A WEB OF THINGS SYSTEM TO CONTROL PARTY EQUIPMENT

ERAY EROĞLU

MEF UNIVERSITY

JANUARY 2024

MEF UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

MASTER'S IN INFORMATION TECHNOLOGY

M.Sc THESIS

A WEB OF THINGS SYSTEM TO CONTROL PARTY EQUIPMENT

Eray EROĞLU

ORCID NO: 0009-0001-6501-3879

Thesis Advisor: Prof. Dr. İlker BEKMEZCİ

ABSTRACT

A WEB OF THINGS SYSTEM TO CONTROL PARTY EQUIPMENT

Eray EROĞLU

M.Sc in Information Technology

Thesis Advisor: Prof. Dr. İlker BEKMEZCİ

January 2024, 73 Pages

This thesis focuses on the equipment needed to throw a successful party and how to coordinate and synchronize these components seamlessly. It also explores the integration of the Web of Things (WoT) paradigm into party organization to improve coordination and communication between entertainment devices. The research aims to provide a detailed description of the necessary equipment and presents innovative applications of the WoT paradigm to enhance the coordination and synchronization of entertainment elements. By doing so, this thesis contributes to the evolving field of WoT and its potential applications in dynamic and interactive environments such as parties.

Keywords: Party equipment, WebSocket, WoT, IoT, light robot, Raspberry Pi, fog machine, entertainment system, home party system.

Numeric Code of the Field: 92417

ÖZET

PARTİ EKİPMANLARINI KONTROL ETMEK İÇİN BİR NESNELER AĞI SİSTEMİ

Eray EROĞLU

Bilişim Teknolojileri Tezli Yüksek Lisans Programı

Tez Danışmanı: Prof. Dr. İlker BEKMEZCİ

Ocak 2024, 73 Sayfa

Etkili bir parti düzenlemenin temel unsurlarını incelemekte ve başarılı bir etkinliğin düzenlenmesi için gereken ekipmanları ve bu bileşenlerin nasıl koordine edilip bir bütün halinde çalışabileceğini bu tez kapsamında araştırılmış ve geliştirilmiştir. Bu bağlamda, özellikle eğlence cihazları arasındaki koordinasyon ve iletişimin önemi üzerinde durulmaktadır. Bununla birlikte Nesnelerin Web'i (WoT) paradigması üzerinden parti organizasyonuna entegrasyonunun nasıl uygulanabileceğini gösterilmektedir. WoT, nesnelerin interneti üzerinden fiziksel cihazların birbirleriyle iletişim kurmasına ve etkileşimde bulunmasına olanak sağlayan bir iletişim ağıdır. Bu tez, WoT'un parti organizasyonunda nasıl kullanılabileceğini ve eğlence unsurlarının koordinasyonunu ve senkronizasyonunu artırmak için nasıl bir araç olabileceğini ortaya koymaktır. Bu çalışmanın bir diğer amacı da, parti organizasyonu için temel ekipmanların belirlenerek son kullanıcılar tarafından gerekli isteklerin uygulanmasıdır. Hangi cihazların gerekliliği, bu cihazların fonksiyonları ve nasıl entegre edilebilecekleri gibi konular detaylı bir şekilde titizlikle ele alınmaktadır. WoT'un gelişen alanına ve dinamik, etkileşimli ortamlardaki potansiyel uygulamalara katkı sağlamayı amaçlamaktadır. Partiler gibi sosyal etkileşim odaklı etkinliklerde WoT'un nasıl kullanılabileceğini anlamak, bu alandaki araştırmalara yeni bir perspektif kazandırarak literatüre yeni projeleri kazandırabilir.

Anahtar Kelimeler : parti ekipmanları, WebSocket, WoT, IoT, ışık robotu, Raspberry Pi, sis makinesi, eğlence sistemi, ev parti sistemi.

Bilim Dalı Sayısal Kodu: 92417

ACKNOWLEDGEMENT

During the course of this thesis, I would like to express my deep gratitude to Okan Serbest, who I have worked with at every stage of the project and who has played a vital role in the success of this collaborative study. Okan's expertise, guidance and cooperation were critical to the successful completion of this thesis.

Our collaboration with Okan Serbest involved a mutual exchange of ideas at many critical stages, from project design to data collection and analysis. Okan's valuable contributions and suggestions were an important factor in improving the quality of the thesis.

I would also like to thank Prof. Dr. İlker Bekmezci for supporting me throughout this process and ensuring that this project contributes to the academic literature, as well as my colleagues, family and friends. Their encouragement and understanding motivated me throughout this challenging process and helped me to successfully complete the thesis.

I would like to express my sincere thanks to everyone who has contributed to the completion of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Throwing a successful party requires specific equipment, including speakers, light robots, and a fog machine. The challenge, however, is to orchestrate these components so that they integrate seamlessly and synchronize effortlessly with any song. The aim of this thesis is to provide a detailed explanation of the equipment required to throw a party, and a show into how to coordinate these elements in an organized manner. In addition, this research will investigate the integration of the Web of Things (WoT) paradigm into the party setup, exploring how WoT principles can enhance the coordination and communication between these entertainment devices [22]. In addition to the technical aspects of the equipment required for a successful party, it also presents the innovative application of the Web of Things paradigm to improve the coordination and synchronization of these entertainment elements. In this approach, the thesis aims to contribute to the evolving field of WoT and its potential applications in various domains, including the dynamic and interactive setting of a party environment [23].

In the realm of event engineering, orchestrating a memorable and immersive party experience involves the meticulous coordination of essential components. These basic elements include speakers, lighting robots and a fog machine, each of which contributes to the overall sensory experience. While these components individually serve different purposes, their seamless integration is critical to creating a harmonious and captivating party atmosphere.

Figure 1.1 : Transformation Party Box

As the auditory heartbeat of the event, the sound system is responsible for providing the sonic landscape. Choosing a high-quality loudspeaker system that complements the venue's acoustics is paramount. In addition, the implementation of advanced sound processing technologies can enhance the audio experience, ensuring clarity and resonance across different musical genres. Lighting robots, designed to visually dance to the rhythm of the music, play a key role in enhancing the aesthetic appeal of the party. Equipped with intelligent lighting systems, these robots synchronize their movements with the musical beats, creating a visually mesmerizing display. The choice of programmable and responsive lighting technology is key to creating a dynamic and immersive light show.

To add an ethereal and dynamic layer to the party atmosphere, a fog mechanism becomes an integral component. When strategically dispersed, fog interacts with both lighting and sound elements, enhancing their visibility and impact. A well-calibrated fog system ensures that light rays are beautifully refracted, creating captivating visual effects that respond dynamically to the music. In the coming sections of this thesis, an in-depth investigation will be carried out to elucidate the intricate details of each component. This will include the technical specifications required for optimal performance, methods for

synchronizing to music, and the underlying engineering principles that govern their coordinated operation. The aim is to produce a comprehensive.

1. REDEFINING SOCIAL EXPERIENCES AT THE HOME PARTY

The negative impact of the entertainment sector during the pandemic can affect the psychological and emotional health of many people. Factors such as restrictions in the entertainment sector, cancellation of events and social isolation during this period can negatively affect the mental health of many people. Here are the psychological effects of this situation and its effects on the progression of possible diseases.

Restrictions in the entertainment industry can leave many people struggling to plan for future events. This uncertainty can cause stress and anxiety. Event cancellations and social distance rules can limit people's social connections, which can lead to feelings of loneliness and isolation. Being deprived of enjoyable activities can cause many people to become emotionally overwhelmed and depressed. Increased psychological stress can trigger physical health problems or worsen existing health problems. Stress, anxiety and emotional disturbance can affect many people's sleep patterns, which can have a negative impact on overall health. Chronic stress can weaken the immune system, which can reduce resistance to illness. Some people may turn to bad habits to cope with stress, which can have a negative impact on their health. During this challenging time, it is important for individuals to take care of their emotional and mental health. Seeking professional help, maintaining social connections, increasing physical activity, and using emotional support systems can help. It may also be important to find creative and alternative ways to adapt to changes in the entertainment industry.

In addition to the pandemic, there is a growing interest in house parties. Pro Plus survey firm's survey of 2000 people between the ages of 18-34, which is the generation born between 1981 and 1995, defined as Generation Y on the Millennial Generation. According to the results of this survey, millennials are less likely to go out to have fun, both financially and in terms of regulations [9]. The survey emphasizes that the preference for entertainment venues is gradually decreasing. 23% of this age group spend up to £90 per night for entertainment they are willing to spend. Factors such as the closure of hundreds of nightclubs and smoking bans have been reported to have led people to spend

more on nightclubs and bars. It is stated that it causes them to leave entertainment venues in a boring way. Again, according to the survey, only 53% of the community tend to go to the same place. The remaining 47% tend to have fun by forming their own community. In an article published by the world-famous Gentlemen's Quarterly (GQ) magazine, it is emphasized that the way millennials socialize with each other has changed [8]. It is stated that nightclubs and bars can be visited once in a while, but this generation has a different understanding of entertainment. According to the article, this generation believes that house parties can solve the problems created by social media. It is stated that they complain about meeting new people through applications, that they lack emotion, and therefore communication will be even stronger in organizations such as house parties and concerts. Meeting people with different worldviews is more realistic and in line with the flow of life than meeting through the app, and the simplest way to do this is through house parties. It is emphasized that people usually hang out in herds, but house parties reimagine the residential environment on another level, and that such organizations are extremely suitable for people to socialize. The article also emphasizes that the previous economic problems of increasing costs and expensive drinks in nightclubs are an important problem and that such problems can be easily solved through house parties. As can be understood from the above 2 different studies, it is revealed that house parties are a culture and the popularity of entertainment venues such as nightclubs and bars is gradually decreasing due to economic and social reasons. The product in this thesis is positioned in two main markets. These are the Wireless Audio Systems Market and the Home Automation Systems Market globally. While the size of the Wireless Audio Systems Market was 57.3 billion dollars in 2022, it is expected to reach 134.2 billion dollars in 5 years [4]. Especially the American continent has the most important share of this market size, followed by the European continent. As a result of the tests, we conducted in these locations, it is seen that the interest in our product is in the European continent. In addition, the size of the Home Automation Systems Market, which is another market in which our product is included, was 40.8 billion dollars last year, and in 5 years 63.2 billion dollars [5]. The wireless speaker market is poised for significant growth, with an estimated size of USD 34.15 billion in 2024, and is forecast to reach USD 97.54 billion by 2029. This represents a

significant compound annual growth rate (CAGR) of 23.36% during the forecast period from 2024 to 2029. Key drivers of this growth include increased investment in the smart home segment, a growing preference for portable speakers, and continuous product innovation in the wireless speaker market. The high rate of innovation has made these speakers more affordable, contributing to their widespread adoption. In addition, the convenience offered by wireless speakers further enhances the market opportunity [6]. The size of the entertainment devices market is also set to grow over the next 5 years. In 2018, the global home entertainment devices market was valued at USD 225.0 billion and is expected to grow at a compound annual growth rate (CAGR) of 6.3% from 2019 to 2025. The popularity of home entertainment devices is attributed to their convenience and affordability, making them a preferred choice among consumers. The expected growth is further fueled by increasing consumer spending on leisure activities at home, indicating a positive trend in the market over the forecast period [7].

With changing habits, accelerated by COVID, people have started to entertain at home rather than going out. People face various problems when they want to host a large or small party. These include the unavailability of the necessary party equipment in a single product, the inability to manage it in a synchronized way even if purchased separately, the inability to create an audience, and the inability to access the choreographies made during the party from an easy place. Therefore, creating your own party world can offer an enjoyable and personalized experience, especially when you are unable to attend mass events due to pandemic and similar restrictions.

Creating your own party world gives you the freedom to organize events according to your own tastes and interests. By customizing the theme, decorations, music and activities, you can offer attendees a special experience. Organizing your own events gives you the opportunity to bring your friends, family, or community together and strengthen social connections. Organizing your own party gives you more flexibility to control your budget and reduce costs. Instead of organizing an outside event, you can plan a more cost-effective party at home or at a suitable venue. By organizing your own party, you have the

opportunity to organize special events that cater to the interests of your attendees. This allows everyone to enjoy themselves more. By organizing your own party, you can choose special themes and concepts. This helps you create a special experience by personalizing the atmosphere and mood of the event. Organizing your own party gives you the opportunity to make the program flexible according to your wishes. You can customize the program to allow for more interaction with attendees or to spend time in a relaxed environment. Such an alternative can be a powerful tool to strengthen social connections, increase fun and enrich personal experiences, even under limitations. Participants can make more personal connections at your self-organized events and make fun memories in a special atmosphere.

Light robots and fog effects synchronized to music can create an interesting and impressive atmosphere, especially for parties, events, or artistic performances. This concept represents a fusion of technology, entertainment and art and has the potential to offer users an interactive and customizable experience. Users can express their own creativity by creating light and fog effects synchronized with their own music. At parties, events or other social gatherings, users can create fun and impressive visual shows with this system. Users can save and share the choreographies they create, allowing for unique shows that are shared on social media or private platforms. Users can come together on a platform that allows for community interaction and sharing by participating in songs and choreographies by different artists. It will lead to community engagement. With this kind of system, artists and educators can add an interactive and visual dimension to their training and performances. This type of system offers a technologically innovative solution and can enhance users' ability to combine music and visual effects.

It is important to design a user-friendly interface where users can easily interact and select songs and effects. Also, security and privacy concerns about the choreography and performances shared by users should be taken into account. The technical infrastructure and connectivity is somewhat complicated. The device's internet connection should be robust in terms of reliability and performance. A resource should be provided

where users can receive training on interacting with the system and creating their own choreography.

2. PARTY COMPONENTS & TECHNOLOGIES

Organizing a successful party requires careful consideration of several elements to create an immersive and enjoyable experience for attendees. Among the critical components, the loudspeaker stands out as the most important device, serving as the auditory heartbeat of the event. A quality sound system not only provides clear and vibrant music, but also sets the tone for the entire event, raising the energy and atmosphere. In tandem with the auditory experience, lighting robots play a key role in transforming the ambience and enhancing the overall visual spectacle. These dynamic lighting fixtures help create a vibrant and dynamic party mode, synchronizing with the beats of the music to create a mesmerizing dance of color and pattern. The interplay between sound and light becomes an art form in itself, adding to the overall sensory experience of the party environment.

