ESP32 Lighting Controller

CS39440 Major Project Report

Author: Kieran Todd (ktt1@aber.ac.uk)
Supervisor: Richard Shipman (rcs@aber.ac.uk)

20th April 2021
Version: 4.0 (Release)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Kieran Todd

Date: 30th April 2021

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Kieran Todd

Date: 30th April 2021

Acknowledgements

I am grateful to all of the computer science lectures that have helped during my time at Aberystwyth. Specifically I would like to thank Neil Taylor and Richard Shipman. Neil Taylor has helped me and understood me whenever I have had hard times throughout the year and has truly been a supportive head of year. Richard has been a fantastic supervisor during this project, and I am thankful for his support and advice.

I would also like to thank one of my closest friends at university, Ryan Priest, for keeping me sane (mostly) over the last 4 years.

Lastly I am grateful towards my mother who has always supported me thought my time at university and has helped me in whatever way she can.

Abstract

This report aims to outline the process and the decisions made through the duration of my ESP32 lighting controller that was created in the Arduino IDE environment. The aim of the project was to make a fully customisable light emitting diode (LED) controller, with two different LED output options, for the use in dioramas and model scenarios. These LED sequences could either be played constantly, or a trigger could be set on the LED profile. These triggers have parameters that define under what conditions the LEDs would display its LED sequence. The program also has a website front end, that was hosted from the ESP32, that allows the user to create and edit the LED profiles and define the triggers and its parameters.

The program has the ability for expansion to include other types of triggers and outputs. In the planning section of the project, ideas were included to implement servos, motors and sound into a range of the output options. The ideas suggested allowed for moving parts to be displayed via a trigger in a similar way as the LED profiles were activated. Also with the addition of sound, a MP3 file could be triggered when a profile was displayed. On completion, this would allow for the user to create displays like a level crossings with rising barriers, flashing lights and a warning sound preformed when a train passes by. These features did not get implemented however, as they did not fit within the timescale of the project.

Contents

List of Figures

List of Tables

Chapter 1

Background & Objectives

1.1 Background

The aim of the project was to make a fully customisable light emitting diode (LED) controller for the use in dioramas and model scenarios. It was essential to build capacity within the program that allowed for future expansion to accommodate a range of outputs and inputs. The concept of the program was to enable the user to create many lighting sequences. The lighting sequences consisted of a range of brightness levels and an assortment of colours. A feature of the program was to ensure there were minimal constraints on the creation on the lighting sequences as possible. The user set this lighting sequence to display on a specific LED. Also, the user was able to define the conditions under which the LED profile would play (the LEDs trigger). Some of the base triggers initially formulated were time, temperature, weather, button and constant.

The concept behind the customisable LED controller came from the current marketplace for model LED setups. Malcom Miniatures are the manufacturers of one of the most popular LED controllers on the marketplace [1]. However, these controllers are limited as they display only two lighting sequences i.e. candles and fire. The lighting sequences can only be displayed on standard LEDs. The controller has no functionality or expandability to allow for the use of assignable LEDs. Also, these LEDs were restricted to either being ‘on’ or ‘off’; there was no way of assigning a condition to the program that allowed the user to control how the lighting sequence was displayed. This was undesirable as having LEDs powered constantly could be seen as an eyesore, unimaginative or not engaging for the user. Also, turning the unit on and off could have been a burden.

1.2 Analysis

The program was created on the ESP32 microcontroller. The reason it was used over the other brands of microcontrollers was due to the following reasons:

- In-built WIFI capability - The ESP32 has WIFI funtionality attached to the microcon-

toller. This is benificial as the program was able to host a website front end without the use of an external WIFI card.

- Numerous output pins with pulse width modulation (PWM) enabled - On the ESP32 there were more PWM pins than most other microcontrollers. This allowed the use of a greater number of standard LEDs. The LEDs were assigned a sequence of brightness that were displayed rather than being fixed at an 'on' or 'off' state.
- End-user specification - In the initial project specification the microcontroller that was identified as a 'must have' was the ESP32. This was due to the reasons outlined above.

The main reason the ESP32 microcontroller was selected over the other brand of micro controllers (other than the in-built WIFI capacity) was because, as stated above, the ESP32 has more PWM pins. Comparatively, if we look at other micro controllers like the ESP8266(4 PWM pins) and the Arduino Nano(6 PWM pins) have a fraction of the PWM pins comparted to the ESP32 (25 PWM pins).

The program created allowed for two different LED outputs: standard LEDs and addressable LEDs. The operation of the LEDS by a user was via a website. Standard LEDs are regular LEDs with a fixed colour. When powered, these LEDs display a brightness which is fixed to either a High (on) or Low (off) state. Addressable LEDs display a colour. The use of three different variables, red, green or blue, defines the colour output on the addressable LEDs. For example, if a purple output was required then the red and blue values were set to a high value while the green value was set to zero.

The standard LEDs and addressable LEDs were connected to the ESP32 microcontroller output pins. Some of the output pins on the ESP32 have pulse width modulation (PWM) enabled, which allowed us to display a brightness from 0 (off) to 255 (brightest setting) on the standard LEDs. PWM allowed us to reduce the power delivered to the standard LED by having it powered 'off' and 'on' multiple times a second. The longer the time the LED was 'off' compared to 'on' gave a dimmer LED. However, when the LED was powered 'on' for a longer amount of time than 'off', this displayed a brighter LED.

Both standard and addressable LED profiles have a brightness/colour sequence of an undefined length. Each LED profile displayed the next colour/brightness in the array after a specified amount of iterations had passed. An iteration is one of the repetitions of a process that generates a specific outcome. In this program an iteration was one of the many times the program went through the main lighting loop. When creating the LED profile, the user defined how many iterations through the main lighting loop it would take before moving on to the next colour/brightness e.g. each iteration of the main LED array was set to take 200 milliseconds and the next brightness/colour was displayed after three iterations, therefore each brightness was displayed for 600 milliseconds (0.6 seconds) before moving on to the next colour/brightness in the array.

