IoT based Smart Lighting System

Problem Description:

As it can be observed over past decades. The rate of urbanization has increased quite rapidly, and it has led to enhancement of services and applications which makes the lifestyle better (Sikder et al.). IoT has helped to make the idea of intelligent home come true. Smart home systems makes the creates a safer, comfortable, humane living environment (Walia et al., 2016). With the introduction of IoT in our daily lives, new opportunities have emerged to develop new services and integrate different application domains with each other using Information and Communication Technologies but to ensure the seamless services, all the applications have to be maintained with limited energy resources. A smart lighting system can be developed using sensors and communication channels to obtain an autonomous and efficient lighting system. One of the advantages of IoT based systems is that it efficiently manages energy as sue to rapid growth in population, demand for energy like electricity has increased exponentially. The IoT based system could be really helpful as it can help to reduce at least 33% of power consumption and save our natural resources (Sikder et al.).

Industrial locations like warehouses, factories and industrial plants, loading/distribution centres etc have a nature of high-power consumption. Specific lighting is required for these industrial locations, and they are mostly installed with high powered lights which emits a huge amount of heat as a result of high output they need to achieve and maintain. Moreover, in addition to the huge amount of heat, a lot of energy is consumed and also require maintenance and upkeep, which is quite costly and also warms up before achieving its full potential (2020). Additionally, these lights have their own power source, and they can be controlled remotely, and also can be equipped with sensors capable of ingesting a variety of information. Basically, sensor-based lights reduce energy usage through features like dimming or turning off light based on the motion detected or intensity of the environment . For example: Bluetooth Low Energy or radio frequency identification sensors can be used in light fixtures which can read BLE or RDIF tags on assets and track their position and movement. And currently there are open-on-top and open-on-bottom solutions for sensor-based lights which means that a variety of sensors can be integrated at the bottom and APIs can be created to a variety of software solution in the top which opens the door for unlimited possibilities (Piunno).

Basically, the aim of this project is to develop a Smart Lighting System which could be maintained for large buildings with many thousands of buildings and switches. The report concentrates on developing a system that implements the smart light system for large building without affecting any other attributes of the building and maintain the scalability to control the intensity of lights and also provide alerts for faulted lights in the building.

Requirements of the system:

Functional requirement and non-functional requirements of the smart lighting system will depend upon Stakeholders. Basically, functional requirements define the basic system behaviour like what system does or what it is supposed to do while non-functional requirements specifies the how the system should do what it is supposed to do (2021) .

Some of the general functional requirements that any IoT system should have:

- Energy Management
- Data management
- Event Management
- Risk Management
- Privacy Management

Some of the function and non- functional requirements related to the system are:

- The smart lighting system should be scalable for large building which uses thousands of lights and switches which means the reliability and performance of the system should not be affected with increase in lights and switches.
- The system should change the intensity of light or turn off and on with respect to the lighting conditions of the environment.
- The system should create alert when some of the LEDs stop working or show unusual behaviour. For example, the system can use MQTT with Complex Event Processing to report faulted LEDs and create alert so that those lights could be repaired or replaced.
- The system should provide the user which information the intensity of the room and status of LED with respect to it. For, example if the light intensity of the room is high which means there is no need to turn the LEDs on, so the system should keep the LEDs off and when the light intensity of the room goes below the threshold then of should automatically turn the LEDs on.
- The system should provide visual representation of the data collected from the smart lighting system to the user so that the user could analyse how the lights are working and how the energy consumption could be reduced.
- The system’s performance should not posses’ latency due to increased lights in the building and should have the capacity to provide the same output. So, the system can used EC2 instances and Load balancer to balance the load and reduce the pressure on server.

- The data stored for all the room and their LEDs and switches must be stored securely in online databases which could be only accessed by authorised people of the company and should provide good secure protocols to avoid hacking of the data.

Risks associated with the system which could occur and should be taken care of while developing the prototype for the smart lighting system:

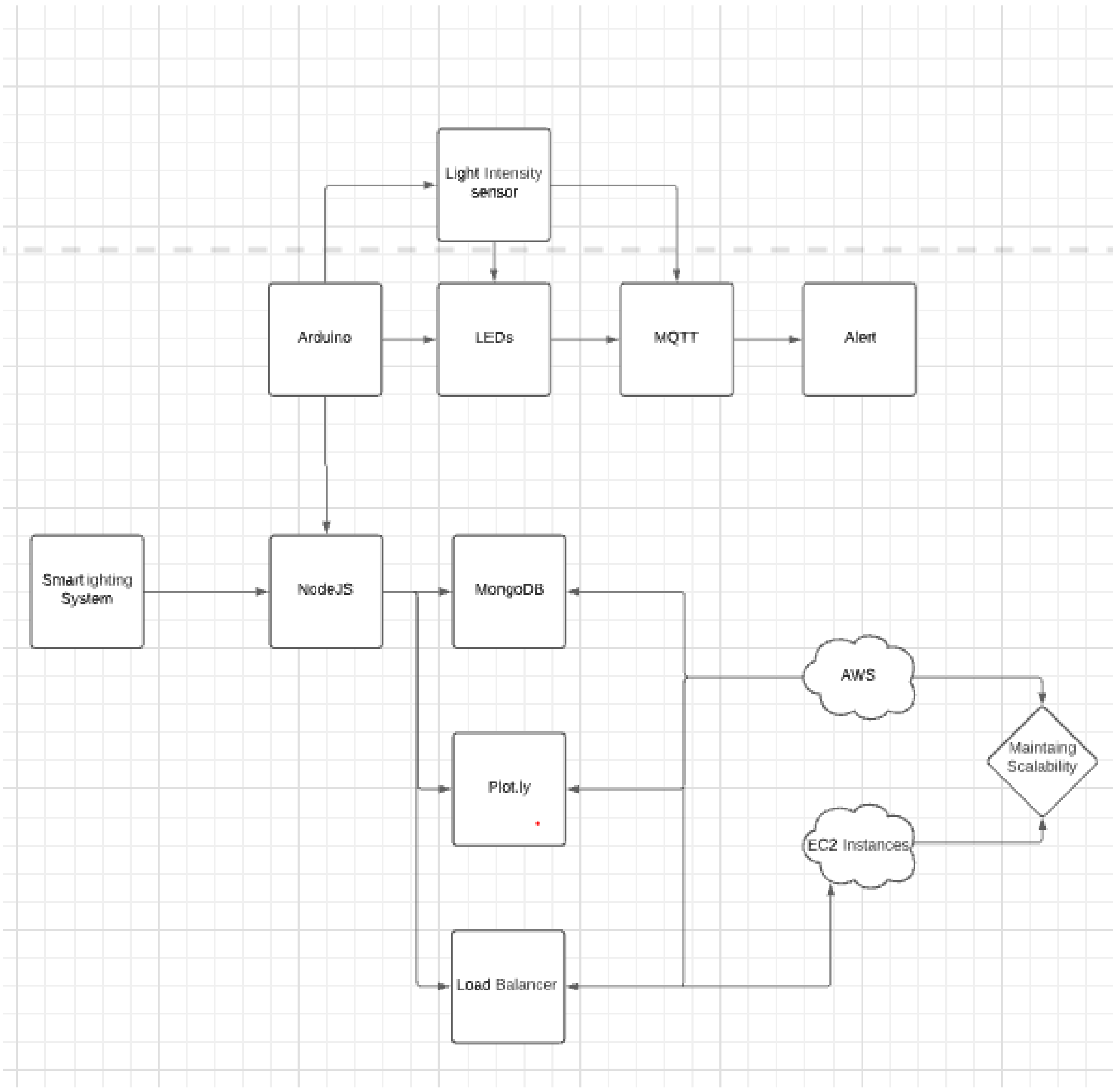
- Hacker's attack
- Botnet attack
- Rogue IoT devices
- Unsafe interfaces
- Insufficient data protection
- Lack of regular patches and updates
- Poor IoT management (2021)

System design:

This protype is created using Arduino (C++) and Node.js framework. Arduino will be connected to LEDs and Light sensor(a photoresistor that reads light intensity) and will be running using C++ language on Arduino IDE and sending data on a serial port related to the status of LED and light intensity of the room of the building. The server will receive the data generated from Arduino and sensors using serial port library. This system is using MongoDB to store the data and plot.ly to visualise the data. Even socket.io can be used to generate a live graph which updates in real time. The data generated from Arduino will be directly saved in a csv file and in the code will be used in the form JSON format. The light intensity sensor will be used to measure the intensity of the room so that the intensity of the LED can be changed accordingly. When the intensity of the room is already high, it will make the LED go off and vice-versa. I will be showing 1 room of a building with 3 or 4 LEDs, and which could be controlled using web interface using Photon particle or Argon particle or could be controlled using the automatic smart light system which changes the status of light using the data obtained from sensors. The web interface can be only accessed by authorised person using security protocols. The code will use MQTT to publish the intensity of each LED and then it checks for faulty LEDs using Complex Event Processing. The user will have the option to add more rooms and LEDs without affecting the scalability of the system as the system is using Load balancer and multiple EC2 instances (created using same AMI images) to balance the load and create a separate collection for each room.

