

# ADAPTIVE NIGHT LIGHT

## *Final Research Paper*

November 21, 2020

### GROUP JAMLER

Authors \_\_\_\_\_

*Trevor Byler* | [tdbyler0@frostburg.edu](mailto:tdbyler0@frostburg.edu)

*James Ritchie* | [jpritchie0@frostburg.edu](mailto:jpritchie0@frostburg.edu)

### COSC 365-001

Professor \_\_\_\_\_

*Dr. Joy Zheng* | Frostburg State University



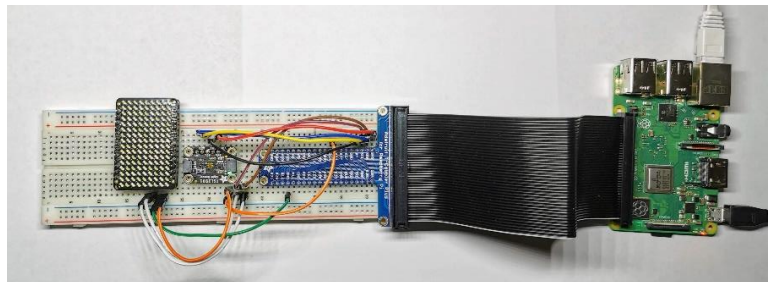
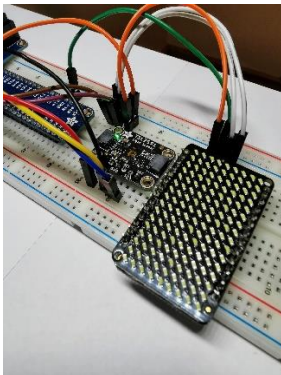
## TABLE OF CONTENTS

Table of Contents .....	2
I. Introduction .....	3
II. Hardware.....	4
A. Hardware List .....	4
B. Wiring .....	4
III. Literature Review .....	5
A. General Sensor Network.....	5
B. Double Light Sensor .....	5
C. Light Sensor in Agriculture .....	6
D. Comparison of Light Sensors .....	6
III. Software & Communication.....	7
IV. Result.....	8
V. Issues Faced .....	9
VI. Work Distribution .....	9
A. Trevor Byler .....	9
B. James Ritchie .....	10
VII. Source Code & Screen Shots .....	11
References .....	13

## I. INTRODUCTION

The Raspberry Pi is a portable computer that is inexpensive, compact, and customizable. It can function as a normal computer. It can be loaded with a full operating system with a GUI and downloadable programs. It is compatible with standard peripherals, such as keyboard, mouse, and computer monitors. The new model of the Raspberry Pi: Raspberry Pi 4, has up to 8 GB of SDRAM, a quad-core processor, ethernet, Wi-Fi, and Bluetooth functionality, and a variety of other ports. Raspberry Pi can be hooked up to a plethora of sensors as well. Sensors can read basic user input, environmental properties, incoming wireless signals, and biometrics. These sensors, the Python programming support built into Raspberry, its low cost, and small form-factor make it a great tool for developing and testing sensor networks and embedded systems.

For phase one of our project, we chose to combine a Raspberry Pi (3B+) with a sensor (Adafruit TSL2591 High Dynamic Range Digital Light Sensor) and light output (LED Charlieplexed Matrix - 9x16 LEDs in conjunction with a IS31FL3731 16x9 Charlieplexed PWM LED Driver) to construct a night light. We wanted our output light to turn on once the amount of visible light being picked up by the sensor dipped below a set threshold. We also wanted it to work in the opposite way so that the light would then turn off if the amount of visible light was greater than that of our set threshold. Phase two of our project was then to create a webpage for the project. The purpose of the webpage is to display an overview of the project, real-time data from the Raspberry Pi sensor and output, and include all our documentation.