The program allowed for the use of servos to move parts within a diorama e.g. a level-crossing barrier or a character. If the user were defining a servo profile, the user would set the degree (angle) the servo would turn to and the speed at which it would move. Also, the program allowed for the use of motors. The motors allowed for scenes such as lighthouse lights or a merry-go-round to revolve around a point at a defined speed for a specified

length of time. Both the servo and the motor had a trigger, just like the LED profiles, to which it performed the desired reaction when the trigger was true.

Each LED was assigned a LED profile by the user which performed a specific action when a specified trigger came into fruition. Each of these profiles were given a 'trigger' which defined what conditions the LED started displaying its lighting/ brightness sequence. The standard triggers were:

- Time- The user defines a start and an end time. When the geographical time falls between the start and end time, the LED profile plays continuously. Once the geographical time surpasses the end time, the LED switches off.
- Weather - The user selects a location and is given a list of different weather states. When the weather in the defines location matched the correct state the LED profile played until the weather changes.
- Temperature - The user defines a temperature in degrees Celsius. They also define whether they would like the profile to trigger when the current temperature is above or below the defined trigger.
- Button – The button used by the program has two states. When activating the button (in its 'on' state) the profile plays. When deactivating the button (in its 'off' state) the profile ceases to play.
- Constant – The LED plays continuously when the trigger is set to constant.

The aim of the website was to give the user a manageable way of changing the triggers and LED brightness/colour levels as stated above. The user was able to change, retrieve and edit the LED profiles and then assign LED profiles to the respective LEDs easily. The main purpose of the website was for any user to be able to understand and navigate the website controls even if they have a limited computer knowledge.

1.3 Process

The process that was adopted to complete the project was the agile methodology. Other methodologies considered were incremental development and continuous integration. The agile methodology was preferred as its main features include rapid production of working code and emphasis on what the customer would want. Specifically, it was decided that Scrum agile methodology would be used. However, this methodology had to be adapted slightly as Scrum was created for use in small teams not a singular person. After reading a helpful and easy to read article on "Scrum for One" by Alex Andrews [2], it was decided to adopt Scrum with the following changes:

- Daily scrum - This is a personal five-minute review of what was completed yesterday, specifically the elements that did not get completed, then plan the task(s) for the day prioritising what was not completed previously.

- Sprint - This is completed in the same way as a normal scrum; it is just trying your best to accomplish what was proposed during the daily scrum.
- Story time - Normally completed at the end of the week, Story time was time spent away from the desk thinking about the implementation of features and tasks for the next sprint week.
- Release - This time was spent with the customer (my supervisor) explaining the tasks completed, future tasks and problems encountered.
- Sprint plan- This was where the plan for the next week's sprint tasks were completed.

Chapter 2

Requirements

At the beginning of this project, a list of requirements was defined. It was then decided that a MoSCoW analysis would be performed on the list of requirements. Each requirement was given one of the four letters which are:

- M - Must have - This is the highest priority letter. These requirements were necessary to the completion of the project and must be prioritised.
- S - Should have - These requirements were not vital to the completion of the project but were features and additions that were expected to be completed by the end of the project.
- C - Could have - The 'could have' requirements were desirables that were not vital to the completion of the project but were additions to the project that made it function better.
- W - Will not have - The lowest priority letter. these requirements were normally out of the scope of the project and were too ambitious for the time frame so they would not get completed.

This analysis helped me understand what requirements should be focused on and gave a better picture of what the final product may look like. Requirement one to seven are requirements of the code, while requirement eight to sixteen are website requirements.

2.1 RQ1 - ESP 32 must react to the changes made by the user (M)

When a LED configuration is changed an any way (trigger/trigger parameters changed, profile changed, LED attached) the ESP32 responds to these changes and displays the new LED configuration correctly.

2.2 RQ2 - Display the brightness and the colours of the LEDs correctly (M)

When the brightness or the colour of the LEDs change, for example when the display moves on to the next section of the array, the LEDs will display the new brightness/colour in the intended way.

2.3 RQ3 - Perform defined output when trigger is met. (M)

While a trigger is active, the LED brightness/colour sequence is displayed to the LED.

2.3.1 RQ3.1 - Time trigger(S)

The user defines a start time and an end time, when the geographical time falls between these two times the trigger is considered active.

2.3.2 RQ3.2 - Weather trigger(C)

The user defines a weather state and the location they are situated in. When the weather specified is happening in the defined location, the trigger is considered active e.g. set trigger to active when it is raining in Bridgend.

2.3.3 RQ3.3 - Temperature trigger(C)

The user defines a temperature threshold and whether they would want the trigger to be active when the surrounding temperature is above or below the specified temperature threshold.

2.3.4 RQ3.4 - Button trigger(C)

When the attached button is pressed, the trigger is set to active. When the button is pressed again the profile is now considered not active.

2.3.5 RQ3.5 - Sensor trigger (C)

When movement is detected by the attached sensor, the profile is active until the display sequence finishes.

2.4 RQ4 - Will have 2-3 pre-set standard and addressable profiles for fire, constant and strobe(W)

The completed program will have two or three defined profiles for some of the standard lighting scenarios that the user may want to display. These profiles are fire (a flickering light to simulate flickering flames), constant (a permanant brightness that simulates a room light) and strobe (a flashing light to simulate a warning light).

2.5 RQ5 - Play MP3 file when trigger is met (W)

When the trigger is considered active and starts playing the lighting sequence, a defined mp3 file is played. For example, when the flashing LEDs for the level crossing is displayed, the warning sound associated with it will be played.

