DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Multi-Room Light Usage Analysis Web App

submitted to
Professor Iyad Obeid

ECE 3824: Computation III
Temple University
College of Engineering
1947 North 12th Street
Philadelphia, Pennsylvania 19122

April 29, 2024

Prepared by: Jonathan Isely, Jalen Guan, Leo Berman, Yuan Nghiem, Jake Grinshpun

Emails: Jonathan.Isely@temple.edu, Jialiang.Guan@temple.edu, Leo.Berman@temple.edu, Yuan.Nghiem@temple.edu, Jacob.Grinshpun@temple.edu

A. GOAL

The goal of our project was to develop a system that does the following:

- Hardware component that checks the status of the room lights
- Database for our hardware component to store its results
- Backend to connect all components
- User-friendly website for our database to be searched

B. HARDWARE COMPONENT

Our hardware component requirements:

- Monitor a voltage in pin and make decisions
- Access the internet and make requests

The ESP32, in conjunction with a light sensor module, is a relatively inexpensive option that fulfills all of our requirements.

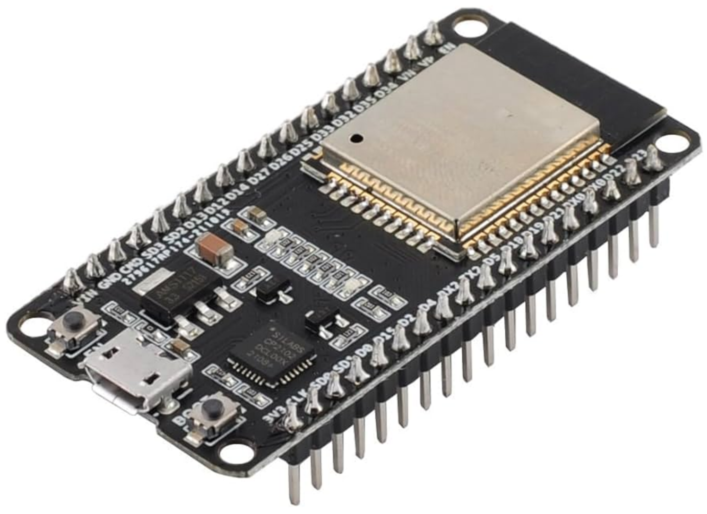
Figure 1: ESP32

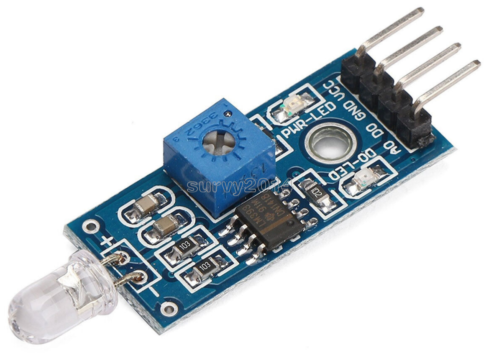
Figure 2: Light Sensor

C. DATABASE COMPONENT

Our database component requirements:

- Multi index searching using room and date/time
- Date/Time granularity to the minute
- Quick-return on multi-query requests for graphics purposes

We ended up using Amazon RDS to host a MySQL instance with one vCPU, 2 GiB of memory, 20 GiB of SSD storage. Due to the short-term nature of our system, we chose an OnDemand pricing model with RDS Proxy to minimize disruptions and increase scalability for future use of the application.

Schema with hour entries as minutes the light is on:

SCHEMA	room	date_entry	hour0	hour1	...	hour23
SAMPLE	603	2024-04-10	1	2	...	3

Figure 3: Schema Example

D. BACKEND

Backend requirements:

- Dynamic web templating for ease of future maintenance.
- SQL querying

A Python webapp using Flask and Jinja2 is a current industry standard that we were able to utilize. hosting on Google Cloud's App Engine, allows us to host our backend and frontend virtually free of charge.

Defining our requests using the following format:

```
@app.route("/date")
def date():
    ...
    return render_template('date.html')

@app.route("/increment_column")
def increment_column():
    ...
    return jsonify(success=True, status_code=200)

@app.route("/")
def index():
    ...
    return render_template('base.html')
```

Figure 4: Pseudo-code for Flask framework

E. FRONTEND

Our frontend requirements:

- User friendly
- Future maintainable
- Aesthetically pleasing

With our frontend being hosted off of the python Flask/Jinja2 framework, HTML/CSS/JavaScript was an obvious choice that allowed us to create readable code with an aesthetically pleasing look.

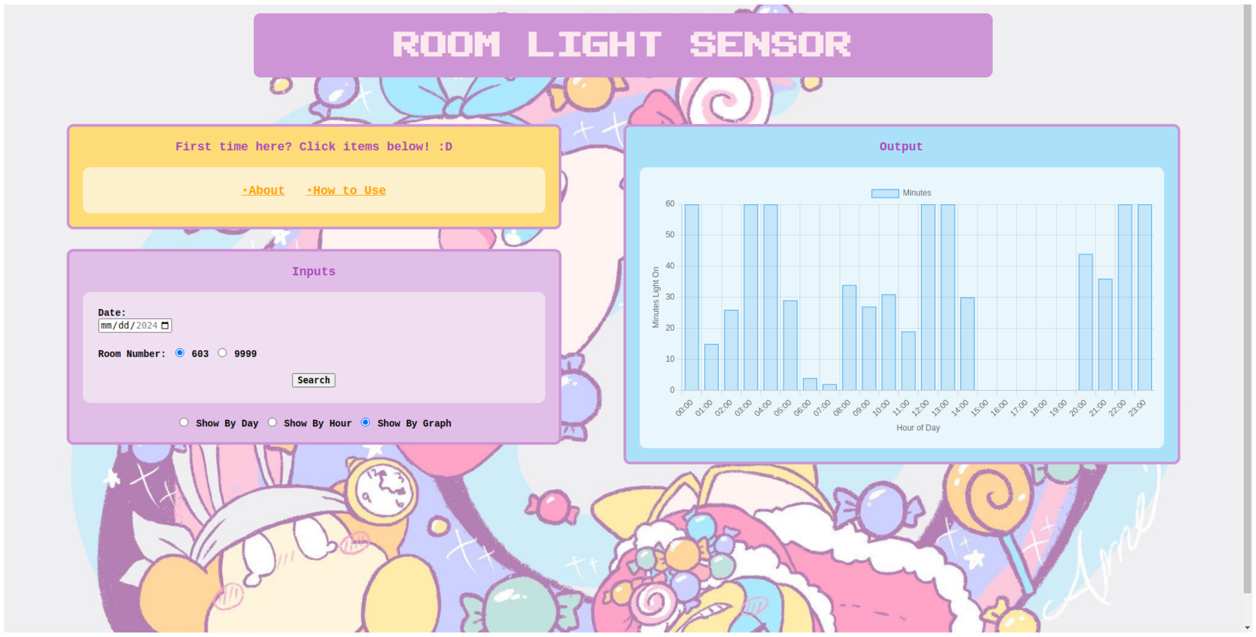

Figure 5: Image of Web App

F. COST ANALYSIS

Item	Cost [$]	Source
ESP32	6.33	Amazon Marketplace
Light Sensor	2.00	Amazon Marketplace
RDS MySQL	24.82 / Month	Amazon Pricing Calculator
RDS Proxy	21.90 / Month	
RDS Storage	2.30 / Month	

Figure 6: Itemized Cost

Upfront Cost: $8.33
Monthly Cost: $49.02