## II. HARDWARE

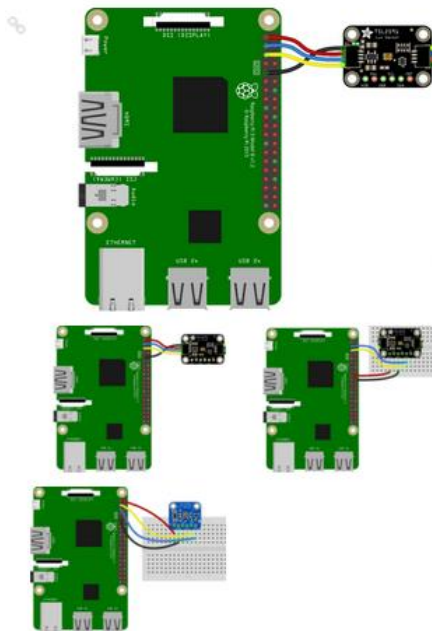
### A. Hardware List

- Raspberry Pi 3 B+
- 5V 2.5A Switching Power Supply with 20AWG Micro USB Cable
- Full sized breadboard
- Premium Male/Male Jumper Wires - 20 x 3" (75mm)
- SD/MicroSD Memory Card - 16GB Class 10 - Adapter Included
- Adafruit 2028 Assembled Pi T-Cobbler Plus - GPIO Breakout
- Adafruit TSL2591 High Dynamic Range Digital Light Sensor
- IS31FL3731 16x9 Charlieplexed PWM LED Driver
- LED Charlieplexed Matrix - 9x16 LEDs - Cool White

### B. Wiring

Adafruit TSL2591 High Dynamic Range Digital Light Sensor:

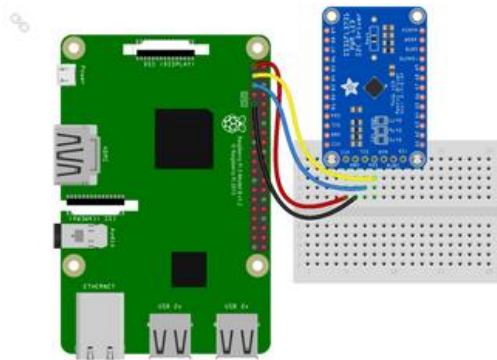
Here's the Raspberry Pi wired with I2C:



- **Pi 3V3 to sensor VIN (red wire on STEMMA QT version)**
- **Pi GND to sensor GND (black wire on STEMMA QT version)**
- **Pi SCL to sensor SCK (yellow wire on STEMMA QT version)**
- **Pi SDA to sensor SDA (blue wire on STEMMA QT version)**

IS31FL3731 16x9 Charlieplexed PWM LED Driver:

Here's the Raspberry Pi wired to the breakout with I2C:



- **Pi 3V3 to sensor VIN**
- **Pi GND to sensor GND**
- **Pi SCL to sensor SCL**
- **Pi SDA to sensor SDA**

[1]

### III. LITERATURE REVIEW

#### *A. General Sensor Network*

Much research has been done in the application of Raspberry Pi's, especially when connected in a network of devices. Researchers at the University of North Texas created a sensor network built with Raspberry Pi and Arduino. They designed and built a wireless network consisting of 3 sensor nodes, 3 router nodes, and a base station. Nodes communicated with an XBee radio receiver. The Raspberry Pi and Arduino acted as microcontrollers that managed the temperature and humidity sensor connected to the node. The engineers had to write applications for the sensor nodes, a database using MySQL, and a web server application for the base station. [1]