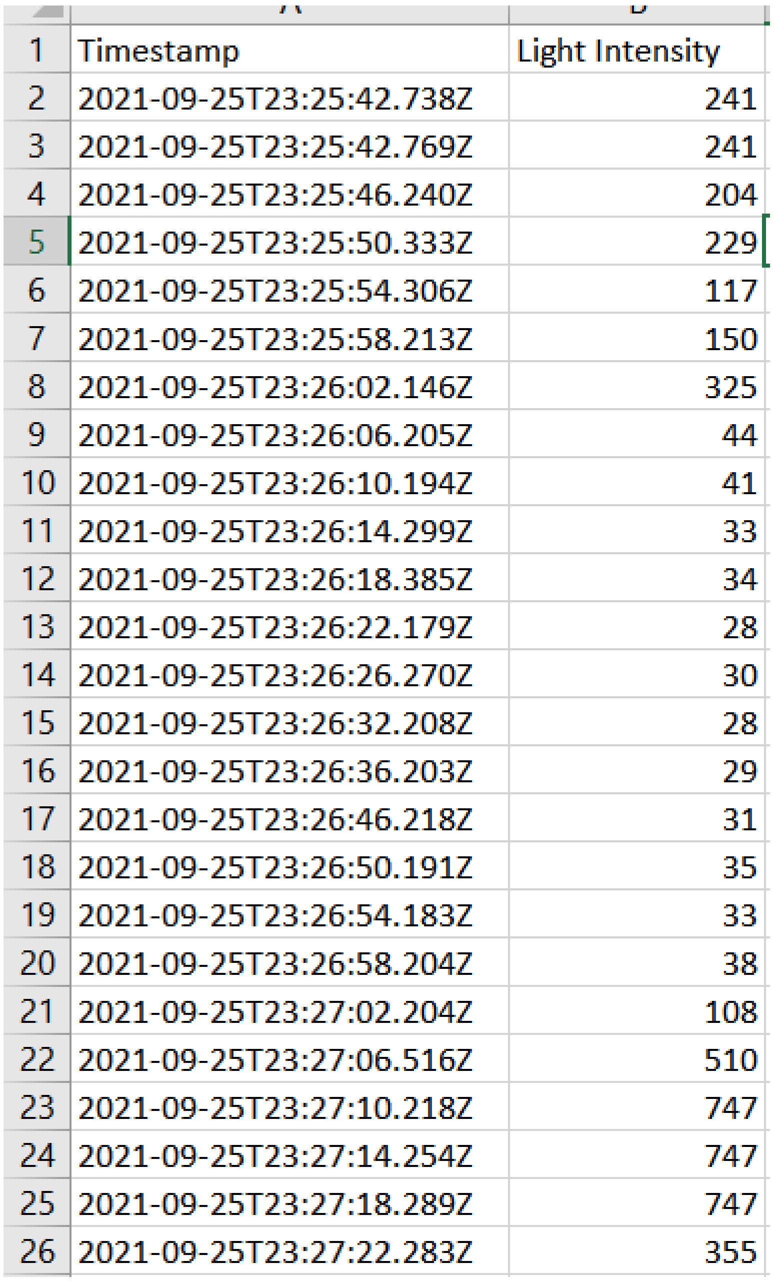
The LEDs can be controlled using Photon or Argon particle which makes it easy to control the LEDs from anywhere in the world using webpage over internet. Plot.ly will be used to visualise the data and it makes it quite easy for the user to analyse the energy consumption and mange resources.

Architecture Diagram of the system:

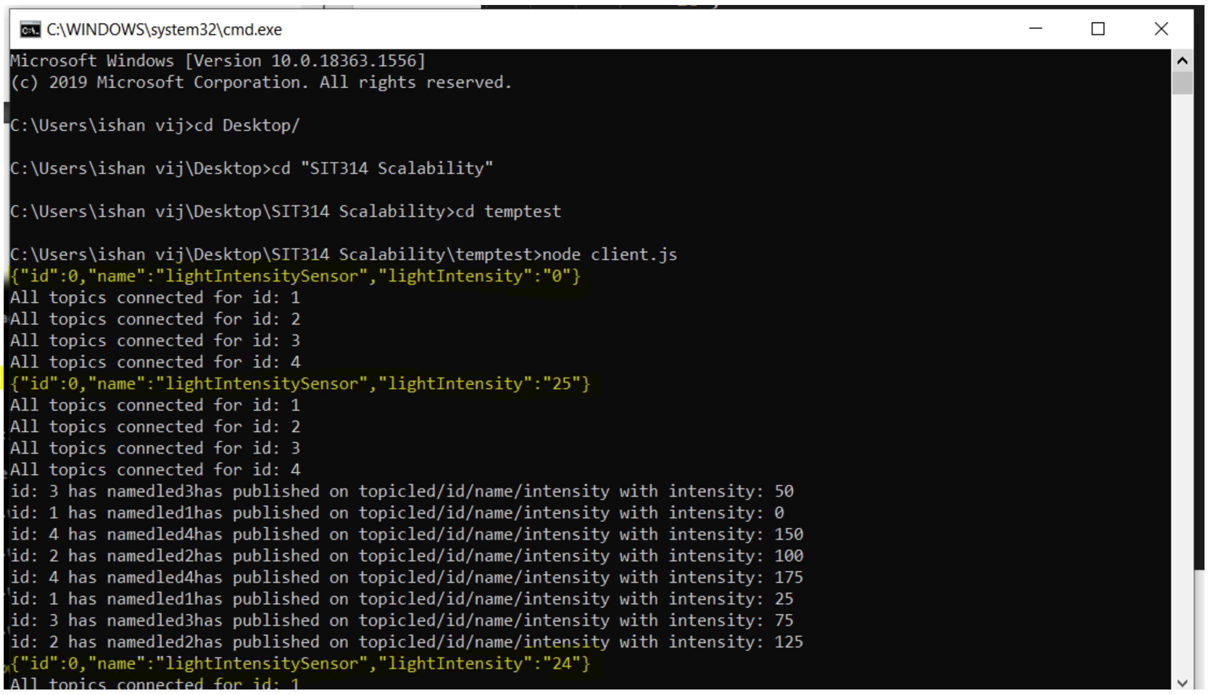


Data Formats:

