ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Mimicry Stage Lighting Control System

Team #29

RUIQI LI
(ruiqili4@illinois.edu)
HAOZHE CHEN
(haozhe3@illinois.edu)
ZHAOHUA YANG
(zhaohua2@illinois.edu)
ANQI TAN
(anqitan2@illinois.edu)

Sponsor: Timothy Lee
TA: Yuchuan Zhu

May 24, 2022

Abstract

The lighting effect is widely used nowadays, which is realized by Stage Lighting Systems (SLS). Current SLSs are mainly used on large stages. However, there are some problems on some smaller and amateur situations, such as home parties, small bars, and discos. The professional SLSs have low portability, complicated operations, and lack of liveness. Therefore, It is vital to design a mimicry stage lighting control system that is portable and user-friendly enough to deal with these problems.

In this project, we designed a **Mimicry Stage Lighting Control System**, including a joystick, a central server and a light array. To use this light system, the operator need to hold the joystick and wave his arm. The joystick will track the movement and send position information to central server. The signal is then processed and redirected to the robot arm and control the actual directions of light. Besides, the color of the light is determined by the features of music. It turns out that our design satisfies all the requirements and show great practicability.

Contents

1 Introduction

1.1 Purpose

Stage lighting system (SLS) [1] is used to control lanterns and related equipment to perform artistic lighting to achieve specific stage effects. Current SLSs are mostly designed for large stage performance, related protocols are also very mature [2]. However, under some smaller and amateur circumstances, like home parties or discos, a complete SLS may encounter some problems:

1. **It's too unwieldy**. A complete SLS is cumbersome for small and amateur scenarios. Only a small-scale portable light array with a user-friendly controller is needed. Also, the domestic power support system is not compatible with DMX cables [2].
2. **It's difficult to operate**. An SLS needs professional stage lighting engineers to install and operate. The cost of learning for amateur use needs to be reduced.
3. **It lacks liveness**. In real practice, live manipulation and improvisation are unavoidable. A special mechanism is needed to allow performers to control the color and direction of the light at will without paying much attention.

This project aims to build a **mimicry stage lighting control system**. It's an SLS that the light array will follow the direction of the operator's arms and imitate immediately, where the operator needs to hold two miniature joysticks.

1.2 Functionality

1. **Precision**. For a good mimetic stage effect, the orientation of lights should have a small difference from joysticks. Specifically, the difference of angles should be less than 10° and the delay should be less than 200 ms.
2. **Portability**. The system should be friendly for carrying and installation for amateur users. It should have a reasonably small size, light weight, low energy consumption, and be wireless connected. The minimum physical components configuration (one joystick and one light) should be able to be contained in a 40L bag and less than 10 kg. Endurance of 60 minutes for each device under continuous usage should be guaranteed.
3. **User-friendliness**. Firstly, the user should be able to control the orientation of the light intuitively. The light array should have an alternative waving mode, like imitating the joysticks or randomly moving, which can be switched by simply clicking buttons. The color of light will also be automatically encoded according to music. The time cost of the music analyzer should be lower than 1% of the length of the song, even if the color sequence is pre-computed.

1.3 Subsystem Overview

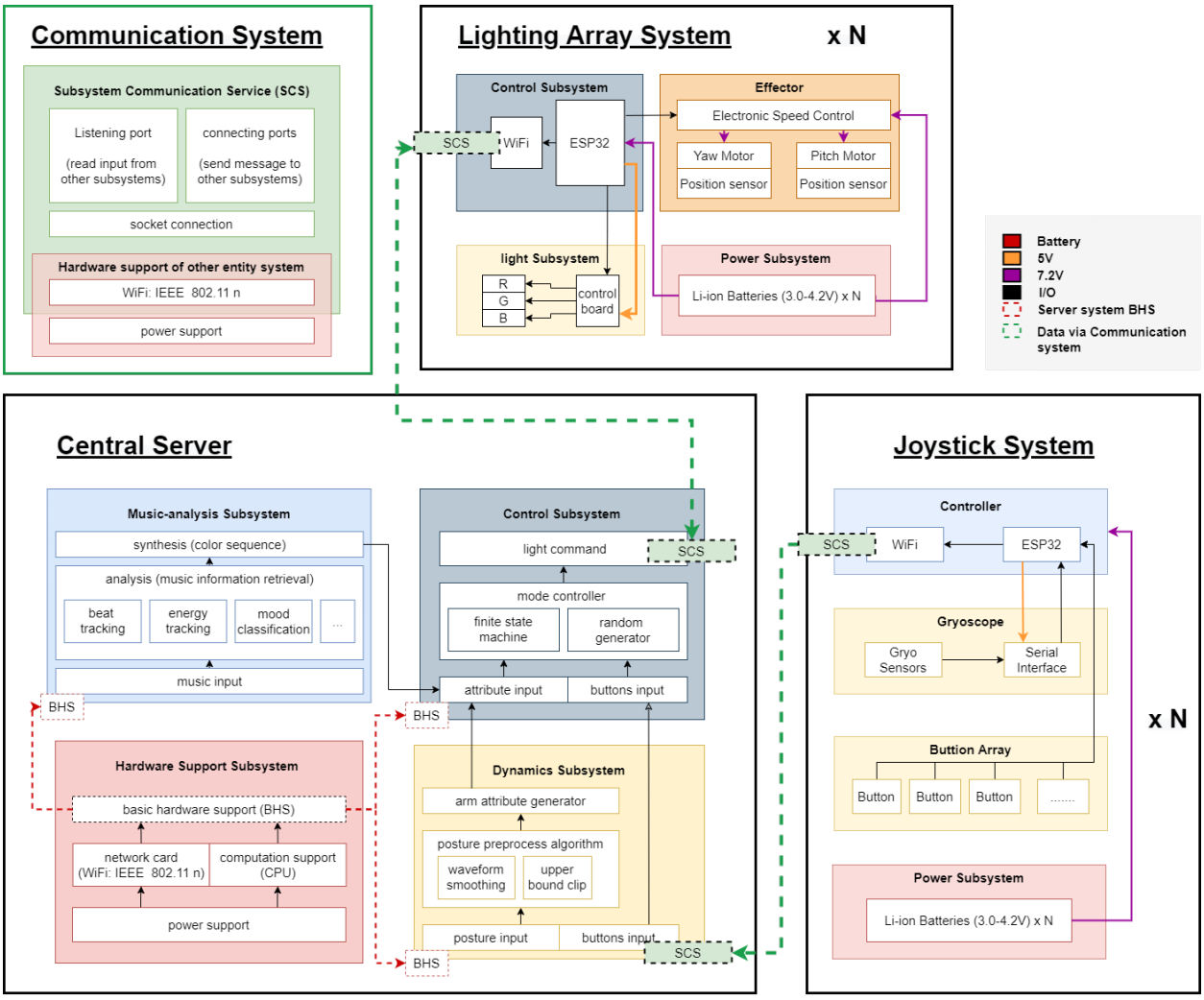


Figure 1: System Block Diagram

Our project consists of 4 high-level subsystems: communication system, lighting array system, central server, and joystick system. The joystick reads the movement signals and the button instructions, which are sent to the central server (PC). The central server process the raw data transmitted from joystick to generate the servo movement commands. Also, it decides whether to play the music according to the button command information. If so, the central server fetches color commands which are pre-computed. These commands are then transmitted to the lighting array and are performed by the latter. The communication system provides support for wireless communication between the other three subsystems, where a reconnection mechanism is implemented for stabilized connection.

