Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Summer 2021

The Lighters
LightHouse

Dipika Giri
Dhruv Patel
Parker Skinner
David Trimino

Revision History

Revision	Date	Author(s)	Description
0.1	8.7.2021	DG,DP,PS,DT	document creation
0.2	8.16.2021	DG,DP,PS,DT	First draft complete
0.3	12.07.2021	DG,DP,PS,DT	Final draft complete

Contents

List of Figures

List of Tables

1 INTRODUCTION

The overall product concept is a web application acts like a controller which has multitude of functionality. The main functionality that the system will need is to take in the user's customization preference and display them with the LED lights. The system should also store the user's customization bundles in a cloud repository for others to have and share.

2 System Overview

The system will essentially consist of three layers: an application layer, middle-ware, and the hardware layer. The two layers Application and Middle-Ware are separate layers as both accomplish and handle different aspects although all taking place in the pi itself. The user layerâs function is to deal with the presentational and interactional functions in regard to a Dispatcher, Store, Action, and View. The user layer will be using a flux architecture. The middle-ware with input validation of the user, and to process the JSON objects and translate them into readable code for the LED lights. The validation step is essential to ensure the information being sent to the LED lights is valid. The hardware layer is responsible for displaying the LED lights to the user's preferences.

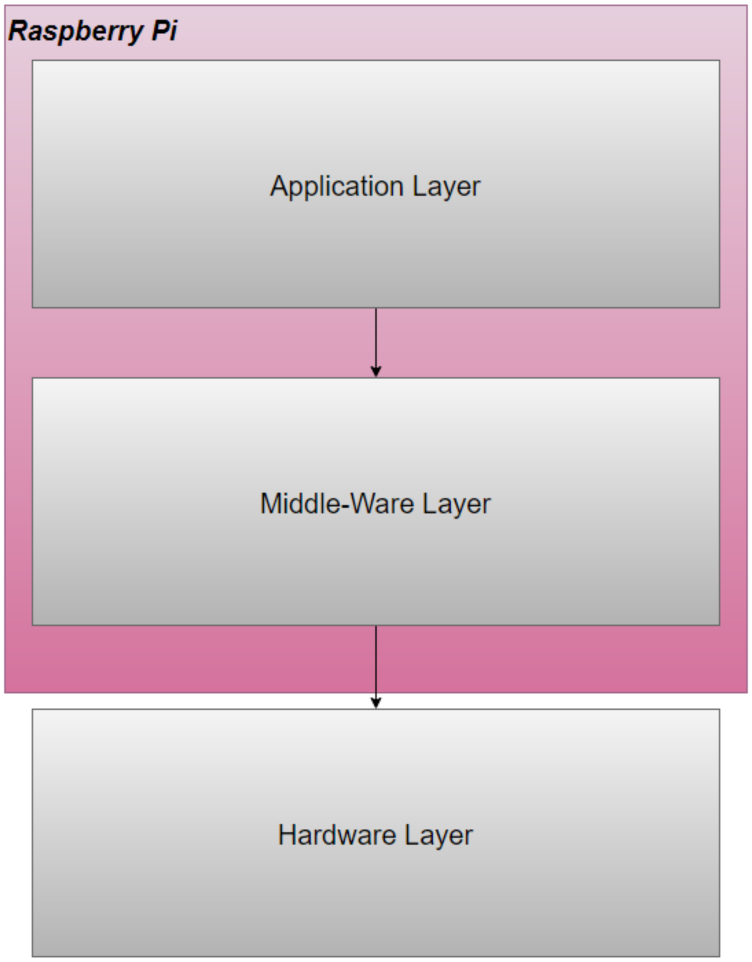

Figure 1: A Simple Architectural Layer Diagram

2.1 Raspberry Pi Layer Description

The raspberry pi layer is the user interface for the user using the LightHouse LED Web Application and the background processes that deliver the displays that the user requested. This layer consists of 4 subsections, action, dispatcher, store, view, along with any constants that will be used. Overall this will essentially be the application layer consists of two sub layers: user and back. The user layer is responsible for what the user sees and interacts with. The back layer is responsible for changing configuration of LEDs, sending data to LEDs, and handles any other background processes.