To further enhance the immersive nature of the celebration, the inclusion of a fog mechanism adds an extra layer of excitement. Spreading a fine mist throughout the venue not only enhances the visual impact of the lighting effects, but also adds a sense of mystery and anticipation, enveloping attendees in an ethereal atmosphere. The synergy between the sound, light and fog mechanisms works in harmony to create a multi-sensory experience that captivates, engages, and leaves a lasting impression.

As the principles of party organization evolve, the incorporation of advanced technologies and innovative equipment is essential to stay at the forefront of entertainment trends. Understanding the intricate interplay between speakers, light robots and fog mechanisms provides valuable insights into the art of creating unforgettable party experiences.

As can be seen in figure 2.1, the components to be used in our party box are shown. The light robot, the liquid pump to make the fog mechanism to receive liquid from outside, then the fog resistor to convert it from liquid to fog, the speaker, our electronic card to

balance the voltage and ammeter values of all components, and then the microprocessor, namely Raspberry pi, to control all these components via the web [3].

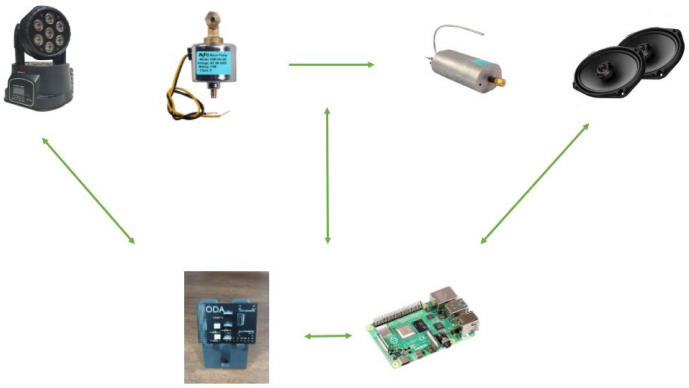

Figure 2.1.: New Party Box Interior Design

The scope of this thesis is to explain the method and context through which the devices shown in Figure 2.1 exchange data with each other. In line with this scope, each device within this Internet of Things (IoT) ecosystem adheres to the principles of the Web of Things (WoT) paradigm, using the WoT Thing Description (TD) standard for semantic identification. The properties and interaction points defined in the TD standard meticulously articulate the sensor data obtained from the Raspberry Pi, providing a comprehensive and standardized representation of each device's capabilities and functionalities.

The exchange of data between the backend and frontend components is facilitated by a carefully designed and adhered to standard API. In this case, the WebSocket protocol serves as the conduit for data transfer, embodying a well-defined interface that ensures interoperability and consistency. This adherence to standard APIs increases the overall robustness and maintainability of the entire IoT system. The WoT-inspired architecture promotes a bi-directional interaction model, enabling devices to engage in meaningful

exchanges. For example, the frontend has the ability to send control commands to the Raspberry Pi via the intermediary backend. This organization of commands and responses emphasizes a dynamic and symbiotic relationship, enabling not only the reception of data, but also the initiation of actions across the IoT ecosystem.

The autonomy and modularity of each device in the IoT framework is paramount, exemplified by the divergence of programming languages and technologies. The Raspberry Pi, which uses Python, the backend, which uses Node.js, and the frontend, which relies on React.js, demonstrate a conscious design decision to encapsulate specific functionality within each device. This deliberate separation improves maintainability, scalability, and facilitates future expansion or replacement of individual components without compromising the integrity of the overall system. The independence and modularity of each device contributes significantly to the adaptability and extensibility of the entire IoT ecosystem.

To explain the scenario in this thesis, there is a WoT scenario through a scenario that communicates with WebSocket using Raspberry Pi (Python), backend (Node.js) and frontend (React.js). The Raspberry Pi is used to collect environmental information using various sensors (e.g. temperature, humidity). It then uses Python to process this sensor data and send it to a WebSocket server. At the backend, a Node.js-based WebSocket server receives the data from the Raspberry Pi. It processes the data and sends it to the frontend application in a specific format (e.g. JSON). When it comes to the frontend, a React.js based web application receives data from the backend via WebSocket. This data can be used to create different visualizations on the user interface or displayed to the user. The next step is to analyze the technologies chosen on the frontend, backend and microcontroller side and the reasons for them [22].

2.1. Frontend Technologies for Enhanced User Experience

In the frontend development space, a myriad of options unfold, requiring a crucial decision at the outset the architectural choice between a traditional multi-page application (MPA) and a single page application (SPA). This decision has profound implications, as it forms the basis of the project's structural design and has a significant impact on both the user experience and the development workflow. Despite the merits of both approaches, a careful evaluation is essential to determine which best suits the unique requirements and objectives of this project [14].

The traditional multi page application model is based on a structure in which each distinct section or functionality of the application resides within separate HTML pages. Navigation in this paradigm involves complete page reloads, with each page taking responsibility for rendering its content. This method, ingrained in conventional web development practices, offers simplicity in certain aspects of development. However, it can introduce latency associated with reloading entire pages.

Conversely, the Single Page Application paradigm adds dynamism to the user experience. SPAs load a single HTML page and dynamically update content as users navigate through the application, eliminating the need for full page reloads. This model leverages client-side rendering and promises a noticeably faster and smoother user interaction. However, it requires a robust client-side framework, which adds a layer of complexity to the development process.

Single Page Applications (SPAs) provide a smoother and more seamless user experience by loading the entire application on the first request. Subsequent interactions, such as navigating between pages or accessing content, are handled dynamically without full page reloads. SPAs use techniques such as pre-fetching and caching, resulting in faster page loads after the initial load. In addition, SPAs use client-side routing to update the URL without making a server request, contributing to a more responsive feel. SPAs often

involve fewer server requests because only data and resources need to be fetched from the server, rather than entire HTML pages. This can lead to reduced server load and bandwidth usage. SPAs are typically easier to maintain because the frontend and backend can be developed independently. Updates to the frontend can be deployed without affecting the entire application. SPAs are well suited for applications that require rich and dynamic user interfaces. They enable the use of advanced UI components, real-time updates, and interactive features without the need for full page reloads. They provide a more app-like experience, similar to mobile applications, with smooth transitions and instant responsiveness [17].

Often mistaken for a client-side framework, ReactJS is actually a powerful library renowned for its extensive feature set. It has gained popularity due to its versatility in providing a wide range of functionality. Positioned as a lightweight solution, ReactJS excels at rapidly developing single-page applications through its innovative component-based architecture. Leveraging features such as two-way data binding and the Virtual DOM, ReactJS accelerates website performance to unprecedented speeds [18].

The choice of React as the optimal framework for single page application (SPA) development is based on several key attributes. These are lightweight, separation of concerns (SoC), JSX – XML like syntax, component-based framework, virtual DOM, object oriented paradigm, highly scalable and flexible.

React is renowned for its lightweight nature, ensuring efficient and streamlined development without unnecessary overhead. It adheres to the principle of separation of concerns, promoting a modular and organized code structure for improved maintainability. Its use of JSX, an XML like syntax, allows developers to seamlessly render HTML code using JavaScript, simplifying the integration of markup into the application logic. At the core of React's architecture is a component-based paradigm that promotes reusability and flexibility in application design. Components encapsulate modular functionality, encouraging a structured and scalable approach. The implementation of a virtual DOM

mechanism contributes to React's high performance by streamlining the rendering process, resulting in faster updates and an improved user experience. React follows the principles of object-oriented programming, providing developers with a familiar and powerful paradigm for building scalable and maintainable applications. Its commitment to performance is underscored by its efficient rendering strategies that ensure optimal speed and responsiveness in single page applications.

In conclusion choose the ReactJS each of them fort his thesis. React is known for its component-based architecture, which promotes modularity and reusability. Components encapsulate specific functionality, making applications easier to manage and scale. It is lightweight and flexible, allowing developers to choose additional libraries and tools based on project needs. This flexibility appeals to developers looking for a tailored and efficient development experience. React's Virtual DOM optimizes rendering and improves performance by selectively updating only changed parts of the DOM. React has a large and active community that contributes a wealth of resources, tutorials, and third-party libraries. This community support can be valuable for troubleshooting and staying up to date with best practices. React is easy to integrate with other technologies and libraries. Its compatibility with tools such as Redux for state management increases its appeal for complex applications. React has been widely adopted by industry leaders, giving a sense of confidence in its reliability and future development. Many well-known companies use React for their frontend development [16],[20],[21].

2.2. Backend Infrastructure for Seamless Integration

Although there are many options on the backend side, NodeJS was used because of its advantages in certain areas such as event-driven, non-blocking I/O, fast execution, large ecosystem (NPM), scalability, real-time applications, community support, cross-platform compatibility, microservices architecture, Raspberry Pi and IoT projects.

To efficiently handle concurrent requests, Node.js follows an event-driven, non-blocking I/O model. This architecture is well suited to handling concurrent connections and asynchronous operations, making it efficient for applications with a high number of concurrent requests, such as real-time applications or APIs [15].

About the V8 engine, Node.js is built on the V8 JavaScript engine from Google Chrome, which is renowned for its speed and performance. This makes Node.js particularly fast and suitable for applications that require fast response times. Node.js has a rich ecosystem of packages and modules available through NPM. This extensive repository makes it easy to integrate third-party libraries and tools into your project, saving development time and effort. For horizontal scalability, Node.js is horizontally scalable, meaning it can handle an increased number of requests by adding more nodes to the system. This makes it suitable for applications that may experience varying levels of traffic. Through WebSocket (SocketIO), Node.js is well suited to real-time applications thanks to its support for WebSocket communication. This is beneficial for building features such as live chat, online gaming, or collaborative editing [16].

Node.js has a large and active community of developers. This means there is a wealth of resources, tutorials, and community-driven support available, making it easier to find solutions to challenges. For platform independence, Node.js is designed to be cross-platform, allowing your application to run consistently across different operating systems. This is particularly useful for projects that need to run on different environments without modification. Through modular development, Node.js lends itself well to a microservices architecture, where applications are made up of small, independent pieces.

2.3. Database Management for Efficient Data Handling

In the context of this master's thesis project, the strategic decision to use MongoDB as the database solution stems from a careful consideration of the project's future scalability and its central reliance on the Internet of Things (IoT) paradigm. Recognizing

the growing importance of IoT in modern technology landscapes, the choice of MongoDB was driven by the expectation that the IoT aspect of the project would play a central role in future developments. The choice of MongoDB was underpinned by the foresight that the IoT component of the project would expand and become more complex over time. The inherent flexibility of MongoDB's schema design allows for the seamless integration and storage of different data formats, a critical feature when dealing with the dynamic and evolving data structures commonly found in IoT applications [21].

As the project envisions a future where IoT devices will generate a significant amount of data, MongoDB's scalability features are paramount. MongoDB's ability to scale horizontally ensures that the database will be able to handle the expected increase in data and device connections, which aligns perfectly with the project's long-term goals.

In addition, MongoDB's JSON-like document model is proving instrumental in addressing the nuances of IoT data representation, providing an intuitive and natural fit for the JSON-formatted data commonly associated with IoT devices. This streamlined approach to data handling not only facilitates development, but also enables the project to efficiently adapt to the evolving data requirements of an expanding IoT ecosystem. MongoDB has many advantages. These include flexible schema design, scalability, JSON-like documents, real-time data processing, geographic indexing, rich query language, ease of development, community and ecosystem, and built-in security features and integration with other technologies. IoT applications often deal with diverse and evolving data formats. MongoDB's flexible schema design, based on the BSON document model, allows you to handle data with varying structures. This flexibility is beneficial when dealing with the diverse data generated by IoT devices. IoT projects typically involve a large number of devices that generate a significant amount of data. MongoDB's horizontal scaling capabilities make it well suited to handle the growing volume of data and increasing number of device connections as your IoT deployment expands. IoT data is often represented in JSON format. MongoDB's support for JSON-like BSON documents makes it easy to store and query data without complex transformations, which

can streamline the development process and improve performance. MongoDB supports real-time data processing through features such as capped collections and customizable cursors. This is beneficial for IoT applications where timely processing and analysis of streaming data is critical.

Many IoT applications involve location-based data, such as tracking the movement of devices. MongoDB supports geospatial indexing, making it efficient to query and analyze location-based information. MongoDB offers a powerful and expressive query language. This is valuable when dealing with complex queries or when you need to perform aggregations on the data collected from IoT devices. MongoDB's ease of use and developer-friendly features, such as a rich set of drivers and comprehensive documentation, contribute to faster development cycles. This is especially important in the rapidly evolving and innovative IoT landscape. MongoDB has a large and active community, which means access to a wealth of resources, forums and third-party tools. The ecosystem around MongoDB can be used to enhance various aspects of your IoT project, from monitoring to analytics. MongoDB offers robust security features, including authentication, authorization and encryption. Security is paramount in IoT applications, especially when dealing with sensitive data and devices. Also MongoDB easily integrates with other technologies commonly used in IoT ecosystems, such as message queues, edge computing platforms, and analytics tools.

In essence, the selection of MongoDB for this thesis project is not just a technology choice for the present, but a forward-looking decision that recognizes the dynamic nature of the IoT landscape. By leveraging MongoDB's strengths in flexibility, scalability and handling diverse data, the project is laying a solid foundation for continued growth, ensuring that the database infrastructure remains resilient and adaptable to the evolving demands of an increasingly IoT-centric future.

2.4. Raspberry Pi: Technical Insights

In the context of this project, Raspberry Pi is a well thought out choice due to its modular expandability, Wi-Fi module, GPIO pins and ease of coding with Python. It also has many advantages in various areas such as modular expandability, ease of coding with Python, community and resources, affordability and accessibility, versatility in project types, integration with web development [20].

