

March 26th, 2021
Dr. Shervin Jennesar, Dr Craig Scratchley, Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, V5A 1S6



Borealis Systems

RE: ENSC 405W/440 Design Specification for Aureora Lights by Borealis Systems Inc.

Dear Dr. Jennesar, Dr. Scratchley, Dr. Rawicz,

This document represents the design specification for the Aureora Lights bias lighting kit. Bias lighting technology emits lights from the back of a monitor to extend the color luminescence of a video, creating a pleasing atmospheric glow for the user. Currently, there are no bias lighting kits on the market for desktop users, nor are there any that feature adjustable light strips for any monitor size. In order to reach a larger bias lighting market, Aureora Lights aims to provide customers with a superior option, with adjustable light spacing and device compatibility in mind. With Aureora Lights, we plan to build the most versatile bias lighting kit on the market.

The design specification documentation describes the different design considerations of the Aureora Lights bias lighting kit and to meet the corresponding requirements specifications. Please note that based on additional research and group decisions, the requirement specification has been revised. Therefore, a summary of the revised requirement specifications is included in Appendix C for reference.

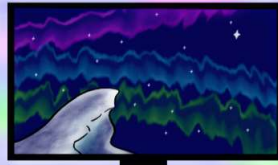
Our team hopes that this document provides to you a clear overview of the product. If you have any additional questions, please feel free to send an email to Justine Kwan, justinek@sfu.ca, the Chief Communications Officer.

Sincerely,

A handwritten signature in black ink that reads "Justine Kwan". The signature is written in a cursive, flowing style.

Justine Kwan
Chief Communications Officer
Borealis Systems Inc.

AUROREA



LIGHTS

Design Specifications for

AUROREA
LIGHTS

By:



Borealis Systems

Team:

Ian Carlsen
Justine Kwan
Daiki Miyanabe
Ryan Kiew

Submitted to:

Dr. Andrew Rawicz
Dr. Craig Scratchley
Dr. Shervin Jennessar

School of Engineering Sciences
Simon Fraser University

Issue Date:

March 26th, 2021

ABSTRACT

This document outlines the design requirement specifications for Aurorea Lights – a bias lighting kit designed to fit most display screen sizes. People who are looking to install bias lighting are left to either buy an entirely new display with bias lighting built into the system, or to build a Do-It Yourself (DIY) version for themselves. We aim to fill this market by providing a bias lighting kit with separable, adjustable modules – thereby allowing users to experience bias lighting on any display of their choice.

The basic system of Aurorea Lights consists of a web server, a Raspberry Pi for video processing, and the LED modules themselves. The web server is implemented using Node JS and web interface via browser the primary interface for users. The Raspberry Pi processes the video stream from a HDMI input using a color averaging algorithm. The LED modules are designed to be easily separable with the use of screws, while maintaining electrical connectivity between the LEDs. The appendices describe the user interface and appearance, supporting test plans, revised requirement specifications, and alternative design options.

CONTENTS

1	Introduction	1
1.1	System Overview.....	1
1.2	Challenges	2
1.3	Document Outline.....	2
2	Hardware/Electrical Design.....	3
2.1	Hardware	3
2.1.1	LEDs.....	3
2.1.2	HDMI Splitter	3
2.1.3	Raspberry Pi	4
2.1.4	Light Diffusion	4
2.2	Electrical Design	5
2.2.1	Power Design	5
2.2.2	Wiring Design	6
3	Physical Design and Module Structure	7
4	User Interface.....	9
4.1	Web User Interface	9
4.1.1	Backend Web Server Software	9
4.1.2	Frontend Software	10
5	Back-end Software Design	11
5.1	Video Processing.....	11
5.1.1	Software Coding Language	11
5.1.2	Supported Video Resolution	11
5.1.3	Capturing HDMI Input.....	11
5.1.4	Processing Border Colors	12
5.2	LED Driver.....	13
5.2.1	LED Driver Library.....	13
5.2.2	LED Refresh Rate.....	13
6	Conclusion.....	13
7	Bibliography	14
Appendix A: User Interface and Appearance.....		15
1	Introduction	15
2	Designs	16
2.1	Software Interface	16

2.2	Module Structure.....	16
3	User Analysis	19
3.1	Software Set Up	19
3.2	Hardware/Electrical Set Up.....	19
4	Technical Analysis.....	19
4.1	Discoverability.....	19
4.2	Feedback	19
4.3	Conceptual models	20
4.4	Affordances.....	20
4.5	Signifiers.....	20
4.6	Mappings.....	21
4.7	Constraints.....	21
5	Engineering Standards	22
6	Usability Testing	23
6.1	Analytical Usability Testing	23
6.1.1	Software	23
6.1.2	Modules	23
6.1.3	Clamping	23
6.2	Empirical Usability Testing	23
7	Conclusion.....	24
8	References.....	25
Appendix B: Supporting Test Plans		26
1	Physical Appearance and Structure	26
2	Hardware/Electrical	27
3	User Interface.....	28
5	Back-end Software Design	30
Appendix C: Requirement Specifications.....		32
1	General Requirements	32
2	Physical Appearance and Structure Requirements.....	32
2.1	General Requirements	32
2.2	Module Requirements	32
3	Hardware/Electrical Requirements.....	32
4	User Interface.....	33
4.1.1	General Requirements	33

4.1.2	Performance Requirements.....	33
5	Video Processing	33
5.1.1	General Requirements	33
5.1.2	Video Processing Requirements	34
5.1.3	Performance Requirements.....	34
6	Sustainability	34
7	Safety.....	34
8	Engineering Standards	35
9	References.....	35
Appendix D: Alternative Design Options		36
1	LEDs:	36
2	Securing Modules Together	36
3	Electric Connectivity Between Modules:	36
4	User Interface Method:.....	37
5	Webserver Software:	37
6	Back-end Software Design	38
6.1	Video Processing.....	38
6.1.1	Software Coding Language	38
6.1.2	Supported Video Resolution	38
6.1.3	Capturing HDMI Input.....	38
6.1.4	Processing Border Colors	38
6.2	LED Driver.....	39
6.2.1	LED Driver Library.....	39
6.2.2	LED Refresh Rate.....	39

LIST OF FIGURES:

Figure 1.1: Hardware System Overview.....	1
Figure 1.2: Software Design Diagram.....	2
Figure 2.1: Adafruit DotStar LED (scale: 1 mm)	3
Figure 2.2: HDMI Splitter	4
Figure 2.3: Power	5
Figure 2.4: Fuses	6
Figure 2.5: Wiring Diagram to safely wire the LEDs.....	6
Figure 3.1: Molex connector with pitch of 2.54 mm	7
Figure 3.2: Screw.....	7
Figure 3.3: MIT Snap-Fit Design Guide	8
Figure 3.4: Internal LED module structure.....	8
Figure 5.1 - Algorithm for processing border colors.....	12
Appendix A	
Figure 2.1: Interface Design for Desktop Browser.....	16
Figure 2.2: Interface Design for Mobile Browser.....	16
Figure 2.3: LEDs, spacer, and corner modules.....	17
Figure 2.4: Electrical contact for modules	17
Figure 2.5: Clamping mechanism.....	17
Figure 2.6: LED and spacer module.....	17
Figure 2.7: Clamping mechanism.....	18
Figure 2.8: Internal spring mechanism	18
Figure 2.9: Attaching the structure to a monitor.....	18
Figure 4.1: Example of Toast Message.....	20
Figure 4.2: Module connector mock-up in all configurations.....	21

LIST OF TABLES:

Table 1.3.1: Design Requirements Document Outline	3
Table 1.3.2: Requirement Specification Notation.....	3
Table 2.2.1: Electrical Requirements	5
Table 2.2.2: Design Requirements for power and wiring	7
Table 2.2.1: Design Requirement for Module Structure.....	8
Table 4.1.1: Design Requirement for Web Server Specification.....	9
Table 4.1.2: Design Requirements for the Front-end Software.....	11
Table 5.1.1: Design Requirements for Video Processing	12
Table 5.2.1: Design Requirements for the LED Driver	13
Appendix A	
Table 4.7.1: Engineering standards and best practices interface and appearance design.....	22
Table 6.2.1: General System Requirements Specifications	32


GLOSSARY

Term	Description
Bias Lighting	Lighting on the back of a display screen that illuminates the surface behind.
LED	Light Emitted Diodes
FPGA	Fully Programmable Gate Array

REVISION HISTORY

Version	Implemented by	Revision Date	Approved by	Approval Date	Reason
1.0	Daiki, Ian, Justine	03/21/21	Justine	03/21/21	Initial User Interface and Appearance Appendix Draft
1.1	Daiki, Ian, Justine, Ryan	03/26/21	Justine	03/26/21	Initial Design Specification Draft Revised User Interface and Appearance Appendix

PRODUCT DESIGN SPECIFICATION APPROVAL

Signature:		Date:	March 26 th 2021
Print Name:	Justine Kwan		
Title:	Chief communications officer		
Role:	Project overseer		

1 INTRODUCTION

Aurorea Lights is a bias lighting kit that can fit a wide variety of display sizes and devices including computer monitors, laptops, and televisions. This product will process video data and use that data to project different colors of light that will match up with the localized border color of the video. Currently, there are no bias lighting kits on the market for desktop users, nor are there any that feature adjustable light strips for arbitrary monitor size or orientations. Therefore, to reach a large bias lighting market, Aurorea Lights aims to increase options for compatibility with adjustable light spacing and device compatibility. With Aurorea Lights, we plan to build the most versatile bias lighting kit.