2.2 Middle-Ware Layer Description

The middle-ware layer is responsible for receiving data packets, from the web server on the raspberry pi, which contain configuration instructions on how to display LED lights. The middle-ware will run through the GPIO pins on the raspberry pi sending data out and is responsible for giving the LED lights the smart functionality, for them to be able to change colors or patterns.

2.3 Hardware Layer Description

The hardware layer includes the LED lights. The lights receive inputs from the GPIO pins as well as the saved patterns from the pi layer and output the requested color and pattern. This includes the brightness information and also the speed at which to run.

3 Subsystem Definitions & Data Flow

This section breaks down your layer abstraction to another level of detail. Here you graphically represent the logical subsystems that compose each layer and show the interactions/interfaces between those subsystems. A subsystem can be thought of as a programming unit that implements one of the major functions of the layer. It, therefore, has data elements that serve as source/sinks for other subsystems. The logical data elements that flow between subsystems need to be explicitly defined at this point, beginning with a data flow-like diagram based on the block diagram.

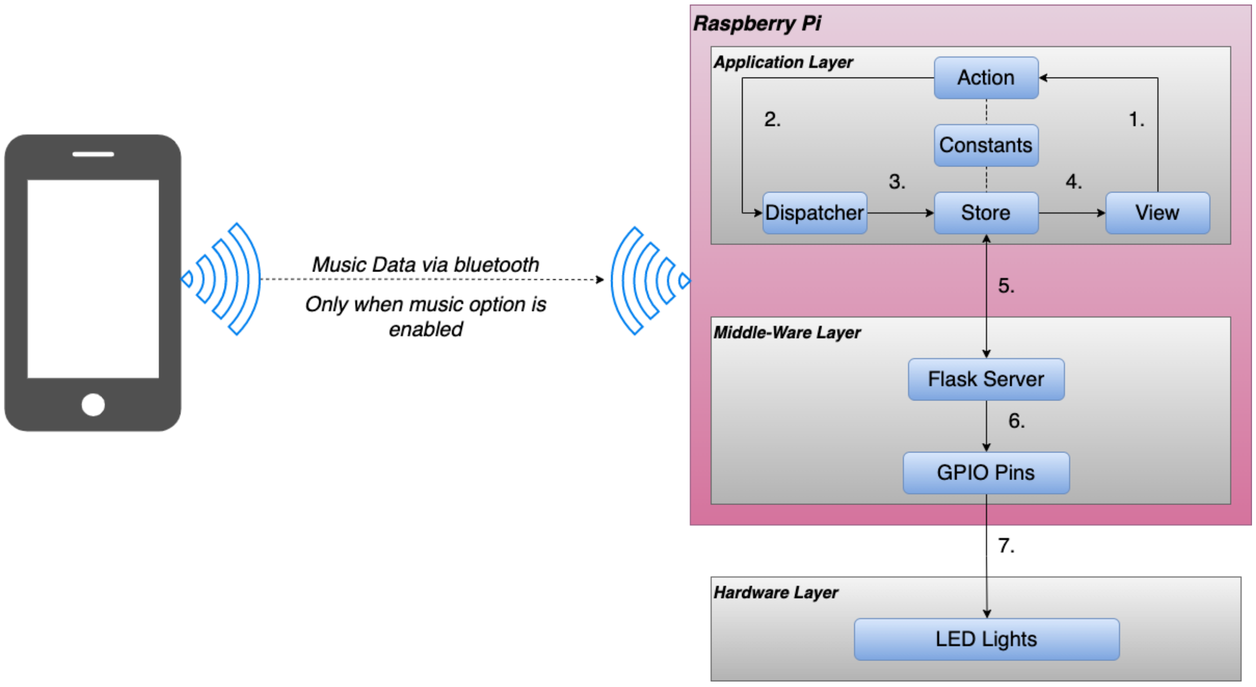

Figure 2: Data flow diagram of system

4 Raspberry Pi Layer Subsystem

The layer consists of 4 subsections, action, dispatcher, store, view, along with any constants that will be used. This layer provides a user interface and communicates with the are layer flask server. The user data is stored in the database and when the patterns are determined, the layer will communicate with flask server. This layer should apply user data and create light patterns in the LED strips.