2 Design

2.1 Joystick System

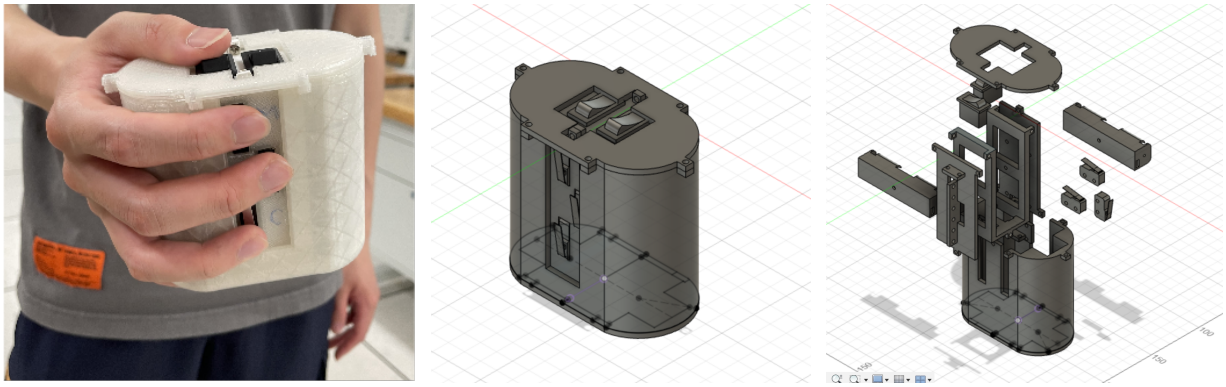
The joystick system is a controller held by the operator to control the actions of the lights.

2.1.1 Control

The software part of the joystick system is based on FreeRTOS. It consists of three parts, the data receiver of posture sensor MPU9250, the TCP client, and the button detection. These tasks work in parallel under the multi-task mechanism provided by the FreeRTOS framework.

- **MPU9250 receiver** MPU9250 interacts with ESP32 via the I2C protocol and sends acceleration and angular velocity to ESP32. To avoid the noise and missing values caused by the sensor itself, a Kalman filter [3] is used to give a great estimation every time. The details of Kalman filter implementation is attached in subsection A.2. The results of the Kalman filter are then stored to some shared variables with lock and wait to be sent by the TCP client.
- **Button Detection** The button array consists of five buttons, with one button controlling the power and the rest of them controlling the finite state machine in the central server. To avoid the bounced signal waveform, a hardware driver of buttons is implemented. The program will wait for the signals to stay constant for some time before setting the logical value.
- **TCP client** The TCP client is introduced in detail in the communication system. For the task implemented in the joystick, a daemon task is constantly fetching shared data produced by the button detection task and the sensor task. Once this attempt succeeds, the fetched data will be sent to the central server via wireless communication.

2.1.2 Mechanical support



(a) Profile display of joystick (b) Joystick mechanical design (c) Exploded-View of Joystick

Figure 2: Mechanical design of joystick

The main framework of joystick is a cuboid shell, where all the components mentioned are placed in it. A smooth cover is produced to wrap up the shell according to ergonomics. Therefore, the only thing the operator should do is to hold the joystick and wave.

The main switch is located at the operator's thumb position. Three microswitches are located as shown in the Figure 2(a). And the spacing between the buttons is properly designed to accord with man-machine ergonomics.

2.2 Communication System

The communication system is an abstract system that handles all the wireless connections between devices. It includes the router, its WiFi environments, and the communication programs on all devices. To meet the precision and portability in the high requirements, stable and reliable multi-device communication via a wireless network is essential. Therefore, to cope with the alterable wireless environment and device configuration, a set of software is developed to adapt to different platforms and guarantee the robustness and stability of the connection, as well as the low latency.

The IEEE 802.11n (WiFi) protocol and the socket protocol based on TCP/IP are chosen for the wireless connections and the ESP32 development board and a Lenovo laptop (XIAOXIN Pro 16) are chosen to support these protocols.

The communication program can be divided into two parts, the listening socket ports, and the sending socket ports. Those ports are maintained in parallel with a multi-thread/multi-task mechanism. The python on the central server and the FreeRTOS framework on ESP32 makes it possible. The disconnection is can be detected in 100ms and automatically repaired with parallel programming. A hotplug re-connection is also realized. Additionally, to guarantee only one message is received each time, a `unsigned long` value, indicating the size of the message content, is attached to the head of the packet.

In the original design, the message delay between devices is about 30s. However, we successfully limit it to 20ms with the following optimizations:

- The socket configuration has been modified. The timer interval is set to 10ms and the buffer size has been set to 3 times the message size.
- Shared data with lock, instead of message queue, is implemented in the ESP32 to avoid redundant sample data. By using the data structure, the sampling rate will be decomposed with the message rate and results in a more accurate result.
- Timestamp is also attached to the message to make sure only a recent message (sent in 100ms before) will be processed. The unnecessary computation can be saved.

With those optimizations, the timestamp difference between the socket client and socket server can be reduced to 15ms on average.

2.3 Central Server

The central server is a computation platform that handles all the computational heavy processes. It can be decomposed to a set of software that processes the message and music file, and the hardware which supports all the computation and network connections. In our case, we choose a Lenovo laptop to be the hardware and deploy the software on it. Since the software is written in python and can be deployed to all the common PC and server, we only focus on the software part in this section. Generally, these different software subsystems are **Music Analysis subsystem**, **Dynamics Subsystem** and **Control Subsystem**. To reduce redundancy, these subsystems will be discussed individually below.

The data-flow in the central server is shown in the Figure 1. Briefly speaking, the dynamics subsystem receives the posture and button data from joysticks and convert it into the gimbal command. The Music analysis subsystem read the music file and generate light command according the music feature. Finally, both two commands are filtered and combined in the control subsystem and sent to the light arrays.

2.3.1 Music Analysis Subsystem

The music analysis subsystem, or the music analyzer, is a `Python` service running on the central server. It takes the music waveform signal as input and generates a color sequence as output. For simplicity, the analyzer precomputes the command sequence before playing, rather than being a real-time computation. The real-time computation and streaming read can be a future work to accomplish.

As discussed in the Design Document, the analysis part extracts 3 features: tempo, beat sequence, and energy sequence. The music signal is sampled at a sample rate $sr = 22050$ Hz. **Besides, an alternative, or addition, is made. We add a neural network classifier to recognize the inherent emotion within the music.** To successfully extract these features, the required theorems and formulas will be discussed below.

Tempo information and beat sequence will be extracted together. This part refers to a well-established algorithm, the beat tracking algorithm by dynamics programming [4]. This algorithm firstly calculates the onset envelope of the input songs. The "onset strength envelope" is a sequence of time, which indicates at what times it has a large chance to be an actual beat. Daniel uses a crude perceptual model on the Mel spectrogram [5] to generate this onset envelope.

Once the onset strength envelope is generated, a global tempo estimate can be made (it's worth mentioning that Daniel's algorithm assumes a constant tempo through the music segment):

$$\text{TPS}(\tau) = W(\tau) \sum_t O(t)O(t-\tau), \quad W(\tau) = \exp\{-\frac{1}{2}(\frac{\log_2 \frac{\tau}{\tau_0}}{\sigma_\tau})^2\} \tag{1}$$

where $\text{TPS}(\tau)$ indicates tempo period strength, $O(t)$ is the onset strength envelope. $W(\tau)$ is a Gaussian weighting function on a log-time axis, where τ_0 is the center of the tempo periods, and σ_τ controls the width of the weighting curve in octaves. The log scale is intended to transform the time axis into an octave measure. τ is then the global tempo prediction, i.e.,

$$\hat{\tau} = \underset{\tau}{\text{argmax}}\, TPS(\tau) \tag{2}$$

Eventually, we can pick peaks in onset strength approximately consistent with the estimated tempo and generate the predicted beat sequence. The red dotted lines in the second figure (Waveform Feature) of Figure 12 are an example of extracted beat sequence.

This algorithm is well-wrapped in a `Python` library `Librosa` [6]. One can leverage the `onset` and `beat` submodules of `Librosa` to generate beat sequence.

Energy sequence is relatively easy to generate. According to Parseval's Theorem, the total energy of a segment of sequence $x[n]$ is $E = \sum_{n=n_1}^{n_2} |x[n]|^2$. Therefore, to compute an energy-time curve, we slice the music signal into small frames, with hop length 512 and frame length 2048.

In the final version of our project, we introduce a new feature: music emotion. The emotion model we use is Russell's Circumplex Model [7]. In this model, human's emotion can be conceptualized as a 2-dimensional bipolar space with four quadrants, as shown in Figure 3. The two dimensions can also be interpreted as arousal (A) and valence (V) values in literature.

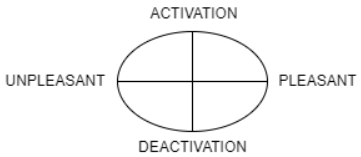


Figure 3: Russell's Circumplex Model for Human Emotions

The neural network is trained on an open-source dataset: 4Q-emotion dataset[8]. This dataset consists of 900 clips of music which gathered from AllMusic API, which are already classified artificially into 4 classes (corresponds to the 4 quadrants). Each clip is about 30 seconds long.