1.1 SYSTEM OVERVIEW

The software component of the kit requires the ability to analyse the video signal and a user interface. The hardware component connects the unit's sub-devices, electrical requirements as well as incorporates the appearance design. To allow the user to fit the product around any monitor size, the hardware and appearance of the LEDs was given modular design which allows the user to choose their lighting arrangement for the monitor.

Figure 1.1 represents an overview of the hardware system as well as the connected sub-devices. The sub-components are a power source, a HDMI splitter, a Raspberry Pi, and the LEDs. The HDMI splitter takes the video source and outputs it to the Raspberry Pi as well as to the user's monitor. The Raspberry Pi will analyse the video signal and send information to the LEDs.

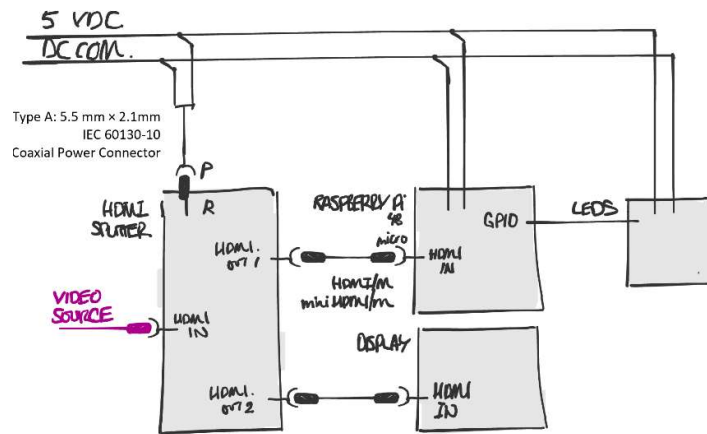


Figure 1.1: Hardware System Overview

The Aurorea Lights kit is made up of three main software components as shown in Figure 1.2. The first software component is the web server user interface, from which the user will be able to access the settings for the device. The second software component is responsible for capturing the video stream from the HDMI source and processing the border colors for the LEDs. The third software component is the LED driver, which awaits input and changes the colors of the LEDs.

The user interface is a web-based program that can be opened from browsers of computer or mobile device. End users can change various settings such as brightness and color saturation of the LEDs, and mode of the lights from the user interface. The web-based program consists of web server and frontend program. The webserver runs on the Raspberry Pi and works as a bridge between the

primary software component and user’s browser. Moreover, the frontend program is a web page that runs on user’s web browser.

The user’s video signal will be processed by a frame-by-frame basis, using an algorithm that first compresses the image, then splits the image into squares depending on the number of LEDs being used, and finally averaging the color of each of the squares. The LEDs are then updated to match their corresponding colors.

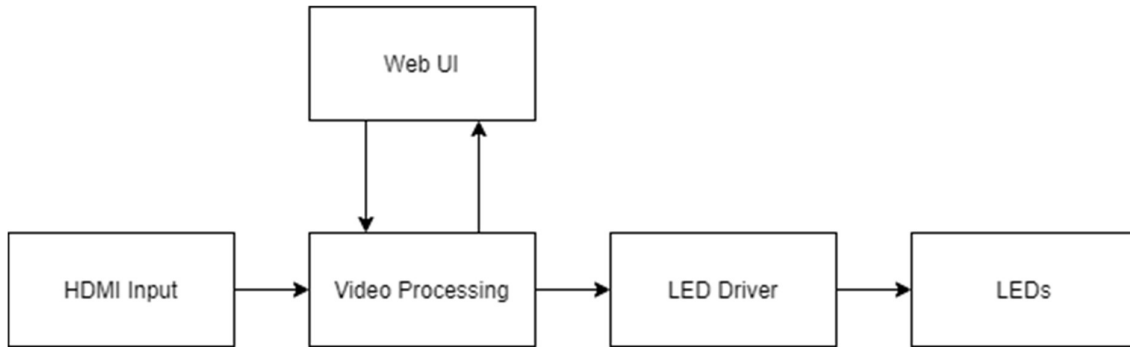


Figure 1.2: Software Design Diagram

1.2 CHALLENGES

For the proof-of-concept, the main challenges for each category is as follows: electrical challenges include maintaining wiring connectivity and ensuring that the system has adequate power throughout. Module design challenges include checking if everything fits together and if the design is compact. Software challenges include matching the video speed with the LED speed such that processing time is the primary concern.

1.3 DOCUMENT OUTLINE

The design specification document goes over the designs to fulfill the requirement specifications. Additional references are included in the appendix, which has four sections. Appendix A: User Interface and Appearance Design describe the software user interface and module structure designs. Appendix B: Supporting Test Plans lists testing procedures to test the design. Appendix C: Requirement Specifications provides updated requirement specification tables for reference. Appendix D contains explicit details on the alternative design choices considered.

Design specifications in this paper will be labelled using the following arrangement:

Des {Section Number}.{Subsection}.{Requirement Number}.{Development Stage}

Section Number	Heading
1	General
2	Hardware/Electrical
3	Physical Design and Module Structure
4	User Interface
5	Video Processing

6	Sustainability
7	Safety

Table 1.3.1: Design Requirements Document Outline

Development Stage	Description
C	Proof-of-concept prototype
E	Engineering prototype
F	Final Product

Table 1.3.2: Requirement Specification Notation

2 HARDWARE/ELECTRICAL DESIGN

2.1 HARDWARE

2.1.1 LEDs

To satisfy [Req 3.2.C], individually addressable LEDs from Adafruit are chosen. To satisfy [Req 2.2.3] and to keep it compact, the 3.5x3.5 mm NeoPixels and the 5x5 mm DotStar was considered, but the 5x5 mm DotStar was chosen ultimately chosen because the solder points is large enough in case it needs to be soldered by hand. The LEDs comes on a strip of PCBs with spacing options of 30 LEDs per meter, 60 LEDs per meter and 144 LEDs per meter. The 60 LEDs per meter option was chosen because it was compact but still large enough to solder with. The dimensioning of the LED is shown in Figure 2.1. The pitch between the pins of the LED is 2.54 mm when scaled.

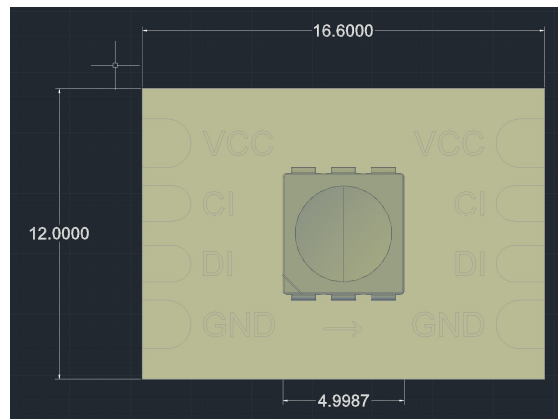


Figure 2.1: Adafruit DotStar LED (scale: 1 mm)

2.1.2 HDMI Splitter

A HDMI Splitter will be used to split the video HDMI signal into two outputs. The first output signal will be sent to the user's monitor to display the video. The second output signal will be sent to the Raspberry Pi for video processing.

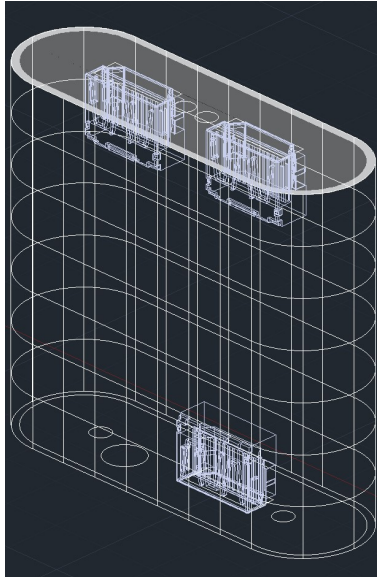


Figure 2.2: HDMI Splitter

2.1.3 Raspberry Pi

The Raspberry Pi will be used for processing the video and driving the LEDs. In previous iterations, the ESP32 microcontroller [1] was used, however it was not fast enough to process video signals at the required rate. This would cause a desynchronization issue which fails to comply with [Req 5.3.2.E]. An FPGA was also considered, but it would significantly increase the cost, making it unviable for commercial marketing.

2.1.4 Light Diffusion

Lights should be diffused as a safety requirement, to reduce the illuminance of the LEDs and prevent blinding. A light diffusion cover was considered, however, videos demonstrating how to encase LEDs into epoxy resin is done as cheaper do-it-yourself approach [2]. As a result, we decided to use this option instead.

Design Choice Summary

Design ID	Description	Corresponding requirement/design specifications(s)	Supplement Material
Des 2.1.1.C	Adafruit 5050 Dotstar LED will be used	[Req 3.2.C]	N/A
Des 2.1.2.C	HDMI Splitter will be used to split the HDMI signal	N/A	Figure 2.2
Des 2.1.3.C	Raspberry Pi will be used to process the video	[Req 1.1.C]	N/A
Des 2.1.4.F	Light diffusion will be done by resin	[Req 8.3.F]	N/A

2.2 ELECTRICAL DESIGN

2.2.1 Power Design

To satisfy [Req 1.3.C] and [Req 3.1.C], a power supply is needed to power all of the sub-devices. Table 2.2.1 shows the electrical requirements for each sub-device and Figure 2.4 shows the wiring diagram for the power supply. The ATX-12V power supply can be paired with ATX breakout board interface to supply 5V of power to each device. The breakout board can allow a current draw of up to 24A. For safety reasons, 30 A fuses will be placed as a safety requirement as shown in Figure 2.4.