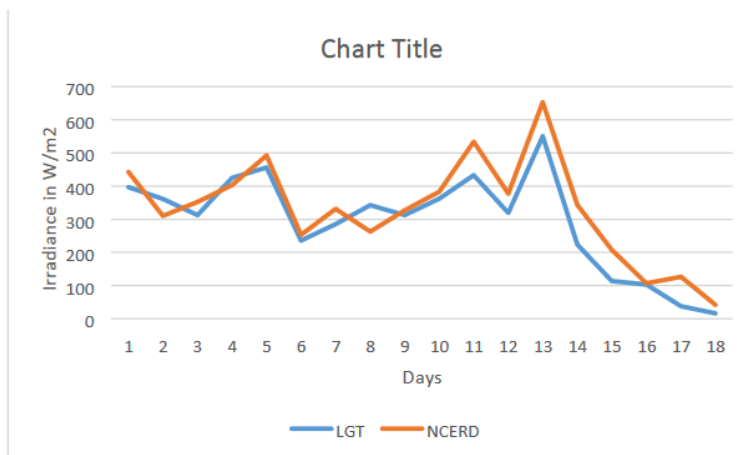
#### *B. Double Light Sensor*

In researching other projects utilizing the same light sensor we are, we found three. The first study being from Openly Published Environmental Sensing (OPeNS) Lab at Oregon State University just briefly touched on the limitations they ran into with using the sensor we chose (TSL2591). Their team was doing environmental data gathering and for the previous design the TSL2591 was sufficient. However, when they created a new design, they wanted to add another light sensor pointing downward to capture albedo from the ground. Because this required two separate light sensors, they had to opt for a different model due to the

addressing constraints of the TSL2591. The TSL2591 only has one address which made it impossible wiring two sensors with two different inputs to a single board for their project. As our project does not require two sensors, we should not run into any addressing issues like this group did. [2]

### *C. Light Sensor in Agriculture*

The second research paper we found was from the Department of Electronic Engineering, University of Nigeria, Nsukka. They were performing an experiment on automation of data collection related to crops drying and needed to use the TSL2591 to log the data regarding light. They chose the TSL2591 because it consists of two photodiodes, one optimized to sense radiation in the visible region and the other radiation in the infrared region. They were able to get their sensor to communicate with their microcontroller using the I2C protocol which we touch on in our communication section. And like we hope to, they were able to accomplish their measuring with libraries provided by the manufacturers. They also compared their data to that of their university's weather station (graph below).



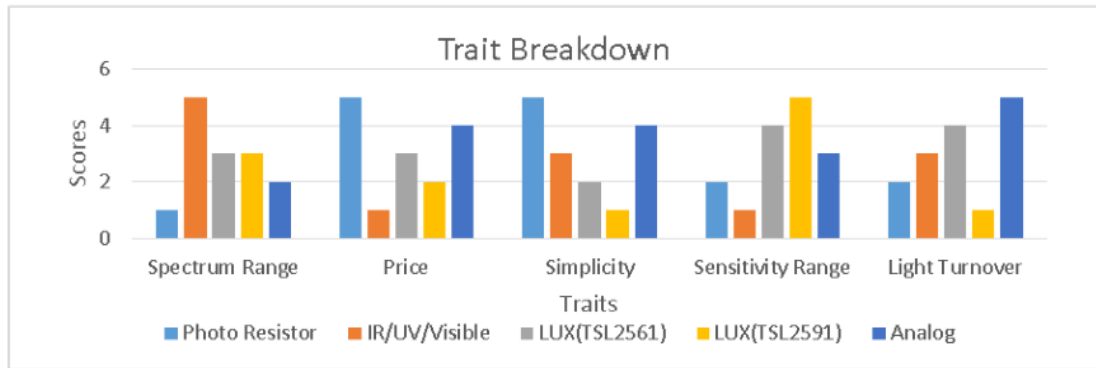
**Figure 8: Graph of solarimeter reading and weather station reading**

[3]

### *D. Comparison of Light Sensors*

Lastly, we read a report from Winona State University in Winona, MN comparing different light sensors to see how they compared to one another. Their group examined 5 different sensors: The Photo resistor, UV/IR/Visible Light (SI1145), High Dynamic Range LUX(TSL2591), Digital Luminosity LUX(TSL2561), and Analog Light (GA1A1S202WP).

Their metrics used for determining the quality of each sensor was: spectrum range, cheapness, sensitivity range, light turnover, command read accuracy, and simplicity of sensor implementation. Below is a trait break down graph of their teams finding based on those criteria:



[4]

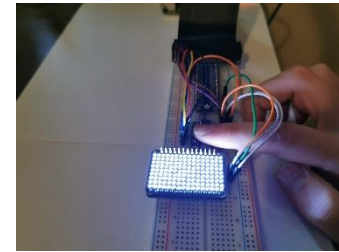
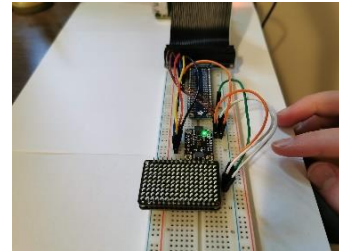
### III. SOFTWARE & COMMUNICATION

Raspberry Pi has support for many Linux-based operating systems. Each flavor of Linux has different strengths and applications. The standard OS officially recommended by Raspberry Pi is Raspberry Pi OS, or Raspbian for short. Raspbian is based on Debian Linux, hence its name. It offers a standard OS experience with a GUI and development tools. It is open source and widely supported among users of Raspberry Pi. Due to the reasons mentioned above, this is the reason our team chose Raspbian as the operating system for our Raspberry Pi.

