CM12002 – Coursework 2: Automatic Disco System

Project Report

1.0. Introduction to the Problem Area

The nightlife entertainment sector, particularly nightclubs and event spaces, has historically been characterized by high energy usage, with significant carbon footprints attributable to non-optimized lighting and sound systems. The challenge is compounded by the absence of systems that adapt to fluctuating occupancy levels, leading to a continuous and often unnecessary consumption of electricity. The motivation behind the development of an automatic disco system is to introduce a technological solution that not only addresses the issue of energy waste but also integrates seamlessly with the dynamic atmosphere that these venues aim to create. This system serves the dual purpose of enhancing environmental sustainability and providing an adaptable, responsive entertainment experience.

1.1. Motivation for Selecting the Problem Area

The selection of this problem area is motivated by the increasing social requirement to reduce operational costs for entertainment venues while adhering to environmental regulations and sustainability goals. By employing a system that automatically adjusts music based on real-time occupancy, energy consumption is going to be substantially lowered, thus contributing to both ecological and economic benefits. Additionally, the feature that moderates sound levels correlates to the growing awareness of hearing health, positioning the system as a health-conscious innovation. This project also reflects the broader societal shift towards intelligent automation.

2.0. Literature and Technology Review

In this project, the concept of the Internet of Things (IoT) is central to achieving an intelligent and responsive nightclub environment. IoT describes devices with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks [1]. Key components like infra-red sensors for monitoring occupancy and microphones for noise level assessment are integral to system functionality. They provide critical real-time data, enabling the system to automatically adjust to varying conditions in the nightclub, such as modifying sound levels or managing the system's use based on the number of attendees present. This approach exemplifies the utilization of IoT for enhancing operational efficiency and satisfying experience in dynamic public spaces.

The IoT plays a significant role in enhancing energy efficiency within buildings. Research has demonstrated that IoT can lead to substantial energy savings in commercial buildings. This technology, involving interconnected devices and sensors, is essential in optimizing energy use, especially in high-energy environments like nightclubs [2, 3].

Sound management is of paramount importance in such settings, due to the health implications of prolonged exposure to high noise levels. Studies show that excessive noise in nightclubs can lead to noise-induced hearing loss. Regulations require sound levels in work environments to be controlled, but nightclub attendees are often unaware of the risks associated with high noise levels reflecting the economic downside of imperfect information present in the industry [4, 5].

3.0. Development Process

3.1 Preparation Phase

Before commencing to work on the project, multiple meetings were organized with the intent to allocate responsibilities and come up with a plan. After analysing features and a creating a plan for our implementation, it was decided that the group would split up in pairs. Each pair took on a share of workload, which then got split between the two individuals in each group. Such division of labour allowed for efficient communication and productive workflow with the assistance of a GitHub repository to collaborate.

3.2. Hardware Development & Implementation

The system consists of three Arduino boards: two "slave" boards and one "master". However, to work autonomously each pair in our group was allocated one slave and one master, with the idea that two masters will be merged later on. The master is the control board that would be in the club's DJ booth (see "Master", fig. 1). Initially, it was supposed to have five components: a speaker outside the booth, pause/play button, manual volume control, SD card reader and an LCD screen showing the current volume level. However, it was later realised that the LCD screen could not be accommodated on the master board due to a lack of space, hence it was moved to one of the slave boards (see "Slave 1", fig. 1). The same slave board also has an array of LEDs representing the lighting ("disco") system, and a microphone for monitoring noise level. The second slave (see "Slave 2", fig. 1) has two infra-red sensors that count the number of people in the room — one at the entrance and one at the exit. Currently, both detectors are configured with a visual range of 1.5 meters and a viewing angle of 180 degrees and require precise placement to ensure correct readings are taken and there is no angle overlap.

Enabling the master to read “.wav” files from the SD card and to output these via the speaker required developing an understanding of the Micro-SD Breakout board [6]. The audio files must satisfy a tight set of requirements: they must be mono, have a sample rate of 16.0kHz and a bit depth of 8 bits. It then became apparent that the digital pins were not providing the speaker with enough current to output the desired volume level. Resolving this issue required implementing a transistor, connecting the 5V power supply (collector) to the speaker (emitter) via a digital pin (base), which increased the current. Moreover, a potentiometer was attached to the base input enabling the user to increase and decrease the volume manually. This is more effective than using the volume method provided by the TMRpcm library because it keeps the code complexity to a minimum.