To train the neural network, a series of preprocessing is needed. We sampled each music signal at the sample rate sr mentioned before. A Mel spectrogram [5] is acquired for each clip by applying a Mel filter bank on the spectrogram of the sampled signal. An example of resulting Mel spectrogram is shown as the first figure of Figure 12. Also, we randomly split the entire dataset in a ratio of 4:1 to generate the training set and verifying set respectively.

The structure of our vanilla classifier is shown as Figure 4. The classifier mainly consists of three parts: a Mel encoder, a bidirectional RNN (LSTM [9]), and a stack of linear mappings. The Mel encoder takes the Mel spectrogram as input, mapping the complicated and lengthy Mel feature to a concentrated and multichannel intermediate feature. In each convolution block (3 blocks in total) in the encoder, we add a `MaxPool` layer to reduce the length of output feature maps, resulting a significantly reduced computation time of the subsequent LSTM. The 2-layer bidirectional LSTM takes the intermediate features from encoder as input, encoding the long-term memories into the hidden layer. Finally, the hidden layer of LSTM is extracted as the input of the stack of linear layers and mapped to the probabilities of the four classes. The loss function is the cross entropy loss.

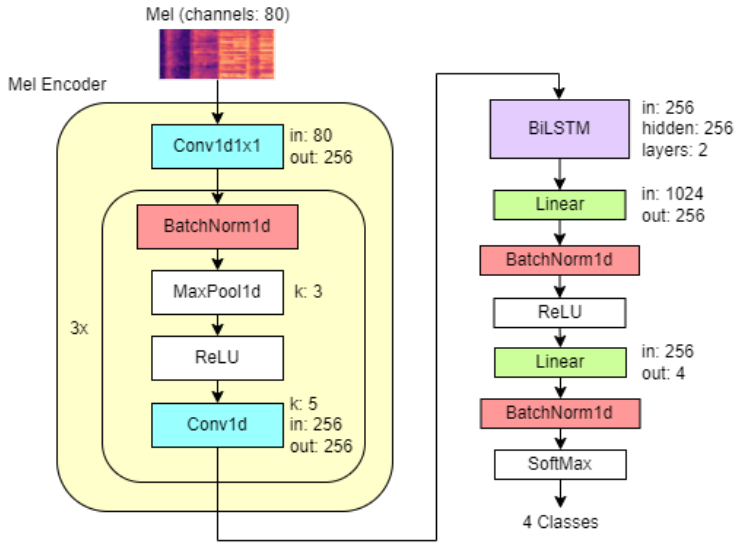


Figure 4: Neural network structure

The synthesis part takes all the features extracted above as input and generates a color sequence. A simple approach is HSV color model [10]. HSV model measures color in three metrics: hue, saturation, and value. Hue is measured with angle (0 - 360°). Therefore, a simple heuristic mapping is: the HSV value has a linear relationship with the energy per frame, while beats have a slightly smaller impact on it. Also, the emotion can be involved

in to concentrate the hue value roughly.

To implement a natural effect of beats, we need to smooth the beat sequence. We convolve a Hamming window with the beat sequence to produce a smoothed beat curve. The Hamming window has a length of 80% of the length of the average beat gap. This can be represented as:

$$\tilde{b}[i] = b[i] * h[i] = \sum_{m=-\infty}^{\infty} b[m]h[i-m] \tag{3}$$

where $b[i]$ is the beat sequence, which equals 1 if there is a beat at that time point and 0 if not. $h[n]$ is the Hamming window. $\tilde{b}[i]$ is the smoothed beat curve.

With these features ready, the heuristic formulas for HSV values can be determined:

$$\begin{cases} H[i] = E[i] + 0.6 \times \tilde{b}[i] \\ H[i] := D_H \times \frac{H[i]}{\max(H[j])} + \text{Stride} \times Q \end{cases} \tag{4}$$

$$\begin{cases} S[i] = E[i] + 0.5 \times \tilde{b}[i] \\ S[i] := D_S \times \frac{S[i]}{\max(S[j])} + (1 - D_S) \end{cases} \tag{5}$$

$$\begin{cases} V[i] = E[i] + 0.4 \times \tilde{b}[i] \\ V[i] := D_V \times \frac{V[i]}{\max(V[j])} + (1 - D_V) \end{cases} \tag{6}$$

where $H[i]$, $S[i]$, and $V[i]$ corresponds values of HSV, respectively. D_H, D_S and D_V remaps the normalized values to a specific space, where $D_H = 240$, $D_S = 0.8$ and $D_V = 0.6$. For example, D_H remaps the normalized H value (between 0 and 1) to 0 to 240, which is the coverage of the specific emotion. Stride = 90 for 4-Q model, meaning that one specific emotion concentrates the Hue value in a 90 range. Q indicates the emotion index, ranging from 0 to 3. $D_V = 0.6$, meaning that Value (brightness) will never be lower than 0.4.

Finally, the HSV values are translated to RGB values and stored in the memory. The overall algorithm can be described as the flowchart Figure 14 in Appendix A.

2.3.2 Dynamics Subsystem

The dynamics subsystem works as a preprocessor and transmitter of the joystick's data. It is the only connection spot of the joysticks. The button data from the joystick is simply transmitted to the control subsystem. For the posture data detected by the sensor, this subsystem will apply temporal and spatial remap on it, converting it to the gimbal command before send to the control subsystem.

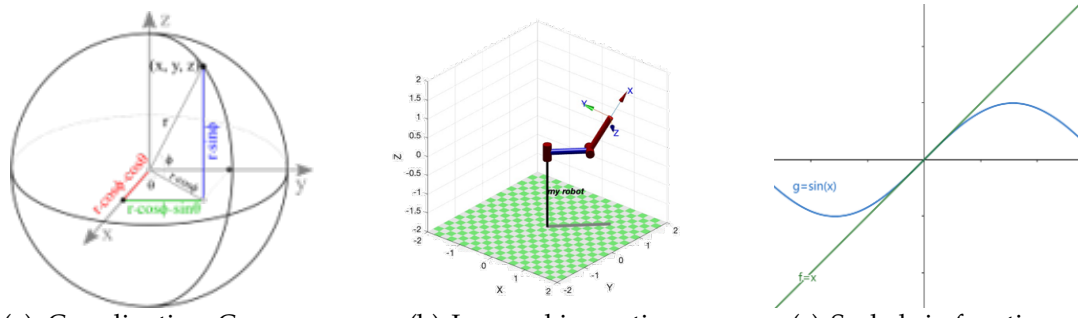
The temporal remap is used to restrain the sensor error caused by oscillation. As mentioned in the joystick system, the pitch and roll angle are derived from acceleration, where

severe oscillation along the y-axis will cause suddenly reversed sample data. When the joystick oscillates along the y-axis violently, the net acceleration may point upwards and the sign of angle will be the opposite during this period. Therefore, a **sign decision rule based on causal convolution** is implemented to decide the sign of a new data. With this algorithm, the sign of a new coming sample is decided by the Equation 7:

$$sgn(X_t) = sgn(\sum_{t=0}^{t=20} X_{T-t} * \delta(t)) \tag{7}$$

With observation and statistics, we found the reversed sample will take up for about 30% in severe oscillation and each period results in about 10 samples. The sum the last 20 samples can reveal the true offset of current posture while not obstruct the data to pass zero point. The Figure 13(a) shows that the algorithm can effectively eliminate reversed samples.

The spatial remap including three parts: the coordinate system conversion, the inverse kinematics and the non-linear upper bound clip.



(a) Coordination Conversion (b) Inverse kinematics (c) Scaled sin function

Figure 5: Spatial Remap

During the coordinate system conversion, the sensor data in Cartesian coordinate is converted to Euler angles, for further inverse kinematics processing. As a trade-off between the high sampling rate and low bandwidth occupation, this part has been finished in joystick in the Kalman filter algorithm. By doing so, accurate posture data can be derived with a high IMU sampling rate, even though the communication rate is relatively low.

To solve the inverse kinematics problem, we simplified the machine into a two-axis three-dimensional mechanical structure. The Denavit-Hartenberg parameter table of each joint of the manipulator (the main body of the robot arm) is obtained through measurement. By plugging in the parameters of this table, we successfully get the inverse kinematics of the manipulator.