Product	Current Capacity (A)	Voltage requirements (V)
Raspberry Pi 4 Model B	3.00	5
HDMI Splitter	1.00	5
LEDs	0.06 per LED	5

Table 2.2.1: Electrical Requirements

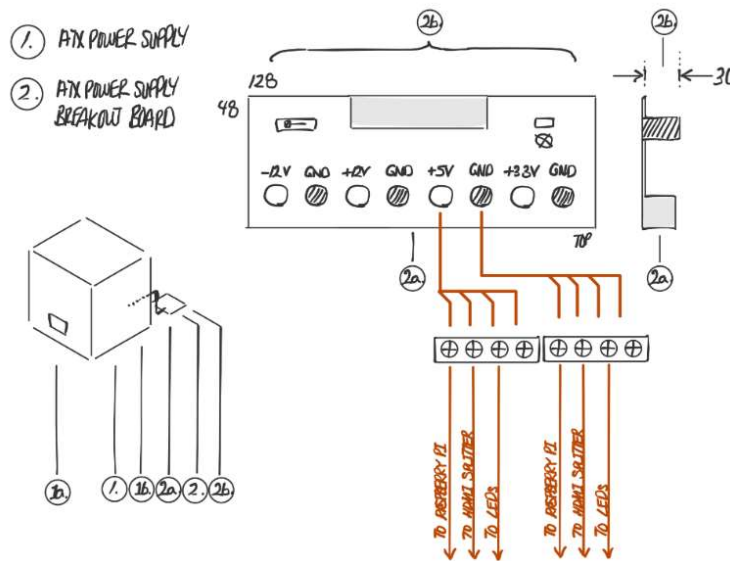


Figure 2.3: Power

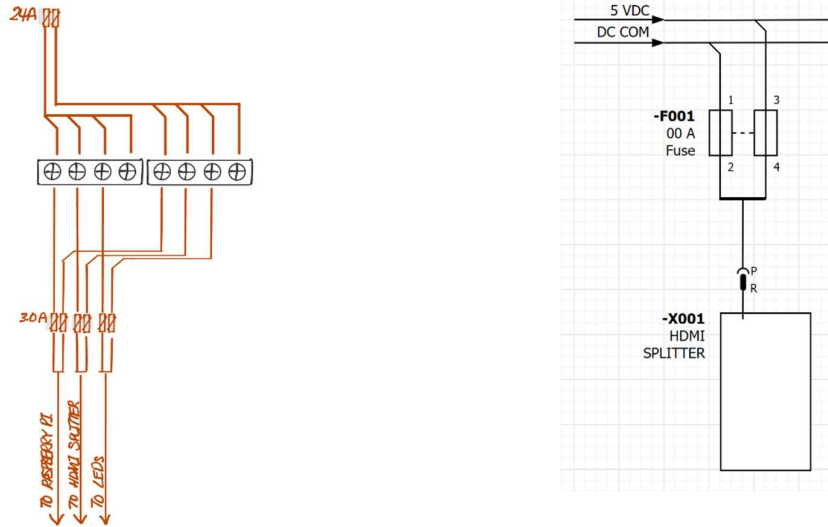


Figure 2.4: Fuses

2.2.2 Wiring Design

Adafruit provides the recommended wiring configurations for the NeoPixels and the DotStars LEDs. They recommend a 1000 μF capacitor before the 5V terminal and a 470 Ω resistor before the Data-In pin to protect the LEDs from current spikes [3]. This diagram is shown in Figure 2.5.

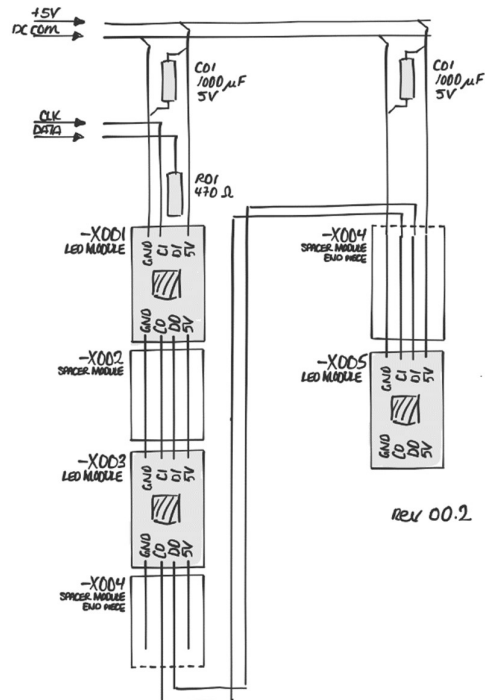


Figure 2.5: Wiring Diagram to safely wire the LEDs

Design Choice Summary

Design ID	Description	Corresponding Requirement/Design specifications(s)	Supplement Material
Des 2.2.1.C	The ATX power supply will be able to power all the components and turn the device on and off.	Req 1.3.C Req 3.1.C	N/A
Des 2.2.3.F	Enclosed wires will connect between the LEDs and the Raspberry Pi	Req 2.2.1.C Req 3.3.C Req 3.4.C	Figure 2.5

Table 2.2.2: Design Requirements for power and wiring

3 PHYSICAL DESIGN AND MODULE STRUCTURE

Modules will be modeled in AutoCAD and 3D printed to a specific sizing, to keep the design small and compact. To connect the modules as per [Req 2.2.1.C], compression connectors were considered due to their compact design. The 3D printed housing will then need to be carefully sized. However, for the proof-of-concept, a Molex 2.54 mm header will be used to demonstrate functionality.

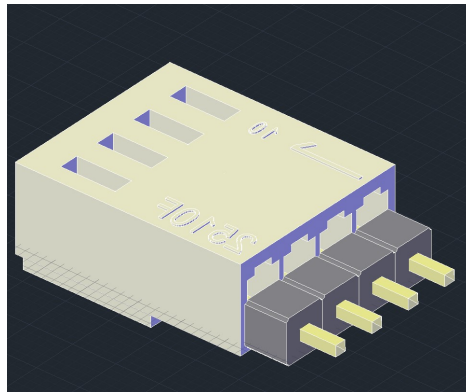


Figure 3.1: Molex connector with pitch of 2.54 mm

To satisfy [Req 2.2.2.C] and further secure modules together, the two methods considered was snap-fit design and screw-in designs. We would like to implement snap-fit using guide as shown in Figure 3.3, however snap-fit design require precision with 3D printing. Therefore, for the proof-of-concept screws are chosen as reliable securing method. M3 x 3mm screws are chosen for due to their small size. Heat sink threads are chosen for the 3D printed modules because they can be directly embedded in the plastic for ease of use, thus satisfying [Req 2.1.2.C].

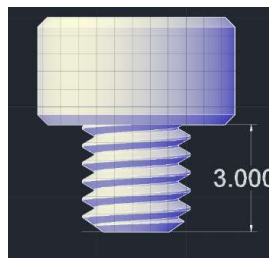


Figure 3.2: Screw

Shape of the cross section Type of design	A Rectangle 	B Trapezoid 	C Ring segment 	D Irregular cross section
1 Cross section constant Over the length	$y = 0.67 \cdot \frac{\epsilon \cdot l^2}{h}$	$y = \frac{a + b_{(1)}}{2a + b} \cdot \frac{\epsilon \cdot l^2}{h}$	$y = K_{(2)} \cdot \frac{\epsilon \cdot l^2}{r_2}$	$y = \frac{1}{3} \cdot \frac{\epsilon \cdot l^2}{c_{(2)}}$
2 All dimensions in direction y, e.g., h or λ , decrease to One-half	$y = 1.09 \cdot \frac{\epsilon \cdot l^2}{h}$	$y = 1.64 \cdot \frac{a + b_{(1)}}{2a + b} \cdot \frac{\epsilon \cdot l^2}{h}$	$y = 1.64 \cdot K_{(2)} \cdot \frac{\epsilon \cdot l^2}{r_2}$	$y = 0.55 \cdot \frac{\epsilon \cdot l^2}{c_2}$
3 All dimensions in direction z, e.g., b and a, decrease to one-quarter	$y = 0.86 \cdot \frac{\epsilon \cdot l^2}{h}$	$y = 1.28 \cdot \frac{a + b_{(1)}}{2a + b} \cdot \frac{\epsilon \cdot l^2}{h}$	$y = 1.28 \cdot K_{(2)} \cdot \frac{\epsilon \cdot l^2}{r_2}$	$y = 0.43 \cdot \frac{\epsilon \cdot l^2}{c_{(2)}}$
1,2,3 	$P = \frac{bh^2}{6} \cdot \frac{E_s \epsilon}{l}$	$P = \frac{h^2}{12} \cdot \frac{a^2 + 4ab_{(1)} + b^2}{2a + b} \cdot \frac{E_s \epsilon}{l}$	$P = Z_{(4)} \cdot \frac{E_s \epsilon}{l}$	$P = Z_{(4)} \cdot \frac{E_s \epsilon}{l}$

Subscript numbers in parenthesis designate the note to refer to.

Figure 3.3: MIT Snap-Fit Design Guide [4]

The internal structure of the modules is shown in Figure 3.4. The wires are bent inwards into the modules to provide enough wire for wiring ease while maintaining compactness to satisfy [Req 2.2.1.C]. To make sure the wires do not cross, position of the connectors will be offset.