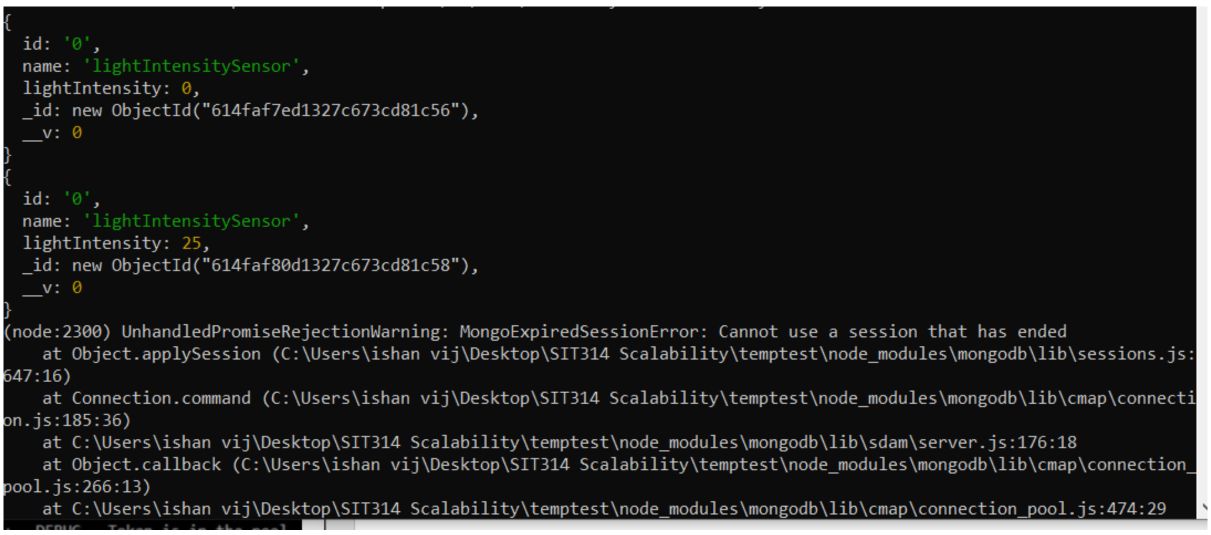
For this task the data is collected in the csv format while collecting from Arduino in an excel file and the same is being used to visualise the data in real time using Plot.ly can be also converted to a JSON format. The advantage of csv file is that it systematically arranges the values from sensors under appropriate columns with time stamps and CSV file is usually smaller in size in comparison to JSON file. Moreover, excel file can make it easy for the user to visualize the data and also allow APIs to post the data easily. Aldo the JSON format is used to upload data to mongo DB.