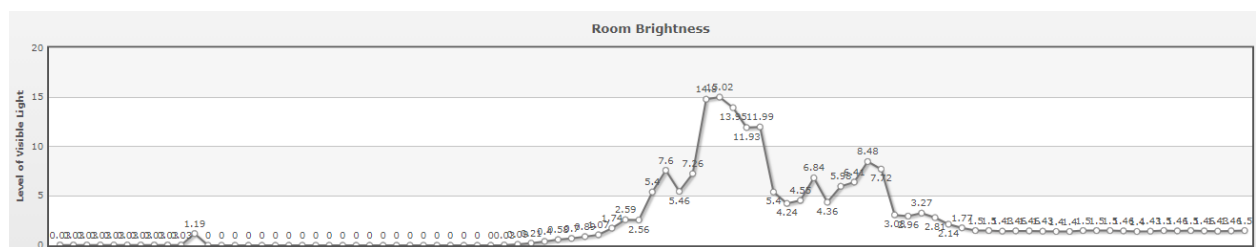
The sensor we used is the Adafruit TSL2591 High Dynamic Range Digital Light Sensor. In addition, we also used output we received from this sensor to then control a LED Charlieplexed Matrix - 9x16 LED display with the help of a IS31FL3731 16x9 Charlieplexed PWM LED Driver, both also offered by Adafruit. Both the light sensor and the driver for the LED matrix communicate using I2C (I-squared-c) and are programmable via Python with the use of additional libraries. For the light sensor we needed to use CircuitPython along with the Adafruit CircuitPython TSL2591 module. As for the LED driver we needed to install the Adafruit CircuitPython IS31FL3731 library on to our board. After necessary setup was complete all the libraries provided comprehensive ways to attain the data from our sensor in a way that we could then manipulate our LED grid.

## IV. RESULT

When starting this project our goal was to have a Raspberry Pi that could collect data from a sensor and then use that data for a separate output and we succeeded. We were able to wire a TSL2591 High Dynamic Range Digital Light Sensor to the Raspberry Pi to analyze the amount of visible light in a room. We were then able to take that data from the light sensor and use it to manipulate an LED grid based on the amount of visible light that was detected by our sensor. All of this was accomplished by writing Python scripts using Python libraries provided from the manufacturer (Adafruit). The provided photos demonstrate the response of the LEDs when the light sensor is off when the sensor is exposed to light, and they turn on when the ambient light with a fingertip.



For phase two of our project we successfully created a webpage to reflect what we had accomplished with our project. The webpage includes a basic overview of the project as well as all the pertinent documentation. Also, on the webpage we have a page to display real time data from our Raspberry Pi. This was accomplished by installing an Apache web server onto the Raspberry Pi as well as a MySQL database. A Python script is used to write the data from the sensor into the database and then that data is presented in the form of a graph on our “Real-Time Data” section of our webpage.





## V. ISSUES FACED

The first issue our group faced was a mistake we made when soldering the pins for the LED Driver. We had soldered the pins the wrong direction on the driver which needed to be fixed. Our solution was to attempt to de-soldering the pins and redo the soldering the correct way. We initially attempted to do this in the classroom, but eventually Trevor had to take the kit home to use more appropriate tools to remove the pins. After removed he was able to correctly solder the pins and get the sensors connected to the Pi.

Another general issue our group faced, as well as many others probably did, was collaboration issues due to the outside issues that arose this semester. With the limitations the pandemic brought on it made in person collaborating difficult during the end of the semester. Because of this, after the initial set up and finishing phase one of the project it pretty much made it so only one member could handle the hardware for our project. We did our best to overcome this by staying in constant contact, like we had all semester, and each just working on what each of us were able to.