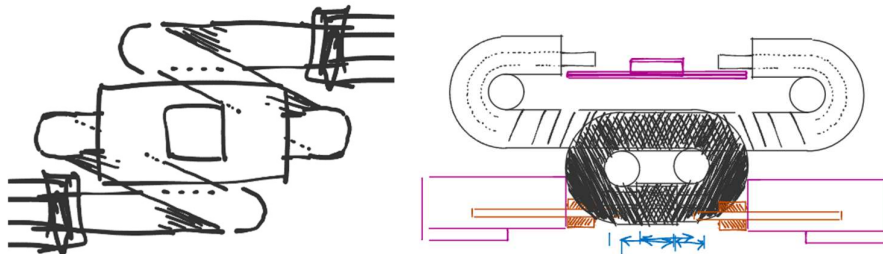


Figure 3.4: Internal LED module structure

Further analysis regarding the module design described in depth in Appendix A: User Interface and Appearance designs.

Design Choice Summary

Design ID	Description	Corresponding requirement/design specifications(s)	Supplement Material
Des 3.1.C	Modules will be modeled in AutoCAD and 3D printed	[Req 2.1.3.E] [Req 2.2.3.E]	N/A
Des 3.2.C	Internal wires will ensure connectivity between the LEDs	[Req 2.2.1.E]	N/A
Des 3.3.F	A clip-on device taken from a selfie stick will secure the modules onto a monitor	[Req 2.1.1.C]	Figure 2.7 Figure 2.8
Des 3.4.C	System will use common connection methods (e.g. screws, plug-in, snap-in)	[Req 2.1.2.C]	N/A
Des 3.5.C	Screws will secure modules together with heat-set insert threads	[Req 2.2.2.E]	N/A

Table 2.2.1: Design Requirement for Module Structure

4 USER INTERFACE

4.1 WEB USER INTERFACE

The user interface for the Aurora Lights is a web-based interface that can be accessed from a web browser of a user's computer or mobile device. To achieve this, the product requires the use of a web server software that can process the requests from the client's device and display a user-friendly front-end web design.

We have considered multiple options for the user interface such as having a small touch screen that can be attached to the monitor or TV and control the settings and using a Bluetooth connection to control from the mobile device or computer. However, we realized that with the touchscreen method, the user will have difficulties changing device settings if the monitor or TV is mounted to a wall or TV, which is otherwise unreachable to the user. Additionally, utilizing the Bluetooth method requires an application to be installed onto the user's computer or mobile device and that needs to be paired with, every time the user wants to change the settings. On the other hand, a web user interface can be accessed from anywhere as long as the device is connected to the same Wi-Fi network, and doesn't require the user to install the application to their computer or mobile device. For these reasons, therefore it was chosen as our new user interface method.

4.1.1 Backend Web Server Software

Web server software is the bridge between the main program which controls the LEDs and the front-end software. The server backend will process the requests from the client's frontend and then update the corresponding settings such as brightness of the light and as well as different lighting display modes. The server backend also replies to the front-end program with any information that is requested.

To communicate between the web server and the frontend program, we have decided to use WebSocket over HTTP because of its fast reaction time. Rapid response time is required to update the setting for the LEDs quickly and to let the user see the result of the selected setting. For the web server software, we have considered Apache, Nginx as well as Node.js and decided to utilize Node.js due to its compatibility with the WebSocket.

Design Choice Summary

Design ID	Design Specification Requirements	Corresponding requirement/design specifications(s)	Supplement Material
Des 4.1.1.1.C	Web server shall communicate with the main program to get or change the setting for the lights.	[Req 4.1.2.C] [Req 4.1.3.C] [Req 5.1.3.E]	N/A
Des 4.1.1.2.C	Web server shall communicate with front-end program via Wi-fi network.	N/A	N/A
Des 4.1.1.3.C	Node JS will be used for web server.	N/A	Appendix B: Webserver Software
Des 4.1.1.4.C	WebSocket will be used to communicate with the frontend software.	N/A	N/A

Table 4.1.1: Design Requirement for Web Server Specification

4.1.2 Frontend Software

To comply with [Req 4.1.1.C], a web-based frontend software will be running on the user’s browser to control some setting for the Aurora Lights such as brightness of the LEDs and mode of the lights.

It is essential that the user will have the ability to control different settings from the user interface such as brightness, color saturation and the lighting mode are essential for the user to control from the user interface also specified by the Requirement Specification. Other functionalities such as mode where LEDs changes color over time and detailed customization can be added in future software revisions. The frontend user interface components are designed using the “Seven Elements of UI Interaction” from Don Norman’s “The design of Everyday Things.” For detailed visual user interface design, see the Appendix A: User Interface and Appearance document.

Design Choice Summary

Design ID	Design Specification Requirements	Corresponding requirement/design specifications(s)	Supplement Material
Des 4.1.2.1.C	Application shall display the status of the LEDs. (On/Off)	N/A	Appendix A Figure 4.1
Des 4.1.2.2.C	Application shall be able to change the status of the LEDs. (On/Off)	[Req 4.1.2.C]	Appendix A Figure 4.2
Des 4.1.2.2.C	Application shall display the brightness setting of the LEDs.	N/A	Appendix A Figure 4.3
Des 4.1.2.3.C	Application shall be able to have component to change the brightness setting of the LEDs.	[Req 4.1.2.C]	Appendix A Figure 4.4
Des 4.1.2.4.E	Application shall display the color saturation of the LEDs.	N/A	Appendix A Figure 4.5
Des 4.1.2.5.E	Application shall be able to have component to change the color saturation of the LEDs.	[Req 4.1.3.E]	Appendix A Figure 4.6
Des 4.1.2.6.C	Application shall display current mode for the LEDs.	N/A	Appendix A Figure 4.7
Des 4.1.2.7.C	Application shall be able to change the modes for lighting from the following: -Bias Lighting: LEDs match the colors on the screen -Uniform Lighting: LEDs set to a single color	N/A	Appendix A Figure 4.8
Des 4.1.2.8.C	When the application is in uniform lighting, the application shall display color picker to change the color of the LEDs.	N/A	N/A
Des 4.1.2.9.C	Application shall notify user with toast message when user changes any configuration.	N/A	Appendix A Figure 4.9

Des 4.1.2.10.C	Application shall notify user with toast message when there is an error.	[Req 4.1.5.C]	Appendix A Figure 4.10
Des 4.1.2.11.C	User interface will be accessed from web browser.	N/A	N/A

Table 4.1.2: Design Requirements for the Front-end Software

5 BACK-END SOFTWARE DESIGN

5.1 VIDEO PROCESSING

5.1.1 Software Coding Language

Python was chosen as the primary coding language for video processing as it is a simple, high-level, object-oriented programming language that is relatively fast and is suitable for data analysis. Python is well-known for its syntax simplicity, which makes code written in Python easy to follow. It has a large assortment of libraries that we can use, and a library called “Python Imaging Library” [5] (which will be discussed later) has functions that do exactly what we need for this project.

Additionally, Python is touted as the preferred language for programming Raspberry Pi’s [6], which means that help is readily available online if we ever get into any issues while programming for the Raspberry Pi.

5.1.2 Supported Video Resolution

Aurora Lights will support an input resolution of up to 3840x2160@30Hz, but the software will only be processing the video at 1920x1080@30Hz. This is due to the hardware limitation of our HDMI video capture card. It is possible to process video of higher resolutions by implementing an on-board HDMI decoder. However, decoding and processing this higher quality video would increase the processing time exponentially, for a minimal improvement in results. The faster option is chosen to keep the LEDs synchronized with the video [Req 5.3.3.E].

5.1.3 Capturing HDMI Input

For capturing HDMI input, the Python Imaging Library [5] was chosen because it contains functions that significantly ease data analysis from the HDMI input. The function `imageGrab.grab()` [7] allows the conversion of a single frame of a video stream into an image, which is needed in the analysis of the display. This is required in accordance to [Req 5.1.2.C] – the software must be able to determine the colors of the LEDs depending on the display. Furthermore, there is also the function `img.thumbnail()` [8], which is a fast and efficient method of compressing an image. We need this speed to achieve Req 5.3.3.E – the software must change the colours of the LEDs synchronously with the display. Both OpenCV [9] and FFmpeg [10] are great libraries for video processing, but for a simple task of capturing and compressing a HDMI input, Pillow provides better performance.

5.1.4 Processing Border Colors

The algorithm we have chosen to process bias lighting colors is to average the colors over predetermined squares of an image. The image is first divided into squares depending on the number of LEDs being used. The LED colors are then determined by the average color of the corresponding border square. For example, **Error! Reference source not found.** displays the boxes that the video processing software will average if the user decides to use 20 LEDs on the top and bottom of the display, and 10 LEDs on the sides. This design choice satisfies [Req 5.2.3.C] – the software must use color averaging algorithms to determine the colors of the LEDs. It also performs faster than color extrapolation, which is essential to sync the colors of the LED to the display [Req 5.3.3.E]. Although color determination based on a relative pixel is faster, it lacks the averaging process, and thus the colors of the LEDs may be inconsistent with the display.



Figure 5.1 - Algorithm for processing border colors

Design Choice Summary

Design ID	Design Specification Requirements	Corresponding requirement/design specifications(s)	Supplement Material
Des 5.1.1.C	The video processing software will be coded using Python.	N/A	N/A
Des 5.1.2.C	The maximum supported input resolution is 3840x2160@30Hz, while the video processing software will process videos of up to 1920x1080@30Hz.	[Req 5.3.3.E]	N/A
Des 5.1.3.C	The video processing software will use Python Imaging Library (Pillow) to capture HDMI video input.	[Req 5.1.2.C]	N/A
Des 5.1.4.C	The video processing software will average colors over squares to process border colors.	[Req 5.2.3.C] [Req 5.3.3.E]	Figure 4.1

Table 5.1.1: Design Requirements for Video Processing

5.2 LED DRIVER

5.2.1 LED Driver Library

The FastLED library was chosen to drive the LEDs because due to its supports with the APA102 LED strip that we are using for the Aurora Lights bias lighting kit. The proper driving of LEDs are required to satisfy [Req 5.3.2E] & [Req 5.4.4.E]. The Adafruit DotStar library may also be used, but FastLED supports almost all commercially available LEDs, while the Adafruit DotStar library only supports Adafruit's DotStar line of LEDs. This versatility would make it easy to change the syntax to work with different LEDs, in the case that we need to change LEDs in the future.