2.6 RQ6 - Will have appripriate storage to store the MP3 audio files(W)

To enable the mp3 file to be played, there will be a storage solution that will contain the desired mp3 files to be played as there is not enough memory on the ESP32. A suitable storage soloution would be a SD card.

2.7 RQ7 - The ESP32 must be easy to set up (S)

The ESP32 controller must be easily operated and set up by users of all abilities incuding those with limited or no coding backround.

2.8 RQ8 - Create a user-friendly website (S) and RQ9 - Easy to use and navigate (S)

The website must be easy to operate. This is to ensure that the program is accessible to all abilities. The use of limited inputs via dropdown boxes and buttons, allow users to easily navigate the website and its functionalities to create the desired effects. These two requirments are paired as they both aim to solve the same problem.

2.9 RQ10 - Using the Website, define the ‘trigger’ for the outputs implemented. (S)

The user is able to change and edit the triggers and its perameters that activate the LED profiles. When the trigger for the Profile is changed, suitable entry methods will appear to define the new trigger perameters.

2.10 RQ11 - Create standard LED profiles (M)

The website should have a section where the user can create a standard LED sequence with as little to no restrictions on the final display as possible.

2.11 RQ12 - Set the standard LED profiles to the LEDs (M)

Once a standard LED profile has been made, the user should be able to assign this new profile to one of the standard LEDs via a dropdown box.

2.12 RQ13 - Create assignable LED profiles (M)

The website should have a separate section where the user can create an assignable LED sequence with as little to no restrictions on the final display as possible.

2.13 RQ14 - Set the assignable LED profiles to the assignable LEDs (M)

Once an assignable LED profile has been made, the user should be able to allocate this new profile to to one of the assignable LEDs via a dropdown box.

2.14 RQ15 - Define the intended motor and servo configuration (C)

The user should be able to specify the desired output for the servos and motors. The user should also be able to set the trigger that performs the defined output.

2.15 RQ16 - Upload a small MP3 file to play when defined by a trigger (W)

The user could be given the option of uploading a MP3 file to the profile. This would then play the MP3 file when the trigger becomes activated.

Chapter 3

Design

This design chapter aims to outline the design ideas and thought processes of the initial planning stage of the project.

3.1 The LED Profile Objects

The LED profile object was the main container of information in the code that stored all of the visual data that was printed to the LEDs. It was essential to have two versions of the LED profile. This was because the standard and addressable LEDs needed to store different content. The initial plan was to have the following elements stored in the LED profile object:

- Profile Name - A modifiable string with no limitations. Initially, this was a base name that the user was directed to change at the beginning of its creation to identify the profile e.g. Fireplace, welding lights, flickering bulb etc.
- Trigger - This element would come in the form of a String which would be restricted to being words which relate to the different types of triggers talked about in 1.2 Analysis e.g. trigger = ”Time”
- Trigger parameters - Preceding the trigger were all of the possible different parameter variables e.g. the start and end time, temperature threshold etc . These supplemented the trigger. They identified the specific environmental conditions that activated the profile and commenced specific activities within the diorama or model.
- LED Brightness array - This was the main array for the standard LED profile. This was an unrestricted array containing all the brightness values available for the main lighting loop to access when activating the LEDs.
- Addressable LED colour array - The addressable LED profiles used a similar container to the brightness array from the standard LED profiles to display the final colour. The brightness array was replaced by three different arrays corresponding to red, green and blue. For example, When the values from the first element in each

array were inputted into the display function of the LED, the LED would display the appropriate RGB colour. This was repeated until the desired colour sequence was achieved.

- LED duration - The LED duration is an integer that defined the number of iterations needed to pass before the next brightness is displayed.

3.2 The LED Object

While the LED profile object (outlined above) contains all of the information on how the LED is displayed, the LED object stores all of the information about the LED itself. These are the elements that would be stored in the LED object:

- LED channel - This is the value that was referenced each time we wanted to print a new brightness to the LED.
- PinOut - An integer to represent which pin was attached to the LED.
- The LED profile - This was how the two objects were connected to each other. The LED object contained a profile object which was the profile assigned to that LED.

3.3 Code Design

One of the biggest design concepts implemented into the coding element of the project was the idea of flexibility. Where there was greater flexibility in the type of profile the user developed, this created a better range of lighting sequences. In turn, this produced a more realistic lighting effect for the diorama/model scenario.

Flexibility within the project were a fundamental part of the design. Customisation of the project enabled the user to fully engage with the final design of the lighting for the diorama. This customisation allowed the user to vary the length of the LED profile and how long the LED profile played for. This was achieved by the user being able to customise the following:

3.3.1 Duration of the iteration

By adding a customisable interaction size to the main code loop of the program gave the user more flexibility in the final display performed by the LEDs. For example, if the iteration size were smaller, this allowed the user to have a smoother transition between colours e.g. the changing colour of the sky. If the iteration size was larger, it allowed for displays such as red flashing lights at a train crossing. This time scale had its limitations however, depending on how long it took for the code to traverse through the main code loop.

3.3.2 Number of iterations before next colour/ brightness display

This allowed the user to have multiple LEDs transitioning at different time scales. For instance, this allowed both of the LED functions talked about in the previous points (fast gradual transitions and slow dramatic transitions) to be displayed at the same time.

3.3.3 Unrestricted length of profile array(s)

By having an unrestricted profile length this allowed the user to have any number of brightness/colours before the end of the array. However, this restricted how often the profile could be activated by its trigger and was stuck completing the entire sequence before it was able to be played again e.g. The lighting sequence cannot be stopped midway through the display.