The main technical roadblock was dealing with the AJAX request to the PHP script. A policy called CORS was blocking the request to the PHP file. This is a security measure that is put in place by most browsers and web servers. There are procedures to deal with this that involve authentication keys. For our purposes, we could just update the Apache configuration file with a header that would allow all traffic to access the PHP file. This is not recommended if one wants to make the web server publicly available, since it poses security risks. However, this project was for educational purposes, so we will not be hosting a domain for the entire Internet to see.

## VI. WORK DISTRIBUTION

### *A. Trevor Byler*

Trevor wrote several sections of the Phase 1 Research Paper, including the introduction, part of the literature review, the operating system overview, and the conclusion. He handled the soldering of the light sensor and LED matrix. He wired the sensor and light to the output

pins of the Raspberry Pi, and then downloaded and configured the correct python libraries for the LED matrix and the sensor. He developed and tested scripts that ran the phase 1 goal of the project: create an adaptive night light. For phase 2, he installed an apache web server on the raspberry pi, as well as a MySQL database. He also wrote a python script that stores the light sensor info in the MySQL database. With some HTML pages from James, he created a basic web server that would access the data from the database using PHP and JavaScript and graph it on the webpage.

### *B. James Ritchie*

James ordered all the necessary hardware and materials for the project. He also handled the installation of the Raspbian Operating System on the Raspberry Pi. He wrote various sections of the Phase 1 Research Paper including the section on memory, communication, and subsections of the literature review. He also handled writing the weekly reports due for the project throughout the semester. James also provided the HTML and CSS coding for the webpage except for the real-time data page which Trevor handled.

## VII. SOURCE CODE & SCREEN SHOTS

Full source code can be found here: <https://github.com/TrevorByler/LiveSensor>

```
# normalizes output to approximately the percentage of max light
n_factor = 10/2147483647.0

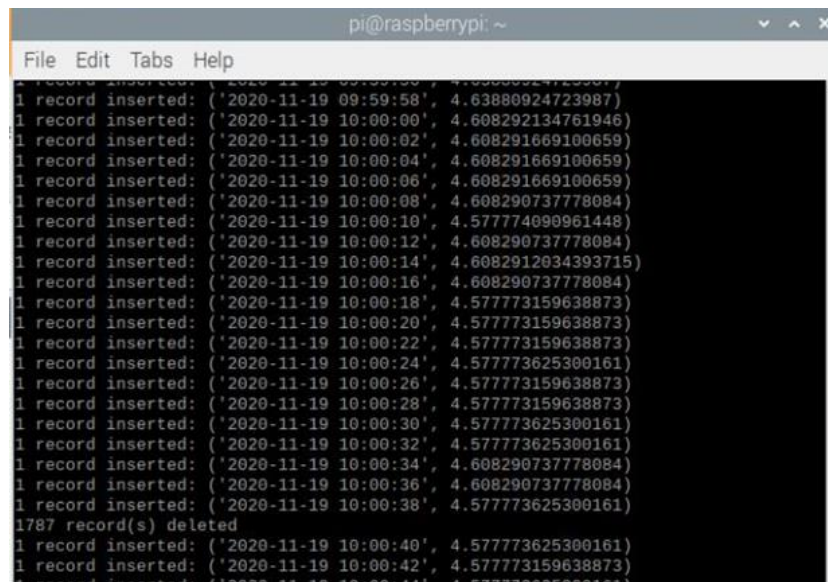
last_del = time.time()

while True:
    visible = sensor.visible*n_factor*100
    timestamp = time.strftime('%Y-%m-%d %H:%M:%S')

    sql = """INSERT INTO raw_data (time, visible_light)
            VALUES (%s, %s)"""
    vals = (timestamp, visible)
    mycursor.execute(sql, vals)
    mydb.commit()
    print(mycursor.rowcount, "record inserted: {}".format(vals))

    if (time.time() - last_del)/60 > 60:
        sql_delete = "DELETE FROM raw_data WHERE time < CURRENT_TIMESTAMP - INTERVAL 1 DAY"
        mycursor.execute(sql_delete)
        mydb.commit()
        print(mycursor.rowcount, "record(s) deleted")
        last_del = time.time()

    time.sleep(2)
```



```
pi@raspberrypi: ~
File Edit Tabs Help
1 record inserted: ('2020-11-19 09:59:58', 4.63880924723987)
1 record inserted: ('2020-11-19 10:00:00', 4.608292134761946)
1 record inserted: ('2020-11-19 10:00:02', 4.608291669100659)
1 record inserted: ('2020-11-19 10:00:04', 4.608291669100659)
1 record inserted: ('2020-11-19 10:00:06', 4.608291669100659)
1 record inserted: ('2020-11-19 10:00:08', 4.608290737778084)
1 record inserted: ('2020-11-19 10:00:10', 4.577774090961448)
1 record inserted: ('2020-11-19 10:00:12', 4.608290737778084)
1 record inserted: ('2020-11-19 10:00:14', 4.6082912034393715)
1 record inserted: ('2020-11-19 10:00:16', 4.608290737778084)
1 record inserted: ('2020-11-19 10:00:18', 4.577773159638873)
1 record inserted: ('2020-11-19 10:00:20', 4.577773159638873)
1 record inserted: ('2020-11-19 10:00:22', 4.577773159638873)
1 record inserted: ('2020-11-19 10:00:24', 4.577773625300161)
1 record inserted: ('2020-11-19 10:00:26', 4.577773159638873)
1 record inserted: ('2020-11-19 10:00:28', 4.577773159638873)
1 record inserted: ('2020-11-19 10:00:30', 4.577773625300161)
1 record inserted: ('2020-11-19 10:00:32', 4.577773625300161)
1 record inserted: ('2020-11-19 10:00:34', 4.608290737778084)
1 record inserted: ('2020-11-19 10:00:36', 4.608290737778084)
1 record inserted: ('2020-11-19 10:00:38', 4.577773625300161)
1787 record(s) deleted
1 record inserted: ('2020-11-19 10:00:40', 4.577773625300161)
1 record inserted: ('2020-11-19 10:00:42', 4.577773159638873)
```