Raspberry Pi is equipped with General Purpose Input/Output (GPIO) pins for easy connection to a variety of external devices. This modular design makes it easy to connect sensors, actuators, and other hardware components to extend the functionality of your project. Raspberry Pi built in Wi-Fi module provides wireless connectivity, allowing your project to communicate with other devices or connect to the Internet without the need for additional hardware. Raspberry Pi is well supported by the Python programming language, renowned for its simplicity and readability. Python's extensive libraries and community support make it an excellent choice for coding applications and interacting with hardware on the Raspberry Pi. In addition, Raspberry Pi has a large and active community of developers and enthusiasts. This community-driven ecosystem provides a wealth of resources, tutorials, and solutions to common challenges, making it easy for developers to find support. Raspberry Pi comes with extensive documentation covering hardware specifications, programming guides, and troubleshooting tips. This documentation is valuable for both novice and experienced developers. It boards are cost-effective compared to many other computing platforms, making them accessible to a wide range of projects, including hobbyist and educational endeavors. It offers several models with different specifications, so you can choose the one that best suits your project requirements and budget.

On the other hand, Raspberry Pi is often used in Internet of Things (IoT) projects due to its compact size, low power consumption and versatile connectivity options. Its GPIO pins make it suitable for DIY electronics projects, enabling the creation of custom

electronic setups and control of external hardware. It can be used to create web-based interfaces to control and monitor connected devices. This is particularly useful for projects that require user interaction via a web browser.

3. RASPBERRY PI AUTO HOTSPOT

In the context of this thesis, as shown in Figure 2.1, there are party equipment, electronic card, and Raspberry Pi. Raspberry Pi in particular is the device that enables data exchange between such party equipment. The first thing to decide is how these components will communicate with Raspberry Pi. Data can be exchanged using 2 methods: Bluetooth and Wi-Fi. Although both have their own advantages, it is worth comparing them to decide which is the right one to use in this work [2].

3.1. Comparative Analysis and Selection of Data Transfer Methods

In this work, we propose a system for the exchange between devices such as light robots and fog machines, which are essential components needed when giving a party. One of the key decisions in the development process is the choice of communication protocol, in particular whether to use Bluetooth or Wi-Fi for data transfer. This decision has a significant impact on the overall performance, range, and efficiency of the data transfer.

The choice between Bluetooth and Wi-Fi depends on several factors, each with its own set of advantages and considerations. Bluetooth, known for its low power consumption and ease of pairing, can be a suitable choice for short-range interactions, promoting a seamless connection between devices in close proximity. On the other hand, Wi-Fi, known for its higher data rates and greater range, becomes an attractive option when the choreography involves devices scattered across a larger performance space.

Examining the trade-offs between these two communication technologies is critical to optimizing the choreography system. Bluetooth's simplicity and energy efficiency can be advantageous in scenarios where power conservation and simplicity are paramount. Wi-Fi, with its higher bandwidth, may be preferable for applications requiring rapid and high-volume data exchange, potentially meeting the demands of complex and synchronized choreography.

This consideration goes beyond the purely technical, as factors such as power consumption, latency and resilience also play a crucial role in determining the most appropriate communication protocol. By addressing these considerations, this paper aims to provide insight and guidance to developers navigating the nuanced landscape of device interactions, facilitating informed decisions regarding the integration of Bluetooth or Wi-Fi in the creation of choreography systems.

Wi-Fi operates in the 2.4 GHz and 5 GHz frequency bands and offers relatively high data rates. The range can be up to several hundred feet, making it suitable for applications that require extended coverage. The IEEE 802.11 standards, such as 802.11n or 802.11ac, offer increased throughput for data-intensive tasks. Wi-Fi natively supports the TCP/IP protocol stack, enabling seamless integration with Internet-based communications. This makes it ideal for scenarios where Raspberry Pi needs to communicate with a web server or web-based services. It connections are more complex to set up due to network configuration, SSID and security protocols such as WPA2 or WPA3. However, this complexity allows for robust security implementations, which are essential when transmitting sensitive data over the network.

Bluetooth, specifically Bluetooth Low Energy (BLE), is designed for low power consumption. BLE allows Raspberry Pi to maintain communication with other devices while conserving power, making it suitable for battery-powered or energy-efficient applications. It operates in the 2.4GHz frequency band and uses different profiles for different use cases. Common profiles include the Serial Port Profile (SPP) for serial

communications and the Generic Attribute Profile (GATT) for BLE applications. These profiles define how devices interact with each other.

Bluetooth generally has simpler pairing processes than Wi-Fi. Devices can connect without extensive configuration. This simplicity is advantageous for projects where quick and easy device pairing is essential. It is well suited to short-range communication, typically up to 100 meters. This makes it ideal for scenarios where the Raspberry Pi needs to interact with nearby devices in close proximity.

Ultimately, Wi-Fi emerged as the preferred choice for data interaction for a number of compelling reasons. Chief among these considerations is Wi-Fi's ability to offer high data rates, ensuring fast and efficient communication between devices. This is particularly important in the context of choreography systems, where real-time data transfer is essential for the synchronized execution of performances.

In addition to high data rates, Wi-Fi's wide coverage area seamlessly accommodates the spatial requirements of choreography setups. Whether devices are distributed across a stage or in different locations within a performance space, Wi-Fi's wide coverage ensures consistent and reliable connectivity, facilitating the seamless coordination of light robots and fog machines.

The inclusion of Internet access capabilities further enhances the versatility of Wi-Fi in choreographic systems. This connectivity feature opens the door to the use of online resources, enabling dynamic and interactive elements within choreographies. It also allows for remote monitoring and control, giving choreographers greater flexibility in managing and adapting performances.

The adoption of standard protocols is another key factor in favor of Wi-Fi. Standardization ensures interoperability and compatibility with a wide range of devices, fostering an ecosystem where different components can communicate and collaborate

seamlessly. This contributes to a more cohesive and extensible choreography infrastructure.

Wi-Fi's innate support for multiple devices is paramount in the context of choreography systems, where numerous light robots, fog machines and other elements need to synchronize their actions. The ability to accommodate multiple devices simultaneously facilitates a cohesive and intricately coordinated choreographic experience.

In addition, the advanced security options offered by Wi-Fi play a critical role in protecting the integrity and privacy of data exchanged between devices. This is particularly important in performance scenarios where secure and reliable communications are essential.

Finally, Wi-Fi is ability to simultaneously access the Internet adds an extra layer of functionality. This allows for the seamless integration of online content, expanding the creative possibilities for choreographers. The ability to simultaneously access the Internet while maintaining device-to-device communication adds dynamic, multimedia elements to the choreographic experience.

In summary, the choice of Wi-Fi for data interaction in choreographic systems is underpinned by a comprehensive set of features, including high data rates, wide coverage, Internet access, standard protocols, multi-device support, advanced security and simultaneous Internet access. Together, these contribute to a robust, flexible and powerful infrastructure that enables choreographers to create compelling and technologically enriched performances.

3.2. Enabling Automatic Wi-Fi Connection on Raspberry Pi

AutoHotspot on Raspberry Pi is a versatile tool that streamlines and automates the complex task of managing Wi-Fi connectivity. It enables Raspberry Pi to seamlessly switch between two critical roles: that of a wireless access point providing a hotspot for devices to connect to, and that of a client device connecting to an existing Wi-Fi network. This feature-rich solution greatly enhances the adaptability of Raspberry Pi, making it ideal for a wide range of scenarios. Whether you're setting up a mobile hotspot on the move, creating a standalone network for a specific project, or enabling Raspberry Pi to easily join an existing Wi-Fi network, AutoHotspot simplifies the process with its easy-to-use and robust functionality.

One of the notable benefits of AutoHotspot is its ability to intelligently detect the availability of preferred Wi-Fi networks, ensuring a smooth transition between creating a local hotspot and connecting to an existing network. This dynamic capability is particularly useful in situations where the Raspberry Pi needs to operate autonomously, adapting to changing environments without manual intervention. Indeed, incorporating AutoHotspot into our project represents a significant leap forward in terms of control and connectivity. This feature-rich tool gives us the ability to effortlessly manage and coordinate the various components of our project, including speakers, lighting robots and the fog mechanism, all through a centralized and user-friendly web interface.

AutoHotspot's low latency ensures that commands from the website are transmitted quickly, allowing us to respond in near real-time when controlling the various elements of our setup. This is critical to creating a seamless and immersive user experience, especially when dealing with dynamic elements such as speakers, lights and fog mechanisms that require precise and synchronized control. By harnessing the power of AutoHotspot, our project gains a level of versatility that simplifies the integration of these diverse components. The ability to control them through a unified web interface not only improves the user experience, but also opens up possibilities for remote management and automation.

Whether AutoHotspot's orchestrating a synchronized light and sound show, triggering fog effects at specific times or dynamically adjusting the behavior of robotic elements, AutoHotspot becomes the backbone of this interconnected system. Its adaptability and ease of use make it a key enabler for our project, allowing us to focus on creating engaging and interactive experiences rather than getting bogged down in the complexities of managing wireless connectivity. In essence, AutoHotspot acts as a hub that facilitates the seamless integration and control of our project components, bringing us one step closer to creating a sophisticated, interactive, and user-friendly system.

In addition, AutoHotspot offers a comprehensive set of configuration options, allowing users to tailor the wireless connectivity settings to their specific needs. This level of customization ensures that the Raspberry Pi can be seamlessly integrated into a wide range of use cases, from Internet of Things (IoT) projects to educational setups and beyond. AutoHotspot on Raspberry Pi is a comprehensive solution that not only simplifies the complexities of Wi-Fi management, but also extends the functionality of the Raspberry Pi, making it an invaluable tool for projects that require versatile and automated wireless connectivity.

3.3. Automated Hotspot Installation Raspberry Pi

This installer script simplifies the process of configuring your Raspberry Pi to seamlessly switch between a Wi-Fi network and an access point setup based on your preferences. It provides an easy and automated solution for Raspberry Pi users who want the flexibility to connect to their home network when in range, or automatically set up a Wi-Fi access point when away from known networks. The script eliminates the need for manual intervention by allowing both manual execution and timer-based switching without requiring a system reboot. The installation script has been designed to support different use cases, which are outlined in three comprehensive guides available on this website. These guides provide step by step instructions for setting up a permanent static access point and two different setups that facilitate a smooth transition between a

Raspberry Pi access point and a Wi-Fi network connection. By combining these guides into one installer, users are provided with an easy-to-use tool that streamlines the entire setup process and makes it accessible to users with varying levels of technical expertise. The installer's features allow users to easily customize the Wi-Fi behavior of their Raspberry Pi, increasing the overall convenience and functionality of their Raspberry Pi-powered projects.

To use the installer, follow these steps in a terminal environment step by step:

- Download the AutoHotspot-Setup.tar.xz archive to the current folder using the following command curl "https://www.raspberryconnect.com/images/hsinstaller/Autohotspot-Setup.tar.xz" -o AutoHotspot-Setup.tar.xz
- Unpack the file in the current folder with the command tar -xvJf AutoHotspot-Setup.tar.xz
- Change the directory to the AutoHotspot folder.

After running the script you will be presented with the following menu options:

- Install AutoHotspot with eth0 access for attached devices
- Install AutoHotspot without eth0 access for attached devices
- Install Fixed Access Point with eth0 access for connected devices
- Uninstall AutoHotspot or Fixed Access Point
- Add a new wireless network to the Pi (SSID) or update the password for an existing one.
- Force AutoHotspot: Force an access point or connect to a Wi-Fi network when a known SSID is in range.
- Change access point SSID and password
- Exit

According to this project used 1. option. After installation and a subsequent reboot, the Raspberry Pi will automatically connect to a router that has been previously connected to and is listed in the configuration file located at in etc. in wpa_supplicant at wpa_supplicant.conf. If there is no router in range, the Raspberry Pi will automatically create a Wi-Fi access point.

If an Ethernet cable is connected to the Raspberry Pi, which provides access to the internet, devices connected to the access point will be able to connect to the internet or local network. After successfully connecting to the access point, the Raspberry Pi can be accessed remotely using the following method with SSH (Secure Shell).

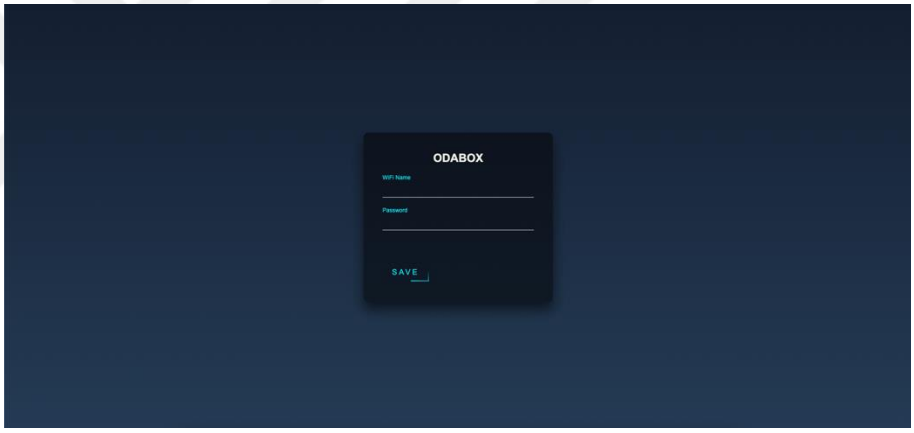

Figure 3.3.1 : Raspberry Pi User Interface for the Autohotspot

With the HTML file successfully written, it will now be displayed at the specified IP address, 192.168.5.50. In addition, a Flask application was developed to store the information and trigger the activation of a hotspot when the Raspberry Pi is first booted. This integration not only ensures a seamless presentation of the HTML content, but also allows the Raspberry Pi to efficiently collect and manage data via the Flask backend. The dynamic combination of the HTML interface has shown in Figure 2.3.1 and the Flask backend enhances the functionality of the system, providing a user-friendly experience while enabling the Raspberry Pi to serve as a robust and versatile hotspot-emitting device from the moment it is powered on.

4. LIGHT ROBOT INTEGRATION

There are essential elements that allow the creation of a personalized choreography on a single unit, including lighting robots, a fog machine, and speakers. The lighting robots are particularly noteworthy as they use the DMX512 protocol, allowing them to be seamlessly controlled from a DMX table. DMX512, a digital communication standard widely used in stage lighting and effects, was originally established by the United States Institute for Theatre Technology (USITT) as a common lighting communication protocol.

Prior to the USITT standardization of DMX, different companies used their own proprietary communication protocols between the lighting board and the dimmers. DMX differs from other digital communication protocols in that it is a balanced signal. This unique feature ensures that any interference in the cable or transmission process is less likely to affect the signal quality, as the system is designed to ignore signal noise.