Table 1: D-H parameters and tables

i	θ_i	d_i	a_{i-1}	α_{i-1}	rotation/translation
1	θ_1	212mm	0	$-\frac{\pi}{2}$	0
2	θ_2	0	90mm	0	0

Since the manipulator is an open-chain structure, after obtaining the joystick's position and attitude, the central processor leverages the Newton-Laffson method to solve its inverse kinematics. This process is essentially an application of a numerical iteration algorithm. As the initial value of the selected joint angle is close to the real value, the computation result converges. Therefore, the information on each joint angle can be obtained, and the orientation analysis and path planning can be executed.

For the third part, the non-linear upper bound clip is achieved with a scaled $\sin$ function as shown in Figure 5(c). The physical structure limits the max angle the servo can reach. To protect the servo, a threshold is set for each servo. Comparing the input and the output of the equation, we can find that, the more severe the posture diverges from the original position, the more the value declined. The shape of the sin function can be scaled by changing a and b in Equation 8 according to the threshold and domain. Parameters of our implementation have been mentioned in the R&V section.

$$f(x) = a\sin(b * x) \tag{8}$$

2.3.3 Control Subsystem

This subsystem plays the role of the mode controller and control frame synthesizer of the whole system. A control frame consists of the timestamp position of music, light color command, and movement command. It allows the light to move synchronously with music, which is important for User-friendliness in High-level Requirements.

Since this subsystem is the most expandable and flexible part of the project, we didn't make too much restrict requirement on it. So, we will focus on its design and function in this section. However, as the interface of the project, it can easily collaborate with other projects. For the example, the command can be converted to adapt DMX512 protocol with the third party package PyDMX[11].

By continuously receiving data from the message queue of the dynamics subsystem and music analysis subsystem, the control subsystem can get the in-time processed data. Inside this subsystem, a random generator and two finite state machines are maintained.

According to the light color command, the random generator would be able to translate the RGB color into HSV color and remap the Hue to the pitch and roll angle with a random factor. As a result, the light will move to the music when entering random mode. The generator can also generate random color commands by mapping the RGB light to HSV

light, rotating the Hue for a random angle, and mapping back to RGB. From the user's perspective, the color of lights can switch among several different types.

The data from the subsystem and that from the random generator is then filtered by the finite state machines. With the button data transmitted by the dynamics subsystem, the states can transit in different paths. Take the movement mode state machine as an example, a single click on the movement mode button will cause successive transitions of movement mode state. Specifically, it is in order of **mimic**, **repeat**, and **random**. If the instant state is **mimic** mode, the user can record its posture by holding the record button. This period of posture will be played repeatedly in the repeat mode. The color mode can be changed similarly by singly clicking the color mode button.

The instant states of the finite state machines will then be used as a filter of two sets of candidate commands as shown in the Figure 6.

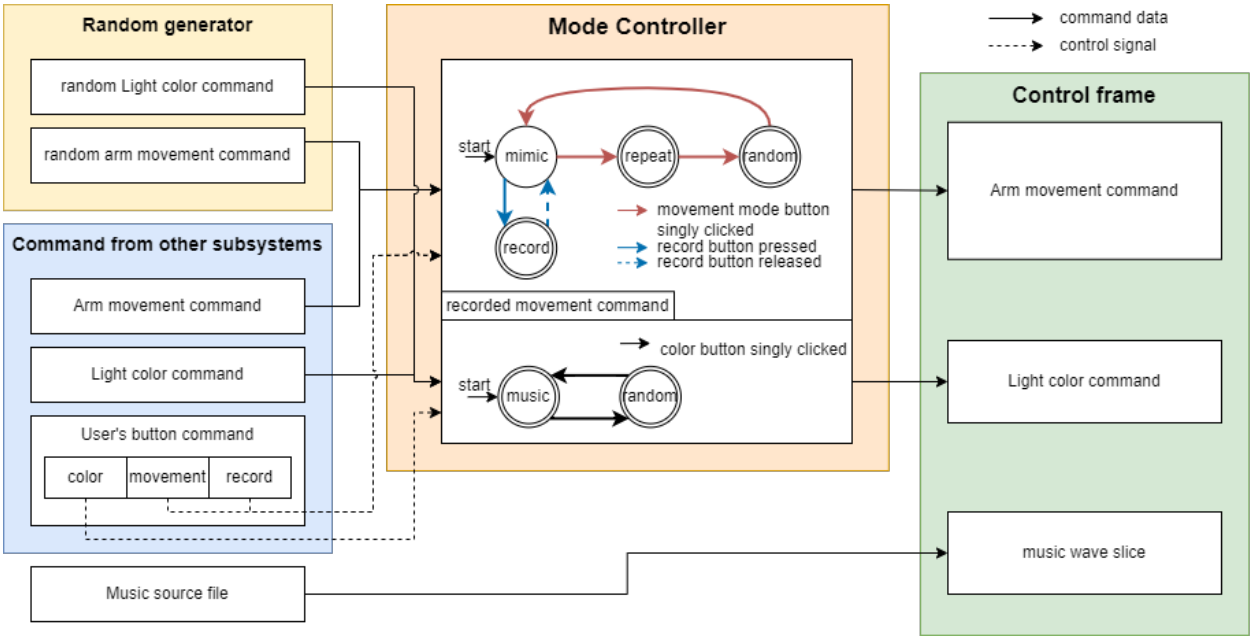


Figure 6: Dataflow chart of control subsystem

2.4 Light Array Subsystem

The light array subsystem consists of three parts: control, robot arm (or manipulator/effector), and light strip. The mechanical structure support physical support for the light gimbal.

2.4.1 Control

As discussed before, we use ESP32 as the controller, where FreeRTOS provides support for multi-threading. Four threads are running on the MCU, which are respectively: 1) TCP communication thread, 2) thread of pitch servo, 3) thread of roll servo and 4) thread

of LED strip. The TCP thread writes commands received from the PC to a message queue with the highest priority. The other 3 threads request access to the queue (request the lock) and fetch the corresponding commands (angle commands or color commands) and perform their functions.

The ESP32 is powered by two 18650 Li-ion batteries [12], which provides about 7.4 Volt. The regulator built on the ESP32 board will stabilize the power supply.

2.4.2 Robot Arm

The robot arm has two degrees of freedom (DOF), where each degree is supported by a motor. **We made two modifications in this part:**

1. In our original design, the robot arm is built along with the pitch and yaw axis, which is changed to the pitch and roll axis in the final design. This is because of the potential misalignment and instability. If we build the robot arm along with the pitch and yaw axis, the workspace is mainly upturned. The lamp tube would be disjointedly installed on the pitch axis, introducing unnecessary misalignment. If we build the robot arm along with the pitch and roll axis, the workspace is mainly antrorse, which is also conforming to usage habits.
2. In our original design, the yaw (now roll) axis is implemented by an RM6623 stepping motor. The RM6623 stepping motor is controlled by a CAN bus, which needs a CAN bus transceiver to communicate with ESP32. Unfortunately, we cannot acquire any CAN bus transceiver due to the pandemic. The production of RM6623 is also suspended. Therefore, we use another Dsservo servo to replace the stepping motor.

We use two Dsservo DS3218 [13] servos to support the 2-DOF robot arm. The MCPWM peripheral (Motor Control Pulse Width Modulator) [14] of ESP32 provides support for PWM waveform generation. In this project, we only use `Timer` and `Generator` submodules of MCPWM to generate control waveforms.

We use servos with different workspaces. The roll servo has an angle range of 360°, while the pitch is 270°. In practice, the workspace will be further limited and remapped. The maximum pulse width for DS3218 servo is 2500 ms, while the minimum is 500 ms, which corresponds to the full duty cycle and zero respectively.

The servos are directly powered by two 18650 Li-ion batteries. The signal wire is connected to one of the GPIOs on the ESP32.

2.4.3 Light Strip

We use WS2812 [15] as the light source. WS2812 is an intelligent control LED light strip. Each pixel of RGB colors can achieve a 256 brightness display, or a completed 16777216 full-color display.