4.1 Action

The action subsystem layer is the interaction with the user. The goal is to get user data needed to run the simulation on the Raspberry Pi and create various light pattern. The action will be selected and sent to the dispatcher. The helper methods collected which create an action from parameters and assign it to a type and provide to a dispatcher.

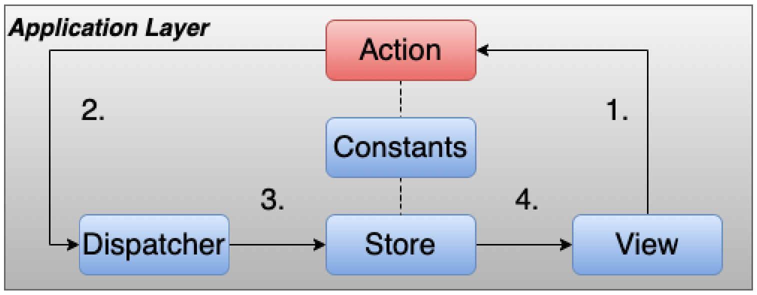

Figure 3: Raspberry Pi Subsystem 1

4.1.1 Assumptions

The user will be able to connect the LED strip to the raspberry pi and will be able to run the web application using the raspberry pi. The user will be able to select an effect from various options including color and nine color patterns in the user interface.The light effect data will saved and back-end will use that data.

4.1.2 Responsibilities

The action subsystem layer will include interface to connect the devices, select various colors with or without pattern. The patterns have the ability to either run the previously selected color or have its own color, for example, rainbow pattern contains its own color which includes variety of color whereas color wipe pattern has either previously chosen color or random color if color is not chosen previously . The responsibility of this layer will include user data and saving the data to use in the back end.

4.1.3 Subsystem Interfaces

Table 2: Action Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Selecting Color and Pattern	Web Page	Send to dispatcher

4.2 Dispatcher

This dispatcher layer allows is to trigger a dispatch to the store and include a payload of data which is the action. The data includes hex form of color and the pattern chosen from the user interface.

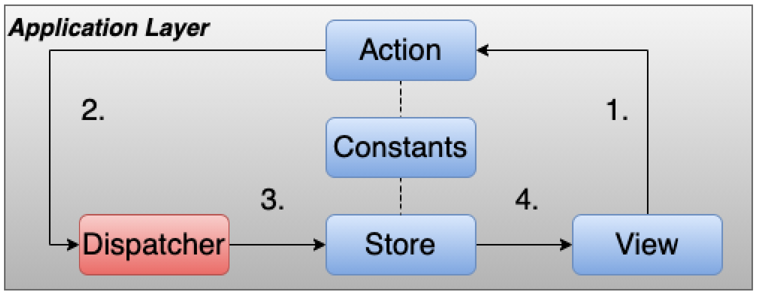

Figure 4: Raspberry Pi Subsystem 2

4.2.1 Assumptions

The layer dispatches the data from action layer. The data will be used in store layer.

4.2.2 Responsibilities

This layer is responsible for dispatching actions to the store subsystem. Every key is analyzed and managed. Any changes to the data should be managed using the action subsystem layer.

4.2.3 Subsystem Interfaces

Table 3: Action Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Sending the Action to Store	Action	Send to store

#1 Store user data user data data table #1 Manage user data user data updated data table

4.3 Store

The store subsystem layer contain application state and logic. Stores manage application state for a particular domain within your application. From a high level, this basically means that per app section, stores manage the data, data retrieval methods and dispatcher callbacks.