In the context of this thesis, a novel approach to the control of lightweight robots has been undertaken, as mentioned in reference [10]. This innovative methodology not only increases the precision and flexibility of choreography creation, but also represents a departure from traditional control methods. By exploiting the capabilities of DMX512 and incorporating a new control paradigm, this thesis contributes to the advancement of the field of lightweight robotics and paves the way for a more sophisticated and user-friendly experience in choreography design on a single device.

4.1. Designing a Network for Light Robots

DMX512 is a widely used standard in digital communications, particularly for the management of stage lighting and effects . Originating from the United States Institute for

Theatre Technology (USITT), DMX, or DMX512, serves as a standardized communication protocol for lighting. Prior to the establishment of the DMX standard, individual companies used their own communication protocols to interact between lighting boards and dimmers. DMX is unique among digital communication protocols in that it is a balanced signal. This feature ensures that any interference in the cable or transmission process is likely to be ignored as signal noise [1],[12].

The IP network packets used for this application are USB packets. USB is the chosen method for transmitting DMX lighting information over a wired connection, specifically for events such as parties. USB was chosen over ArtNet because it is more suited to the dynamic and adaptable nature of party environments. Unlike DMX, which typically allows for one universe of 512 channels, USB offers distinct advantages in the context of a party environment. It is capable of supporting multiple universes, although its capabilities are often limited by the speed and bandwidth of the USB connection.

While ArtNet-like systems are often used in larger installations for their distribution capabilities via Ethernet-based systems, USB is a practical choice for party setups. USB is favored for its ease of installation and use, especially in dynamic and temporary setups such as party environments. In addition, USB works well with the wide range of devices commonly used in party scenarios.

In essence, the choice of USB over ArtNet is driven by its adaptability, simplicity and compatibility with the specific requirements of hosting events such as parties [11].

4.2. Hardware Design for Light Robots

The Raspberry Pi was chosen for its cost effectiveness, low power consumption, solid state storage, portability, and overall affordability. ENTTEC offers a product called ODE that serves the same purpose as the Raspberry Pi and Open DMX USB combination. However, going the Raspberry Pi route was seen as a more economical and versatile

choice than using ENTTEC's ODE. In addition, using OLA on the Raspberry Pi provides greater flexibility.

The specific Raspberry Pi model chosen is the Model 4, which has both Wi-Fi and USB inputs. For the USB to DMX512 functionality, ENTTEC's Open USB DMX device was used. This is a standard, open source, and open hardware device.

It's important to note that the chosen USB to DMX device belongs to the non-smart category of converters. Unlike smart converters, which can generate the DMX signal independently, this device simply repeats one-to-one what it receives from the USB. OLA allows variable DMX speed, with a rate of 44Hz being used for this project. OLA recommends a speed range between 30 and 44Hz.

Open Lighting Architecture (OLA) provides a versatile framework for managing lighting control data. Compatible with multiple protocols and supporting more than a dozen USB devices, OLA offers flexibility in handling different lighting setups. It can operate independently as a standalone service, facilitating signal conversion between different protocols. Alternatively, it can be used as a backend for lighting control software using the OLA API. OLA is cross-platform and runs on a variety of systems, including ARM architecture, making it particularly suitable for low-cost Ethernet to DMX gateways [13].

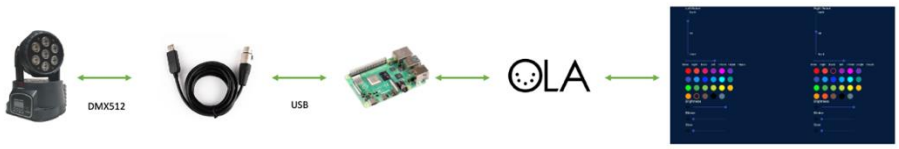

Figure 4.2.1 : Light Robot Design

4.3. UI UX Design for Controlling Light Robots

Extensive research was undertaken to better understand the user experience of the application. The aim was to improve usability and effectiveness. The results of this research led to a strategic decision to adopt a leaner, simpler and more elegant design, resulting in the removal of the DMX tables. In essence, the redesign seamlessly integrates primary and auxiliary colors into the user interface.

Beyond color considerations, there was a specific need for a design that would facilitate the dynamic movements of the lighting robots in both horizontal and vertical planes. This requirement was met with a thoughtful and deliberate approach, resulting in the design shown in the figure. In addition, the design aims to make it easy to adjust the brightness, flashing and rotation speed levels of the light robots.

The revised design not only prioritizes a minimalist aesthetic, but also optimizes functionality, ensuring that users can easily and intuitively manipulate both the color schemes and spatial orientations of the light robots. This holistic approach to design represents a commitment to providing a user-friendly and visually appealing experience within the application.

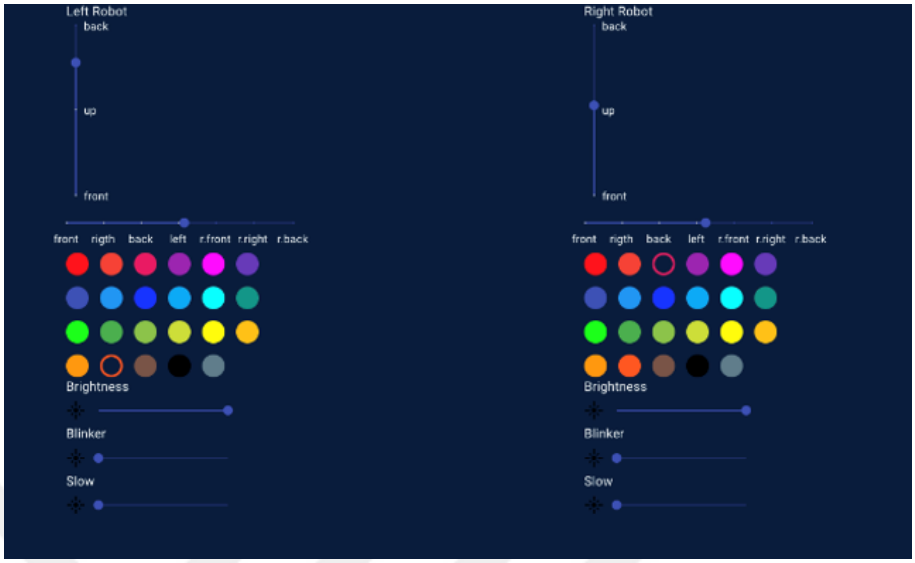

Figure 4.3.1 : Light Robot Control User Interface

4.4. Software Strategies for Controlled Light Robots

The successful deployment and configuration of the Open Lighting Architecture (OLA) on Raspberry Pi marks a significant milestone in the development of the backend API. The backend API, hosted on port 9090, serves as a critical component in the overall system architecture. It allows seamless communication and integration between the Raspberry Pi and the frontend.

In the context of the master's thesis, particular attention is paid to the intricacies of this connection process. As part of this process, the Raspberry Pi, which is on the same local network as the frontend application, actively communicates its unique IP and MAC addresses to the frontend. This exchange of network information is strategically orchestrated during connection initiation.

This information transfer is not just a technical detail; it plays a vital role in creating a dynamic and responsive system. The frontend, armed with the Raspberry Pi's IP address, gains the ability to send requests directly to the Raspberry Pi. This direct communication is a fundamental element in the real-time operation of the light robots by frontend.

Such an architectural approach is not only a technical solution, but also a subject of investigation and analysis within the master's thesis. Factors such as communication efficiency, security implications and the overall impact on the user experience are explored and evaluated in detail. The dissertation explores the complexities of integrating hardware (Raspberry Pi) with website and the impact on the overall performance and functionality of the lighting control system.

5. FOG MACHINE INTEGRATION

Once the basis of the fog mechanism is established complete with a fog resistor and a pump for liquid intake the next step is to integrate these components with a Raspberry Pi. This crucial connection requires a carefully designed electronic board. In the Figure 5.1 illustrates the specific design that facilitates this seamless integration.

In the diagram at the figure 5.1.2, note the functions assigned to the various pins. The pins labelled 14 and 28 are used to control the pump, dictate its movement and regulate the liquid intake process. These pins act as the dynamic drivers behind the rhythmic pulses of the pump, ensuring the precise delivery of liquid required for the fog mechanism.

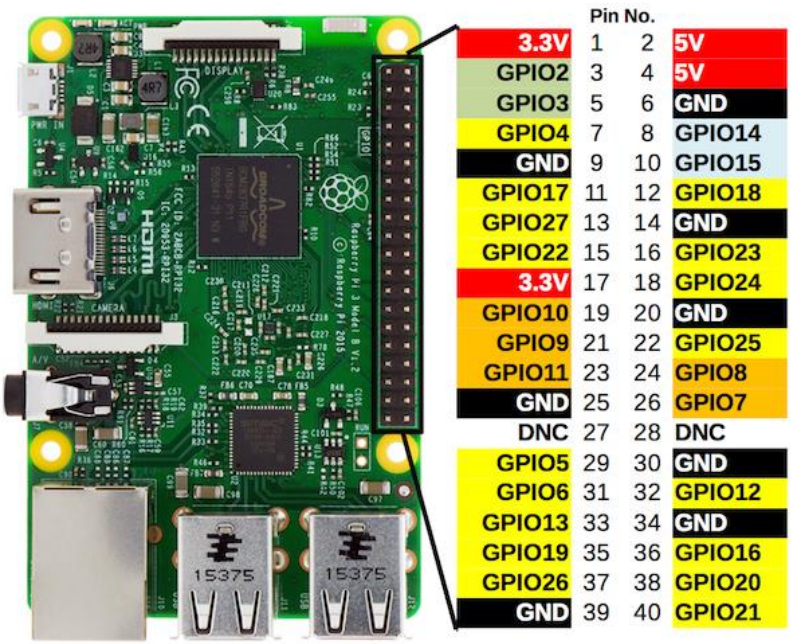

Figure 5.1 : Raspberry Pi Pinout

On the other hand, pin number 32 has a special role in the design it is dedicated to managing the fog resistance. This pin acts as the control center for the heating element within the resistor, orchestrating the temperature required to create the fog effect. By

modulating the current through pin 32, we can fine-tune the heat applied to the resistor, optimizing the fog creation process.

The synergy between these components and the Raspberry Pi is at the heart of our fog mechanism. The electronic card design acts as a translator, enabling effective communication between the Raspberry Pi and the fog system. It's this careful integration that transforms a collection of hardware components into a cohesive and functional system.

In summary, the journey from the fog resistor and pump to the Raspberry Pi involves not only the selection of the right components, but also the development of a well-thought-out electronic card design. This design, illustrated in the figure, shows the strategic allocation of pins, each of which plays a vital role in the synchronized operation of the fog mechanism.

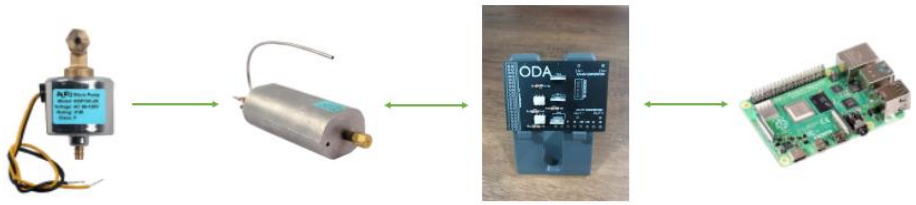

Figure 5.2 : Fog Machine Hardware Design

5.1. Control Mechanism for Fog Machine

As shown in Figure 5.1.1, activation of the Heat Switch button on the top bar of the web page initiates a sequence in which a heating command is transmitted from the pin connected to the fog resistor. This deliberate heating process is essential to raise the temperature of the resistor to the threshold required to convert the liquid coming from the pump into vapor. It is therefore essential to maintain the temperature at a specific level to ensure that the system reaches the required conditions for each fog pumping request.

In the technical underpinnings of this mechanism, the heating command is meticulously orchestrated to achieve optimal and consistent performance. The interaction involves a works of electronic signals, with the heat switch button triggering a sequence that activates the pin connected to the fog resistor. The resistor in turn responds by heating up to the prescribed temperature, facilitating the phase transition of the liquid introduced by the pump into a gaseous state.

Maintaining the temperature at a specific level is a critical aspect of this process as it directly affects the efficiency and reliability of the fog generation. The system is designed to ensure that when a fog pump request is received, the required temperature conditions are met before the fog generation process begins. This meticulous control mechanism not only guarantees the consistency of the fog output, but also contributes to the longevity and optimum performance of the fog resistor and associated components.

The temperature control system acts as a safeguard, preventing the fog generation process from starting unless the system is at the ideal operating temperature. This level of precision is critical to delivering a seamless and responsive fog generation experience, in line with the high standards set by the master's thesis project. The technical intricacies involved in this temperature control process underline the sophistication and reliability embedded in the fog mechanism and are a significant contribution to the overall success of the project.

5.2. Thermocouple Integration into Website for Fog Control

The integration of a thermocouple adjacent to the fog resistor plays a key role in optimizing the performance of the fog mechanism. This thermocouple, intricately connected to the Raspberry Pi, provides a seamless channel for real time data exchange. The purpose of this connection is to precisely monitor and regulate the temperature of the fog resistor. The Raspberry Pi continuously measures temperature fluctuations through the thermocouple, providing insight into the current thermal state. This information is used to effectively orchestrate the fog production process. In essence, the system dynamically adjusts the fog output command based on the observed temperature, ensuring that the liquid from the fog pump is heated to the correct temperature before being sprayed. This meticulous temperature control not only improves the quality of the fog effect, but also contributes to the overall safety and reliability of the fog mechanism in the choreography production setup.

To facilitate user control and ensure optimal functionality, a heat button has been strategically placed in the design to allow users to manipulate the resistance of the fog mechanism. When activated, this heat button triggers the Raspberry Pi to initiate a controlled heating process, gradually increasing the temperature of the fog resistance. The system is configured to reach a certain threshold, 440 degrees to be precise, at which point the fog mechanism is primed and ready to emit fog. This deliberate user involvement in the heating process not only provides a hands-on approach to choreography production, but also allows for customization and adaptation based on specific performance requirements. The 440-degree target ensures that the fog mechanism operates within a defined temperature range, guaranteeing both efficient fog production and equipment longevity. This interactive feature also allows users to fine-tune the fog production to suit the creative nuances of their choreography.

