Webcam Lighting Studio: A Framework for Real-Time Control of Webcam Lighting

Karan Gonagur*, Shobhit Aggarwal*, David Fillmore, Jr.†, and Asis Nasipuri*
*Department of Electrical & Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223
Email: {kgonagur, saggarw4, anasipur}@uncc.edu
†Department of Theater, University of North Carolina at Charlotte, Charlotte, NC 28223
Email: D.M.Fillmore@uncc.edu

***Abstract*—We present the design of a multiuser networked wireless system to remotely configure and control the lighting of multiple webcam users at different locations. This system makes use of a Raspberry Pi and a wireless DMX transmitter as the wireless interface that can be used to control the DMX webcam lights. A lighting control software called OLA is used on the Raspberry Pi. A web interface is designed to issue commands to OLA API running on the Raspberry Pi to control DMX lights associated with Raspberry Pi. Multiple wireless interfaces, each for a specific user at a different location, can be simultaneously configured and managed using the web interface. The interactive web interface can be used to control the intensity and color of the DMX lights. The web interface follows a model controller view design and makes HTTP calls to the OLA software running on Raspberry pi. The proposed system enables an operator to provide optimum and artistic lighting effects for a group of online presenters.**

***Index Terms*—webcam lighting, DMX, system design.**

I. Introduction

The COVID-19 global pandemic at the start of the year 2020 forced many businesses, educational institutions, meetings, and conferences to go virtual. This ushered in the need for remote meetings and presentations across a broader set of users involving new requirements [1]. The most common format for online meetings involved multiple individuals at different locations, each controlling their own audio-visual parameters (e.g., lighting, sounds) from their own teleconferencing applications such as Zoom, Google Meets, etc. However, the pandemic also forced concerts and theatrical performances to go virtual, which brought in a new set of challenges and advantages [2]. Remote theatrical performances extended the reach of users to different geographical locations, providing access to the disabled and the elderly at affordable costs. This has given rise to a demand for quality, professional-looking lighting that can be controlled remotely. Just as in live theatrical performances, concerts, and professional presentations, lighting is controlled by a lighting designer or operator who is present at the venue and controls the lighting systems using lighting control hardware, online performances and presentations often need a lighting control mechanism that can be controlled remotely.

The standard industry protocol used to control the lights for commercial light shows and stage lighting shows is the DMX512 protocol [3]. DMX512 was created in 1986 by an engineering commission USITT (United States Institute for Theater Technology) to control dimming channels on lights. DMX512 protocol uses the RS-485 standard at the physical layer to overcome electrical noise and send data over lengthy cabling distances. The DMX protocol makes use of XLR connectors to make physical connections. The 3-pin XLR connectors use Data+, Data- and ground, and the 5-pin uses 2 additional data pins which are mostly unconnected. At the Data link layer, the DMX layer sends out 513 frames out of which 512 frames are data frames and 1 frame is used for synchronization. The operator uses a DMX light controller to issue DMX signals to control the lights. These lights usually have physical connections to the controller. The operator can therefore control the parameters of different fixtures by adjusting the DMX values of these fixtures.

In a virtual performance, the performers are usually located at different locations, each having their own lighting system set up to provide adequate illumination for the webcam operation. However, current webcam lighting systems lack the ability to provide special effects and variations of lighting that would be required at various instances during live performances that are normally controlled by a lighting operator. There is a need for a centralized interface where a single operator can have access to the webcam lighting of multiple users present at different locations and control the lights at various instances during the performance. The operator must be able to do so in a convenient manner and in real-time.

We propose a real-time multi-user networked interface to control DMX lighting fixtures at multiple locations, which we refer to as the Webcam Lighting Studio (WLS). This paper presents the detailed prototype design of WLS, which involves the development of a web-based user interface, database, and a networked hardware interface that receives control signals from the web interface via the internet and wirelessly controls the DMX-enabled lighting setup present with each performer. The web-based user interface provides the operator to access the set of lights (light universe) at each user location and transmits control signals to the WLS Interfaces located in each user location. The database maintains current and dynamically changing parameters entered by the operator. The WLS hardware interface receives signals via the Internet and transmits wireless DMX signals to the set of lights for that user's universe. The system allows operators to control multiple lighting systems in real-time as if they are present

Fig. 1. Ring light (Source: adapted from [**?**])

there physically.