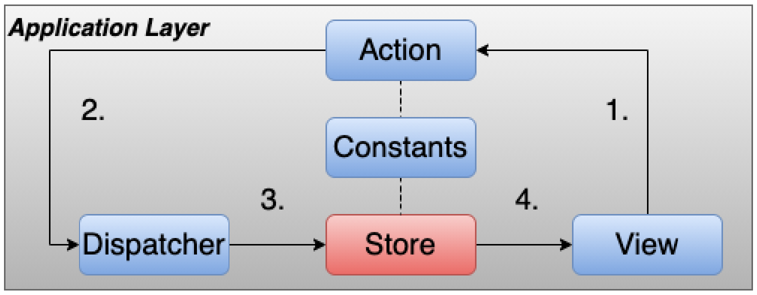

Figure 5: Raspberry Pi Subsystem 3

4.3.1 Assumptions

We assume the layer has the proper data from the dispatcher which is the action data.

4.3.2 Responsibilities

This layer is responsible for analyzing the user data to create the light pattern selected and execute it on the LED strips.

4.3.3 Subsystem Interfaces

Table 4: Store Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Communicate with Flask	Color and Pattern Data	Color and Pattern Data

4.4 View

View subsystem layer will provide various selection of effect a user can select. It is the extended version of the web application subsystem layer including hex and patterns which can be selected.

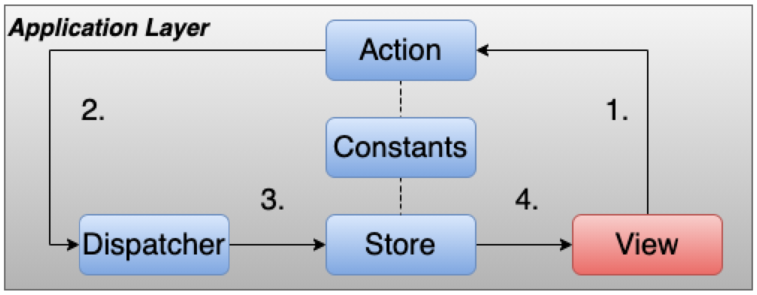

Figure 6: Raspberry Pi Subsystem 4

4.4.1 Assumptions

The layer provides displays the hex and pattern information to the user. The layer includes a user interface with selection of light color and pattern. The layer interacts with the action and store.

4.4.2 Responsibilities

The layer properly displays the color and patterns for the user to select at all times it is needed.

4.4.3 Subsystem Interfaces

Table 5: View Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Displaying the color and pattern	Display	Send to store or action

5 Middle-Ware Layer Subsystems

The Middle-Ware Layer is the flask server and the GPIO Pins that will directly interface with the led lights and send instructions in regard to color and pattern information to change the LED states.

5.1 Flask Server

The Raspberry Pi receives JSON objects from the front end web server and is parsed by the Flask server to extrapolate color and pattern data.

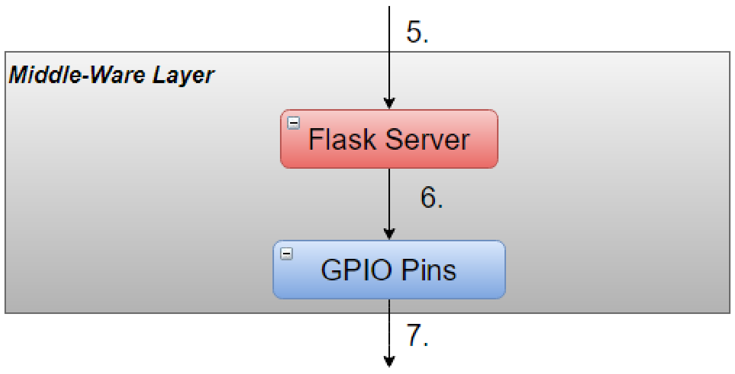

Figure 7: Middle-Ware Subsystem 1

5.1.1 Assumptions

We are assuming that all packets sent to the back end will be from the front end only and nothing else will try to interface with it.

5.1.2 Responsibilities

This subsystem has the responsibility of receiving packets and parsing the data contained in those packets, such as the color and pattern information to be displayed onto the lights.