The Remote Control (RMT) [16] peripheral built-in ESP32 is used to generate the control

signal. The WS2812 is controlled by a carrier modulated sequence waveform shown as Figure 7. We leverage the carrier modulation function built in the RMT peripheral of ESP32. A RMT adapter can be implemented to translate the RGB values and write the `T0H` ticks, `T0L` ticks, `T1H` ticks, and `T1L` ticks to the RMT buffer. An example implementation can be found in [17].

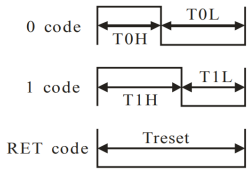
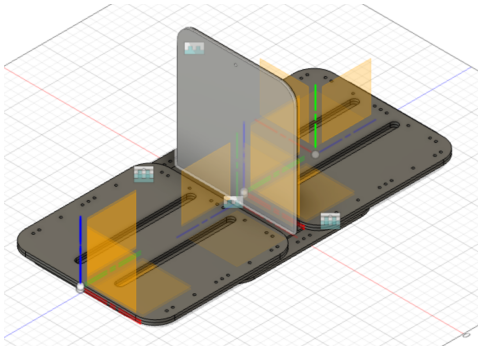


Figure 7: WS2812 Sequence Chart

We use a small LED strip with 16 pixels, mounted on a 3D-printed carriage. The LED strip is also powered by two Li-ion batteries. The signal wire is connected to one of the GPIOs on the ESP32.

2.4.4 Mechanical Support

In order to convert the yaw axis and pitch axis into pitch axis and row axis, we need to rotate the Angle of the manipulator by 90°. And because the end of the mechanical arm will form a large moment of inertia under high-speed movement, our base needs to have enough weight to support the movement of the mechanical arm above us.



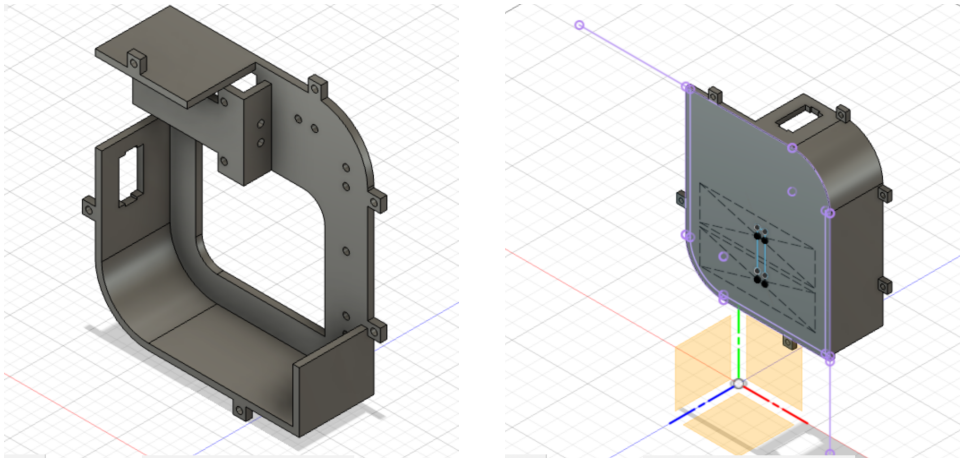
(a) Mechanical design of light base

(b) Profile display of light

Figure 8: Base constructed by acrylic sheet and mechanical structure

Due to material limitations, we use two steel bookshelves as the main body and assemble it through acrylic panels with properly planned 3 mm holes. Depending on the length of screws we have, we need to assemble our brackets with more complicated mechanisms and arrangements. In addition, we arrange our electronic components through reasonably designed boxes to ensure safety and improve aesthetics. Reasonable connection and discharge structure ensures the convenience of battery replacement.

All design use rounded corner and chamfer structure to ensure aesthetically pleasing appearance and user safety.



(a) upper box mechanical design (b) lower box mechanical design

Figure 9: Mechanical design of storage box for electronic components

We link pitch servo and row servo through cross roller bearings to ensure that the rotation axes of pitch servo and row servo cross each other on the basis of improving the stability of operation and reducing the mechanical sound, so as to avoid unnecessary jitter and mechanical loss.

In the assembly of mechanical arm, we try to use standard parts and screws for steering gear. So that even if there is any failure, our users can view and repair.

Through the arc structure and special inner grain, the lighting intensity is enhanced and the light is concentrated.

It is able to be contained in a 20 litre bag and weights 3.3 kg. It also has a single-hand pull structure. It's very comfortable to extract the arm with one hand.

(a) A structure that gathers light energy (b) lower box mechanical design

Figure 10: Photo of pull by single hands

3 Cost and Schedule

3.1 Cost

Name	Hourly Rate	Hours	Total	Total*2.5
Ruiqi Li	¥180	300	$8640 (=¥54000)	$21600 (=¥135000)
Haozhe Chen	¥180	300	$8640 (=¥54000)	$21600 (=¥135000)
Zhaohua Yang	¥180	300	$8640 (=¥54000)	$21600 (=¥135000)
Anqi Tan	¥180	300	$8640 (=¥54000)	$21600 (=¥135000)
Total				$86400 (=¥540000)

Table 2: Labor

Description	Quantity	Manufacturer	Vendor	Cost/Unit	Total Cost
ESP32-WROOM-32	4	espressif	Taobao	$3.44	$13.76
Thicken bearing	4	ZCD	Taobao	$3.6	$14.4
Servo	2	Dsservo	Taobao	$20.928	$41.856
RM6623 stepping motor	2	DJI	Taobao	$143.68	$287.36
LED ws2812 RGB borad	5	Aobie	Taobao	$8.47	$42.35
MPU9250	5	Mumu	Taobao	$7.25	$36.25
Screws, nuts and copper posts	100 each	Huitai	School lab	$0.21	$63
3D printed consumables	23582mm	PMAX	School lab	$0.01613	$38
Laser cutting consumables	5	Shenmei	School lab	$4.03	$20.16
Miscellaneous	n	/	School lab	/	$293.14
Total					$364.73

Table 3: Parts

Section	Labors	Parts	Grand Total
Total(s)	$86400 (=¥540000)	$364.73 (=¥2,279.6)	$86757.37 (=¥542233.6)

Table 4: Grand Total

3.2 Schedule

Week	Task	Responsibility
03/14/22	**Project Proposals Due (Mon.)**	All
	Development board decision and purchase	Zhaohua Yang
	Prepare design review	Haozhe Chen
03/21/22	Preliminary analysis of music signals	Ruiqi Li
	Research and select all hardware modules	Anqi Tan
	Purchase hardware & all parts, rough 3D modeling	Anqi Tan
03/28/22	Study datasheet for sensor, microcontroller, bluetooth modules	Haozhe Chen
04/04/22	Program microcontroller & control button	Zhaohua Yang
	Overall 3D modeling and preparation for 3D printing	Anqi Tan
	Assemble and test communication module	Haozhe Chen
04/11/22	**Individual Progress Report (Mon.)**	All
	Complete the printing, machining and assembly of non-standard parts	Anqi Tan
04/18/22	Test the lighting system and the movement of gimbal	Anqi Tan
	Test the joystick system and the connection to central server	Zhaohua Yang
04/25/22	Black-box testing on individual components	Ruiqi Li
	Overall control system code debugging	Haozhe Chen
05/02/22	**Prepare Mock Demo**	Haozhe Chen
	Confirm the achievement of the overall function	Ruiqi Li
	Finally debug the remaining problems	Haozhe Chen
05/16/22	**Finalize Demonstration**	All
	Finalize Presentation (Fri.)	All
05/23/22	**Final Report due (Tue.)**	All
	Lab checkout and finalize final paper	Anqi Tan

Table 5: Schedule

4 Requirements, Verification and Results

4.1 Joystick Subsystem

Requirements	Verification
posture sensor 1) The mechanical frequency of gyroscope is X_Axis: 33kHz, Y_Axis: 30kHz, Z_Axis: 27kHz.	1) The delay of movement between operator's arm and robot arm should be no longer than 500ms.
MCU 1) work at about 3.3V with a minimum current 0.5A[18]. 2) Button should be debounced and when pressed or released. 3) The situation where MCU stop working due to the lack of voltage should be avoided.	1) Use multimeter to check if the voltage and current satisfy the requirements. Use an oscilloscope to check if the voltage signal is steady. 2) One side of the button is connected to a PULL-UP-ONLY port of ESP32 and another to the ground. We define pressed=1, release=0. Using oscilloscope to measure the bounce time. The debounce succeeds if the debounce time is less than 10ms. 3) we need a power supply supervisor, to lower the CHIP_PU if detect that the voltage is less than 2.3V. Then we just see if ESP32 will shut down when working.