II. Related Works

In this section, we present a review of existing technologies to provide lighting for webcam applications and existing projects developed to control DMX lights.

A. Ring Light as a basic solution for webcam lighting

The Ring Light [**?**] as shown in Figure 1 aims to improve the appearance of users by providing illumination through a ring-like LED light. The webcam version of the ring light aims to provide lighting for video calls and conferencing by embedding LED lights around the webcam. However, the ring light cannot be controlled over the internet and must be controlled manually by the user. One of the major disadvantages that were found in our survey of the ring light users is that the users have to stare into the webcam surrounded by the bright LED lights while using the setup, which is not a comfortable experience, especially for longer durations.

B. Raspberry Pi as a DMX light controller

This project by Flashular [5] makes use of Raspberry Pi to control DMX lights. In this project, a simple user interface is generated on the Raspberry Pi to control DMX lights (see Figure 2). It uses a Raspberry Pi and a DMX controller to control a DMX light. One end of a DMX controller is connected to the Raspberry Pi and the other end of the controller is connected to the DMX light. This project makes use of two programs that run on Raspberry Pi. The first program generates the user interface and contains the function calls to set DMX channels on a shared memory location. The second program runs in the background and performs serial communication to send data to the DMX controller connected to the Raspberry Pi. Whenever the user makes changes to the color wheel on the user interface, the Raspberry Pi transmits the corresponding data to the DMX controller and changes the color of the DMX light. Although this project provides a simpler mechanism to control DMX lights it has a lot of limitations. Firstly, the code is configured to work only for a specific DMX controller, namely, the Vellman controller [6]. One can use cheaper alternatives to reduce costs. Secondly, to control multiple fixtures, one has to make significant changes to the code. One has to have access to Raspberry Pi to control the lighting fixtures. Most importantly this project can only control lighting fixtures connected to a single Raspberry Pi.

Fig. 2. Raspberry pi as a DMX light controller (adapted from [4])

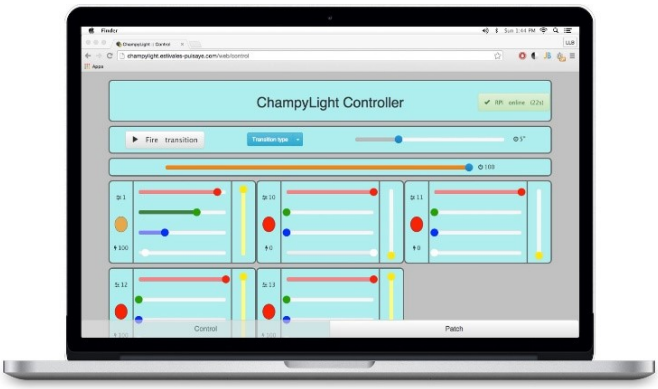

Fig. 3. Champylight web interface (adapted from [**?**])

C. Champylight

The Champylight [7] makes a significant improvement to the project by Flashular. It provides an additional web interface to control lighting fixtures connected to the Raspberry Pi. The web interface runs on a computer, so one does not need to connect to the Raspberry Pi to control the lights. This interface also makes it possible to control multiple lighting fixtures connected to a single Raspberry Pi without having to make significant changes to the code. The main limitation of this improvement is that the web interface can only control lighting fixtures connected to a single Raspberry Pi.