Research revealed that electrical noise could disrupt the audio signal which the microphone, attached to Slave 1 (fig. 1), needs to pick up [5]. Hence, it was connected to the 3.3V power supply to mitigate noise problems and because this provides a more accurate depiction of the frequencies being picked up than the 5V power supply.

We utilised I2C to allow our master and slave boards to communicate, with an extra breadboard implemented between them to handle the connections between Arduinos. We communicate in as simple a way as possible, sending Boolean values rather than longer messages which would require further processing upon receival.

3.3. Software Development & Implementation

The SPI, SD and TMRpcm libraries, amongst others, are required to communicate with and play .wav files from the Micro-SD Breakout board. Implementing the audio functionality required extensive research to develop an understanding of these. [8, 9, 10] Furthermore, storing the ".wav" filenames raised scope problems because it was initially achieved using a list initialised in the "setup()" function. However, this resulted in it being outside the scope of the "loop()" function. This problem was solved by pre-defining a key-value map data structure in the "setup()" function and using this to map file names to an index ready for the user to select. The Master receives volume readings from Slave 1 (fig. 1) via I2C communication. However, minimising the number of bits transferred enables effective distribution of resources, such as time and processing power. Consequently, the Master only requests volume readings every five seconds and achieves this without delaying execution by making use of the "millis()" function.

The microphone, attached to Slave 1 (fig. 1), produces an analogue value corresponding to the electrical signal produced by the change in capacitance caused by soundwaves as they hit the diaphragm of the microphone. [11] To convert this electrical signal to the amplitude of the soundwave, Slave 1 (fig. 1) calculates the difference between the maximum and minimum voltage detected over a two second period. The calculated value is compared to a pre-defined range, if it exceeds the range then Slave 1 (fig. 1) sends "true" to the master board which is interpreted as an indication the volume needs to be lowered.

As mentioned, Slave 2 (fig. 1) has two infra-red sensors, one representing people entering the nightclub and another representing people leaving. It was initially planned to use interrupts to perform functionality upon a change of state for either infra-red sensor, however after experimentation with this it was decided to use a regular check function in the main loop, as the interrupts had too much delay, bottlenecking other parts of the system. A counter is used to record how many people are in the nightclub, if this counter is zero, Slave 2 (fig. 1) sends a "true" to the Master board (fig. 2) to indicate that music should be turned off.

4.0. Evaluation of the Implementation and Future Improvements

Throughout the project development, numerous challenges were encountered, particularly due to the limitations of the Arduino Uno. These challenges included the quality of components, limited space on the boards, and other hardware constraints inherent to a less powerful microcontroller. Below is an evaluation of how the impact of these limitations was mitigated or minimized in the final product.

4.1. Evaluation of the Hardware Implementation

As mentioned in 3.2, it was required to perform various conversions to audio files to allow compatibility with the micro-SD reader Converting the .wav files to this format proved difficult and significantly lowered the quality of the audio. As a result, this practise may not be suitable for real-world use. Given more time, an alternative such as USB flash memory would be considered.

Originally, the Master was intended to increase and decrease the volume depending on the data received from Slave 1. However, testing revealed that the size of the components means that the microphone fails to pick up music from the speaker unless they are in close proximity. The Master would often maximise the volume beyond safe levels and thus its ability to increase the volume was removed. In the future, more powerful components would be utilized to effectively re-implement this feature.

As mentioned in Section 3.2, a migration of the LCD display to Slave 1 (fig. 1) was necessitated due to space limitations on the master board. While relocating the LCD display close to other visual outputs, such as lighting, was deemed logical, there were backend issues encountered with this implementation. Currently, the LCD display indicates when music is playing, shows the current volume level, and alerts when the volume is lowered. In the initial design, it was intended for the LCD to display the name of the currently playing song, along with corresponding metadata such as the duration of the song and the next song in the queue. This feature would be modified with fewer space limitations and more time available for development.