We set wireless communication the highest priority and enhance the frequency of CPU. The overall delay is about 200ms. The working current is about 100 mA, which enables the joystick to work for about 15 hours. For the debounce, the last state and current state are recorded. We read after signal stays constant for a period of time. Finally, given all wires configured well and charged batteries, the situation where ESP32 stops working did not happen.

4.2 Communication Subsystem

Requirements	Verification
Wireless environment 1) The WiFi router should support IEEE 802.11n protocol. 2) The wireless signal should be strong enough for maintaining stable communication. The typical demonstration environment can be simplified as an 8m*8m*3m room containing all devices and the WiFi router. The router should ensure the bottom threshold of WiFi signal strength of all devices is higher than -70dBm in RSSI (Received Signal Strength Indication).	1) Lenovo XIAOXIN Pro16 is used to establish the wireless network. The network card: `Intel Wi-Fi 6 AX200` supports IEEE 802.11n IEEE WLAN standard and can be used as router as mentioned in [19]. 2) **(Modified)** The ESP32 is used for signal strength detection. 1. **The FreeRTOS, instead of the MicroPython** [20], is used as the framework for embedded programming. The test program is written to record the signal strength and the instant position by integrating the accelerations from the IMU. 2. Put the ESP32 and the router into an 8m*8m*3m room. Start the WiFi router and the ESP32 and establish connection. 3. Hold the ESP32 board and walk around the room, and analyze the signal strength record.

The test program reads the RSSI from network module on ESP32 and integrates the position with accelerations from the IMU MPU9250. By analyzing the record data, a 3D signal strength heatmap can be drawn, as shown in Figure 11. The minimum signal strength detected is -67.5dBm, which meats the requirement.

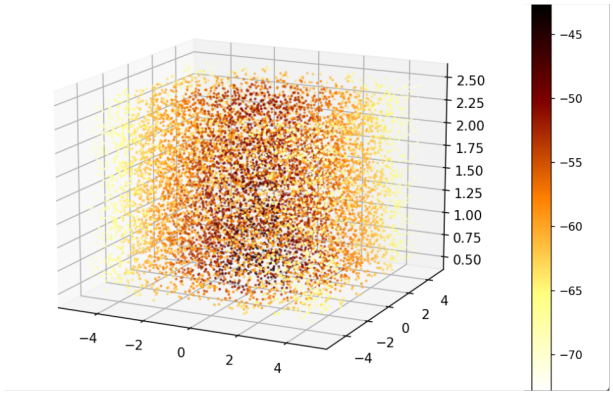


Figure 11: Scatter of WiFi signal strength in RSSI

Requirements	Verification & Results
Central server 1) The central server should provide a network card which supports IEEE 802.11n protocol. 2) The operating system of central server should support network and parallel programming.	The programs are coded with Python 3.7 and successfully passed the tests on the following devices: Lenovo XIAOXIN Pro16 (Windows11), Microsoft Surface Pro5 (Windows10) and is Mac pro 2021 (Monterey). All the devices support IEEE 802.11n IEEE WLAN standard and parallel programming.
Embedded development boards 1) The development board should provide hardware that supports the IEEE 802.11n protocol. 2) The development board framework should support network I/O programming.	According to [21], the ESP32 WiFi Radio and Baseband support the 802.11 b/g/n IEEE WLAN standard. **The FreeRTOS framework** is chosen to support network I/O programming and multi-task parallel programming. The sensor data reading, communicating, effectors controlling has been set to different tasks.
Communication stability 1) The service on a central server should about to maintain at least 32 connections concurrently and handle them in parallel. 2) The delay between two devices should be limited to 30ms. 3) The connection failure can be detected in 100ms and new connection will be automatically established for repairing.	1) Threading programming is utilized in central server programs to make parallel execution possible. A front-end application will be developed to monitor the running threading pool. 2) **(Modified) The delay has been mainly reduced by modifying the timer interval and buffer size in socket configuration.** The timestamp is still acquired from SNTP server with LwIP protocol and used for avoid redundant messages. 3) **(Modified)** There will be a daemon listening program in socket server that will always try to accept new connections from other devices. **A hotplug re-connection mechanism has been achieved. All devices can automatically reconnect without any other action after restart.**

4.3 Music-analysis Subsystem

Requirements	Verification
Music Analyzer Service The required parameters are specified in the Design part and the Design Document. The analysis part involves 3 music features: tempo, beat and energy. The synthesis part consists of a series of heuristic formulas. The colors should change visually with the music. Also, the computation time should be less then 3 seconds.	**(Modified) A neural network classifier for music emotion recognition is added.** One can install the requirements specified in Design Document and run the algorithm. The color change is obvious. The computation time can be measured by `Python` standard library `time`.
Development Environment 1) We use `Python` as development language. We recommend `Anaconda` as the package and environment manager. The environment is constructed on a PC with Microsoft Windows 10 operating system. 2) To specify the environment and version of libraries, we use: 1. `python==3.7`. 2. `librosa==1.18.5`. 3. `numpy==1.18.5`. 4. `ffmpeg`. We recommend installing this using `conda-forge`.	1) The OS type of PC and the configuration of `Anaconda` is easy to verify. If you configure the environment variable for `Anaconda`, you can easily launch environment by typing `conda activate` in `Anaconda prompt CMD` or `power shell`. 2) Firstly activate the corresponding `Anaconda` environment in the `Anaconda prompt`. Then type `conda list` to generate all the packages installed and check if the requirements are met.

The simulation of color synthesis can be shown in Figure 12. The color is changing according to the energy and beats successfully. The emotion recognized also constrains the range of variation.

The neural network is trained from scratch, which achieves an accuracy of 70%. The training on a PC is extremely time-consuming. To accelerate convergence, we add multiple. `BatchNorm` layers [22]. In our test, the computation time for a regular pop song (4 minutes long) is at most 0.5 seconds on an Intel gen 11 i5 CPU.

The average overall computation time is listed as:

Process	loading files	beat	energy	emotion	color mapping
Time(s)	6.12	1.39	0.16	0.40	0.03

Table 10: Average Time Cost

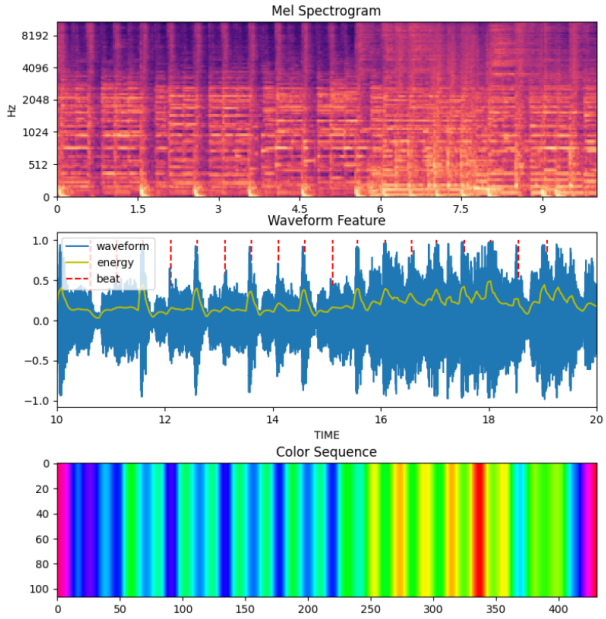


Figure 12: An example of color sequence output

The overall computation time is below 2 seconds. It is worth mentioning that a large proportion of the time cost is consumed to load the file. It is because we use MP3 files as music source, which needs a long time to load and decode. This can be improved by using streaming reading in the future work. Streaming reading also allows real-time processing.