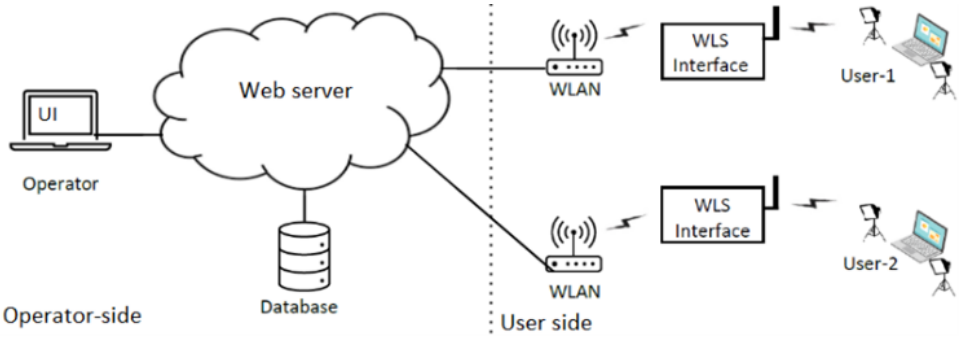

Fig. 4. System block diagram

III. System Design

We propose an IP-enabled solution that allows an operator to control the webcam lighting of multiple users at different locations. The specific design considerations for the proposed system are as follows:

- The current implementation requires each presenter at a different venue to have a wireless interface for a set of DMX lights as shown by Fig. 4
- The wireless interface should be controlled by the operator through a web application. The operator interacts with the front end or the user interface of the web application.
- The operator can register new users by adding their IP address, a friendly name, and the number of lighting fixtures the user has.
- This information should be stored in a database. The user interface displays the list of registered users who can be identified by their friendly names.
- The User interface should provide a way to manage the registered users and the settings of each light associated with that user.
- The operator can set the color and intensity of the lights through the user interface.

The proposed wireless interface is implemented using a Raspberry Pi and a wireless DMX transmitter. The operator enters control requests that are relayed to the user's wireless interface over the internet via the web server. The server relays the requests to the Raspberry Pi's Ip address. The Raspberry Pi has a lighting control software called Open lighting architecture (OLA) [8] running on it. The software's API receives the request and sends DMX data to the transmitter connected to the Raspberry Pi via the USB to DMX cable [9]. The transmitter transmits the data to the light fixture and changes its state

A. Hardware Used

1) DMX Lights: The lights are milight DMX controllable 9W LED bulbs (Figure 5). These bulbs have wireless DMX receiver embedded inside them. To control them, we use a DMX 512 transmitter (see Figure 6). Pairing between the transmitter and the lights can be done by pressing the set button on the transmitter 3 times. Once a light is paired

Fig. 5. Milight DMX LED bulb (adapted from [10])

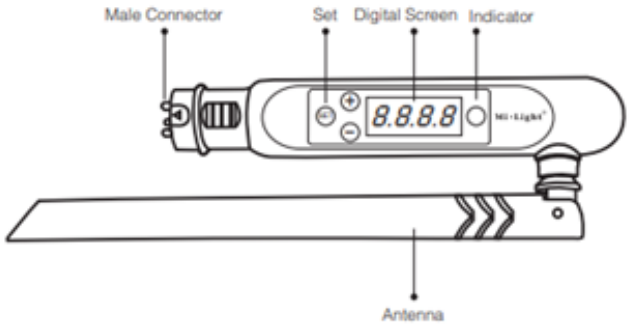

Fig. 6. Wireless DMX transmitter (adapted from [**?**])

the light bulb settings can be controlled by transmitting the corresponding DMX data. The bulb has 5 consecutive DMX channels corresponding to red, green, blue, warmth, and coolness parameters of the bulb. The channels can have a value ranging from 0 to 255. Settings 0 and 255 corresponds to the lowest and highest intensity levels of that parameter, respectively.

2) Wireless Interface: A wireless interface is present at each presenter's location. The wireless interface consists of 3 components: a Raspberry Pi zero, wireless DMX transmitter and the USB to DMX cable.

- Raspberry Pi Zero W: The Raspberry Pi Zero W [11], as shown in Figure 7, is a tiny but powerful computer about

Fig. 7. Raspberry pi zero w (adapted from [**?**])

Fig. 8. USB to DMX cable

half the size of a credit card. It has Bluetooth and wireless LAN connectivity, 512 MB ram, a micro USB power port, and a mini HDMI port. The Raspberry Pi zero w has the Broadcom BCM2835 CPU with a clock speed of 1 GHz. It supports a micro SD slot. The Raspberry Pi zero runs on the Raspbian operating system, Debian based linux os. The Raspberry Pi is assigned a unique IP address and this allows the operator to make http requests to the Raspberry Pi to control the lights via the web application.