Python script updating MySQL database 24/7

JavaScript function that sends an ajax request to the PHP script on right

```
<script>
function callPHP() {
    var option = $('input[name="timespan"]:checked').val();
    console.log(option);
    $.ajax({
        type: "POST",
        url: "http://10.0.0.234/chart_data.php",
        dataType: 'json',
        data: {args: option},
        success: function(obj) {

            if(!('error' in obj)){

                var chartData = obj;
                const chartConfig = {
                    type: 'line',
                    renderAt: 'chart-container',
                    width: '100%',
                    height: '400',
                    dataFormat: 'json',
                    dataSource: {
                        // Chart Configuration
                        "chart": {
                            "caption": "Room Brightness",
                            "xAxisName": "Timestamp",
                            "yAxisName": "Level of Visible Light",
                            "theme": "fusion",
                        },
                        // Chart Data
                        "data": chartData
                    }
                }
                var apiChart = new FusionCharts(chartConfig);
                apiChart.render();
            }
        });
    });
}
</script>
```

PHP script will query the database for data from last 5 minutes or 24 hours, and return the data to the HTML page

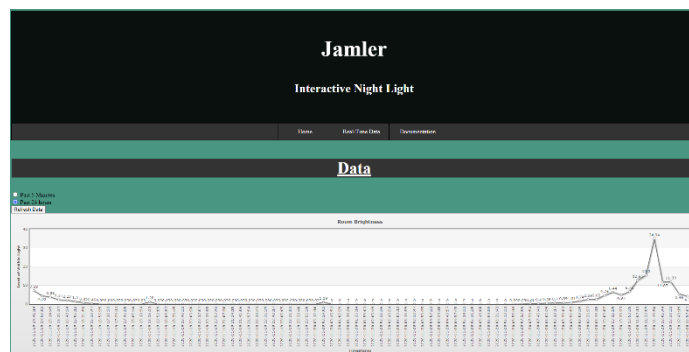
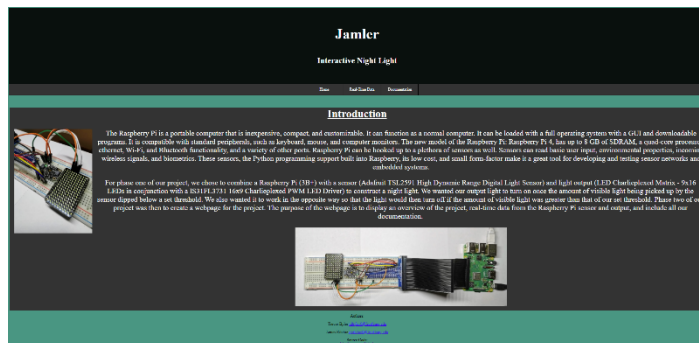
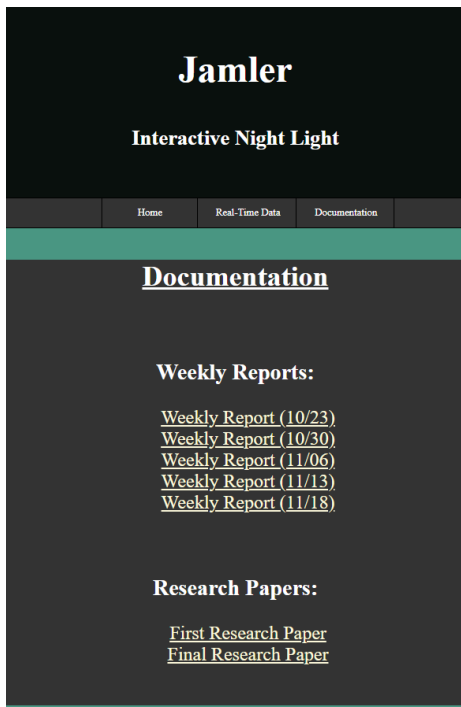
```
<?php
header('Access-Control-Allow-Origin: *');
$options = $_POST['args'];
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "jamler";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if($options=="day") {
    $query = "SELECT * FROM (SELECT @row := @row + 1 AS rownum, " .