The temperature of the fog mechanism is conveniently displayed to the user via a dedicated bar on the user interface. This intuitive visual representation allows users to

easily monitor the current temperature status of the fog resistor. As the resistor heats up, the bar provides a dynamic and real-time display, giving users immediate feedback on the progress of the heating process. This visual cue not only increases user awareness, but also ensures a seamless and user-friendly experience when managing the temperature of the fog mechanism. Whether activating the heat button for fog production or simply keeping an eye on the thermal status of the system, the temperature bar serves as a valuable tool for users to make informed decisions when fine-tuning their choreography production settings.

Figure 5.2.1 : Temperature User Interface Design

6. SPOTIFY INTEGRATION

The heartbeat of any great party is undoubtedly the music. As technology continues to shape the way we experience entertainment, the choice of music platform becomes paramount. This thesis explores the strategic decision to adopt Spotify as the centerpiece of a party setup, examining not only its extensive user base but also the opportunities it presents for developers. The focus on Spotify is based on its unique ability to transcend device boundaries, coupled with its open-source nature, which opens the door to customization for an enhanced party experience.

Before looking at the technology, it's important to recognize the fundamental role of music in social settings. Music has the power to set the mood, evoke emotions and create a shared experience among participants. Understanding the psychological and sociological impact of music in gatherings lays the foundation for further exploration.

The decision to use open source in the area of music platforms stems from the desire to provide a tailored and unique experience. This section explores the principles of open source, highlighting its collaborative nature and the potential for developers to innovate and customize features for specific use cases.

A comprehensive analysis of Spotify serves as the core of this thesis. Examining its massive user base provides insights into the platform's popularity and its relevance to social events. Furthermore, the focus shifts to Spotify's developer-friendly environment, exploring the tools and opportunities it provides to developers interested in extending the platform's capabilities.

One of Spotify's outstanding features is its ability to play music seamlessly across devices. This section explores the technical aspects of this capability and discusses how it contributes to the convenience of playing curated playlists on your own party device. Diving into the open-source nature of Spotify, this section explores how developers can

leverage this aspect to tailor the music experience for a party environment. By discussing APIs, SDKs, and other developer tools, it highlights the potential for creating unique music experiences.

6.1. Website Authentication for Spotify Account Login

To integrate Spotify into a website, developers often use the Spotify authorization process to allow users to seamlessly access their Spotify accounts. Spotify uses the OAuth 2.0 protocol for authorization, ensuring a secure and user-friendly experience. The process typically involves redirecting users to the Spotify Accounts service, where they log in and grant permission for the website to access their Spotify data. Upon authorization, Spotify generates an access token and an update token, which the website can then use to make authorized requests on the user's behalf. This authorization mechanism not only enhances the functionality of the website by enabling features such as playlist integration and personalized recommendations, but also ensures the privacy and security of user data. Overall, the Spotify authorization process acts as a bridge between the website and the user's Spotify account, enabling a seamless and integrated music experience [12].

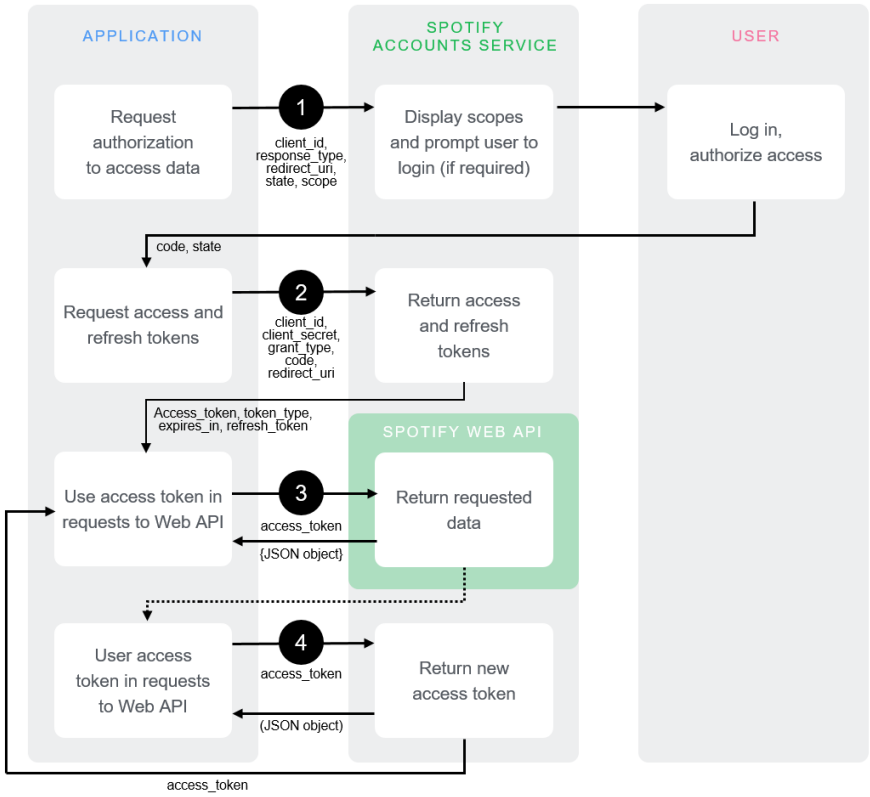

Figure 6.1.1 : Spotify Login Architecture [26]

When starting the login process on the website, users are seamlessly taken to the Spotify login screen where they can securely authenticate their identity using their Spotify credentials. Upon successful login, users are prompted to enhance their experience by clicking on 'Open Device Authorization (ODA)' in the 'Other Devices' section within the Spotify application. This crucial step establishes a secure connection between the website and the user's Spotify account, giving the website the necessary permissions to access Spotify data.

Furthermore, this integration goes beyond a simple login mechanism. As users navigate the Spotify application and select a track of their choice, the website uses this interaction to dynamically retrieve information about the selected track. Using Spotify's

API, the website can retrieve details such as the song title, artist, album, and other relevant metadata. This real-time synchronization allows the website to display accurate and up-to-date information about the selected song, creating a seamless and immersive user experience. This integration not only streamlines the user's interaction between the website and the Spotify app, but also adds a layer of personalization by displaying the user's music preferences on the website interface.

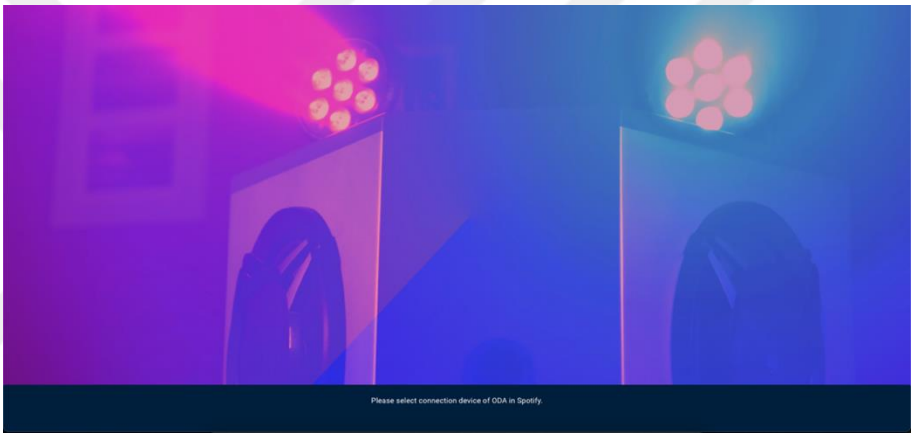

Figure 6.1.2 : Navigate to Spotify in Thesis App

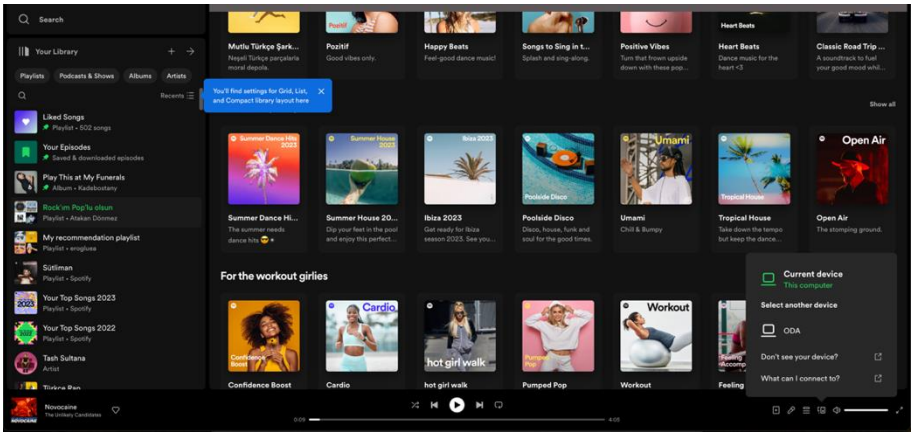

Figure 6.1.3 : External Speaker in Spotify

6.2. Retrieving Track Features for Enhanced Spotify Integration

There is a lot of data that can be retrieved from the Spotify API. In this project, a song needs certain information. These are track duration, genre information, singer/artist name, album name. Spotify's API provides accurate information about the duration of each track. By retrieving this data, you can display the exact duration of the song currently playing on your interface. This is valuable for users who want to throw a party while using light robots and fog machines. Spotify's extensive music database includes genre tags for every track. This information allows you to categories and display the genre of the song you are listening to. This feature enhances the user experience by allowing listeners to explore and understand the variety of music genres at the party. The name of the singer or artist is displayed prominently on the interface, helping users to identify and appreciate the musical talent behind the track. A track's album information is also available through Spotify's API. Including the album name in your interface provides context and allows users to discover more tracks from the same album. It adds a visual and informational layer to the listening experience, making it more engaging.

In addition, designing an intuitive and aesthetically pleasing interface is critical to a positive user experience. Include a section or panel in your interface that displays track details. Consider using images, icons or visual elements associated with the genre, artist or album to enhance the visual appeal. Spotify's API supports real time updates, ensuring that your interface accurately reflects the latest track information. By implementing real-time syncing, users can see changes instantly as the party playlist progresses, creating a seamless and dynamic experience.

6.3. Incorporating Spotify API into Our Website

In addition to its superior control over request handling, Axios introduces a layer of security and reliability through its automatic handling of request and response transformations. This feature is invaluable when dealing with different data formats,

allowing Axios to seamlessly convert requests and responses between JSON and other formats without manual intervention. This automatic transformation capability simplifies the integration process and ensures consistent and standardized data exchange between the application and the Spotify API.

Axios also excels at handling cross-browser compatibility issues through built-in support for XMLHttpRequest, which can be particularly beneficial in different web development environments. This ensures a more consistent experience across browsers, mitigating potential challenges that could arise from relying solely on the Fetch API.

Axios also facilitates the organization of API calls through the use of interceptors, allowing developers to structure code for authentication, error handling or any other custom logic in a modular and reusable manner. This organizational flexibility promotes cleaner and more maintainable code bases, making it easier for developers to manage and scale their applications over time.

In summary, Axios not only extends the capabilities of the Fetch API, but also introduces features that improve the security, reliability, and maintainability of communication between applications and the Spotify API. Its comprehensive toolset enables developers to navigate the intricacies of web development more efficiently, resulting in a more robust and adaptable integration with the Spotify service.

7. CREATE A CHOREOGRAPHY

In this comprehensive section, a meticulous investigation is undertaken to define the essential communication protocol required to create the choreography. The choice of communication protocol is based on its ability to facilitate seamless interaction between the backend, frontend, and Raspberry Pi components. This careful selection will ensure optimal data transfer efficiency and system responsiveness. The rationale behind the selection is explained, taking into account factors such as latency, reliability and bandwidth requirements inherent in the intricacies of the choreography design.

A granular analysis of the choreography file structure is then presented, elucidating the constituent elements and their corresponding parameters. The file serves as a central artefact, encapsulating the intricate instructions that govern the synchronized orchestration of the performance. Each parameter within the file is meticulously examined for its role in shaping the dynamic choreographic sequence.

The data exchange mechanisms between the backend, the frontend and the Raspberry Pi are systematically demystified in the ensuing discussion. The role of each component in the collaborative creation of the choreography file will be examined. The intricate processes of sending and receiving data will be explained, emphasizing the interplay between algorithms, protocols, and hardware resources.

It also describes the technical intricacies of how the backend processes and refines the choreography instructions before delivering them to the frontend and Raspberry Pi. This includes a detailed exploration of the algorithms used for real-time analysis, optimization, and synchronization. It also explains the role of the frontend in rendering the choreography based on the received instructions, highlighting the graphical rendering techniques and computational methods involved. The unique position of the Raspberry Pi in this ecosystem is dissected, with a focus on its role in executing the choreography. The hardware-specific considerations, such as GPIO interactions and real-time constraints, are

addressed in detail to provide a comprehensive understanding of the Raspberry Pi's contribution to the overall choreographic framework.

7.1. Deciding Communication Protocols for Data Exchange

WebSocket is a communication protocol that enables bi-directional, full-duplex communication between a client (typically a web browser) and a server over a single, long-lived TCP connection. Unlike traditional HTTP connections, which are stateless, and request-response based, WebSocket allows both the client and server to send messages independently. This real-time, low-latency communication is ideal for applications that require instant updates, such as online games, financial trading platforms or collaborative editing tools.

WebSocket works by establishing an initial handshake through an HTTP request, after which the communication channel remains open for the duration of the session. This persistent connection minimizes overhead and reduces latency compared to traditional polling mechanisms.

Now equipped to exchange data over Wi-Fi, the next critical decision is to determine the communication protocol, with options such as WebSocket or RESTful API. Each approach brings its own set of advantages and considerations that affect the dynamics of how data is sent and received within the choreography system.

WebSocket, known for its real-time, bi-directional communication capabilities, offers a compelling solution for scenarios where instant updates are paramount. This protocol enables a persistent connection between devices, facilitating the seamless flow of data in both directions. WebSocket is particularly beneficial for choreography systems that require low latency and fast response times, ensuring that the actions of lighting robots and fog machines are precisely synchronized with the performance.

On the other hand, a RESTful API, while traditionally request-response oriented, provides a standardized and stateless communication model. This approach is well suited to scenarios where choreography commands can be encapsulated in discrete HTTP requests. RESTful APIs are often chosen for their simplicity, scalability, and ease of integration with a variety of platforms and programming languages.