- USB to DMX cable: The USB to DMX cable (see Figure 8) converts the incoming USB data from the Raspberry Pi to DMX data. The USB to DMX cable has a USB and DMX interfaces to perform the conversions. This DMX data is given to the input of the transmitter.
- Wireless DMX transmitter: Wireless DMX Transmitter has an input DMX port and a transmitter. The input DMX port is used to receive DMX data from an external source. The transmitter transmits the received input DMX data wirelessly. The Wireless DMX Transmitter is used as a replacement for wired DMX connections. This project uses the Milight FUTD01 wireless DMX Transmitter [12]. The Transmitter uses the 2.4GHz ISM band to transmit the DMX 512 data wirelessly. The Transmitter's DMX input is connected to the Raspberry Pi via a USB to DMX cable. This allows the Transmitter to receive the DMX data from the Raspberry Pi. The transmitter that we are using can support 16 independent zones. Each zone is a collection of 5 DMX 512 channels. The lighting fixture that we are using has 5 DMX channels. Thus, The Transmitter can be used to control up to 16 lights at the same time.

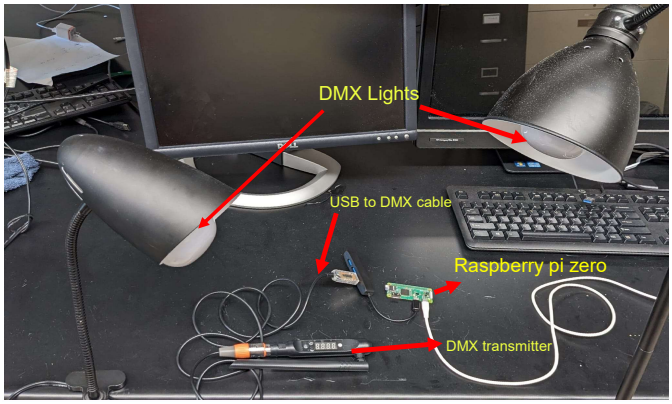

Fig. 9. Hardware setup of WLS for an user with wireless interface and two LED lights.

An experimental setup for the proposed WLS equipment for an user with two DMX lights in illustrated in Figure 9.

B. Software Used and Developed

1) User side: The Raspberry Pi Zero W present at the presenter's location uses a set of software in order to control the associated hardware. It runs on Raspberry Pi OS [13] which is a Debian-based Linux OS. The OS is available in both 32-bit and 64-bit versions. It also has a lite version and the desktop version. The desktop version has a Graphical User Interface (GUI), while the lite version has a Command Line Interface (CLI). WLS uses the 32-bit Raspberry Pi OS lite since we do not require a graphical user interface thereby optimizing the performance of the Raspberry Pi.

To control the lighting equipment an open-source framework known as Open Lighting Architecture (OLA) [14] is used. It implements the lower-level software and protocols that are needed to control DMX lights. The OLA consists of 3 important parts: OLA daemon, OLA client library, and OLA plugins. The OLA daemon runs as a background process and is responsible for handling the DMX512 communication between different hardware devices. The OLA client library provides applications with APIs to interact with the OLA daemon. OLA defines plugins that support DMX over different hardware devices. The FTDI USB to DMX is of particular interest as it allows us to transmit DMX data over the USB to DMX cable. Each plugin device must be assigned to a universe id and a port, this process is called patching. The FTDI USB to DMX plugin is patched to universe three and configured as an output port. The OLA web server runs on the local host at port 9090. It provides the set_dmx [14] API to interact with the webserver and set the DMX values for the specified universe. The set_dmx API takes in universe id and DMX values for channels 1 to 512 as the input and sets the corresponding values for that universe. The set_dmx API call can be made to the OLA running on the Raspberry Pi