5.2.2 LED Refresh Rate

Although the device can only process videos at a 1920x1080@30Hz, it can still process videos at a higher refresh rate but at a smaller resolution. To take this framerate into account [Req 5.2.4.F], the LEDs will be updated 60 times a second. Going any higher than 60 would simply increase processing time with minimal improvement. Going lower would cause an obvious discrepancy between video and LED output, which we would like to avoid [Req 5.3.4.E].

Design Choice Summary

Design ID	Design Specification Requirements	Corresponding requirement/design specifications(s)	Supplement Material
Des 5.2.1.C	The LEDs will be driven using the FastLED library.	[Req 5.3.2.E] [Req 5.4.4.E]	N/A
Des 5.2.2.C	The LEDs will be updated 60 times per second.	[Req 5.2.4.F] [Req 5.3.4.E]	N/A

Table 5.2.1: Design Requirements for the LED Driver

6 CONCLUSION

The chosen hardware design will utilize an HDMI splitter to send the video signal into the Raspberry Pi that will then send the color signal to the LEDs. The system will be powered by an ATX power supply with the ATX breakout board.

The system will use a modular structure. Screws will be used to connect the modules together and the 2.54 mm header connector will be used for connections between modules. A clamping mechanism will be used to secure the module structure onto the monitor.

The user interface is hosted by a webserver that can be accessed from a web browser on the same network as the Raspberry Pi. The user interface will provide status information as well as allow the user to adjust the brightness and saturation of the lights. The HDMI video input will be processed using the Python Image Library, and the LEDs will be driven using the FastLED library, updating at a rate of 60 times per second.

For the proof-of-concept, the power supply will generate power to the devices. The basic software functionalities that will be demonstrated is the color averaging for the LEDs, and the basic user interface. Module structures will be 3D printed with the LEDs, connectors, screws and wiring in place. To refine the product after the proof of concept, some methods may be swapped out for more refined methods. This includes snap-fit connectors and compression connectors.

7 BIBLIOGRAPHY

- [1] "ESP32," Espressif, [Online]. Available: <https://www.espressif.com/en/products/socs/esp32>. [Accessed 23 03 2021].
- [2] "Embedding LED Strip Lights in Epoxy Resin," [Online]. Available: <https://do-daddy.com/how-to-embed-led-strip-lights-in-epoxy-resin/>. [Accessed 26 03 2021].
- [3] "Powering NeoPixels," [Online]. Available: <https://learn.adafruit.com/adafruit-neopixel-uberguide/powering-neopixels>. [Accessed 4 04 2021].
- [4] "Snap-Fit Joints for Plastics," [Online]. Available: http://fab.cba.mit.edu/classes/S62.12/people/vernelle.noel/Plastic_Snap_fit_design.pdf. [Accessed 21 03 2021].
- [5] "Pillow," Alez Clark and Contributors, [Online]. Available: <https://pillow.readthedocs.io/en/stable/#>. [Accessed 23 03 2021].
- [6] "Raspberry Pi FAQs," Raspberry Pi Foundation, [Online]. Available: <https://www.raspberrypi.org/documentation/faqs/>. [Accessed 23 03 2021].
- [7] "ImageGrab Module," Pillow, [Online]. Available: <https://pillow.readthedocs.io/en/stable/reference/ImageGrab.html>. [Accessed 23 03 2021].
- [8] "Image Module," Pillow, [Online]. Available: <https://pillow.readthedocs.io/en/stable/reference/Image.html>. [Accessed 23 03 2021].
- [9] "OpenCV," [Online]. Available: <https://opencv.org/>. [Accessed 23 03 2021].
- [10] "FFmpeg," [Online]. Available: <https://www.ffmpeg.org/>. [Accessed 23 03 2021].

APPENDIX A: USER INTERFACE AND APPEARANCE

1 INTRODUCTION

This document outlines the user interface and appearance design of Aurorea Lights bias lighting kit by Borealis System Inc. The scope includes the software interface as well as the hardware structure of the bias lighting.

The software interface of the device allows for the user to directly control the lights. Users can change the settings of the lights by accessing a web browser through their wireless network. It features the ability to change the brightness, saturation, and the ability to select between the bias lighting or static lighting modes.

The hardware interface is integrated with the shape and appearance of the product. The kit's electrical components will be arranged and slotted together by the user using common mechanics (e.g., snap-in, screw-in, plug-in). Once the product is fully configured to the individual monitor size, the main hardware interface is the power button.

Technical analysis of the design shown, references the concepts described in *The Design of Everyday Things* by Donald Norman [2].

2 DESIGNS

2.1 SOFTWARE INTERFACE

Figure 2.1 and Figure 2.2 shows the interface designs for both desktop and mobile users. These interfaces can be accessed from a web browser and is used to control brightness, the color saturation of the lights, as well as the different modes of the product. The two modes are uniform lighting and bias lighting: uniform lighting has all the LEDs set to a single color, while bias lighting has the LEDs match the colors on the screen. The sliders for brightness and the color saturation allow the users to have intuitive control of both parameters with the icons on the sides that display the effect of the sliders. The same interface blocks are used between the desktop and mobile versions of the software so the user can use both interfaces interchangeably without confusion.

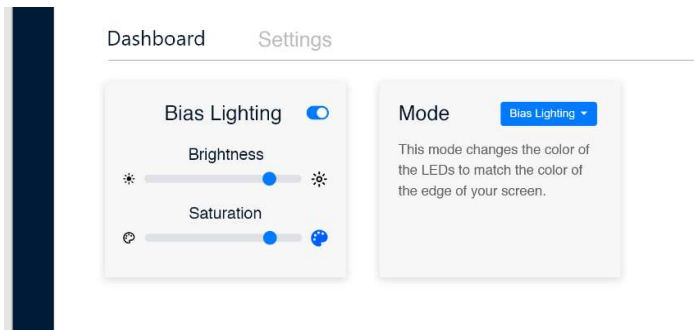


Figure 2.1: Interface Design for Desktop Browser

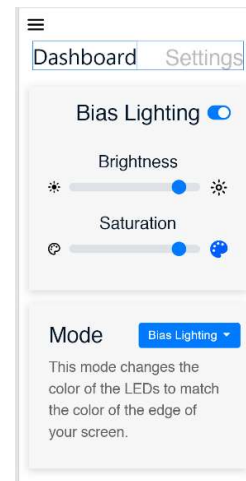


Figure 2.2: Interface Design for Mobile Browser

2.2 MODULE STRUCTURE

The considerations of the module structure include the arrangement of the modules, fitting of the modules, the connectivity of power to the LEDs, and methods to attach it to a monitor. Figure 2.3 shows the arrangement of the LED, spacer, and corner modules. The LED module will house the lights. The spacer modules will evenly space the structure and will have different sizing options to increase spacing combinations. The corner module will create the orthogonal bend at the corners. There are two proposed methods of implementing the corner modules: one method uses an angled spacer to create the corner bend, while the other method uses a dedicated corner LED module. The optimal corner module style has not been decided yet. When fully connected, the modules will make a rectangular structure. Figure 2.4 shows the electrical connectivity between modules and Figure 2.5 shows the clamping mechanism to attach it to the back of a monitor.

- ⑦ LED MODULE
- ② spacer module.
- ②④ small
- ②⑧ medium
- ②⑩ large.
- ③ corner LED MODULE
- ④ corner spacer module
- ⑤ solder points
- ⑥ pads
- ⑦ traces
- ⑧ connector
- ⑧① male
- ⑧② female
- ⑨ clip on

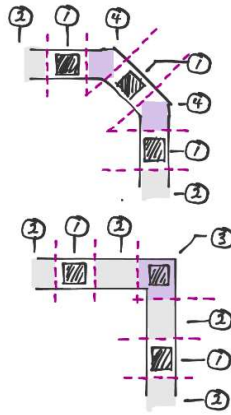


Figure 2.3: LEDs, spacer, and corner modules

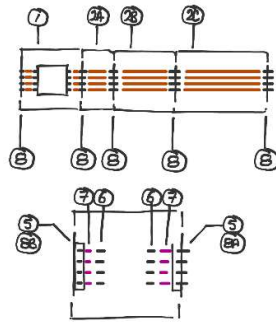


Figure 2.4: Electrical contact for modules

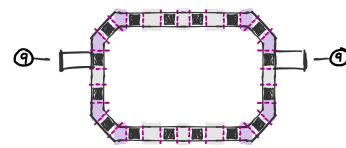


Figure 2.5: Clamping mechanism for modules

Figure 2.6 shows the proposed design for an individual LED and spacer module. For the LED module, a plastic enclosure made of plastic will encase the LEDs and the PCB board. Wires will be soldered to internally connect the PCB board to the external plug-in connectors to ensure electrical connectivity as shown in Figure 2.4.

The proposed method to connect the modules together is with screws. The shape of the connection side has orthogonal edges to provide alignment. The screws on both sides provide a strong, balanced connection, and prevents the electric connectors from being loosened. The holes for the screws will have threaded nuts embedded into the plastic by heat, reducing the number of parts for the user to potentially mishandle or lose.