The choice between WebSocket and RESTful API depends on the specific requirements and characteristics of the choreographic system. WebSocket excels in situations that require constant communication and instant updates, while RESTful APIs offer a more structured, resource-centric approach that fits well with certain choreographic workflows.

Given the intricacies of choreography, the decision between these communication protocols should be based on factors such as the need for real-time updates, the complexity of the data exchange, and overall performance requirements. Both WebSocket and RESTful API options offer unique advantages, and the choice should be made with a thorough understanding of how each protocol fits with the goals and requirements of the choreographic experience.

7.1.2. Rationale for Utilizing SocketIO

Socket.IO is a JavaScript library that abstracts the WebSocket protocol and provides additional functionality to simplify real-time communication in web applications. It is built on top of WebSocket, but also includes fallback mechanisms to support environments where WebSocket may not be available. Socket.IO is designed to be easy to use and includes features such as automatic reconnection, transport abstraction, level of abstraction, compatibility, and flexibility.

Socket.IO can automatically attempt to reconnect if the connection is lost, ensuring a more robust and reliable real-time experience. It allows clients to be grouped into

"rooms", enabling targeted message delivery to specific groups of clients. It can use different transport protocols, including WebSocket, HTTP long polling and others, to adapt to different network conditions and browser capabilities. It simplifies event-based communication between clients and the server. Clients can send events and the server can respond to or send events to multiple clients. WebSocket operates at a lower level, providing a basic communication channel, while Socket.IO operates at a higher level, abstracting WebSocket and adding additional features for ease of use. Socket.IO is designed with compatibility in mind, addressing issues related to WebSocket support across different browsers and network environments. It is generally easier for developers to use due to its simplified API and built-in features, making it suitable for rapid development of real-time applications. WebSocket is lighter and suitable for scenarios where a minimalist, efficient protocol is sufficient. Socket.IO, with its additional features, is chosen when ease of implementation and compatibility across different environments are critical [19].

Use the Socket.io library on both the ReactJS (client) and Raspberry Pi (server) sides. On the Raspberry Pi, set up a SocketIO client using Python. Socket.io allows real-time, bi-directional communication between the ReactJS frontend and the Raspberry Pi backend. Define custom events to pass data from the Raspberry Pi to ReactJS and vice versa. Set up a RESTful API on the Raspberry Pi using a web framework such as Express.js for Node.js or Flask for Python. Define endpoints (URLs) that the ReactJS application can send HTTP requests to, specifying the desired actions. Use standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on the Raspberry Pi. For example, send a GET request to retrieve data or a POST request to update data. Decide on a data format for communication (e.g. JSON). Serialize the data on the ReactJS side before sending requests, and deserialize it on the Raspberry Pi side when processing incoming requests.

In summary, WebSocket provides a basic communication protocol, and Socket.IO builds on top of it to provide additional functionality, making it a convenient choice for

developers looking for a more abstracted and feature-rich solution for real-time communication in web applications. The choice between WebSocket and Socket.IO depends on the specific requirements and constraints of the project at hand. While which method to use depends on the requirements of your project, SocketIO was preferred due to the need for instant updates from the Raspberry Pi frontend for choreography production and requirements such as real-time interaction.

Socket.IO acts as a central middleware that enables continuous bi-directional communication between clients (Raspberry Pi and ReactJS) and the server (Node.js) using the WebSocket protocol. The architecture of the project revolves around three different stations, consisting of two clients (Raspberry Pi and ReactJS) and a server implemented in Node.js. The need for a backend component arises from the requirements of persisting and retrieving choreography operations and handling user-related functionalities.

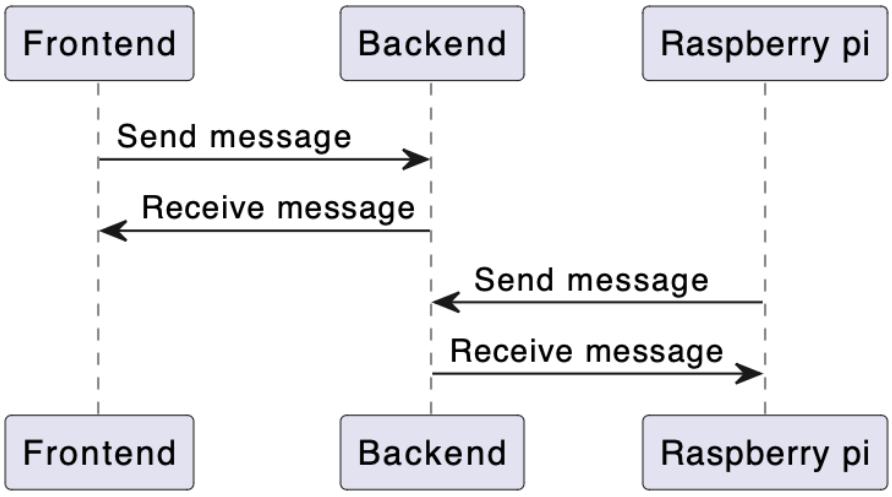

Figure 7.1.2 : In Thesis Data Architecture

In the context of Socket.IO, communication is organized in "rooms". The server side, implemented in Node.js, takes on the role of the Socket.IO server and manages these rooms. Upon user login, automatic integration into the designated room is orchestrated, establishing a shared context for real-time data exchange. This architectural decision

streamlines the implementation of emit and on operations within the confines of the same room. Emit operations allow the server to send real-time updates or messages to all participants in the room, while on operations allow clients to listen for and respond to specific events within the room.

The room-based communication model offered by Socket.IO is advantageous for choreography operations. It ensures that all relevant devices (Raspberry Pi and ReactJS clients) are synchronized and updated in real time, enhancing the collaborative nature of choreography creation. At the same time, user operations such as logins and user-specific updates benefit from this room-based approach, facilitating efficient and targeted communication within the Socket.IO ecosystem. This nuanced architecture takes full advantage of Socket.IO's capabilities to provide a robust, real-time communication framework for choreography and user-related functionality.

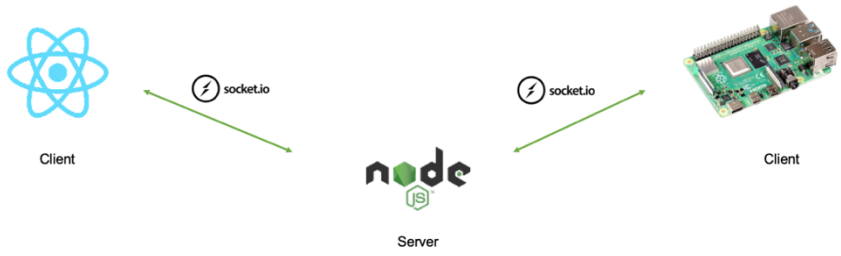

Figure 7.1.2.1 : Socket Architecture

7.2. Data Transfer Among Website, Backend, and Raspberry Pi

In this thesis, an advanced data exchange infrastructure has been developed using Node.js on the backend and React.js on the frontend. The communication protocol used is Socket.IO, a powerful tool that enables real-time, bi-directional communication

between the server and the client. This strategic implementation not only ensures fast and immediate data transfer, but also creates a highly responsive and dynamic user interface.

The architecture of this data exchange process is illustrated in Figure 7.2.1 . At the backend, Node.js acts as a server, managing data requests and processing. At the same time, React.js, at the frontend, oversees the user interface and communicates seamlessly with the server via the built-in Socket.IO interface. This architectural design optimizes the capabilities of both technologies, creating a cohesive and efficient connection between server and client, especially over Wi-Fi.

Using Socket.IO over Wi-Fi not only increases the speed and reliability of data exchange, but also meets the scalability requirements of today's web applications. This choice of technology is in line with the current trend for real-time, responsive user experiences, ensuring that data updates are immediately reflected on the frontend, providing users with a smooth and engaging interaction.

The following sections of this thesis explored the integration between Node.js and React.js, as well as the intricacies of the Socket.IO implementation. This in-depth analysis aims to provide valuable insights into the effectiveness and efficiency of using this technology stack for seamless data exchange in sophisticated web applications.

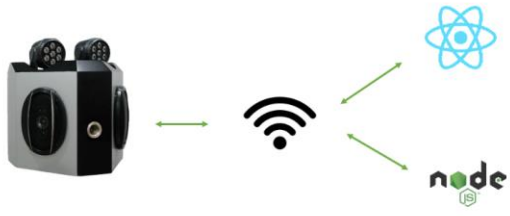

Figure 7.2.1 : Connection Diagram

7.3. Frontend Roles in Choreography Creation

The frontend plays a key role in the choreography creation process and its responsibilities can be divided into several sections. First, when a user logs into the site, the frontend performs a series of actions to initiate the choreography creation process. This includes establishing a secure WebSocket connection with the backend server, facilitated by the Socket.IO library. The frontend triggers authentication events, securely sending user credentials to the server for validation, and upon successful login, the frontend subscribes to relevant choreography channels, indicating an active participant status.

The choreography creation process then unfolds as users interact with the frontend interface. The frontend provides an intuitive graphical interface that allows users to define choreography elements such as light robot movements, color schemes and fog activation patterns. User interactions are captured by event listeners that generate Socket.IO events encapsulating the choreography details. These events are then sent to the backend server in real time.

On the backend side, the server receives these Socket.IO events, parses and validates the incoming choreography data. The server processes user-defined choreography actions and updates the central choreography file accordingly. The server also co-ordinates the distribution of these updates to all connected clients, ensuring real-time synchronization. The backend acts as a central hub for choreography management, orchestrating the collaborative efforts of multiple users contributing to the creation process. At the same time, the Raspberry Pi, acting as a client, establishes its own WebSocket connection to the backend server. It subscribes to the same choreography channels, allowing it to receive real-time updates. When the backend receives custom choreography events from the frontend, it sends updates to the Raspberry Pi. The

Raspberry Pi interprets these updates and extracts relevant information, such as light robot movements and fog activation cues. It then executes the prescribed choreography actions, effectively bringing the choreographed elements to life in the physical environment.

In summary, the frontend acts as the user interface and interaction gateway for choreography creation. It facilitates user engagement, captures choreography inputs and communicates them to the backend server and Raspberry Pi via the WebSocket protocol. This complex communication flow ensures a collaborative and synchronized approach to choreography creation, where user actions on the frontend are seamlessly translated into real-world choreographed performances orchestrated by the Raspberry Pi and managed by the backend server.

7.3.1. User Interface and User Experience Design

The easy-to-use choreography production interface combines a timeline on the left, allowing users to navigate to specific seconds, and a control panel for light robots and fog mechanisms on the right. Each second on the timeline acts as a clickable button leading to a detailed control interface where users can program the movement, speed and color of the light robots, as well as adjust fog density and duration. To avoid complexity, the system ensures a minimum 2-second interval between commands, preventing overlap between the completion of one full movement of the light robots and the initiation of another. The interface also features an auto-sync function to harmonize elements and a preview mode for real-time visualization. With a visually appealing design and intuitive controls, choreographers can effortlessly create captivating performances while having the flexibility to fine-tune their creations.

The user-friendly choreography production interface has been meticulously designed to provide a seamless and intuitive experience, combining a left-side timeline and a right-side control panel dedicated to the management of lighting robots and fog mechanisms. The timeline, conveniently positioned on the left, acts as a navigation tool,

allowing users to effortlessly navigate through specific seconds of their performance. Complementing this, the right panel provides a comprehensive set of tools for orchestrating the dynamic elements of the choreography.

Within this intuitive interface, each second on the timeline acts as a clickable button, providing access to a detailed control interface. Here, users can fine-tune the movement, speed and color of the light robots, as well as parameters such as fog density and duration. This granular level of control allows choreographers to precisely shape and synchronize every aspect of their performance.

To streamline the creative process and avoid unnecessary complexity, the system incorporates a thoughtful design element a minimum 2 second pause between commands. This deliberate pause prevents overlap between the completion of one full movement of the light robots and the initiation of another, ensuring a coherent and visually appealing choreographic sequence. The interface also includes Auto Sync, a powerful feature designed to bring disparate elements together seamlessly. This feature intelligently aligns movements, colors, and fog effects, eliminating the need for manual adjustments and allowing choreographers to focus on artistic expression rather than intricate synchronization.

To support the creative workflow, the interface includes a preview mode that provides a real-time visualization of the choreography. This allows choreographers to assess and fine-tune their creations on the fly, providing instant feedback on how different elements come together in the performance space. With its visually appealing design and intuitive controls, the Choreography Production Interface empowers choreographers to effortlessly bring their creative visions to life. The flexibility built into the interface allows not only the creation of captivating performances, but also the fine-tuning of each element to ensure a dynamic and engaging audience experience.

Figure 7.3.1.1 : Creating Choreography User Interface

Seamlessly integrated with Spotify, the algorithm dynamically retrieves song duration data to match the choreography to the music. Synchronization occurs at two-second intervals, ensuring that commands to the lighting robots and fog mechanism are precisely timed to the music's beats. For ease of use, selections are automatically reset to defaults within each two-second interval, giving choreographers a clean slate to begin their creative input. This reset feature not only simplifies the choreography process, but also allows for a systematic and organized approach to refining each segment of the performance. In addition, the user has the flexibility to customize and fine-tune the default settings based on their creative vision, providing a good balance between automation and user control in the choreography production process.

7.3.2. Selecting Tracks on Spotify

The integration of Spotify functionality into the choreography creation interface adds another layer of sophistication. When the user selects a song from Spotify, the interface dynamically retrieves and displays key characteristics of the selected song at the bottom of the interface. This includes essential information such as the song's title, artist and duration. The retrieval of this metadata serves as a contextual anchor for

choreographers, providing relevant details about the musical canvas on which the choreography will unfold.

In the context of recording choreography, the extraction and display of song attributes is essential for several reasons. First, capturing the name and artist of the selected song enriches the choreography documentation process by linking each creative effort to a specific musical composition. This link contributes to the holistic documentation of choreographies and facilitates future reference and replication.

In addition, the duration of the selected song is a critical parameter during the choreography recording phase. This temporal information becomes an integral aspect of choreography synchronization, allowing users to precisely time choreographed sequences with the progression of the music. By prominently displaying these song characteristics, the interface not only provides the user with a visual cue to the selected musical backdrop, but also aids in the strategic planning and execution of the choreography.