Additionally, the implementation of a more complex lighting system was considered. The initial plans included the potential to synchronize lighting patterns with the BPM of a song. However, this would have involved communication from the master board (fig. 1) to Slave 1 board (fig. 1), potentially disrupting the established communication mechanisms. Although workarounds were feasible, the speaker and micro-SD reader experienced slight latency due to lower quality hardware. This latency made it challenging to synchronize the start of the song with the initiation of the lighting sequence, aligning with the BPM. Future improvements would involve investing in higher quality hardware for both the speaker and micro-SD reader to reduce latency and enhance synchronization capabilities.

4.2. Evaluation of the software implementation

A significant challenge was encountered when attempting to facilitate multitasking on the Slave 1 (fig. 1) board. The initial design involved simultaneous operation of lighting sequencing and microphone functionality. However, due to the hardware limitations of the Arduino Uno, difficulties were faced in realizing this functionality. Subsequent research into the ATmega328 [12] revealed that its single-core architecture with a 20MHz clock speed at 5V did not possess multitasking ability like multi-core, higher-spec processing units [13]. The ‘Threading’ library was utilized to create a separate thread for the microphone functionality, resulting in a temporary halt of the lighting while the microphone was active. This approach managed to create smooth transitions between functions, with lighting pausing for only 2 seconds before resuming. Interestingly, it was observed that using a threaded approach led to a shorter delay in switching functions compared to sequential execution in the main loop.

Experimentation determined that a 2-second interval was adequate for obtaining precise volume level readings, followed by a resumption of lighting effects for approximately 5 seconds. The constraints of the given hardware limited the extent to which this challenge could be addressed, but further exploration into the ‘Protothreads’ C library [13] might offer a more efficient solution. For future iterations of the project, the use of a microprocessor capable of light multitasking may be appropriate, or further experimentation with other threading techniques could be explored given additional time.

Storing the song file names found itself to be more complex than we first thought. After attempting to use a regular array, we found string corruption occurred after the program had run for a few minutes. We mitigated this by creating a map-type data structure which pairs an integer to a string as a key-value pairing. This allowed us to pre-allocate memory before inserting strings into the data structure, as-well as simplifying the user song selection process. However, communication between the slave boards and master board proved problematic; upon waiting for a message from an I2C bus, the main master loop would pause. This was resolved by checking

the bus was ready prior to requesting data from it, allowing other parts of the program to continue and due to the speed at which the main loop runs, it is a matter of milliseconds before the bus is checked for data again meaning there is still minimal delay.

5.0. Conclusion

The development of this system has been met with great satisfaction. Efficient workload and time management was achieved, and the collaboration within the group was notably successful. While there are minor improvements that could be made, as detailed in Section 4.0, the final product meets the standards of the project scope with comfort. The research, analysis, and implementation of the initial plans have significantly enhanced the understanding of Arduino architecture and the C++ language. Moreover, the project has honed skills in transforming an initial concept into a real-life product. The end result is an efficient and practical system, equipped to manage a nightclub in an environmentally friendly manner on a scaled down level.

6.0. References

[1] “Internet of things”, Wikipedia, Available from: https://en.wikipedia.org/wiki/Internet_of_things [Accessed 5 December 2023].

[2] Feng H., 2022, “Investigating energy savings when using iot devices in buildings: a case study in the uk”. In B.G Soto, 24-26 July 2022, Ixia, Rhodes, Greece. Published at researchgate.net.

[3] Vadruccio R., 2022, “Increasing energy efficiency in Smart Building through Internet of Things retrofitting intervention”. In B.V. Elsevier, 2022, Milan, Italy. Published at sciencedirect.com.

[4] “Excessive noise levels in nightclubs may cause noise-induced hearing loss”, Cambridge Core Blog, 2014. Available from: https://www.cambridge.org/core/blog/2014/02/25/turn-down-the-volume-an-examination-of-the-effects-of-nightclubs-on-hearing/ [Accessed 5 December 2023].

[5] “Noise Levels in Nightclubs May Increase Risk of Hearing Loss”, AOC, 2016. Available from: http://www.aocphysicians.com/blog/noise-levels-in-nightclubs-may-increase-risk-of-hearing-loss [Accessed 5 December 2023].

[6] “Micro SD Card Breakout Board Tutorial”, Adafruit Ind., New York, 2023. Available from: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-micro-sd-breakout-board-card-tutorial.pdf [Accessed 6 December 2023].

