ESP32 Lighting Controller

CS39440 – Major Project

Kieran Todd

Ktt1

Richard Shipman

G400 – Computer Science

1.0 – for Review

I’ll make a real cover page after review

# Project Description

The aim of the ESP32 lighting project is to create a piece of software that easily gives the user a way of configuring and displaying multiple different LED and motor sequences when a condition is met. The lighting controller will have an easy-to-use web based front end for the user to assign lighting profiles to the different LEDs and the conditions they will start lighting up under, when and how motors will spin and to create and edit lighting profiles that the LEDs display.

When setting the LEDs profiles and changing the motors configuration you will have the option of setting the LED/motor to be timed or reactive. When a motor/LED is timed it will carry out what is intended (whether that is the LED profile or the motor spinning a certain way) repeatedly after the start time has passed until the end time. When a motor or a LED is reactive it will carry out the intended task a set number of times after the reactant specified is met. The reactants are the weather (e.g. When it rains) or when a sensor is tripped. The main target audience for this product is model makers looking for custom LED and motor+ configurations in their model book nooks and model train sets.

# Proposed Tasks

Firstly I need to thoroughly look through the process of making and hosting a website that is visually appealing on the ESP32, to do this I will have to brush up on my CSS and do some spike work and pseudocode planning as to how I will run the lighting controller and the website in parallel.[1]

Also, as the LEDs and motors will have to react and preform their desired actions when it is a certain time, or a weather condition is met I will have to look up an API or an SDK for the weather [2] and the Clock functionality [3].

# Project Deliverables

Simple and easy to use website that looks good – As the intended audience (Model Creators) may not have in depth knowledge on websites and coding, it has to be able to be used by everyone, with a good look and feel.

LEDs and motors respond to changes the user makes on the website

Final system – A system that meets all of the points discussed earlier in the document which will be programmed in the Arduino IDE. I will be able to demonstrate the system fully meeting all of the points discussed and with working examples.

Documentation – The documentation will cover all aspects of setting up and operating the system, including the initial setup of the system, creating new LED profiles, how to assign them to LEDs and the conditions you can set the LEDs and Motors to activate.

Testing documentation – Documents containing the outcome of the System, how it meets my initial expectations of the system and if it falls short, where and why it does

Final Report – The final report is a full report containing the process I would have went trough making the system, the changes I would have made along the way and then a full description on what the system does and how it operates. The report will also contain a full comparison from the final report to what I expected to make, including the testing documentation and how I would develop the system further if I had the time following this project.

# Bibliography

[1] When it comes to hosting the webserver from the ESP32 I have done some research into how to host the website and how it can be accessed, most of the information I have obtained has come from <https://randomnerdtutorials.com/esp32-web-server-arduino-ide/>

I have yet to do any research into CSS.

[2] When it comes to the weather API I have had a look around and one of the better more recognised weather APIs would be OpenWeather : <https://openweathermap.org/>

This will return hourly weather temperatures and weather forecasts for the users area.

[3] I have done some research into some Clock APIs that will return the time one of them being: <http://worldtimeapi.org/>

I have yet to decide and may do some research into some different APIS