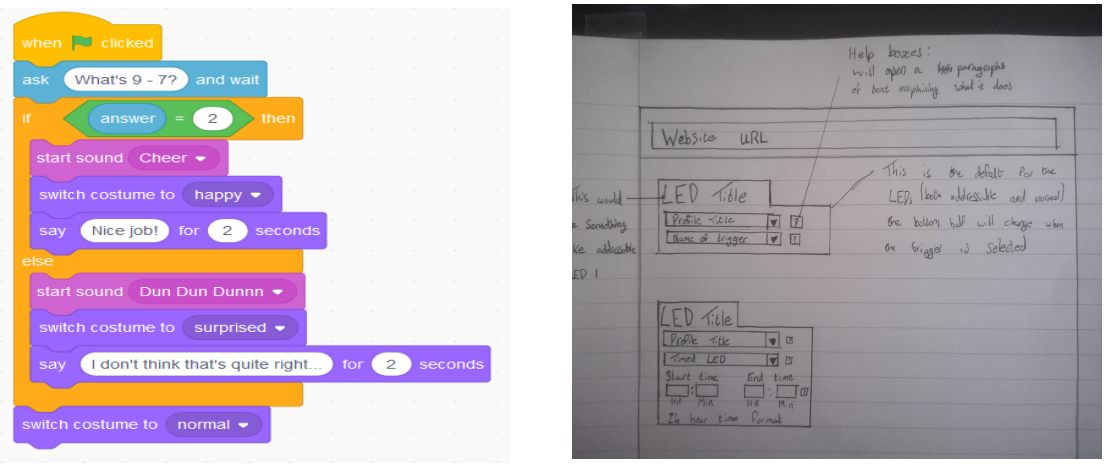
3.3.4 The design of the main lighting loop

If the main lighting loop , which is what sets the LEDs to their specified colour/brightness, was coded in the correct way, it allowed for any number of LEDs to be played at any one time. However, the more LEDs active at one time limited how fast the iteration could be completed.

3.4 Website Design

The main aspect that needed to be considered with the website design was its ease of use. As some of the clientele for the program may not be tech-savvy or experienced in coding, it was imperative that the program webpage was clearly laid out and the profiles were easily modified. The overall design chosen is loosely based on the code blocks from a programming learning tool called Scratch.

It was decided the use of code blocks, which is a similar concept on which Scratch is based on [3], would make it easier for the user to understand what they were currently working on. When a trigger was chosen for the LED, additional input options were added to choose the parameters of the trigger. The additional input options depended on what trigger was chosen. For instance, if the time trigger were selected, four additional boxes appeared so that the user could enter the start and end time in hours and minutes. However, if the temperature trigger was selected, only one box would appear along with an additional drop-down box. This enabled the user to enter the threshold in which the temperature would trigger and then whether they wanted it triggering above or below this threshold. Also, to make it easier for the user to understand the expected input, there was a help box that the user could access for a description and a explanation. This showed the user a paragraph of text explaining what they were choosing and what the expected input would look like.



(a) scratch example

(b) Drawing of intended UI Design

Figure 3.1: Comparison Between Scratch and My UI Design

Chapter 4

Implementation

This implementation phase shows the process of my work. Showing how the project was implemented and the problems that were encountered during its creation along with realisations that caused me to adapt my final product from the initial design. The main resource used throught this project was several guides on the ESP32 by a website that offers guides and projects called 'Random Nerd Tutorials' [4]

4.1 Start of the Implementation Phase (Sprint One)

When starting the project, the code produced had some initial iterations that were not up to the level expected. One of the biggest problems encountered was around information storage and how to do this effectively and in the correct manner. One such example of this was the standard LED profile array. The main help used in this sprint was a random nerd tutorial on using PWM pins on a ESP32 [5].

4.1.1 LED profile array size

When the program was still in the first weeks of development, there were problems with the standard LEDs displaying a dim brightness even though the brightness value that had been assigned to the LED was quite high. The problem persisted even after the hardware (the LEDs, resistors and wires) had been replaced. My supervisor suggested using printout statements to identify what value the brightness array was giving to the main lighting loop which revealed the values coming back were a lot lower than the values that had been set. The problem was the array was never initialised which meant values were not being stored in the correct place in the memory which caused random values to be generated instead of the expected stored values. As a solution, a function was created within the object. This function declares the size of the array that is being used before any values are inputted. However, this is limiting due to the fact that once a profile is created, the size of the profile array cannot be edited. A new LED profile has to be created to have a longer profile array.

4.1.2 LED and LED profile objects

As stated in the design and analysis chapters of the report, all the information was stored on objects that were named LED profiles e.g. the display array, duration, trigger and name of the profile. Initially, the plan was to have an object for the LEDs and an object for the LED profile. The LED profile would store everything about what was being displayed and how e.g. the display array, duration, trigger and name of the profile. On the other hand, the LED object stored the LED channel, the pin the LED its attached to and the LED profile object. However, it was decided that the information for the profiles was better stored on the LED profile. This was because when it came to the main lighting loop of the controller, it quickly became confusing when trying to access the information needed to display the correct brightness/colours. This was due to the fact that in order to access the brightness array, the array inside of an object which was inside of another object had to be referenced. This created long lines of code that quickly became very confusing and hard to read.

```
class LEDprofile{
  public:
  String trigger;
  int shr;
  int smin;
  int ehr;
  int emin;
  float degC;
  bool below = true;

  String id;
  int leng;
  int *brightnes;
  int lengPos = 0;
  int duration;
  int pinOut;
  int count = 0;
  int LEDchannel;
  void arrayInitalise(int l){
    leng = l;
    brightnes = new int[leng];
  }
};
```

4.2 Implementing the Addressable LEDs (Sprint Two)

After these two problems were resolved, the next implementation stage of the project was the use of addressable LEDs. This stage of the project needed to be revised considerably when comparing it to the initial design ideas. The addressable LEDs were not a single LED like the standard LEDs, but a group of LEDs connected in a ring. The solution was to accommodate this by having another object that had an array containing all of the LED profiles for that ring light. In this instance, the ring light that was being tested, had eight LEDs on the ring. So the ring light object had eight assignable LED profiles stored within the LED profile array.