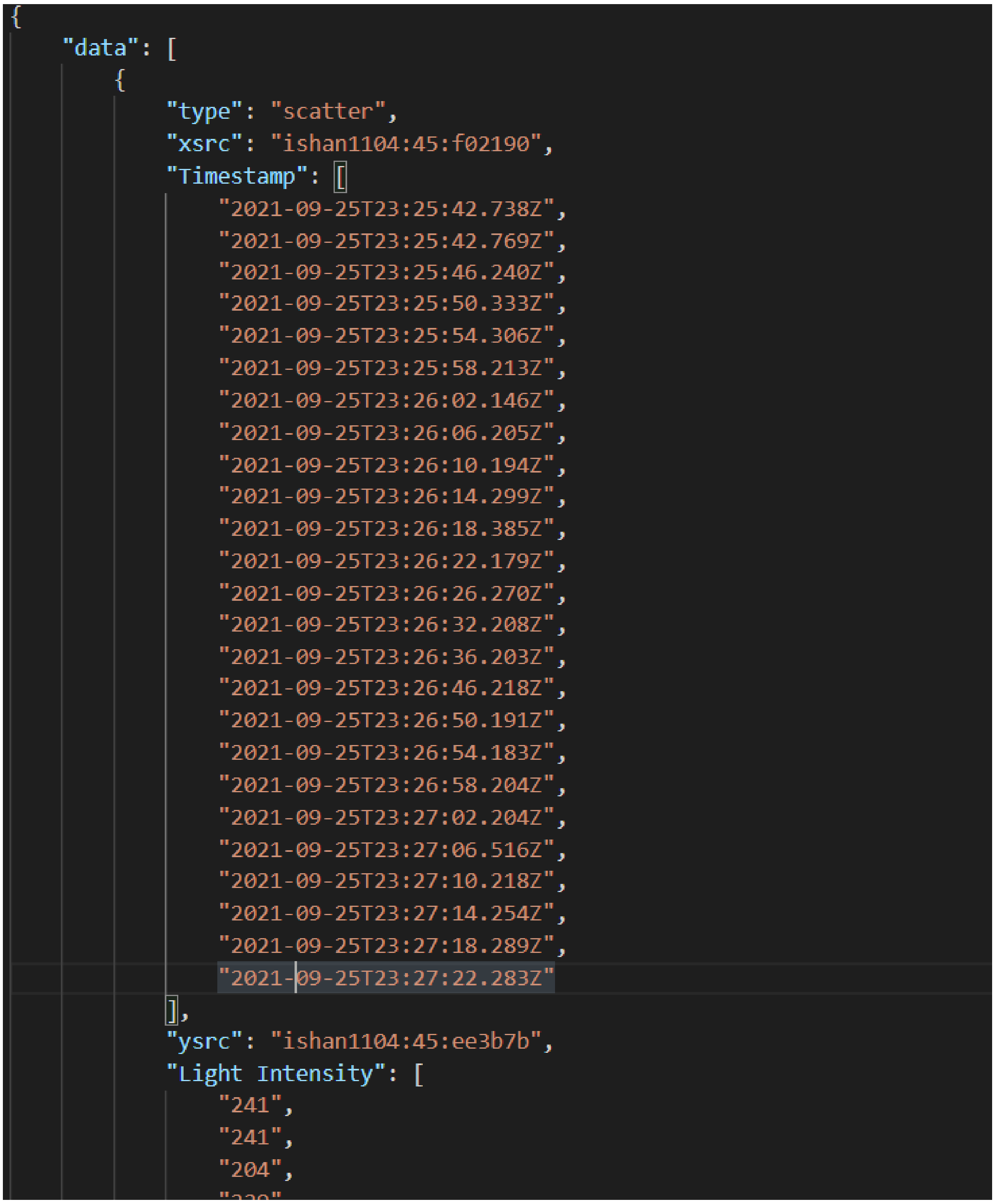
1	Timestamp	Light Intensity
2	2021-09-25T23:25:42.738Z	241
3	2021-09-25T23:25:42.769Z	241
4	2021-09-25T23:25:46.240Z	204
5	2021-09-25T23:25:50.333Z	229
6	2021-09-25T23:25:54.306Z	117
7	2021-09-25T23:25:58.213Z	150
8	2021-09-25T23:26:02.146Z	325
9	2021-09-25T23:26:06.205Z	44
10	2021-09-25T23:26:10.194Z	41
11	2021-09-25T23:26:14.299Z	33
12	2021-09-25T23:26:18.385Z	34
13	2021-09-25T23:26:22.179Z	28
14	2021-09-25T23:26:26.270Z	30
15	2021-09-25T23:26:32.208Z	28
16	2021-09-25T23:26:36.203Z	29
17	2021-09-25T23:26:46.218Z	31
18	2021-09-25T23:26:50.191Z	35
19	2021-09-25T23:26:54.183Z	33
20	2021-09-25T23:26:58.204Z	38
21	2021-09-25T23:27:02.204Z	108
22	2021-09-25T23:27:06.516Z	510
23	2021-09-25T23:27:10.218Z	747
24	2021-09-25T23:27:14.254Z	747
25	2021-09-25T23:27:18.289Z	747
26	2021-09-25T23:27:22.283Z	355