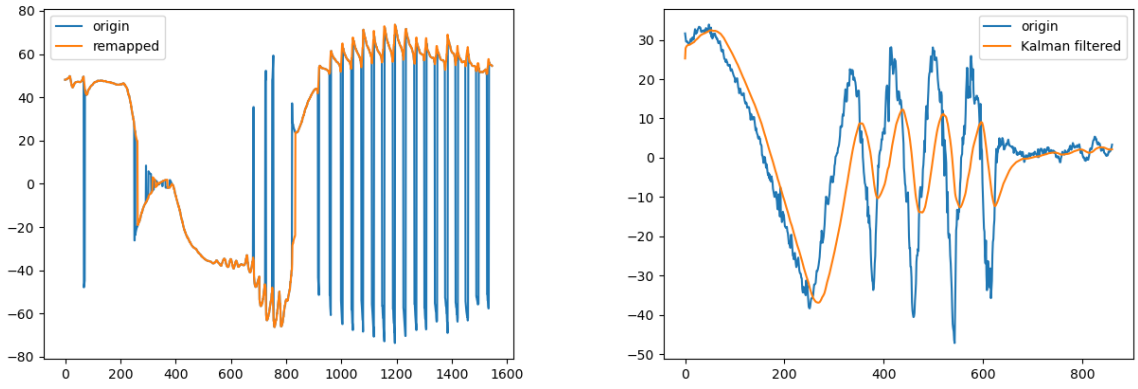
4.4 Dynamics Subsystem

Requirements	**Verification & Results**
Temporal remap 1) Smooth the waveform of the posture sequence to avoid fierce movement. The frequency of orientation change should be less than 5/sec.	1) **(Modified)** ~~A low pass filter will be utilized to inhibit the posture signal whose frequency is above 5Hz.~~ A causal convolution based sign decision rule is used to eliminate the reversed sample caused by oscillation. The non-linear clip also avoids the violent movement near the servo movement threshold. The 13(a) gives a visual result.

Requirements	Verification & Results
Spatial remap 1) Predict and fill missing values using known values. The predicted value should differ from the real value with less than $\pm 1\%$. 2) Discard values that are far away from the mean. With a fixed-width sliding window of 0.1 seconds, the point with ± 3 standard deviation will be identified as an outlier and will be filled with a new value with the method mentioned in 1). 3) The non-linear projection function should be clip the upper bound to any given variable. It also should almost-linear in the gimbal movable range and become gentle when near the boundary.	1&2) 10% of posture data points will be removed from a sequence before feeding to the algorithm. The generated results of those missing points will be collected and the percentage mean of the difference between the real generated data will be calculated. 2) **(Modified)** ~~A modified log function would be able to handle the clip requirement. $f(x) = a * \log(b * x + 1)$~~ **A scaled sin function is utilized to clip the upper bound.** $f(x) = a * \sin(bx)$ where a and b are adjusted according to servo movement threshold. In the implementation, a is set 180 for roll servo and 90 for pitch servo, b is set to be $\frac{\pi}{2*360}$.

The temporal remap has been achieved with a causal convolution based sign decision rule mentioned in design section. The reversed samples has been effectively eliminated as shown in the Figure 13(a).

Kalman filter [23] is used to predict the missing or wrong data point. The test mentioned in verification has been successfully passed and Figure 13(b) shows its effect. Comparing the filtered samples versus the original samples, the noise of sensor has been reduced and the wave form is more smooth.



(a) Causal convolution based sign decision rule (b) Kalman filter

Figure 13: Dynamics Remap Results

4.5 Lighting Array Subsystem

Requirements	Verification & Results
Servo for pitch shaft on the gimbal Given a PWM pulse width, the SERVP output Angle can be locked, reducing the delay for arm movement following. It has a large working Angle. In addition to mechanical foot collision, it can restore human arm movements as much as possible. **Motor for yaw axis** A RM6623 stepping motor is used to support the yaw axis of the robot arm.	Using the digital servo rather than the analog servo, the minimum control Angle is below 0.18°. **(Modification)** ~~Choose servo with motion Angle over 360 °.~~ **We use servos of 270° and 360°.** **(Modification) First, we cancel the yaw axis of the robot arm due to essential non-axiality and instability. Second, we replace the RM6623 with another servo due to the lack of CAN bus transceiver.**
Light Strip 1) Intelligent back connection protection, power back connection will not damage IC. 2) The three primary colors of each pixel point can achieve a 256-level brightness display and the scanning frequency is not less than 400Hz/S. 3) Can directly connect to DMX512 lighting control protocol.	**LED ws2812 rgb development board** 1) **(Modification)** ~~Built-in power-on reset and power-off reset circuit to protect the circuit.~~ **We use the power regulator built in ESP32 to protect the circuit.** 2) Naked eye observation of light waveform changes following the change of music rhythm. 3) RS-485 bus interface is required (mentioned in the Control section).
Bearing with bearing force at the bottom 1) It can absorb more than 300kN vertical pressure while maintaining smooth rotation. 2) The inner and outer rings can be fixed with the bottom and head respectively in a relatively simple way, while saving the installation space. 3) High speed and excellent rotation accuracy, full softening.	**crossed roller bearing** 1) Because the roller is cross arranged, so only one set of cross roller shaft rings can bear the load in all directions. The rigidity is improved 3 to 4 times. 2) Cross roller bearing is fixed with the cross roller shaft ring to prevent mutual separation, making the operation simple when installing the cross roller shaft ring. 3) Because the inner ring or outer ring of the cross roller bearing is two separate structures, the bearing clearance can be adjusted.

5 Conclusion

5.1 Accomplishments

1. **Precision**. We successfully make the robot arm of the light mimic the movements of the joystick within a certain range. Due to the limitation of the servos' workspace, the actual range of movements is clipped and remapped. The delay between the joystick and the robot arm of the light is reduced down to 200 ms, while the delay between pure TCP clients (without effectors) is about 20 ms.
2. **Portability**. All the components of our project are portable and wireless. The joystick and the light are powered by rechargeable Li-ion batteries. One pair of joystick and light weigh only 4 kg and can be contained in a 20L bag. The joystick can last 15 hours and the light can last 4.5 hours. The program of the central server is tested on multiple machines like MS Windows 10/11 and Apple macOS.
3. **User-friendliness**. The control of the light relies entirely on intuition. Users do not need professional training to start the operation. We successfully implement the alternative control mode: imitation, recording, repeating, and randomly moving. The transition between these modes is achieved by simply pressing the buttons. The color of the light can be determined by the inherent feature of the music being played. One can easily distinguish the hue, the saturation, and the brightness of the color varying with the music. The analysis of the music is done before the music starts playing, which has a time cost of less than 0.5% of the length of the song.

5.2 Uncertainties

1. The workspace of the robot arm is limited. The servo for the pitch axis has a rotation range of 270°, while the servo for the roll axis is 360°. It is worth mentioning that each servo is unable to rotate infinitely in one direction. Therefore, users need to control the light within a limited workspace.
2. Due to the uncertainty inside the servos, it is possible to rotate over 360°. In this case, the wires will get twisted and cause damage. A preliminary solution is to add an electric slip ring in the center of the bearing so that the upper and lower portions of the wires become isolated and the twisting is prevented.
3. Although the overall delay is successfully reduced to 200 ms, it is still a considerable latency. The pure communication delay only contributes 20 ms, while the majority of the time is caused by the computation of control units and execution of motors. As tested, it takes 100 ms for a filtered instruction to be fully executed (without communication).

5.3 Future Work

Hardware	Software
1) **Delay** The response time of the servo can be further reduced by replacing the servo with a better type or a step motor with a PID control algorithm. 2) **Rotation range** The chassis can be made to rotate continuously 360 degrees by using a step motor. The winding problem of the power supply can be solved with an electric slip ring. 3) **Beam focus** The beam of the light can be further focused. A lens with a better focus effect and lighter weight can be utilized. By doing so, the wobble problem of the robot arm will also be solved. 4) **Array** More lights can be built to form an array to get a better stage performance.	1) **Decoupling** The subsystems can be further decoupled to adapt requirements more flexibly. 2) **Parallel Program** The parallel program model of central server can be optimized to save computer resource; and the music analysis can be made in real-time, using streaming read. 3) **Front-end App** The front-end App can be designed more user-friendly. Functions likes supporting switching songs, adding new songs, creating song list can be added.

5.4 Ethics

Ethics of our product

Our group guarantees that the robot will not use strong light that can harm people's eyes by properly controlling light energy and divergence. We will reinforce the gimbal and try our best to avoid potential safety hazards caused by the falling of the light during use. We will do our best to remind our users to avoid using lights late at night, to avoid disturbing others' sleep.