4.2.1 Function for the assignable LEDs

The assignable LEDs used a function called 'Adafruit Neopixel' to display the colours to the ring light. This function required an object to be made and 'called to'. The function was then able to display colours on the assignable LEDs. Initially, when first implementing the assignable LEDs, it was thought that this object could be created and then 'called to' within the ring light object. However, when implemented in this way, the assignable LED always displayed two blue LEDs and one white LED. This did not alter even if by changing the input variables. Following extensive research (or trials) it was found that there were no simple/acceptable solutions to this problem. There was no good acceptable solution to this problem. In order for the assignable LED to work, the object was created outside of the object. As a result the program was restricted to having a specified amount of ring lights prior to the final release to the user. The guide that helped me understand the use of the addressable LEDs was a guide on arduino learning that showed how the libary functioned [6].

```
class AssignLEDprofile {
  public:
  String trigger;
  int shr;
  int smin;
  int ehr;
  int emin;
  int degC;
  bool below = true;

  String id;
  int lengPos = 0;
  int count = 0;
  int duration;
  int pin;
  int leng;
  int index;
  int *red;
  int *green;
  int *blue;
  bool active = true;
  void arrayInitalise(int l){
    leng = l;
    red = new int[leng];
    green = new int[leng];
    blue = new int[leng];
  }
};
```

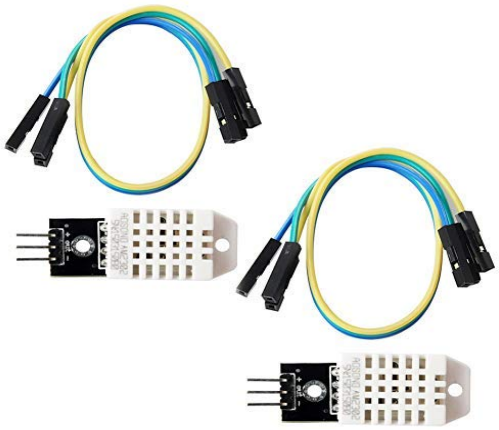
4.3 Implementing Triggers (Sprint Three)

The next implementation stage in the project were the triggers. The triggers allowed the activation of the LED profiles. Even though this stage was less problematic than other sections of the project, there were some realisations when it came to how the trigger could be activated with the button and the weather sensor.

4.3.1 Removing the button trigger

During this stage it was concluded that the button trigger was not required. One of the negative features of the ESP32 was that there are limitations on the number of pins that can be connected to it. The use of a pin to detect when there was a signal coming from the

ESP32 for a trigger seemed very complicated when there was an easier alternative that did not take up an extra pin. When wiring up the LED to the ESP32, a button was added, and the trigger set to constant. The outcome of this was that when the button was pressed the LED is played the profile. However, when the button was pressed, the LED profile did not start at the beginning. It started wherever it was at in the brightness/colour array as the main lighting loop was displaying the colour/brightness to the LEDs even when the button was set to off.



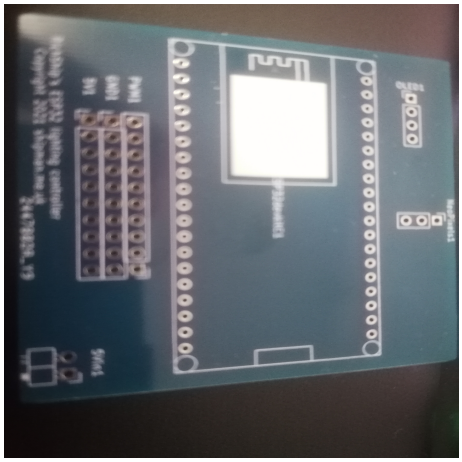
(a) The tempurature sensor used

(b) The sensor used for trigger

Figure 4.1: The two peices of hardware used for the customisable tempurature trigger and the sensor trigger

4.3.2 Getting and implementing the circuit boards

During this week, the circuit boards that were supplied by my supervisor arrived. This did not take too long to implement and reduced the number of wires connected to the board. This made working on the board a lot easier.



4.4 Website Creation (Sprint Four)

Before the beginning of the project, the complexity of HTML and CSS was underestimated on my part. Initially, the plan was to use file management within the ESP32 to have a document for the code, a document for the CSS style sheet and a document for the HTML sheet. After struggling with the concept of communication between the HTML webpage and the code for a couple of days which delayed the production of other parts of the code, it was decided that the HTML would be coded in the C++ file which is not as efficient as HTML but does work effectively enough for this design.