[7] “MAX4466 Microphone Preamplifier Module”, Proto Supplies, 2023. Available from: https://protosupplies.com/product/max4466-microphone-preamplifier-module/ [Accessed 6 December 2023]

[8] “Arduino & Serial Peripheral Interface (SPI)”, Arduino, 2023. Available from: https://docs.arduino.cc/learn/communication/spi [Accessed 5 December 2023]

[9] “SD”, Arduino, 2023. Available from: https://github.com/arduino-libraries/SD [Accessed 6 December 2023]

[10] “TMRpcm”, TMRh20, 2023. Available from: https://github.com/TMRh20/TMRpcm [Accessed 6 December 2023]

[11] Mitchell, G., “How does a microphone pick up sounds?”, BBC Science Focus. Available from: https://www.sciencefocus.com/science/how-does-a-microphone-pick-up-sounds [Accessed 6 December 2023]

[12] “ATMega328(P)”, Wikipedia, 2023. Available from: https://en.wikipedia.org/wiki/ATmega328 [Accessed 6 December 2023]

[13] “Why multi-threading is not possible on an Arduino”, Robots Backend, 2022. Available from: https://roboticsbackend.com/how-to-do-multitasking-with-arduino/ [Accessed 6 December 2023]

7.0 Appendix

Figure 1: The Master and Slave Boards

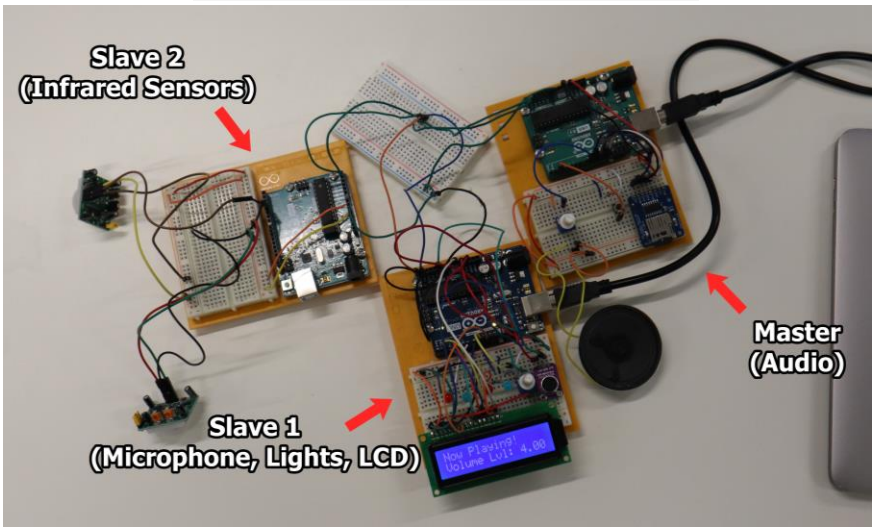

Figure 2: The Master Board

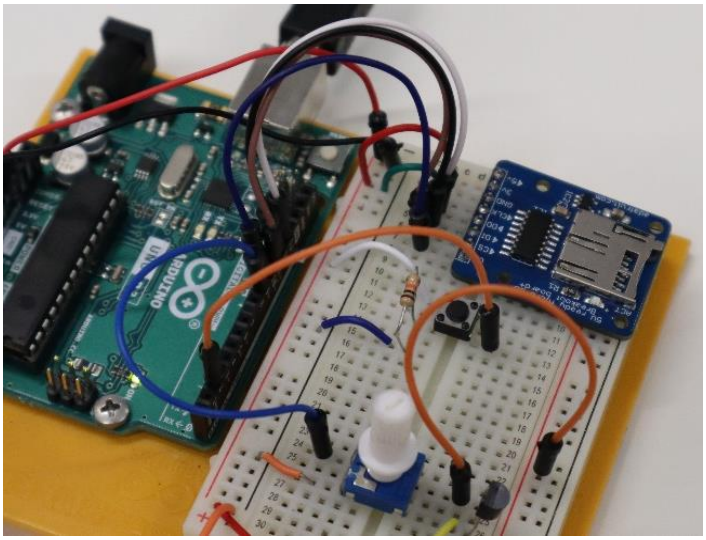

Figure 3: Fritzing Sketch of our implementation