        "DATE_FORMAT(time, '%Y-%m-%d %H:%i:%s') AS time, visible_light" .
        " FROM (SELECT @row := 0) r, raw_data) ranked" .
        " WHERE rownum % 500 = 1 AND time > CURRENT_TIME - INTERVAL 1 DAY";
} else {
    $query = "SELECT DATE_FORMAT(time, '%Y-%m-%d %H:%i:%s') AS time, visible_light" .
        " FROM raw_data WHERE time > CURRENT_TIME - INTERVAL 5 MINUTE";
}

$result = $conn->query($query);
if (!$result) {
    trigger_error('Invalid query: ' . $conn->error);
}

//initialize the array to store the processed data
$jsonArray = array();
//check if there is any data returned by the SQL Query
if ($result->num_rows > 0) {
    //Converting the results into an associative array
    while($row = $result->fetch_assoc()) {
        $jsonArrayItem = array();
        $jsonArrayItem['label'] = $row['time'];
        $jsonArrayItem['value'] = $row['visible_light'];
        //append the above created output into the main array.
        array_push($jsonArray, $jsonArrayItem);
    }
}
//Closing the connection to DB
$conn->close();
//set the response content type as JSON
header('Content-type: application/json');
//output the return value of json encode using the echo function.
echo json_encode($jsonArray);
```



The 3 tabs of the website

## REFERENCES

- [1] S. Ferdoush and X. Li, "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications," *Procedia Computer Science*, vol. 34, pp. 103-110, 2014.
- [2] M. N. D. C. U. D. J. S. Thomes DeBell, "Evaporometer Upgrades: Improved Weatherproofing and Accuracy in Teal-Time Enviromental Data Gathering," Corvallis, 2018.
- [3] E. S. U. S. I. F. Edward Anoliefo, "Automation Of Procedures For Experiments On Hybrid Crop Drying," Nsukka, 2019.
- [4] J. Carpenter, "Light Sensor Performance Comparisons," Winona, 2015.
- [5] "Adafruit," [Online]. Available: <https://www.adafruit.com/>. [Accessed 17 10 2020].