4.4.1 Website Simplification

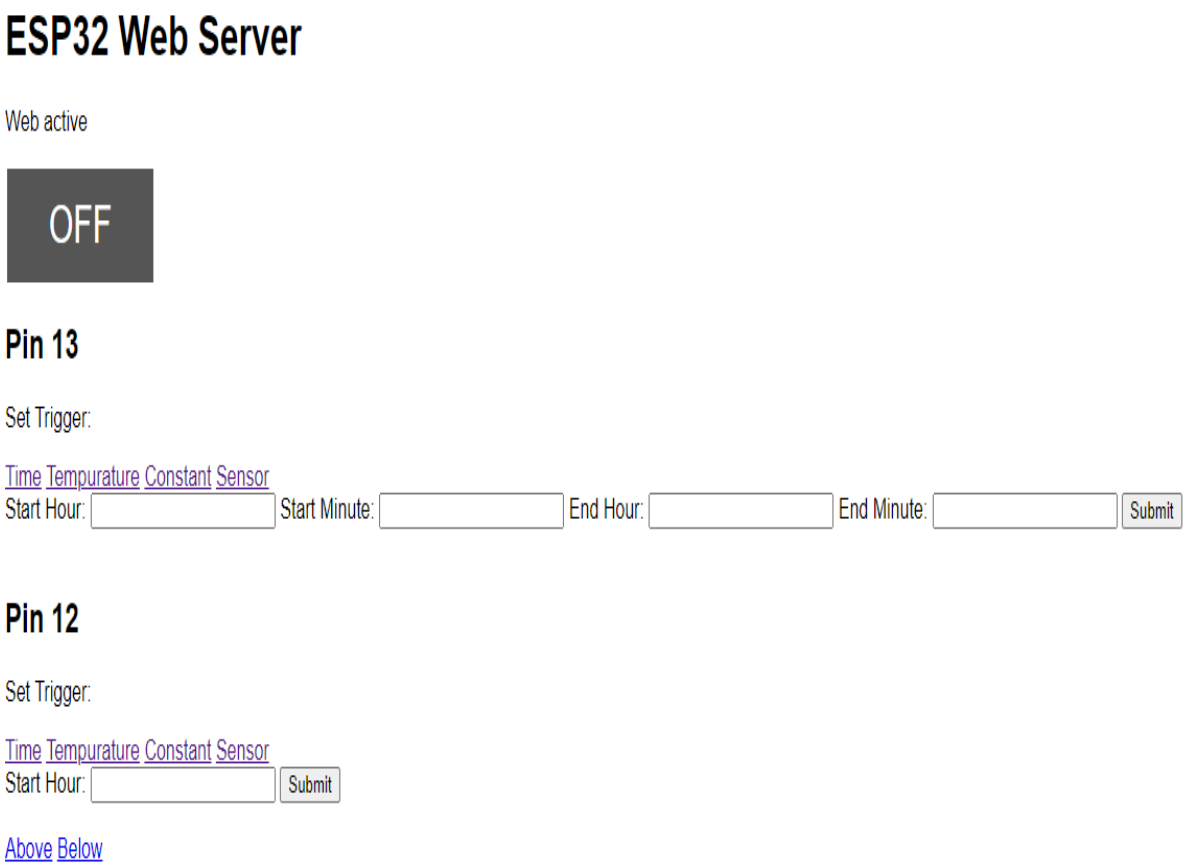
Due to bad planning and not keeping to the Scrum methodology as the project went on, there was not enough time left on the project to fully complete the web side of the project. Therefore, there was some sacrifices that had to be made to the functionality and the design of the website to allow for a semblance of a completed website.

4.4.1.1 Functionality changes

When the fourth sprint started, it was realised pretty quickly that the time scale left for the project did not allow for the time needed to implement all of the must have features identified in the MoSCoW analysis. So in order for a functioning website to be made, the functionality that was excluded from the website was the creation of the profiles. This was an undesirable but necessary step this far into the project with the time scale left in the project. Now, to create a profile the user must code it in the setup of the code.

4.4.1.2 Design chagnes

When it came to the implementation of being able to change the triggers of the LED profiles on the website, there was some difficulties getting the dropdown boxes communicating what was selected to the main code. It was realised that you are unable to add ‘¡a href¿’ statements to the dropdown box implemented and to do it this way, a lot of CSS styling and HTML defining of drop-down boxes had to be made. As the time left before the project deadline was short and functionality had already been cut from the website, it was decided that the functionality of the website was more important than the look of the website. So instead of using drop-down boxes, simple links were implemented instead as a bad looking but functional website is better than a good looking one that does nothing.


ESP32 Web Server
Web active
OFF
Pin 13
Set Trigger:
Time Tempurature Constant Sensor
Start Hour: Start Minute: End Hour: End Minute: Submit
Pin 12
Set Trigger:
Time Tempurature Constant Sensor
Start Hour: Submit
Above Below

Chapter 5

Testing

This testing section is used to identify how the testing phase of the project was performed.

5.1 Overall Approach to Testing

The approach I used to testing was to use test tables to test if the program met the requirements that was created at the beginning of the project. Each test case will have:

- Test ID - An identifier to refer back to if needed. The ID will be displayed as TCx, with TC meaning test case and x being replaced with the number the test is e.g. TC1.
- Reference ID - The ID of the requirement the test is testing to if complete.
- Process - What is the process that is performed to test the requirement. If there is more than one Step to the process they are enumerated.
- Expected Result - What would have to happen for the test case to be considered a pass.
- Pass/Fail - Whether the test passed or failed.

The test will be split up into two different tables. One of the test tables refers to all of the tests for the code side of the program and the other is tests for the web side of the program. I have only performed tests for all of the testable requirements. I have only performed test cases for all of the implemented requirements with the MoSCoW priority of M (must-have), S (should-have) and C (Could-have).

5.2 Code Side Test Cases

The table that follows is the Test cases for the code side of the project.