7.3.3. Determining Movements for Party Equipment

In order to effectively manage the light robots, we recognized the paramount importance of providing users with an intuitive and user-friendly interface. Our approach to interface design was meticulously crafted, based on the findings of detailed frontend research and analysis. The goal was to create an interface that not only met the functional requirements of controlling light robots, but also enhanced the overall user experience.

Through this in-depth research process, we identified the specific needs and preferences of our users. This included understanding their expectations, preferred interaction patterns and the visual elements that resonated with them. These insights played a key role in shaping the design principles that guide the user interface.

For the get colors, this is a function that starts by creating a new copy (newRobotColour) of the songCor array. This is done to avoid directly modifying the state,

which is good practice in React to ensure immutability. This retrieves the current color information for the selected second from the copied array. Determines the type of robot (left or right) based on the value of the robot variable. Updates the RGB color values and the hexadecimal color value in the color object according to the robot type. The modified color information is reset in the copied newRobotColour array for the selected second. The setSongCor status variable is then updated with the modified newRobotColour array to reflect the color change. This function sets the hexadecimal color value in the state using setValue(event.hex). This value can be used elsewhere in the application. As a result, this function makes it easy to dynamically handle color changes for a particular robot (left or right) at a particular second in a musical choreography. It adheres to React's immutability principles by making a copy of the array and updating the state accordingly.

The process of capturing light robot movements and configuring the fog mechanism within the choreography file is integral to the user interaction paradigm. The values extracted from the sliders, including leftHorValue, leftVerValue, rightHorValue, rightVerValue, brightnessValue.L, brightnessValue.R, blinkerValue.L and blinkerValue.R, accurately represent the desired positions, brightness levels and blinking characteristics of the light robots. These slider values are dynamically captured from the user interface and seamlessly incorporated into the choreography file construction process within the saveCoreography method.

User interaction also extends to the activation of the fog mechanism, a key element of the choreography. The inclusion of check boxes for each second interval in the left hand seconds list provides a user-friendly interface for selecting specific moments when the fog mechanism should be activated. The user's selection of these checkboxes is translated into binary values within the choreography file, indicating whether the fog mechanism should be active (1) or inactive (0) at each corresponding second. This interactive feature allows the user to precisely orchestrate the interplay between the robot's light movements and the fog effects, contributing to the dynamic and expressive nature of the choreographed performance.

In essence, the user's engagement with sliders and check boxes serves as an intuitive means of articulating creative intent within the choreography. The mapping of user input to choreography file parameters is a central aspect of the UI design, ensuring a seamless and expressive choreography creation experience. This user-centric approach enhances the versatility of the choreography file, allowing users to define the spatial dynamics of the light robots and the temporal activation patterns of the fog mechanism in a visually intuitive manner.

7.4. Backend Responsibilities in Choreography Creation

The backend side actually acts as a bridge between the front end and the Raspberry Pi. On the backend side, this is where the WebSocket connection actually takes place. If the verification keys on the Raspberry Pi and the front end are correct, then the room is opened, and data can be exchanged in the room. Until someone leaves the room. WebSocket connection and data exchange in the context of your choreography system.

In the backend, our system serves a dual purpose: not only does it perform the vital task of persisting user interactions by writing choreography data to the database, but it also acts as a central hub for coordinating communication between both client sides the frontend and the Raspberry Pi via a unified socket connection. This orchestration is critical to ensuring real-time synchronization and seamless collaboration between the different components of our system.

As users engage in creating choreography via the frontend, the backend not only facilitates the storage of this choreography data in the database, but also plays a pivotal role in bringing both client sides into a shared virtual space, commonly referred to as a 'room', through the use of socket connections. This room concept enables a dynamic exchange of information between the frontend and the Raspberry Pi, fostering a cohesive and synchronized environment for choreography execution.

Importantly, the backend provides users with the flexibility to modify the choreography even after it has been initially created. Once choreography data is written to the database, users retain the ability to revisit, edit or reuse it for other actions. This versatility is made possible by the bi-directional communication established via the socket connection, allowing seamless updates and changes to be transferred between the frontend and the Raspberry Pi in real time. The role of the backend goes beyond mere data storage; it actively facilitates a collaborative and interactive environment for users to iteratively refine their choreography. Users can make adjustments, experiment with different sequences, and see the immediate impact on the Raspberry Pi side, all while benefiting from the persistence and accessibility of their choreography data.

In summary, our backend system serves as the lynchpin of our choreography management ecosystem. It not only ensures the longevity and accessibility of the choreography data in the database, but also facilitates dynamic, real-time collaboration between the frontend and the Raspberry Pi, giving users the flexibility to shape and adapt their choreography even after it has been created.

The backend is implemented using a server-side technology capable of handling WebSocket connections, such as NodeJS. When a user tries to connect (frontend or Raspberry Pi), he provides a verification key. The backend checks the validity of the key against a pre-defined set of keys that are securely stored. If the key is valid, the user is granted access, otherwise the connection is denied. The room management Once verified, the backend creates a unique 'room' for the connection. This room is identified by a unique identifier, ensuring that each performance has its own isolated communication space. Information about the room, such as connected users and performance details, is maintained on the backend. In WebSocket connection handling, the backend establishes a WebSocket connection with both the front end and the Raspberry Pi, allowing bi-directional communication. The WebSocket protocol enables real-time data transfer between connected parties. The frontend sends data, such as choreography instructions or

real-time updates, to the backend via the WebSocket connection. The backend receives and processes this data, performing any necessary validation checks. Validated data is then sent to the Raspberry Pi within the designated room. The backend acts as an intermediary, ensuring that the instructions from the front end are accurately passed to the performance hardware.

The WebSocket connection allows real-time updates to be sent from the frontend to the Raspberry Pi. This is critical for maintaining synchronization between the choreography and the performance elements. The backend includes robust error handling mechanisms. If the backend detects any discrepancies, it can send error messages back to the frontend and provide feedback to users about issues that may need attention. Encryption protocols (e.g. SSL/TLS) are implemented to secure the WebSocket connection. Measures such as rate limiting and IP filtering can be used to prevent malicious activity. The backend monitors user activity in the room. When all participants leave the room or the session ends, the backend initiates room closure, freeing up resources and ensuring a clean state for the next session. The backend logs relevant data, including successful connections, data exchanges and any errors that occur. Analytics tools can be integrated to track performance metrics, helping to analyze performance and drive future improvements. The backend architecture is designed to scale, allowing multiple rooms to be created simultaneously. Load balancing strategies can be used to distribute incoming connections across multiple server instances.

The backend acts as the nerve center of your choreography system, managing connections, facilitating secure data exchange, and ensuring the smooth orchestration of performances. The WebSocket connection forms the backbone of real-time communication, enabling dynamic control over the synchronized elements of the performance. The detailed processes and security measures in place contribute to a reliable and secure experience for users and performers alike.

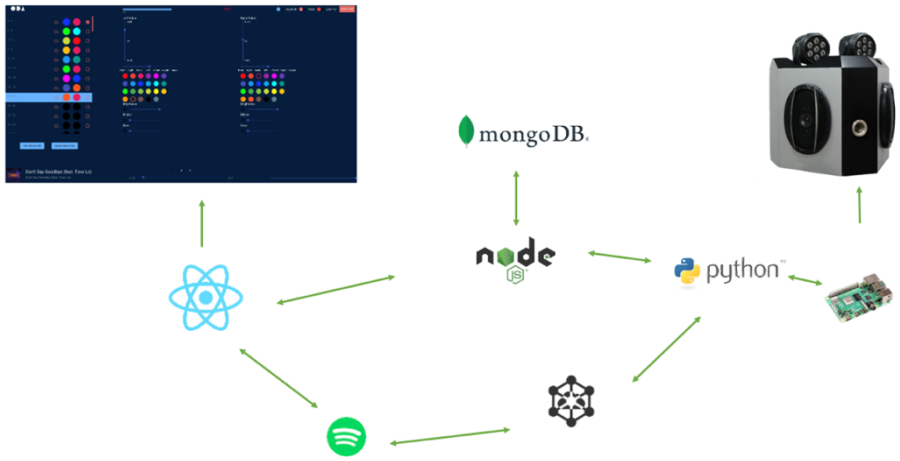

Figure 7.4.1 : General Architecture

7.5. Raspberry Pi Functions in Choreography Implementation

The Raspberry Pi side also acts as a client on the web socket side, just like the front end. As both are clients, the room will be served when they make requests to the appropriate place on the backend using the same key. Both the frontend and the Raspberry Pi initiate WebSocket connections independently. Each WebSocket connection is associated with a verification key. This key is used to ensure the authenticity of the connection and prevent unauthorized access. The frontend and Raspberry Pi connect to the backend server using the appropriate WebSocket URL (e.g. 'wss://your-backend-url/'). During the connection process, each client (frontend and Raspberry Pi) sends its verification key to the backend for validation. Once the backend has validated the verification keys and ensured the authenticity of both connections, it creates a shared 'room' for communication. The room is identified by a unique key or identifier.

The frontend and the Raspberry Pi join the same room by providing the unique key during the WebSocket connection process. This effectively places both clients in a shared room where they can exchange messages via the backend. Messages, commands, or choreography data sent from the frontend are received by the Raspberry Pi in the same

room. Similarly, replies, status updates or any data sent from the Raspberry Pi will be received by the frontend in the shared room. The WebSocket connection enables real-time collaboration, allowing synchronized actions between the frontend and the Raspberry Pi. For example, the front end can send choreography instructions and the Raspberry Pi can execute them in real time.

In summary, both the frontend and the Raspberry Pi, acting as WebSocket clients, connect to a central backend server. Through the use of verification keys, the backend ensures the authenticity of the connections and creates a shared 'room' where both clients can collaborate in real time. This architecture enables seamless communication and coordination between the user interface (front end) and the hardware control (Raspberry Pi) in your choreography system [23].

7.5.1. Implementing Algorithms on Raspberry Pi

The algorithm seems to control various aspects of hardware, including GPIO pins, based on choreography data. I will explain the key components and the overall structure. The algorithm seems to control various aspects of the hardware, including GPIO pins, based on choreography data. I will explain the main components and the overall structure. The first section imports the necessary libraries and modules, including SocketIO for WebSocket communication, base64 for base64 encoding/decoding, and several custom modules (oda_running, oda_only_smoke_new, oda_termometer, live_try_for_robots), and initializes a SocketIO client called sio. It's configured with logging for debugging purposes. This defines a function to handle the connect event. When the socket connection is established, it prints a message and calls the my_message function with the argument 'sample'. Similar event handlers are defined for connection_message, temperature, corData, onlySmoke and liveTryForRobots. These functions are executed when the corresponding events are triggered on the WebSocket server. The main class of function is an asynchronous function that connects to the WebSocket server ('sample.com/'). It then enters a loop to handle communication within the WebSocket connection. Within the loop,

it attempts to send a message and receive a response within a timeout of 2 seconds. If a ConnectionTimeoutError occurs (indicating a problem with the connection), it prints an error message, sets live to False and exits the loop. The asyncio.ensure_future(call_dapi()) and asyncio.get_event_loop().run_forever() are used to start the asynchronous task and run the event loop indefinitely.

Start analyzing the choreography class, which is one of the imported classes in the main class. The main function is Action and takes two parameters. These are data and isStop. Data means choreography data, isStop means when the choreography suddenly stops sending Boolean value from the front end. Start analyzing the choreography class, which is one of the imported classes in the main class. The main function is called Action and takes two parameters. These are Data and isStop. Data means choreography data, isStop means when the choreography suddenly stops sending Boolean value from the frontend. The choreography data has a second value. Then the current second in the choreography is calculated. The elapsed time is divided by 2 to convert it to seconds. And check if the choreography should stop. If the dead flag is set or the nowSecond is equal to the total number of elements in the choreography (len(data.corData)), a stop procedure is initiated. It sleeps for 1 second, sends a POST request to reset DMX values (assuming it's controlling lighting or other equipment) and then cancels the interval (inter.cancel()). This is a busy wait loop. It waits until dead is False, indicating that the script should continue. This section checks if the smoke value for the current second is 1. If true, it reads the temperature using oda_termometer.temperatureRead() and sets the GPIO output (pin 21) based on the temperature. These lines print more information about the current state, including the current second, the robot value from the choreography data and the total length of the choreography. This checks if the current second is the last second in the choreography. If it is, it prints a message and sets GPIO pin 21 to LOW. This checks if the current second minus 1 equals 0. If true, it prints a message, sends a POST request to set DMX values for the first second, then sleeps for 1 second. Finally, a POST request is sent to set DMX values based on the robot value for the current second. In summary, these lines control the execution flow based on the choreography data, setting GPIO outputs and

sending requests to control external devices. The logic is based on the timing information from the choreography and the values of 'smoke' and 'robot'.

To sum up, the technical infrastructure established during the choreography creation process is shown in Figure 7.2, providing a visual representation that illustrates the seamless dynamics of data exchange between the frontend, backend and Raspberry Pi components. The intricately designed architecture provides a clear and efficient communication framework that facilitates coherent and synchronized orchestration of data flows throughout the system. Figure 7.2 serves as a visual guide to provide stakeholders with a comprehensive understanding of the interconnectivity and synergy that underpins the collaborative functioning of the frontend, backend and Raspberry Pi within the choreography creation environment. This visual representation not only enhances understanding, but also serves as a valuable reference for future development, optimization and troubleshooting efforts within the established technical infrastructure.