For the proof-of-concept, the proposed design for securing the modules is with screws since they are reliable. However, another design consideration is using the torsion snap-fit implementation, which would make assembly faster. Torsion snap-fit is recommended for parts that may be disassembled and reassembled frequently [2]. However, concerns regarding the snap-fit approach include: finding the optimal way to keep the snap-fit design compact, avoiding strain on the plastic enclosure when connected, and keeping it durable and flexible for reattachability. If proven to be reliable, a snap-fit design will be shown in the final product.

- ① Enclosure (Top)
- ② Enclosure (Bottom)
- ③ Led
- ④ Window to expose LED
- ⑤ PCB Board
- ⑥ Connector
- ⑥① Male connector
- ⑥② Female connector
- ⑦ Wires (internal)
- ⑧ Screw hole (x4)
- ⑨ Snap fit mechanism to enclose top and bottom

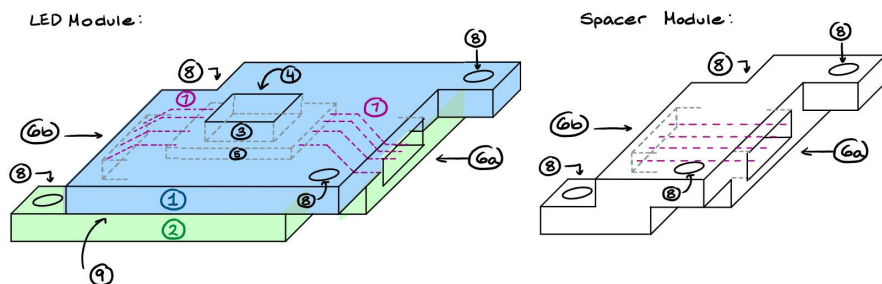


Figure 2.6: LED and spacer module

To attach the structure to the monitor, the kit will use a clamping mechanism. This is the same clamping method that holds phones in selfies sticks, which are cheap and easy to disassemble. The clamping mechanism is made by two sliding components as shown in Figure 2.7. The mechanism is extendable, but an internal spring pushes the two components together as shown in Figure 2.8.

There is also a rubber component that will contact monitor so that it can be held up by friction. The clamps will be attached to both sides of the monitor to hold up the structure.



Figure 2.7: Clamping mechanism

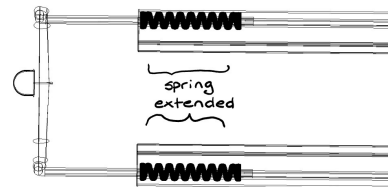
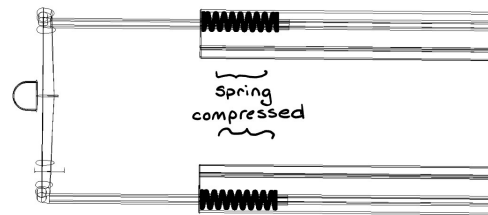


Figure 2.8: Internal spring mechanism

Figure 2.9 shows the full structure and how it will attach to the back of a monitor.

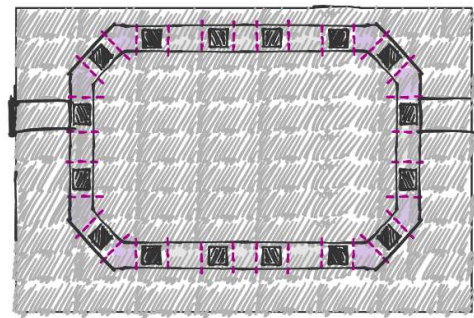


Figure 2.9: Attaching the structure to a monitor

3 USER ANALYSIS

This product is aimed towards an average consumer such that the technical knowledge required is minimized. The complexity is on par with the set-up of other common household appliances.

3.1 SOFTWARE SET UP

The user needs to know how to connect their computer or mobile device to the Wi-Fi, access the configuration page of the device and how to connect the Raspberry Pi to their Wi-Fi.

3.2 HARDWARE/ELECTRICAL SET UP

The user is required to know how to use a screwdriver, and can follow snap-in, plug-in and clamping instructions.

4 TECHNICAL ANALYSIS

4.1 DISCOVERABILITY

Norman describes good discoverability as *“it is possible to determine what actions are possible and the current state of the device”* [1].

Within the software interface, actions that the user can take include turning the lights on and off, adjust the brightness and color saturation of the lights. These actions are represented as switches, sliders, and dropdown menus, and are highlighted in vivid blue to make it stand out from the background.

The module structure features many visual signifiers that indicate to the user how things can be connected. All the components are uniquely shaped for its purpose and will be packaged in a way that they will be easily identifiable.

4.2 FEEDBACK

Feedback is used to notify the user of any changes that have occurred to the system or device.

The software interface gives feedback to users by showing a small toast message at the bottom of the screen explaining what changes have been made and disappears after few seconds. Figure 4.1 shows the example of the toast message that appears when a user changes the brightness using the slider.

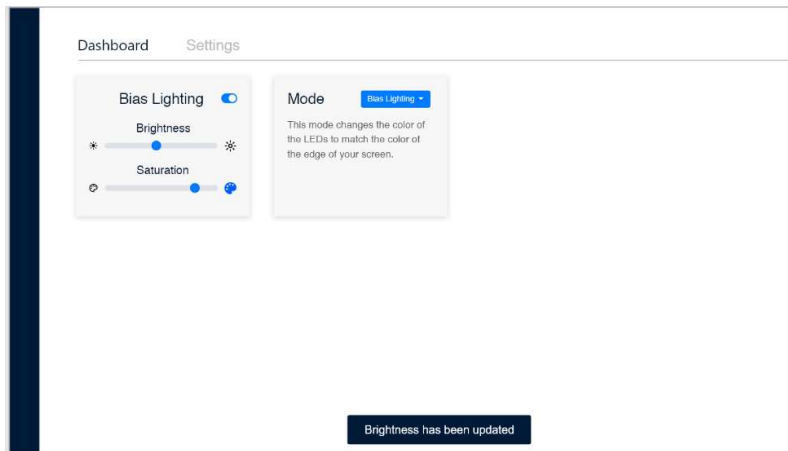


Figure 4.1: Example of Toast Message

For the module structure, indicators of whether it is correctly assembled include: if the screws are fully embedded to secure the modules, and if the snap-fit and clamping mechanisms snaps to its default position. When the user powers the lights for the first time, a quick visual test will show if the lighting works.

4.3 CONCEPTUAL MODELS

The software interface uses different icons such as the icon of the sun and the palette to show the user which direction leads to brighter setting and a higher saturation for the two sliders. These are common symbols used in other interface products and is expected that they will be intuitive for the user.

For the module structure, the user manual will aid in the user's conceptual understanding of how to fit the components together. It will follow conventional mechanics such as the insertion of screws and the clamping structure already found in selfie sticks allowing the user to have an intuitive understanding of how the structure is fitted. The shape of the module structure mimics the rectangular frame of the monitor for an intuitive approach of how to evenly distribute the LEDs.

4.4 AFFORDANCES

Affordances are the perceived actions and properties of an object that help determine its operation.

The switch, sliders, and dropdown menu in the software interface shows the users that they can interact with these components by either clicking or dragging them.

Clues for how to assemble the module structure are visually implied based on the shape of the components. When two pieces easily fit, it is conveyed that they are intended to be arranged in that manner. The screw threads indicate where to insert the screws.

4.5 SIGNIFIERS

For software, labels for brightness and saturation icons clarify the actions that can be done with the corresponding slider.

For the modules structure, signifiers for how to connect the parts are implied by the shape of the connectors. Signifiers of a complete and correct connection are whether it fits properly and if it holds the structure properly.

4.6 MAPPINGS

By using conventional switch and sliders, users can intuitively understand how to interact and change the settings, which includes the direction of the slider. As a result, the user can move the cursor to the right to get the maximum value, which is conventional for most software interfaces.

To assemble, an assembly manual will include illustrative diagrams that clearly indicate how they are mapped together. Visual mapping is also indicated by the shape, clearly identifying what is the corresponding connection piece.

4.7 CONSTRAINTS

In the software interface, the current dashboard design is very simple with interactive controls to change the common settings for the bias lighting. The more involved settings will be in the settings page and can be accessed by clicking on the “Settings” link. This method prevents the user from being overloaded with extra information and allows the user to focus on the core features of the product.

When connecting the modules together, the unique interlocking shape only allows it to be fitted one way. Figure 4.2 shows the cardboard mock-up for how to connect modules together, of which only one configuration fits. The modules, plug-in connectors, and clamping mechanisms all have unique connector styles, allowing the user to only connect them in the way intended.



Figure 4.2: Module connector mock-up in all configurations

5 ENGINEERING STANDARDS

Below describes the relevant standards for networking and Wi-Fi from IEEE. Additional sources for best practices are described in *Material Design*, a website for software interface. An online guide of how to create snap-fit joint for plastics is also referenced.

IEEE 802.11	Wireless LAN Working Group [3]
CCNC.2004.1286913	Consumer electronics industry standards for in-home entertainment networking and device connectivity [4]
Material Design	Material is an adaptable system of guidelines, components, and tools that support the best practices of user interface design [5]
Bayer Material Science	Snap-Fit Joints for Plastics [6]

Table 4.7.1: Engineering standards and best practices interface and appearance design

6 USABILITY TESTING

This section details the analytical and empirical tests that will provide feedback to check if the system is working properly. Software feedback is primarily visual. Testing for the module structure involves whether the components fit and hold their structure well. Critical care should be taken to ensure connectivity throughout the structure to make sure the lights will turn on. The tolerance and the shape of the module may need to go through several design iterations to improve its reliability. For 3D printing, a tolerance of 0.3mm is recommended for the print settings [7]. For fitted parts, a 0.08mm gap is recommended between each wall [8].