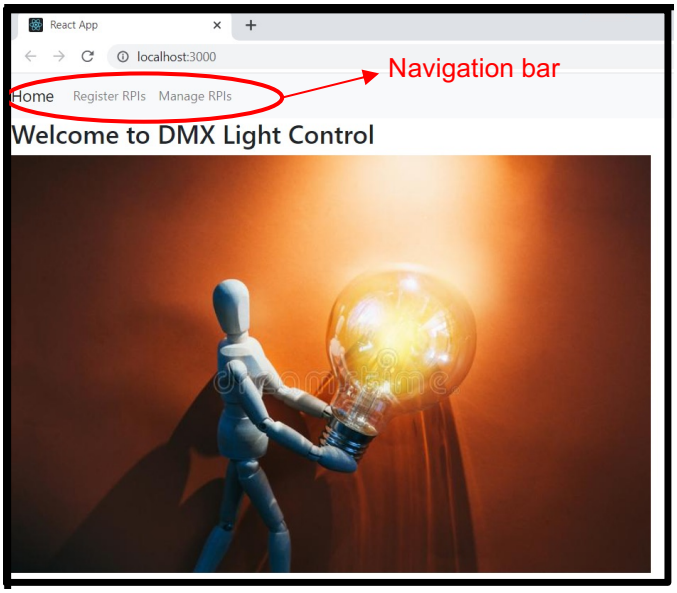

Fig. 10. Homepage

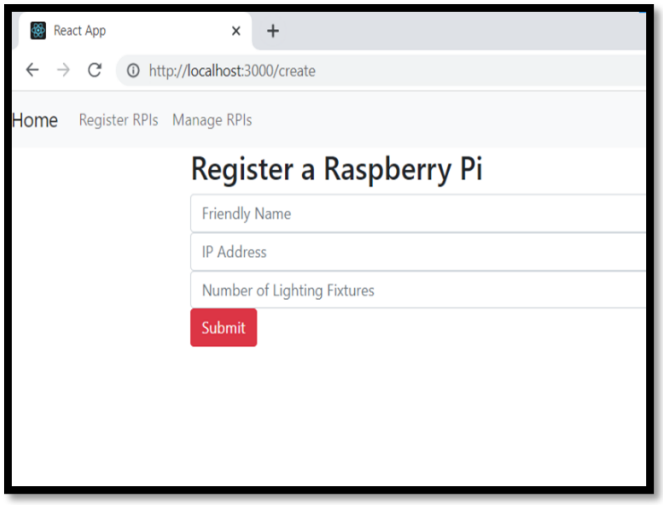

Fig. 11. Raspberry PI registration page

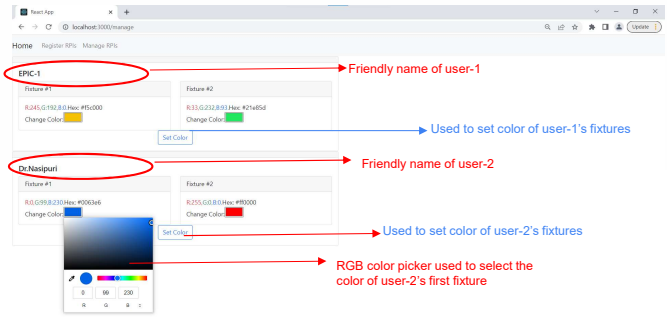

Fig. 12. Manage Rpi page

if we know the Raspberry Pi's IP address. This allows us to use a web application to control the OLA server running on Raspberry Pi over the internet.

2) Operator side: The operator uses a web application to control the user-side hardware. This web application was developed using the MERN stack [15] and provides the operator with an interface to control the lights. The MERN stack is composed of Mongo DB, Express, React, and node technologies.

The front-end application and the user interfaces were developed using React [15]. The server side was developed using express and node frameworks of javascript. For the database, Mongo DB [15] was used. Mongo DB is a non-relational database used to store data in the form of documents and collections. The document is the simplest unit of data supported by MongoDB and it consists of key-value pairs.

Typically, all applications define a set of interfaces to access the application's data or functionality. Application Programming Interfaces (APIs) allow multiple applications to interact with one another without exposing the details of the interfaces. Web APIs are typically used by the client application to access the functionality of the Web server. The client sends a request to the API of the server. The server's API processes the request and returns a response. The response contains the data requested by the client. The client's request to the server's API contains endpoints, resources, and the HTTP method used. In our design, we use a URL to specify the endpoint and the resource. Resource refers to the data requested by the client and the endpoint is used to identify the address of the resource. The method specifies the action to be taken by the API.