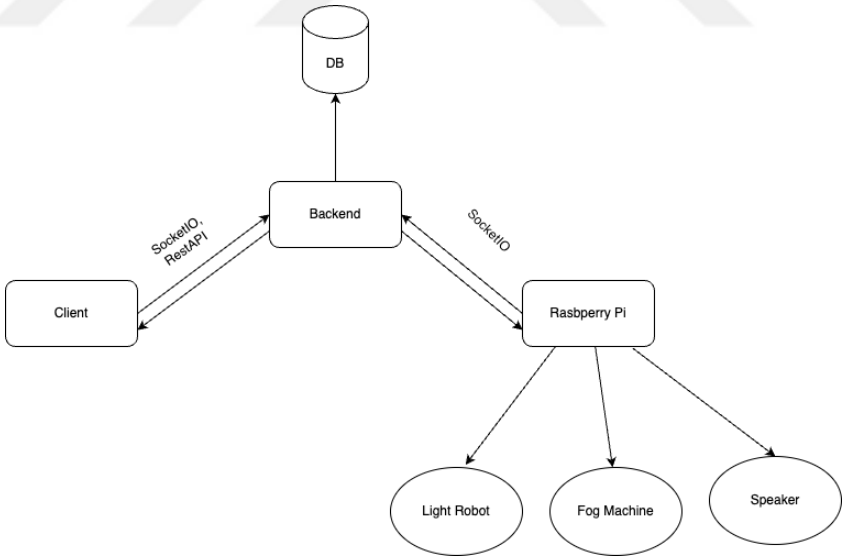

Figure 7.5.1.1 : System Design of The Party System

7.6. Preparing Choreography Files for Execution

Given the significant size of the choreography file, the use of a robust encoding mechanism is essential for secure and efficient data transfer. JSON objects, particularly those representing complex choreography instructions, can be encoded to mitigate potential data corruption and improve the overall integrity of the transmission process.

A common technique for securing the transmission of large JSON payloads is to use Base64 encoding. Base64 encoding converts binary data, such as a JSON object, into an ASCII string format, ensuring compatibility with various communication channels and mitigating issues related to special characters or binary data corruption during transmission.

By using Base64 encoding, the JSON choreography file is transformed into a string of ASCII characters, effectively encapsulating the information in a format that is both secure and resistant to data loss. This encoding approach not only ensures the integrity of the transmitted data, but also provides an efficient means of handling large payloads, optimizing bandwidth usage during communication between the server and clients.

On receipt at the client end, the Base64-encoded string can be easily decoded back into its original JSON format, ensuring the fidelity of the choreography data. This technical approach not only enhances the security and reliability of choreography file transfers, but also aligns with industry best practices for handling large data sets in real-time applications. Choreography file has 6 basic objects. These are startTime, lightRobots and fog.

Choreography Name (name): This name given by users to personalize their choreography.

Selection Track Name (trackName): The track name that users choose from Spotify to create their choreography.

Selection Track Id (trackId): The track id that users choose from Spotify to create their choreography.

Choreographed Seconds (startTime): The choreography file encapsulates temporal information through the inclusion of choreographed seconds, denoted by the StartTime attribute. This temporal reference serves as an anchor for orchestrating different events within the choreography. It defines the starting point for each sequence, allowing precise synchronization of the choreographed elements.

Light Robot Position and Color Codes Object (lightRobots): The choreography file contains a dedicated object to define the position and color codes of the light robots during each choreographed second. This object is a complex combination of spatial and chromatic attributes, specifying the exact locations and colors for each light robot. Position information can include Cartesian coordinates or other spatial descriptors, while colour codes can be represented in formats such as RGB or HEX, providing a comprehensive set of instructions for orchestrating visually compelling performances.

Fog Mechanism Activation (fog): Another integral part of the choreography file is a JSON object that determines the activation status of the fog mechanism. This JSON structure acts as a binary switch, indicating whether the fog mechanism is active or inactive during specific choreographed seconds. It encapsulates important control information to ensure that the fog effect is precisely synchronized with other choreographic elements. The JSON format can include attributes such as "isFogActive", providing a clear and programmatically accessible indication of the fog mechanism's operational status.

In essence, the technical design of the choreography file meticulously combines temporal, spatial and operational information to provide a comprehensive and programmatically accessible representation of the choreographed performance. Each object serves a specific purpose and contributes to the overall precision and synchronization of the choreography as executed by the designated devices.

This section will detail the steps taken on the frontend side, where the choreography involves synchronizing lights and fog effects to a selected song. Let's break down the steps involved in preparing this choreography file.

Track Selection on Spotify: Implement Spotify authentication using OAuth 2.0 to allow users to log in with their Spotify credentials. Request the necessary permissions from the Spotify API to access the user's song data. Provide a search interface that allows users to search and select a song. Retrieve the metadata of the selected song, including its duration.

Seconds Listing and Fog Checkbox: Create a dynamic list of seconds based on the duration of the selected song. Each second is displayed as a string on the left side of the page. Create a checkbox for each second to allow users to indicate whether they want fog to be present at that particular second. Implement toggle functionality for these checkboxes.

Light Robot Movement on the Right Side: Design a user-friendly interface on the right side of the page for users to specify the movement of light robots. Allow users to drag, drop or select predefined movements for light robots. Implement a visual representation of the positions or movements of the light robots.

Choreography Data Preparation: Define a structured data format to represent the choreography. This could be a JSON object or an array. As users interact with the interface, dynamically fill the startTime, lightRobots and fog objects for each second interval.

File Preparation for Transfer: Serialize the choreography data into a format suitable for transfer. JSON is commonly used for this. Ensure that the data format is well defined and documented so that both the frontend and Raspberry Pi can understand it.

Real-time Communication: If real-time synchronization is required, implement a WebSocket connection between the frontend and Raspberry Pi for bi-directional communication. Use a WebSocket library in your chosen programming language at both ends.

Raspberry Pi Implementation: Parse and interpret the received choreography data on the Raspberry Pi. Extract information such as startTime, lightRobots and fog for every second. Implement control logic using GPIO or other mechanisms to control the light robots and fog machine based on the choreography.

User Feedback: Provide visual feedback to the user upon successful data transfer to the Raspberry Pi. Display information messages for further action in case of problems.

This process involves seamless interaction between the frontend and the Raspberry Pi to create a visually compelling and synchronized choreography experience. The key is to ensure clear communication of instructions in the choreography file and correct interpretation on the Raspberry Pi side.

7.7. Playing Your Choreography

In this section we will focus on the practical implementation of the choreography files we have painstakingly created, exploring the steps involved in playing these choreographies alongside the corresponding songs on the designated device. We will also unravel the transformative process of orchestrating a lively party environment - culminating in the realization of the system's intended purpose.

The playback of the choreography files seamlessly integrates the artistic vision conceived during the creation phase with the immersive experience of the accompanying music. Using the capabilities of our system, users can effortlessly synchronize the choreography to the rhythm and beats of the chosen song, creating a harmonious fusion of light robotics and musical composition. This synchronization ensures a captivating and synchronized spectacle, enhancing the overall aesthetic and entertainment value of the performance.

To facilitate the execution of the choreography, the system uses the Raspberry Pi as the designated device. Coordinated by the backend, the Raspberry Pi retrieves the choreography data and interprets the specified movements for the light robots. This interpretation process is intricately synchronized with the playback of the corresponding song, creating a seamless and visually enchanting display that reflects the artistic intent encoded in the choreography files. Beyond the technical intricacies of the execution, the ultimate goal is to create a vibrant party environment in line with the overarching purpose of the system. The amalgamation of choreographed light robot movements and music converge to create an engaging and dynamic ambience. The synchronized interplay between lights, robots and music creates an immersive experience that transforms any space into a vibrant party venue.

This section of this thesis comprehensively explores the practical application of the choreography files within the system. By examining the steps involved in playing choreography alongside music on the Raspberry Pi device, we gain valuable insight into the real-world implementation of the system we have developed. We also explore the transformative impact of this integration, highlighting how it contributes to the creation of an electrifying party atmosphere. Through this exploration, we not only validate the technical robustness of our system, but also underscore its potential to enhance and redefine the experiential landscape of social gatherings.

Figure 7.7.1 : Play Choreography Screen

CONCLUSION

In summary, this thesis represents a comprehensive exploration of the technical innovations and creative methodologies employed to develop a cutting-edge product that enables individuals to create their own immersive party experiences. A ground-breaking product has been created by systematically enhancing existing devices with state-of-the-art microprocessor technology and using contemporary frameworks such as ReactJS, NodeJS, SocketIO and Python. This device seamlessly integrates fog and light combinations, allowing users to dynamically express their unique party worlds at any moment within a song, all through a unique, user-friendly interface.

The culmination of this research and development effort positions the newly created product as a formidable competitor to established market offerings such as the JBL PartyBox [24] and LG XBOOM [25]. This achievement not only underscores the technical prowess employed in the creation of the device, but also validates its potential as a game-changer in the field of party entertainment technology.

Moreover, the implications of this thesis go beyond technical innovation. The product serves as a versatile tool accessible to a wide range of users, including home party organizers, individuals looking to have fun with friends, as well as aspiring DJs and content producers. The democratization of such advanced party technology ensures that the joy of creating personalized party environments is no longer limited to professionals but is readily available to anyone with the desire to create a unique and engaging event.

This thesis not only contributes to the academic understanding of technology integration, but also provides a tangible and impactful solution that enriches the social and entertainment experiences of a wide range of users. The journey from concept to market-ready product serves as a testament to the potential of innovation to redefine and raise the standards of leisure and social engagement.

REFERENCES

[1]Proceedings of the 13th Winona Computer Science Undergraduate Research Symposium, May 1, 2013, Winona. [Online]. Available: [https://cs.winona.edu/cs-website/current_students/Projects/CSConference/2013conference.pdf]. Accessed: Dec. 28, 2023.

[2] "Website Base Robot Control," New Panvel, 2017. [Online]. Available: [http://ir.aiktclibrary.org:8080/xmlui/bitstream/handle/123456789/3242/PE0327.pdf?sequence=1&isAllowed=y]. Accessed: Dec. 28, 2023.

[3] "Conceptualising the Relationship Between Youth, Music and DIY Careers: A Critical Overview," DOI: 10.1177/1749975517750760.

[4] "Wireless Audio Device Market by Product (Headphones, True Wireless Hearables/Earbuds, Speaker) Technology (Bluetooth, Wi-Fi, Airplay), Application (Home Audio, Consumer, Professional, Automotive), Functionality and Region - Global Forecast to 2028," [Online]. Available: [https://www.marketsandmarkets.com/Market-Reports/wireless-audio-device-market-1275.html]. Accessed: Dec. 28, 2023.

[5] "Wireless Speakers Market Size & Share Analysis - Growth Trends & Forecasts (2023 - 2028)," [Online]. Available: [https://www.mordorintelligence.com/industry-reports/wireless-speaker-market?fbclid=IwAR3hgKaC8NbLP3BCnleVW3IvNQwrWNhLoi9lk6NQv31I7680jWqmt7fAl4Y].

[6] "Home Automation System Market with COVID-19 Impact Analysis, By Management, Product (Lighting Control, Security & Access Control, HVAC Control, Entertainment & Other Controls), Software & Algorithm, and Region (2020-2025)," [Online]. Available: [https://www.marketsandmarkets.com/Market-Reports/home-automation-control-systems-market-469.html?gclid=Cj0KCQjw4eaJBhDMARIsANhrQADB1Zh0Kf4w6xw6nALWiM4J34v8fXIB8BXNG7FrmOZcmtPP1v7hYD8aAiQxEALw_wcB]. Accessed: Dec. 28, 2023.

[7] "Home Entertainment Devices Market Size, Share & Trends Analysis Report By Device (Audio, Video, Gaming Consoles), By Distribution Channel (Offline, Online), By Region, And Segment Forecasts, 2019 – 2025," [Online]. Available: [https://www.grandviewresearch.com/industry-analysis/home-entertainment-devices-market]. Accessed: Dec. 28, 2023.

[8] "It’s true: Millennials are Making House Parties Cool Again," [Online]. Available: [https://www.gqindia.com/story/true-millennials-making-house-parties-cool/#group-bonding]. Accessed: Dec. 28, 2023.

[9] "How millennials party differently to Generation X," April 2018, [Online]. Available: [https://www.independent.co.uk/life-style/millennial-nightlife-spending-generation-x-y-baby-boomers-clubbing-drinking-culture-netflix-a8289726.html]. Accessed: Dec. 28, 2023.

[10] R. Cadena, "Automated lighting: the art and science of moving light in theatre, live performance, broadcast, and entertainment," Amsterdam: Focal Press, 2006.

[11] Spotify Developers For Web API, [Online]. Available: [https://developer.spotify.com/documentation/]. Accessed: Dec. 28, 2023.

[12] Raspberry Pi Automatic Hotspot and Static Hotspot Installer, [Online]. Available: [https://www.raspberryconnect.com/projects/65-raspberrypi-hotspot-accesspoints/183-raspberry-pi-automatic-hotspot-and-static-hotspot-installer]. Accessed: Dec. 28, 2023.

[13] "List of 5 Best Single Page Application Frameworks For Web Development," [Online]. Available: [https://www.monocubed.com/blog/top-single-page-application-frameworks/]. Accessed: Dec. 28, 2023.

[14] "Node.js: Using JavaScript to Build High-Performance Network Programs," November 2010. DOI: 10.1109/MIC.2010.145.

[15] A. Mardan, "Socket.IO and Express.js," in Pro Express.js. Apress, Berkeley, CA., DOI: 10.1007/978-1-4842-0037-7_16.

[16] "Modern Web-Development using ReactJS," New Delhi, India, March 2018. [Online]. Available: [http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf]. Accessed: Dec. 28, 2023.

[17] V. A. M. Prathamesh Sonpatki, "ReactJS by Example - Building Modern Web Applications with React," 2019.

[18] D. T. K, J. F. A, D. R, G. S. Josh, "Asynchronous Wi-Fi Control Interface (AWCI) Using Socket IO Technology," DOI: 10.48550/arXiv.1810.05502.

[19] "RaspBot - Raspberry Pi-powered Robot controlled by a WebApp" (2019). ICT. 4., June 2016, [https://arc.cct.ie/ict/4].

[20] "MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data," January 2016. DOI: 10.1109/JSEN.2015.2483499.

[21] D. Guinard and V. Trifa, "Building the Web of Things: With examples in Node.js and Raspberry Pi," 2016.

[22] "The Web of Things: Challenges and Opportunities," May 2015. DOI: 10.1109/MC.2015.149.

[23] JBL Partybox, [Online]. Available: [https://uk.jbl.com/party-speakers/PARTYBOX310-.html?cgid=party-speakers]. Accessed: Dec. 28, 2023.

[24] LG XBOOM XL7S, [Online]. Available: [https://www.lg.com/uk/speakers/xboom/xl7s/]. Accessed: Dec. 28, 2023.

[25] Spotify, [Online]. Available: [https://developer.spotify.com]. Accessed: Dec. 28, 2023.