6.1 ANALYTICAL USABILITY TESTING

6.1.1 Software

- Connect the Raspberry Pi to the Wi-Fi
- Open the dashboard page from a computer or mobile device, which is connected to the same Wi-Fi network as the Raspberry Pi
- Turn the lights on and off from the user interface
 - Check that the toast message appears
 - Verify that the lights turn on and off
- Adjust the brightness using the slider
 - Check that the toast message appears
 - Verify that the brightness of the lights will change according to the input given
- Adjust the color saturation using the slider
 - Check that the toast message appears
 - Verify that the color saturation of the lights change according to the input given

6.1.2 Modules

- Check if the LEDs light up with a brief light sequence when powered on
- Check the current to verify connectivity between modules
- Align the structure with a straight edge to check alignment
- Check to verify that wires are not exposed
- Check to ensure there are no sharp plastic edges

6.1.3 Clamping

- Check if the clamp can be fully extended and easily reverts to its neutral position

6.2 EMPIRICAL USABILITY TESTING

For the software, the empirical usability tests are similar to the analytical usability tests but requires additional feedback from the users. This includes measurements such as the time to complete a step and taking note of the bugs found.

The following tests to be done are:

- Connect the Raspberry Pi to the Wi-Fi
- Open the dashboard page from a computer or mobile device, which is connected to the same Wi-Fi network as the Raspberry Pi
- Turn the lights on and off from the user interface
- Adjust the brightness using the slider

- Adjust the color saturation using the slider

An empirical check for the module structure includes, whether the parts fit together easily, if the LEDs turn on, and if the clamps sufficiently hold it onto the monitor.

7 CONCLUSION

The core concept of this bias lighting kit is to keep the assembly and usability easy and flexible such that an average consumer can use it with minimal technical expertise. With this idea in mind, the core software interface is simple and intuitive, and the module structure uses connection methods that are familiar to an average user.

The current state of the interface and appearance design is a developed interface of the core software features and an overview of the module structure design. A prototype of these core features will be shown in the proof-of-concept. This will include basic interactivity demonstrated through the software interface. Module demonstration for the proof-of-concept include showing a few samples of the different printed modules, how it connects with screws, lights that will turn on based on user input from software, as well as how it will attach to a monitor.

For the final project, future work will largely include; refining the features based on new considerations, testing the effectiveness of the current design methods described, and adding more settings to improve customization features.

8 REFERENCES

- [1] D. Norman, *The Design of Everyday Things*, Philadelphia: Basic Books, 2013.
- [2] "How to connect two parts with 3D printed joints and snap fits," [Online]. Available: <https://www.sculpteo.com/blog/2018/04/25/how-to-connect-two-parts-with-3d-printed-joints-and-snap-fits/>. [Accessed 18 03 2021].
- [3] "IEEE 802.11-2020 - IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks--Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Requirements," IEEE, [Online]. Available: https://standards.ieee.org/standard/802_11-2020.html. [Accessed 21 03 2021].
- [4] "Consumer electronics industry standards for in-home entertainment networking and device connectivity," IEEE, 5 Jan. 2004. [Online]. Available: <https://ieeexplore-ieee-org.proxy.lib.sfu.ca/document/1286913/metrics#metrics>. [Accessed 21 03 2021].
- [5] "Material Design," Google, [Online]. Available: <https://material.io/design>. [Accessed 21 03 2021].
- [6] "Snap-Fit Joints for Plastics," [Online]. Available: http://fab.cba.mit.edu/classes/S62.12/people/vernelle.noel/Plastic_Snap_fit_design.pdf. [Accessed 21 03 2021].
- [7] "How to design snap-fit joints for 3D printing," [Online]. Available: <https://www.3dhubs.com/knowledge-base/how-design-snap-fit-joints-3d-printing/>. [Accessed 15 03 2021].
- [8] M. Forged, "3D Printed Joinery: Simplifying Assembly," [Online]. Available: <https://markforged.com/resources/blog/joinery-onyx>. [Accessed 21 03 2021].

APPENDIX B: SUPPORTING TEST PLANS

1 PHYSICAL APPEARANCE AND STRUCTURE

Test Name:	Module Fitting				
Description:	Test if modules can be fitted together				
Passing Criteria:	Modules structure is not loose when fully attached				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Module Alignment				
Description:	Test if modules alignment is straight when chained together.				
Passing Criteria:	Check if from the straight edge of the first module to the final module does not diverge by 5° per meter				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Check securing methods				
Description:	Check if the snap-in, clamp-in and plug-in mechanisms are working				
Passing Criteria:	Check if the screws can be fully screwed into the heat-set inserts Check if the clamp can be fully extended and easily reverts to its neutral position				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Safety of Modules				
Description:	Check if the modules do not pose a hazard				
Passing Criteria:	Verify modules do not have sharp edges Verify wires are not exposed				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Clip-On Attachment				
Description:	Test if module structure will attach onto a monitor				
Passing Criteria:	Module structure attaches onto a monitor				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

2 HARDWARE/ELECTRICAL

Test Name:	Power On				
Description:	System must turn on				
Passing Criteria:	Raspberry Pi turns on HDMI Splitter's lights turn on indicating that it is powered All LEDs turn on				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Check connectivity between LED Modules				
Description:	Check the current between modules				
Passing Criteria:	Sufficient current is flowing through the modules				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

3 USER INTERFACE

Test Name:	Accessing the user interface from web browser				
Description:	Test if user interface is accessible from web browser on a computer in same network				
Passing Criteria:	Web page loads with user interface displayed on browser				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Turning On/Off LEDs				
Description:	Test if the switch for the LEDs works.				
Passing Criteria:	When the switch is turned on, LEDs would be turned on. Also, when turned off, LEDs would be turned off.				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Adjust brightness of LEDs				
Description:	Test if the brightness control of the LEDs works				
Passing Criteria:	When the brightness control in the UI is increased, the brightness of the LEDs increase. Also, when brightness control is decreased, the brightness of the LEDs decreases.				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Analytical Usability Testing				
Description:	Test if all user interface works as expected. Follow the procedure in Analytical Usability Testing in Appendix A.				
Passing Criteria:	Meets expected results in Analytical Usability Testing in Appendix A.				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

5 BACK-END SOFTWARE DESIGN

Test Name:	Video Input Detection				
Description:	Test if video input is detected				
Passing Criteria:	Video input is detected by the software				
Comments on					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	LED colors are changing				
Description:	Test if LED colors are changing				
Passing Criteria:	LED colors are changing				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	LED colors match video input				
Description:	Test if LED colors match the HDMI video input				
Passing Criteria:	LED colors match the border colors of the HDMI video input				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

Test Name:	Number of LEDs match				
Description:	Test if the LEDs display the appropriate colors when differing the number of LEDs used				
Passing Criteria:	LED colors match the border colors of the HDMI video input even when using the minimum and maximum LEDs supported				
Comments:					
Time/Date:		Tester:		Result:	Pass Fail

APPENDIX C: REQUIREMENT SPECIFICATIONS

1 GENERAL REQUIREMENTS

Requirement ID	Requirement Description
Req 1.1.C	System must utilize a hardware device to analyse the video signal
Req 1.2.C	System must have variable LED spacing
Req 1.3.C	System must be able to provide power to all the sub-components

Table 6.2.1: General System Requirements Specifications

2 PHYSICAL APPEARANCE AND STRUCTURE REQUIREMENTS

2.1 GENERAL REQUIREMENTS

Requirement ID	Requirement Description
Req 2.1.1.C	Structure must be able to clip onto different monitors
Req 2.1.2.C	Device must be easy to set up
Req 2.1.3.E	The device must have a module structure for variable spacing

Table 2.1.1: General Physical Appearance and Structure Requirements Specifications

2.2 MODULE REQUIREMENTS

Requirement ID	Requirement Description
2.2.1.C	Must have electric connectivity between modules
2.2.2.C	Modules must be durably connected to each other
2.2.3.E	Module design must be compact

Table 2.2.1: Module Requirements Specifications

3 HARDWARE/ELECTRICAL REQUIREMENTS

Requirement ID	Requirement Description
Req 3.1.C	The device must have an on/off switch
Req 3.2.C	The LED strips must have individually accessible RGB/RGBW LED's
Req 3.3.C	The device must be able to connect to devices interface connectors
Req 3.4.C	LEDs from the modules must be controllable by the front-end software

Table 2.2.1: Hardware/Electrical Requirements Specifications

4 USER INTERFACE

4.1.1 General Requirements

Requirement ID	Requirement Description
Req 4.1.1.C	The application must have a user interface
Req 4.1.2.C	The application must include an option to adjust brightness of the LEDs
Req 4.1.3.E	The application must include option to adjust color saturation of the LEDs
Req 4.1.4.C	The application must display whether it is connected to the device
Req 4.1.5.C	The application must be able to display any error messages transmitted from the device
Req 4.1.6.E	The application must provide the user an interface to enter aspect ratio information of the monitor, and send that information to the device

Table 2.2.1: General Front-End Software Requirements Specifications

4.1.2 Performance Requirements

Requirement ID	Requirement Description
Req 4.2.1.E	The application must have a start-up time of less than 10 seconds
Req 4.2.2.E	The application must not take longer than 30 seconds to connect to the controller
Req 4.2.3.E	The application must not take longer than 5 seconds to send commands to the device