IV. Implementation

This section presents the details of the implementation and operation of the web interface and the hardware setup for the proposed WLS system. Later in the section, we demonstrate the controlling of DMX lights through the web interface.

The web application is implemented as a three-part application: client application, server application, and database. These applications interact with one another using APIs. The client applications and the server APIs are designed to use the HTTP protocol to send requests and receive responses.

The server application running on the web exposes its functionality through APIs. The client application makes requests to the server's APIs to read or update the data present in the database. The server application provides functionality for CRUD operations that are to create, update, read, and delete data in databases.

The client application displays the list of registered users on the user interface. The web user interface contains three pages. The first page is the home page that has a navigation bar with links to the 'register RPI' and 'manage RPI' pages. This is illustrated in Figure 10. It allows the operator to register new performers and control the color of lighting fixtures for existing performers via the user interface. When the operator wants to register a new Raspberry Pi, its IP address, friendly name, and the number of light fixtures in the user's universe, the user can enter these in the "Register RPI's" page (see

(a) Before (b) After

Fig. 13. Images of a user of Zoom teleconferencing software without lights (a) and after turning on the proposed WLS system (b) at a poorly lit location.

Figure 11).

Once the operator provides these details and clicks the submit button, the client application will send a request to the server's API. The server's API will create a new corresponding entry in the database. When the operator clicks on the "Manage RPI" page, the client application makes a request to the server's API that fetches the most recent data from the database and sends the data to the client in JSON format. The client uses this data and displays the registered Raspberry Pis with their friendly names along with an interface that enables the user to choose colors corresponding to each fixture associated with that Rasperry PI to the operator in an intuitive manner. The operator can choose the color of the fixture using an RGB palette to change the color of the lighting fixture. The designed user interface for setting the colors is illustrated in Figure 12. Once the color is chosen the operator can use the "Set color" button to set the color of the fixture. When the operator clicks the button, the client application sends the RGB values of the fixture and the IP address of the Raspberry Pi associated with the fixture via an HTTP request to the server's API. The server's API uses the information in the request to update the database. This request is simultaneously transmitted to the OLA server associated with the user's Raspberry Pi. The OLA software on the Raspberry Pi receives the request and generates the necessary DMX data to change the color of the light. The operator can register multiple Raspberry Pis and control multiple fixtures associated with each Raspberry Pi using the web application.

V. Demonstration of Performance

To demonstrate the impact of the proposed WLS system on improving the appearance of a webcam user, we illustrate the appearance of a user as observed in the Zoom teleconferencing application at a poorly lit up location, before and after activating our WLS system in Figure 13. In addition to obtaining quality illumination under a wide range of ambient lighting conditions, the main benefits of the proposed WLS system is to achieve special lighting control, as required to achieve artistic effects. For instance, optimum colors and intensity of lighting to match a specific background image is illustrate in

Fig. 14. Appearance of a user in Zoom with optimum lighting to match with a background image.

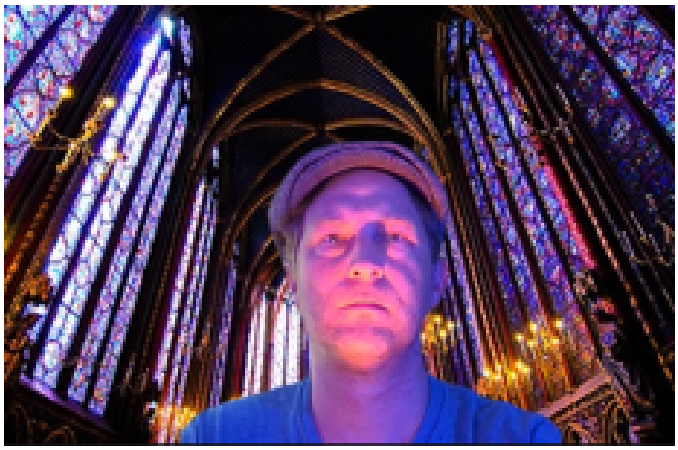

Fig. 15. Illustration of artistic lighting effects achieved using the proposed WLS system.