Ethics of research and development

Our group guarantees that the code of this product does not involve any plagiarism Our group guarantees that this product modeling will not copy any existing modeling solutions Our group states that this product will not use any solution of existing products on the market.

Ethics sources

This project is bound and inspired by the Code of Ethics published by professional societies, such as IEEE and ACM. Members of our group will urge each other to comply with the IEEE Code of Ethics[24] and strive to ensure that the Code is followed without retaliation against individuals who report violations.

References

[1] J. Primrose, *Stage lighting system*, [Online], https://www.theatrecrafts.com/pages/home/topics/teaching/lighting/system/ Accessed March 23, 2022.

[2] USITT, *Dmx protocol basics*, [Online], http://www.dmx-512.com/dmx-protocol/dmx-protocol-basics/ Accessed March 23, 2022.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[4] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[5] Wikipedia, *Mel_scale*, [Online], https://en.wikipedia.org/wiki/Mel_scale Accessed March 23, 2022.

[6] LDT, *Librosa.beat.beat$_t$rack*, [Online], https://librosa.org/doc/latest/index.html Accessed March 23, 2022.

[7] J. Russell and G. Pratt, "A description of the affective quality attributed to environments," *Journal of Personality and Social Psychology*, vol. 38, pp. 311–322, Feb. 1980. DOI: 10.1037/0022-3514.38.2.311.

[8] R. Panda, R. Malheiro, and R. P. Paiva, "Musical texture and expressivity features for music emotion recognition," in *International Society for Music Information Retrieval Conference*, 2018.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[10] A. R. Smith, "Color gamut transform pairs," in *SIGGRAPH '78*, 1978, pp. 12–19.

[11] J. Allen, *Pydmx*, [Online], https://github.com/JMAlego/PyDMX Accessed May 2, 2022.

[12] SmTech, *18650 datasheet*, [Online], https://somanytech.com/18650-battery-specifications-datasheet-18650-battery-specs/ Accessed May 22, 2022.

[13] Dsservo, *Ds3218 datasheet*, [Online], https://dsservo.com/d_file/DS3218%20datasheet.pdf Accessed May 22, 2022.

[14] espressif, *Esp32 mcpwm*, [Online], https://docs.espressif.com/projects/esp-idf/zh_CN/stable/esp32/api-reference/peripherals/mcpwm.html?highlight=mcpwm Accessed May 22, 2022.

[15] Worldsemi, *Ws2812 datasheet*, [Online], https://d2j2m4p6r3pg95.cloudfront.net/module_files/led-cube/assets/datasheets/WS2812B.pdf Accessed March 27, 2022.

[16] espressif, *Remote control (rmt)*, [Online], https://docs.espressif.com/projects/esp-idf/zh_CN/stable/esp32/api-reference/peripherals/rmt.html Accessed May 22, 2022.

[17] ——, *Rmt example*, [Online], https://github.com/espressif/esp-idf/blob/v4.4.1/examples/peripherals/rmt/led_strip/main/led_strip_main.c Accessed May 22, 2022.

[18] ——, *Esp32-wroom-32 datasheet*, [Online], https://www.espressif.com/zh-hans/support/documents/technical-documents Accessed March 23, 2022.

[19] Intel-Corporation, *Intel® wi-fi 6 ax200 cyclone peak 2 product brief*, [Online], https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/wi-fi-6-ax200-module-brief.pdf Accessed March 24, 2022.

[20] MicroPython, *Micropython esp32 firmware*, [Online], https://micropython.org/download/esp32/.
[21] espressif, *Esp32 datasheet*, [Online], https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf Accessed March 23, 2022.
[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Feb. 2015.
[23] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.
[24] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

Appendix A Appendix

A.1 Music Analysis Flow Chart

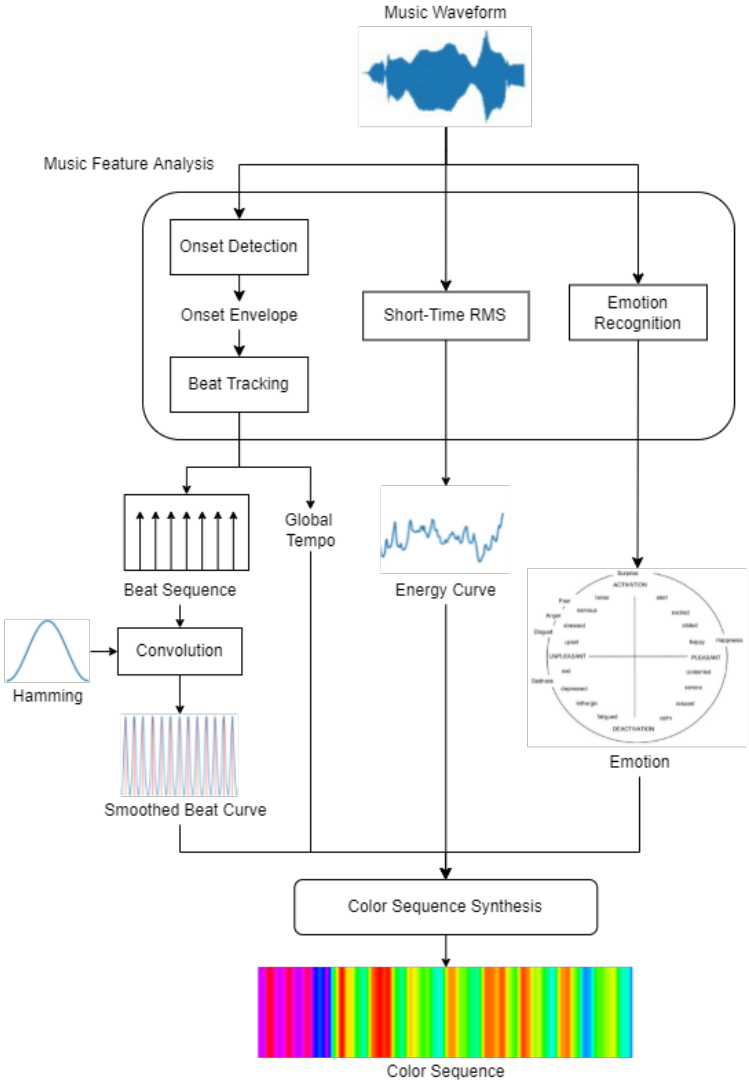


Figure 14: The overall flowchart of the algorithm.

A.2 Introduction of Kalman filter

Kalman filter is a recursive algorithm that can give the best estimation to linear system. It gets Kalman gain by minimizing error to do the posterior estimation.

Time update equation:

$$\hat{x}_{\bar{k}} = A\hat{x}_{k-1} + Bu_{k-1} \tag{9}$$

$$P_{\hat{k}} = AP_{k-1}A^T + Q \tag{10}$$

Status update equation:

$$K_k = \frac{P_{\bar{k}}H^T}{HP_{\bar{k}}H^T + R} \tag{11}$$

$$\hat{x}_k = \hat{x}_{\bar{k}} + K_k(z_k - H\hat{x}_{\bar{k}}) \tag{12}$$

$$P_k = (I - K_kH)P_{\bar{k}} \tag{13}$$

$\hat{x}_{k-1}$, $\hat{x}_k$, respectively represent the posteriori state at $k-1$ and k. It is the result that we want. Actually it is the best estimation at a specific time

$\hat{x}_{\bar{k}}$ is the Prior state estimation at k, and it is the intermidiate step during filtering. That is, it predicts the result at k using the best estimation at $k-1$.

P_{k-1}, P_k respectively represent the posteriori estimation covariance, which is the intermidiate step.

H is the transformation matrix from status variable to measure, and it represents the relationship between status and measurement.

z_k is the measurement, which is the input to the filtering.

K_k is the gain matrix, which is the intermidiate step.

A is the status transformation matrix. Actually it is a hypothesis for the object status transformation.

Q is the Process excitation noise covariance, which used to represent the difference between transformation matrix and the real process.

R is the noise covariance. In reality, R can be measured, so it is the known for filtering.

B is the matrix that transform inputs to status.

$z_k - H\hat{x}_{\bar{k}}$ is the difference between real and predicted value. It adjust priori with kalman gain and get the posteriori.