Table 2.2.2: Front-End Software Performance Requirements Specifications

5 VIDEO PROCESSING

5.1.1 General Requirements

Requirement ID	Requirement Description
Req 5.1.1.C	The software should run with a Raspberry Pi 4
Req 5.1.2.C	The software must be able to determine the colors of the LEDs depending on the display
Req 5.1.3.E	The software must be able to receive and transmit data from and to the application
Req 5.1.4.E	The software must ensure that the system never results in an undefined state
Req 5.1.5.E	The software must ensure connectivity between devices
Req 5.1.6.E	The software must be able to adjust the brightness of the LEDs according to the settings on the front-end software
Req 5.1.7.E	The software should not turn on the LEDs if no input is detected
Req 5.1.8.E	The software must be able to receive information regarding the aspect ratio of the monitor from the application, and change the colors of the LEDs accordingly

Req 5.1.9.E	The software should adjust the brightness of the LEDs depending on the display
Req 5.1.10.E	The software should send appropriate error and warning messages

Table 2.2.1: General Back-End Software Requirements Specifications

5.1.2 Video Processing Requirements

Requirement ID	Requirement Description
Req 5.2.1.C	The software must be able to detect video input
Req 5.2.2.C	The software must be able to interpret and process video data from the input source
Req 5.2.3.C	The software must use color averaging algorithms to determine the colors of the LEDs
Req 5.2.4.F	The software should consider the frames per second of the source and adjust the LEDs accordingly

Table 2.2.2: Software Video Processing Requirements Specifications

5.1.3 Performance Requirements

Requirement ID	Requirement Description
Req 5.3.1.E	The software must have a start-up time of less than 5 seconds
Req 5.3.2.E	The software must change the colours of the LEDs synchronously with the display
Req 5.3.3.E	The software should automatically turn off after 5 minutes if no video data is being transmitted
Req 5.3.4.E	The colors of the LEDs must be visually indistinguishable from the colors of the display

Table 2.2.3: Software Performance Requirements Specifications

6 SUSTAINABILITY

Requirement ID	Requirement Description
Req 6.1.F	The device must be durable enough to handle movement to multiple devices
Req 6.2.F	The variable spacers for the LEDs should not generate excessive waste

Table 2.2.1: Sustainability Requirements Specifications

7 SAFETY

Requirement ID	Requirement Description
Req 7.1.F	The LEDs must come with a light diffuser cover to prevent blinding
Req 7.2.F	The LEDs must comply with the recommendation outlined by IEEE Std 1789-2015
Req 7.3.F	The LEDs must comply with the recommendations outlined by ST 196:2003 and RP 51:1995

Req 7.4.F	The system fully assembled should be completely enclosed and not pose an electrical hazard
Req 7.5.F	The packaging of variable spacers and LEDs should display warnings for choking hazard
Req 7.6.F	The device should not have sharp edges

Table 2.2.1: Safety Requirements Specifications

8 ENGINEERING STANDARDS

Standard	Description
IEEE Std 1789-2015	IEEE Recommended Practices for Modulating Current in High-Brightness LEDs for Mitigating Health Risks to Viewers [1]
ST 196:2003	SMPTE Standard - For Motion-Picture Film — Indoor Theater and Review Room Projection — Screen Luminance and Viewing Conditions [2]
RP 51:1995	SMPTE Recommended Practice - Screen Luminance and Viewing Conditions for 8-mm Review Rooms [3]

9 REFERENCES

[1] "IEEE Recommended Practices for Modulating Current in High-Brightness LEDs for Mitigating Health Risks to Viewers," *IEEE Std 1789-2015*, Vols. 1-80, 5 June 2015.

[2] "ST 196:2003 - SMPTE Standard - For Motion-Picture Film — Indoor Theater and Review Room Projection — Screen Luminance and Viewing Conditions," *ST 196:2003*, pp. 1-4, 20 Oct 2003.

[3] "RP 51:1995 - SMPTE Recommended Practice - Screen Luminance and Viewing Conditions for 8-mm Review Rooms," *RP 51:1995*, pp. 1-2, 1 Jan 1995.

APPENDIX D: ALTERNATIVE DESIGN OPTIONS

1 LEDs:

Design Option	Description
Adafruit NeoPixels	3.5 x 3.5 mm LEDs that are individually addressable
Adafruit DotStars strips, 60 LED per meter (Design Choice)	5 x 5 mm LEDs that are individually addressable

2 SECURING MODULES TOGETHER

Design Option	Description
Snap Fit	Snap-fit design is a faster connection method
Screws (Design Choice)	Screws is a reliable and common securing method

3 ELECTRIC CONNECTIVITY BETWEEN MODULES:

Design Option	Description
Molex Compression Connectors	Molex compression connectors is compact but alignment between the pads is necessary
Molex 2.54 mm connector Header (Design Choice)	Header connectors common and easy to solder in connection

4 USER INTERFACE METHOD:

Design Option	Description
Web Interface	<p>Using webserver hosted in Raspberry Pi and Wi-fi connection to the local network, user can access the UI by accessing the webserver from any browser that is connected in the same local network.</p> <p>Advantage:</p> <ul style="list-style-type: none"> • Able to control the settings such as brightness from anywhere, even the display is attached on the wall and it is out of reach. • Does not need Application to be installed onto computer or mobile device to control. <p>Disadvantage:</p> <ul style="list-style-type: none"> • Only able to control when there is a Wi-fi connection.
Touchscreen Monitor	<p>Attaching a small screen (around 3.5in size screen) onto the Raspberry Pi, and display settings and buttons to configure the system.</p> <p>Advantage:</p> <ul style="list-style-type: none"> • The small screen is always there and easy to use. <p>Disadvantage:</p> <ul style="list-style-type: none"> • Cannot change the setting when the TV is mounted high on the wall and the screen out of reach.
Mobile App via Bluetooth	<p>Using an application installed in mobile device or computer to control the device via Bluetooth connection.</p> <p>Advantage:</p> <ul style="list-style-type: none"> • Able to control the settings such as brightness from anywhere, even the display is attached on the wall and it is out of reach. <p>Disadvantage:</p> <ul style="list-style-type: none"> • Require user to install the application onto mobile device or computer. • Only device with Bluetooth capability can pair. • Need to pair the device every time with mobile device or computer.

5 WEBSERVER SOFTWARE:

Design Option	Description
Apache HTTP Server	Suitable for small setup with a smaller number of requests
Nginx	Suitable for processing large number of queries
Node.js	Suitable for small number of requests, easier compatibility with WebSocket.

6 BACK-END SOFTWARE DESIGN

6.1 VIDEO PROCESSING

6.1.1 Software Coding Language

Design Options	Description
Python (Design Choice)	Preferred language for Raspberry Pi's. Supports a huge set of libraries. Simple to understand. High-level. Object-oriented.
C++	Fast compiling programming language. Limited number of library support. Relatively low-level in comparison. Object-oriented.
Java	Fast compiling programming language. Supports more libraries than C++ but less than Python. Code is convoluted. High-level. Object-oriented.
Arduino	Preferred language for Arduino boards and ESP32. Derived from Processing. Simple to understand. Supports a smaller set of libraries compared to Python.

6.1.2 Supported Video Resolution

Design Options	Description
3840x2160@30Hz input, 1920x1080@30Hz processing (Design Choice)	Hardware limitation of HDMI Capture Card. Good balance of processing time and LED color quality.
3840x2160@30Hz input, 3840x2160@30Hz processing	Achievable by using an on-board HDMI decoder. Significantly increases the video processing time.

6.1.3 Capturing HDMI Input

Design Options	Description
Python Imaging Library (Pillow) (Design Choice)	Friendly fork of PIL – a free and open-source library for Python that supports opening, manipulating, and saving many different image file formats.
Open Source Computer Vision Library (OpenCV)	Library of programming functions that vary from video processing to real-time computer vision.
Fast Forward MPEG (FFmpeg)	Free and open-source library for handling video, audio, and other multimedia files and streams. Designed for command-line-based processing of video and audio files.

6.1.4 Processing Border Colors

Design Options	Description
Color averaging over squares (Design choice)	The image is divided into squares depending on the number of LEDs being used. The LED colors are determined by the average color of their corresponding border square.
Color extrapolation	The current frame and previous frames of the video are analyzed using a motion compensation algorithm, and the next frame of the video is

	predicted. The LED colors are determined by the average color of the predicted frame.
Color based on a relative pixel	The image is divided according to the number of LEDs. The colors of the LEDs are determined by the color of the pixel assigned to the LED.

6.2 LED DRIVER

6.2.1 LED Driver Library

Design Options	Specifications
FastLED (Design choice)	Fast, efficient, easy-to-use library that supports the programming of popular addressable LED strip including Neopixel, WS2801, WS2811, WS2812B, LPD8806, TM1809, and more.
Adafruit DotStar	Adafruit's DotStar library, allowing a broad range of microcontroller boards (most AVR boards, many ARM devices, ESP8266 and ESP32, among others) to control Adafruit DotStars and compatible devices – APA102, etc.
Pololu APA102-Arduino	C++ library that helps control addressable RGB LED strips and panels based on the SK9822/APA102/APA102C RGB LED controller ICs. Provides full access to the 24-bit color register and 5-bit brightness register of each LED.

6.2.2 LED Refresh Rate

Design Options	Specifications
60 times a second (Design choice)	Able to cover lower resolutions at 60 Hz if needed.
30 times a second	Less processing required, but there will be significant loss in quality in LED output if a 60 Hz input is used.