Figure 14. An illustration of artisitc lighting effects is depicted in Figure 15.

VI. Conclusions and Future Work

A system to control webcam lighting of multiple users was designed, developed and tested. The IP-based implementation allows the operator to seamlessly control the lighting systems

of multiple users over the internet. This was achieved by designing a web application that would enable an operator to control the wireless interface of the webcam lights. The wireless interface uses a raspberry pi along with a wireless DMX transmitter to control DMX controllable webcam lights wirelessly. The web application communicated with the ola server by on the raspberry pi by issuing HTTP requests to control the wireless interface.

An initial customer survey process that was conducted as part of this project revealed that the proposed system would benefit multiple segments of online presenters. The survey done on multiple types of users suggested that lighting has a lot of impact on the storytelling experience. This survey included webcam users and service providers working in various industries such as educational institutes, conference hotels, convention centers, business communities, political leaders, theaters, social media influencers, live stream gamers, etc. Our future work includes further development of the web interface and control parameters that meet the requirements of various customer segments. We plan to make the registration process more efficient by designing an app that would allow the users to register their raspberry pi to the web server. Additionally, we aim to add a set of presets of color settings on the user interface that would enable the operator to Furthermore, we aim to add sensors that would detect the change in the ambient light and automatically adjust the lighting to the appropriate preset setting.

ACKNOWLEDGMENT

This work was supported by a grant from UNC Charlotte's Ventureprise Charlotte Launch NSF I-Site program *Zoom Theater Interface* (I-Corps, 2021-2022 Award # 1450417).

REFERENCES

[1] "How zoom became so popular during social distancing," [Online]. Available: https://www.cnbc.com/2020/04/03/how-zoom-rose-to-the-top-during-the-coronavirus-pandemic.html.

[2] "The show must go online: How performance was reinvented for the pandemic," [Online]. Available: https://www.cnet.com/culture/internet/the-show-must-go-online-how-performance-was-reinvented-for-the-pandemic/.

[3] "American national standards institute, "american national standard ansi e1.11 – 2008 (r2018) entertainment technology—usitt dmx512-a asynchronous serial digital data transmission standard for controlling lighting equipment and accessories" [1] american national standards institute , ansi cp/2007-1013r3.1, may 31 2018." [Online]. Available: https://tsp.esta.org/tsp/documents/docs/ANSI-ESTA_E1-11_2008R2018.pdf.

[4] "Namrata gogoi "top 5 webcams with a ring light"," [Online]. Available: https://www.guidingtech.com/best-webcams-ring-light-buy/.

[5] "Flashular: Raspberry pi as a dmx light controller." [Online]. Available: https://www.instructables.com/Raspberry-Pi-as-a-DMX-light-controller/.

[6] "Louis leblin "online dmx light controller"," [Online]. Available: https://www.instructables.com/Online-DMX-Light-Controller/.

[7] "Velleman kits projects," [Online]. Available: http://www.velleman.co.uk/contents/en-uk/p310.html.

[8] "Open lighting architecture," [Online]. Available: https://www.openlighting.org/ola/.

[9] "Lixada usb to dmx interface adapter," [Online]. Available: https://www.lixada.com/p-l0385.html.

[10] "Lgidtech dmx512 led light bulb miboxer 9w," [Online]. Available: https://www.amazon.com/Mi-Light-Temperature-Adjustable-Transmitter-Separately/dp/B07813CHHY?th=1.

[11] "Raspberry pi zero w," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero-w/.

[12] "Milight user instruction – dmx 512 led transmitter model no: Futd01," [Online]. Available: https://milight.pro/manuals/FUTD01_EN.pdf.

[13] "Raspberry pi os," [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.htmlupdating-and-upgrading-raspberry-pi-os.

[14] Ola json api. [Online]. Available: https://wiki.openlighting.org/index.php/OLA_JSON_APISet_a_universe_of_DMX

[15] Mern stack explained. [Online]. Available: https://www.mongodb.com/mern-stack