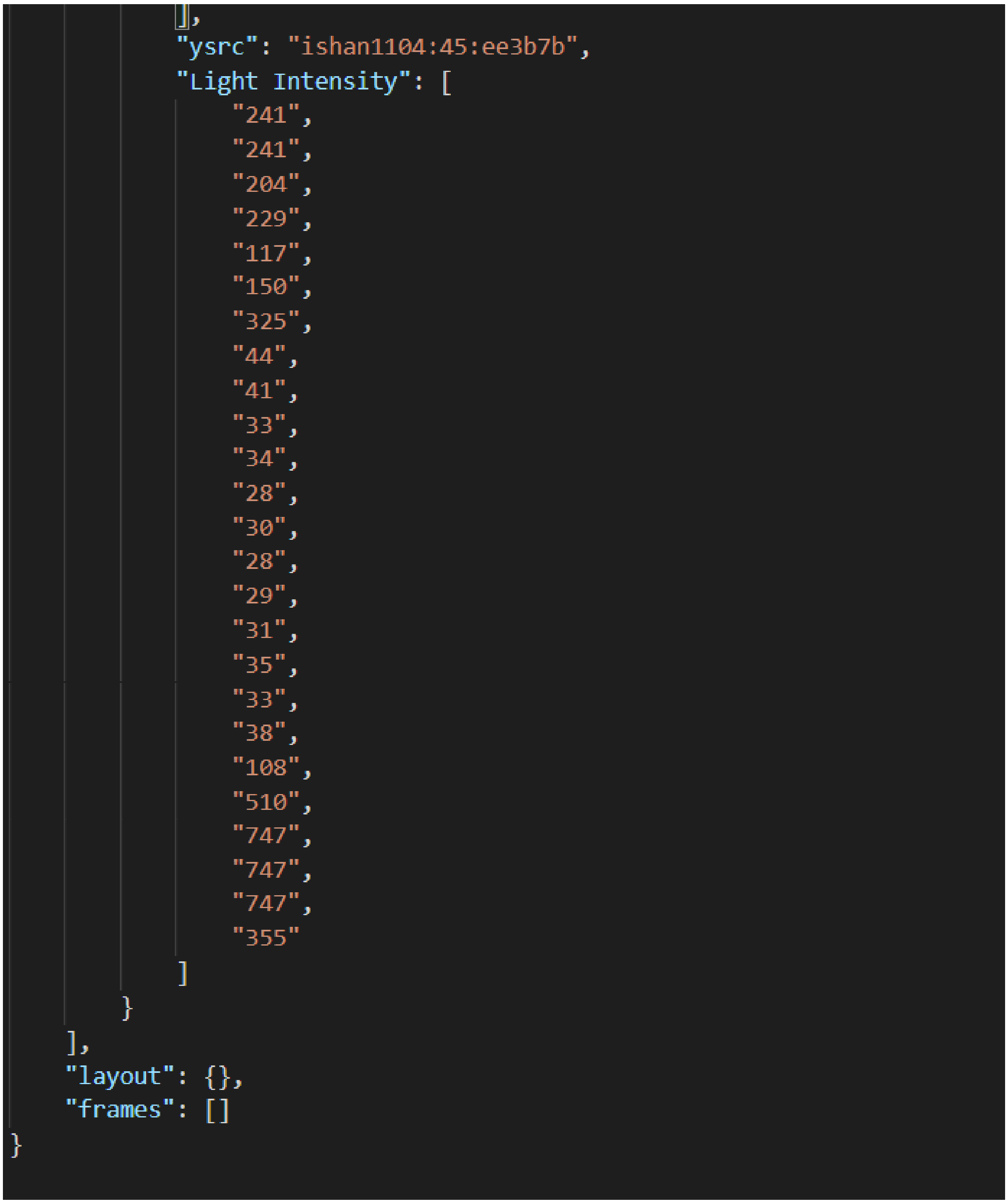
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ishan vij>cd Desktop/

C:\Users\ishan vij\Desktop>cd "SIT314 Scalability"

C:\Users\ishan vij\Desktop\SIT314 Scalability>cd temptest

C:\Users\ishan vij\Desktop\SIT314 Scalability\temptest>node client.js
{"id":0,"name":"lightIntensitySensor","lightIntensity":"0"}
All topics connected for id: 1
All topics connected for id: 2
All topics connected for id: 3
All topics connected for id: 4
{"id":0,"name":"lightIntensitySensor","lightIntensity":"25"}
All topics connected for id: 1
All topics connected for id: 2
All topics connected for id: 3
All topics connected for id: 4
id: 3 has namedled3has published on topicled/id/name/intensity with intensity: 50
id: 1 has namedled1has published on topicled/id/name/intensity with intensity: 0
id: 4 has namedled4has published on topicled/id/name/intensity with intensity: 150
id: 2 has namedled2has published on topicled/id/name/intensity with intensity: 100
id: 4 has namedled4has published on topicled/id/name/intensity with intensity: 175
id: 1 has namedled1has published on topicled/id/name/intensity with intensity: 25
id: 3 has namedled3has published on topicled/id/name/intensity with intensity: 75
id: 2 has namedled2has published on topicled/id/name/intensity with intensity: 125
{"id":0,"name":"lightIntensitySensor","lightIntensity":"24"}
All topics connected for id: 1
```