Test ID	Reference ID	Process	Expected Result	Pass/Fail
TC1	RQ1	TC2 and TC3 given Pass grade	The output changes depending on the changes made by the user	Pass
TC2	RQ2	1.Create a standard LED profile and attach to an unassigned LED 2. Create an assignable LED profile and attach to an unassigned colour LED	The respective LEDs display the correct outputs depending on what was changed	Pass
TC3	RQ3	The implemented triggers are given a pass grade	All of the triggers pass except for weather as it has not been properly implemented	Pass
TC4	RQ3.1	Set the trigger to be Time with appropriate parameters	Time trigger displays profile when trigger is true	Pass
TC5	RQ3.2	Set trigger to be Weather	Not Implemented. Will Fail.	Fail
TC6	RQ3.3	Set trigger to be Temperature with the appropriate values	When the temperature read by the sensor passes over the threshold, the profile is displayed	Pass
TC7	RQ3.5	Set trigger to be Sensor	When motion passed in-front of the sensor, profile is displayed	Pass
TC8	RQ7	Set up the ESP32 with minimal effort	The program isn't as easy to set up as I would have liked to as to set up the website you have to edit the code with your WIFI credentials.	Fail

Table 5.1: Test case table for the code side of the tests

5.3 Test Cases For Website

The table that follows is the test cases for the testable requirements of the website

Test ID	Reference ID	Process	Expected Result	Pass/Fail
TC9	RQ8, RQ9	N/A	The website is simple and clearly laid out .	Pass
TC10	RQ10	Set a different trigger than what has already been set.	The LED now responds to the new trigger when active.	Pass
TC11	RQ11, RQ12	Create a standard LED profile and assign it to a standard LED.	Not Implemented. Will Fail.	Fail
TC12	RQ13, RQ14	Create an assignable LED profile and assign it to an assignable LED.	Not Implemented. Will Fail	Fail

Table 5.2: Test case table for the web side of the tests

Chapter 6

Evaluation

In this chapter of the document, I have reflected upon the initial decisions made, design plans and the overall impact and outcome of the project.

6.1 Planning

As I took the agile methodology approach to producing the program, most of the planning phase was spent understanding what was expected of the final product and doing some spike work and experimenting mainly with hardware that I was going to use. On reflection I feel that this phase of the project is exactly what I needed to understand the scope of the project. In comparison, because most of the hardware that was being used in the project was unfamiliar to me and I have not worked with most of it before, doing a traditional waterfall style planning and design phase would not have been as helpful. This was due to a lack of experience on my part with working some of the aspects of the project. So, when it came to the implementation of the project , I would have known what I wanted the outcome to be, however I would have struggled on implementing the features desired to meet the plans specified in the design brief. This would have led to either a design brief that was not detailed enough or an over enthusiastic design document with requirements that could not have been met.

6.2 Hardware Used

Overall, I have been really impressed with the functionality of the ESP32 and the corresponding hardware (temperature sensor, regular sensor, LEDs and Ring Lights). This is due to the simplicity of how they are used and that ninety percent of the time I did not have any problems with them. From past projects, where I have used other microcontrollers and hardware, most of the time numerous hours were wasted getting the hardware to connect properly and for the code to start reading in the values before you can actually start coding with it. With every piece of hardware used on this project, the setup took at most thirty minutes before I could actually start sending/retrieving data to the different types of

hardware used. I am usure whether this simplicity came from the libraries I used for the different pieces of hardware, the guides I was using [link to guides] or was lucky with good hardware but overall I am very happy with how the hardware operated with my code.

6.3 Requirements

In the planning stage of the project, the MoSCoW analysis that was recommended by my supervisor was very helpful. It helped me identify exactly what was required for the completion of the project, what extra functionalities I could add beyond that to make the project better and what functionalities were unrealistic.

In terms of actually meeting these requirements, I am happy that most of the 'must have' requirements have been met and accomplished. I am content that I have managed to complete some of the 'could have' requirements. However I am disappointed that some of the requirements that I expected the website to be able to do, I have not had time to implement (explained in Following the Methodology and Time Management). Overall I have no discerning opinion on my requirements as I have been able to implement some of the requirements that were considered "extras", while some of the expected requirements were not met on the website. In hindsight I should have stopped working on the extra triggers for the profiles and started focusing on the website, I was enjoying coding with the new hardware I had acquired and as I was already working on the triggers section of the code I thought it would not be too much effort to implement the rest.

6.4 Following the Methodology and Time Management

The Scrum methodology chosen was ideal. It fitted well with how I normally work and when it came to the planning phase of the project, it helped me understand how I was going to complete the objectives I had planned. When it came to implementing the requirements, the daily scrum and the scrum tasks helped keep me working on the first items that were implemented at the start of the project. However, as time crept on I started doing less and less daily scrums and started to slack on the repetitive tasks that kept me on track. This caused daily task to be bundled together into weekly tasks and just doing rough estimations without any proper thought. This ended-up impacting on me significantly near the end of the project as tasks that I thought would take a week took a week and a half, and so did the next task. Therefore, it did not leave me an appropriate amount of time to work on the website, so I did not meet all of the requirements.

In hindsight one of the things that caused this backlog of work near the end of the project was due to daily tasks not being extensive enough. As one of the features of Scrum is to sprint with your task for the day, when I had finished the tasks that I was working on that day's sprint, I considered that day finished and stopped working when ideally I could have started working on the next day's tasks to try and get ahead.

6.5 What I Would Do Differently

If I had to start this project again or had some advice for myself at the beginning of the project. These are the following pieces of advice I would give to myself:

6.5.1 Don't underestimate the things you do not understand

One of the biggest mistakes I made during the planning phase was underestimating the HTML website due to doing some basic HTML a couple years previously. Do not estimate that something is going to be easier or harder depending on a limited previous experience. Two of the biggest misconceptions I had when I started this project was that the HTML would not be that complicated at all and connecting the hardware would be very hard and time consuming. Both of these misconceptions ended up being the opposite of what I thought they were going to be.

6.5.2 Just because something is repetitive and boring does not mean it is not vital

As discussed previously, one of the biggest flaws that happened during the implementation phase of the project was not keeping up to date with my daily scrums and weekly tasks for the Scum methodology. This caused me to be behind on work when I did not realise how far behind I actually was and for me to estimate how long work would take me when I had not made any real estimates.

6.5.3 Balance your time and effort equally

Something that did not take up too much time but is still relevant as I did spend too much time on certain functions trying to perfect them when there was no need. in particular the triggers took up more time than I should have spent on them when I could have relocated that effort into something that was troubling me or that I had not yet started.

