitsubishi Electric Research Laboratories, 2004. [Online] Available:
https://www.merl.com/publications/docs/TR2004-043.pdf [Accessed 1-March-2021]
[6] S. Ehsan, A. Clark, N. ur Rehman, K. McDonald-Maier, 'Integral Images: Efficient Algorithms for Their Computation and Storage in Resource-Constrained Embedded Vision Systems', Sensors 2015 15, 2015. [Online] Available:
https://arxiv.org/ftp/arxiv/papers/1510/1510.05138.pdf. [Accessed 1-March-2021]
[7] Texas Instruments, 'OPT3001 Ambient Light Sensor (ALS)', 2017. [Online] Available:
https://www.ti.com/lit/ds/symlink/opt3001.pdf?ts=1614814503843&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FOPT3001 [Accessed 3-March-2021]
[8] Amazon.com, 'OV7670 Camera Module Supports VGA CIF auto Exposure Control Display Active Size 640X480', 2021. [Online] Available:

https://www.amazon.com/Supports-Exposure-Control-Display-640X480/dp/B07VKFJM1Z [Accessed 4-March-2021]
[9] Alibaba.com, ‘CJMCU-3001 OPT3001 1.6V-3.6V Ambient Light Sensor The Human Eye As A Single Chip Measuring Light Intensity Luxmeter Board Module’, 2021. [Online] Available: https://www.alibaba.com/pla/CJMCU-3001-OPT3001-16V-36V-Ambient-Light-Sensor_62353565429.html?mark=google_shopping&biz=pla&pcy=US