```
{
  id: '0',
  name: 'lightIntensitySensor',
  lightIntensity: 0,
  _id: new ObjectId("614faf7ed1327c673cd81c56"),
  __v: 0
}
{
  id: '0',
  name: 'lightIntensitySensor',
  lightIntensity: 25,
  _id: new ObjectId("614faf80d1327c673cd81c58"),
  __v: 0
}
(node:2300) UnhandledPromiseRejectionWarning: MongoExpiredSessionError: Cannot use a session that has ended
    at Object.applySession (C:\Users\ishan vij\Desktop\SIT314 Scalability\temptest\node_modules\mongodb\lib\sessions.js:
647:16)
    at Connection.command (C:\Users\ishan vij\Desktop\SIT314 Scalability\temptest\node_modules\mongodb\lib\cmap\connecti
on.js:185:36)
    at C:\Users\ishan vij\Desktop\SIT314 Scalability\temptest\node_modules\mongodb\lib\sdam\server.js:176:18
    at Object.callback (C:\Users\ishan vij\Desktop\SIT314 Scalability\temptest\node_modules\mongodb\lib\cmap\connection_
pool.js:266:13)
    at C:\Users\ishan vij\Desktop\SIT314 Scalability\temptest\node_modules\mongodb\lib\cmap\connection_pool.js:474:29
```



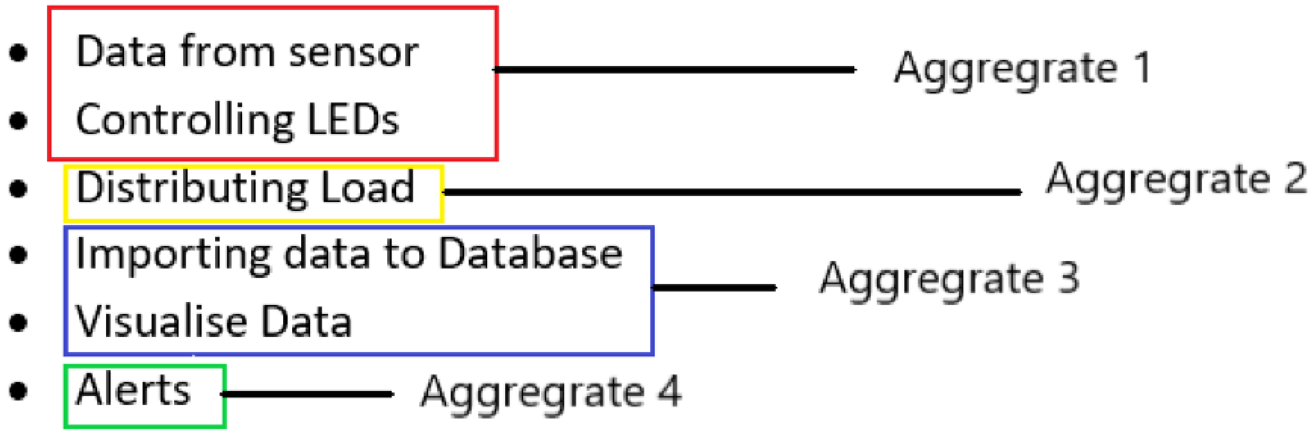
```
{
    "data": [
        {
            "type": "scatter",
            "xsrc": "ishan1104:45:f02190",
            "Timestamp": [
                "2021-09-25T23:25:42.738Z",
                "2021-09-25T23:25:42.769Z",
                "2021-09-25T23:25:46.240Z",
                "2021-09-25T23:25:50.333Z",
                "2021-09-25T23:25:54.306Z",
                "2021-09-25T23:25:58.213Z",
                "2021-09-25T23:26:02.146Z",
                "2021-09-25T23:26:06.205Z",
                "2021-09-25T23:26:10.194Z",
                "2021-09-25T23:26:14.299Z",
                "2021-09-25T23:26:18.385Z",
                "2021-09-25T23:26:22.179Z",
                "2021-09-25T23:26:26.270Z",
                "2021-09-25T23:26:32.208Z",
                "2021-09-25T23:26:36.203Z",
                "2021-09-25T23:26:46.218Z",
                "2021-09-25T23:26:50.191Z",
                "2021-09-25T23:26:54.183Z",
                "2021-09-25T23:26:58.204Z",
                "2021-09-25T23:27:02.204Z",
                "2021-09-25T23:27:06.516Z",
                "2021-09-25T23:27:10.218Z",
                "2021-09-25T23:27:14.254Z",
                "2021-09-25T23:27:18.289Z",
                "2021-09-25T23:27:22.283Z"
            ],
            "ysrc": "ishan1104:45:ee3b7b",
            "Light Intensity": [
                "241",
                "241",
                "204",
```