5.1.3 Subsystem Interfaces

Input is the JSON package and the overall output will be parsed data.

Table 6: Flask Subsystem interfaces

ID	Description	Inputs	Outputs
#5	Incoming Network Packets	JSON Package	N/A
#6	Parsed Network Packet	N/A	Parsed Data

5.2 GPIO Pins

The output will read the parsed data received by the receiver and send commands to the LED Strips based off of that data through the GPIO Pins.

5.2.1 Assumptions

We are assuming here that the data coming from the receiver is already parsed and is in the correct format to be sent to the LED lights and displayed.

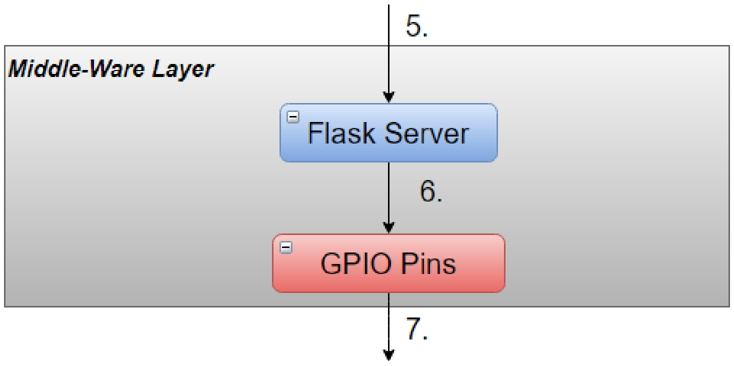

Figure 8: Middle-Ware Subsystem 2

5.2.2 Responsibilities

The Output must send the desired commands to the LED strips to properly change the color, brightness, or any other wanted state of the lights.

5.2.3 Subsystem Interfaces

The input is the parsed data regarding mainly the color and pattern information and the output is to the LED lights to display the color and pattern.

Table 7: GPIO Pins Subsystem interfaces

ID	Description	Inputs	Outputs
#7	Incoming Parsed Data	Instruction Data	N/A
#9	Outgoing LED Commands	N/A	LED Commands

6 Hardware Layer Subsystems

In this hardware layer we have two subsystems including the power supply and the WS281B LED Lights. This layer is solely responsible for the LED lights and the lighting of them. The arrow shown, number 7, comes from the middle-ware layers GPIO Pins subsystem with commands on how the LEDs should be lit up.

6.1 LED Lights

The WS281B subsystem simply contains the LED lights that the whole product is based upon. The lights receive power from the power supply subsystem and inputs on how to light up from the raspberry pi.

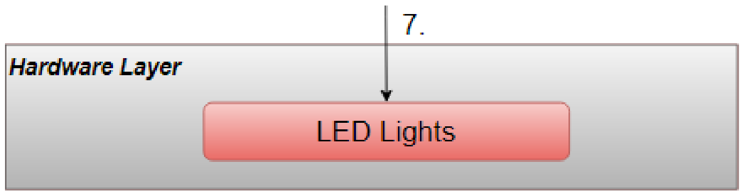

Figure 9: Hardware Subsystem 1

6.1.1 Assumptions

An ample amount of consistent power is provided at all times to the lights ensuring that they are lit at all times consistently and accurately as well as they will have a command sent from the raspberry pi and middle-ware layers GPIO pins on the pattern they are supposed to display.

6.1.2 Responsibilities

The responsibility of this subsystem is to carry out the pattern that the user would like, such as a specific pre-saved pattern from the raspberry pi layer. The LED lights should display the lights with the same intensity throughout the usage or at the brightness level desired.

6.1.3 Subsystem Interfaces

The inputs are from the power supply and from the GPIO Pins subsystem from the middle-ware layer. The outputs for all are the display of the LED lights that are supposed to run.

Table 8: LED Lights Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Power	Power Supply	LED Lights Display
#2	GPIO Pins	Saved patterns	LED Lights Display

References

Gieser, Shawn. "Computer System Design Project I", University of Texas at Arlington.