6.6 Conclusion

The aim of the project was to create a fully customisable LED controller for the use in dioramas and model scenarios. All things considered I feel that I have acheived this this goal. This is because the user is able to create a fully functioning, flexible lighting controller. The final LED display produced by the lighting controller has numerous output possibilities. The user can easily set the duration of the lighting sequence, how long the colour/brightness is displayed and have mutiple LEDs all displaying at different timescales. The user also has a range of different variable triggers that they can assign to the LEDs so that the profiles will display the lighting sequences when the desired conditions are met. However, the user is not yet able to create LED profiles via the website.

6.6.1 If I Had More Time

If I had more time on the project, the first things I would have to implement would be a proper web front end. To do this I would spend more time on realising how to use SPIFFS for the ESP32 and host the website properly without doing it inline so then I could do the more complicated styling with CSS and HTML to make a good looking final website.

On reflection, if I had additional time on the project to futher enhance the outcome, the biggest functionality I would have liked to implement would be the use of servos, motors and sound. The hardest aspect of this would be implementing sound to the program as it needs extra hardware for it to work (Storage and a speaker) while the servos and motors only require the servos and motors respectively to make them work. However when these functionalities would have been implemented, this would have allowed the user futher possibilities when it came to the final display of the diorama/model.

Annotated Bibliography

[1] M. Miniatures, "Malcolm Miniatures main page," http://www.malcolmsminiatures.co.uk/Lighting+controls, June 2021, accessed April 2021.

The lighting controllers sold byMalcolm Miniatures.

[2] A. Andrews, "Guide on using the agile methodology Scrum with a single person," https://www.raywenderlich.com/585-scrum-of-one-how-to-bring-scrum-into-your-one-person-operation, June 2017, accessed February/March 2021.

A guide written by Alex Andrews on how to apply the agile methodology Scrum when working by yourself.

[3] "Scratch," https://scratch.mit.edu/, 2021, accessed March 2021.

A link to the scratch hompage.

[4] R. N. Tutorials, "Random Nerd Tutorials main page." https://randomnerdtutorials.com/, Mar. 2021, accessed February/March 2021.

A series of guides by Random Nerd Tutorials that helped me with different aspects of the project.

[5] R. Santos and S. Santos, "Guide on using the agile methodology Scrum with a single person," https://randomnerdtutorials.com/esp32-pwm-arduino-ide/, 2020, accessed March/April 2021.

A Guide on using the PWM pins on the ESP32 to display different brightnesses to the LEDS.

[6] A. Learning, "WS2812 RGB LED Ring example on arduino," http://arduinolearning.com/code/ws2812-rgb-led-ring-example-on-arduino.php, June 2020, accessed March/April 2021.

An example of the use of the WS2812 ring LEDs that helped me understand how they are operated.

Appendices

6.7 Original MoSCoW analysis

Requirements

MoSCoW – Must have (M), should have (S), could have (C) and will not have (W)

RQ1 - ESP 32 must react to the changes made by the user (M)

RQ2 - Display the brightness and the colours of the LEDs correctly (M)

RQ3 - Perform defined output when trigger is met. (M) The triggers are:

RQ3.1 - Time (S)

RQ3.2 - Weather (C)

RQ3.3 - Temperature (C)

RQ3.4 - Button (C)

RQ3.5 - Sensor (C)

RQ4 - Will have 2-3 pre-set standard and addressable profiles for fire, constant and strobe(W)

RQ5 - Play MP3 file when trigger is met (W)

RQ6 - Will have some sort of storage like an SD card to store the MP3 audio (W)

RQ7 - The ESP must be easy to set up (M)

RQ8 - Create a user-friendly website.(M)

RQ9 - Easy to use and navigate (S)

RQ10 - Define the ‘trigger’ for the outputs implemented. (S)

RQ11 - Create LED standard profiles (Standard LED profiles are the brightness that must be displayed to the user e.g. PWM) (M)

RQ12 - Set the standard LED profiles to the LEDs (M)

RQ13 - Create Assignable LED profiles (Same as the standard profiles however you can set the colour of the LED) (M)

RQ14 - Set the assignable LED profiles to the assignable LEDs (needs specific assignable LEDs) (M)

RQ15 - Define the intended motor and servo config (C)

RQ16 - Upload small MP3 file to play when defined by trigger (W)

RQ117 - Will also have a profile for the servos and motors outputs (C)

Appendix A

Third-Party Code and Libraries

Adafruit NeoPixel - The adafruit neopixel libary is used tocommunicate withe the addressable LEDs. They send an RGB colour value to the LED and it displays the appropriate colour. This libary was used without modification.

WIFI - This is the libary used to host and display the website to the client. This libary was used without modification.

DHT - This is the libary used to recive values releting to the tempurture and mosture of the surrpounding environment. This libary was used without modification.

Appendix B

Ethics Submission

Ref Number 18787

AU Status Undergraduate or PG Taught

Your aber.ac.uk email address ktt1@aber.ac.uk

Full Name Kieran Todd

Please enter the name of the person responsible for reviewing your assessment. Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment rrz@aber.ac.uk

Supervisor or Institute Director of Research Department cs

Module code (Only enter if you have been asked to do so) CS39440

Proposed Study Title ESP32 Lighting Controller

Proposed Start Date 25/01/2021

Proposed Completion Date 1/06/2021

Are you conducting a quantitative or qualitative research project? Mixed Methods

Does your research require external ethical approval under the Health Research Authority? No

Does your research involve animals? No

Does your research involve human participants? No

Are you completing this form for your own research? Yes

Does your research involve human participants? No

Institute IMPACS

Please provide a brief summary of your project (150 word max) The ESP32 lighting project

is 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 for the use in models (e.g. model trains, book nooks, etc.). 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.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material? Not applicable