```
            ],
            "ysrc": "ishan1104:45:ee3b7b",
            "Light Intensity": [
                "241",
                "241",
                "204",
                "229",
                "117",
                "150",
                "325",
                "44",
                "41",
                "33",
                "34",
                "28",
                "30",
                "28",
                "29",
                "31",
                "35",
                "33",
                "38",
                "108",
                "510",
                "747",
                "747",
                "747",
                "355"
            ]
        }
    ],
    "layout": {},
    "frames": []
}
```

Scalability achieved:

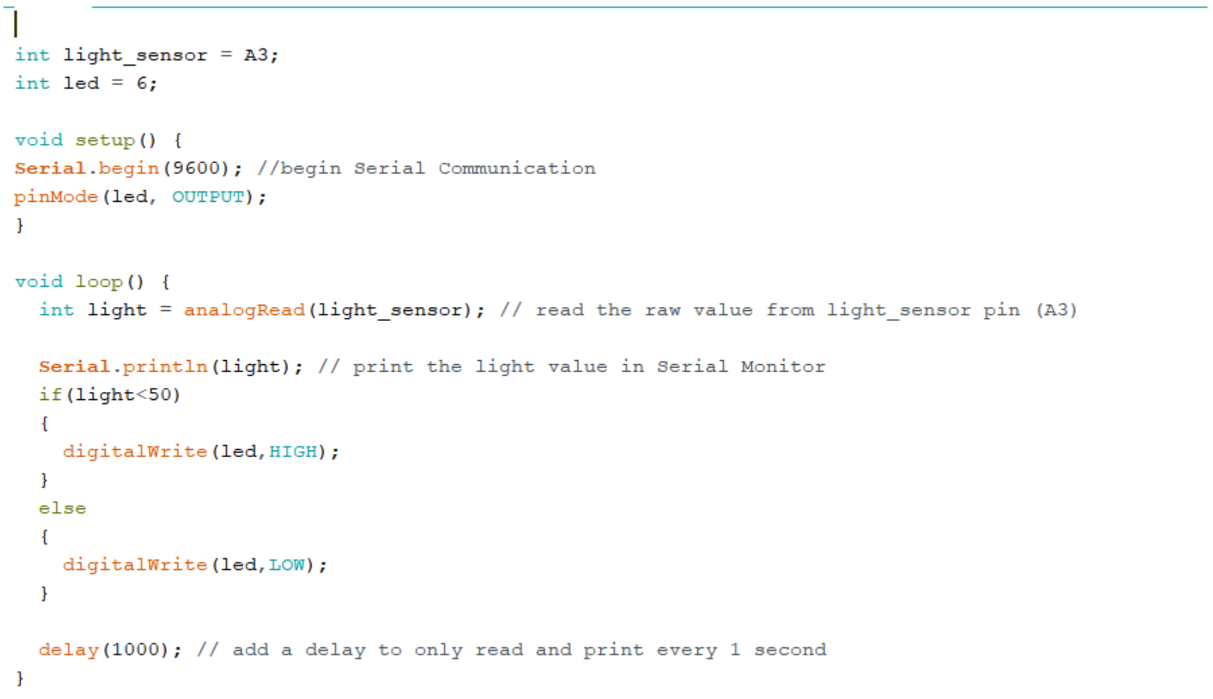
In this project, Arduino will be generating the data for the Led lights with light intensity sensor. So, in this project a Load balancer with multiple EC2 instances is used to distribute the workloads across servers to reduce the pressure and help the system to maintain its performance levels and hence help to improve the scalability of the system. MongoDB and Plot.ly is used to improve the scalability of the IoT data storage. MongoDB supports horizontal scaling and is able to maintain and increase its level of performance under lager operational demands (Maxim, 2016). For each room, MongoDB will create a new collection in the data base and help to distribute the pressure on servers and similarly each room will have separate graphs on plot.ly. Light Intensity is used to automatically turn the lights on and off depending upon light intensity of the room and save resources and load on the servers. The advantage of this prototype is that each part of infrastructure is responsible for a particular action and if one service fails, it no going to completely shutdown the system. I created microservice model for this scenario to improve the scalability of the system.



The benefit of using Light Intensity sensor with LEDs is that alter the intensity of LEDs with respect to the light intensity of the room and helps to save a lot of electricity and resources. Scalability implies that the system should maintain its performance regardless of how many devices and lights are connected to it. To achieve scalability EC2 load balancer is used as it can distribute the workload across multiple EC2 instances and one of the functionalities of load balancer is auto scaling which means that it can turn on and off EC2 instances depending upon the usage and improve the performance of the system. MQTT is used to in this system which is a lightweight messaging protocol that works with a broker based publish-subscribe mechanism. It used complex event processing for this system and create alerts when one of two sensors show faulty data so that the system could be repaired.

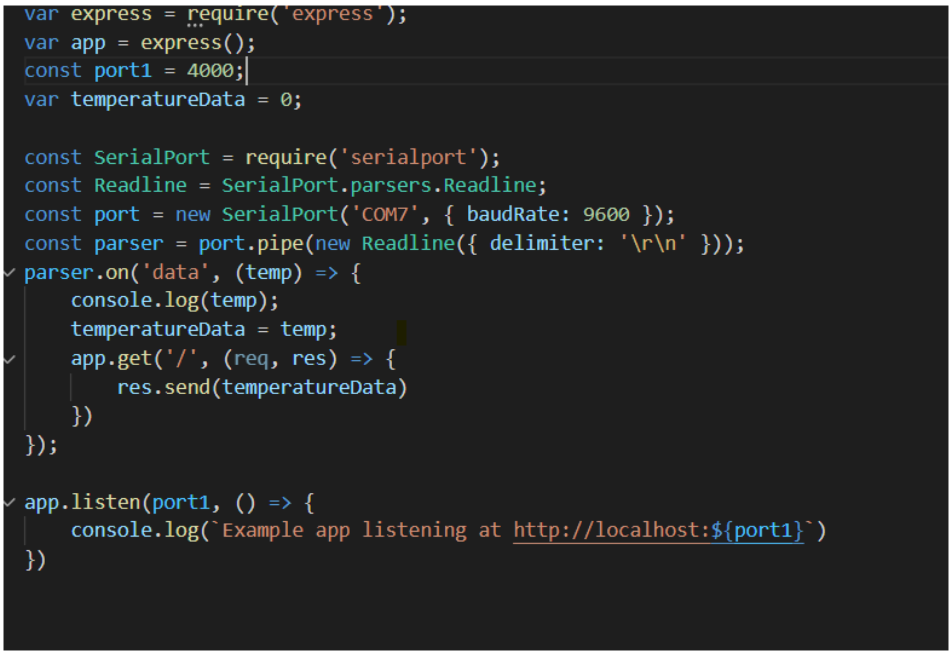
System Demonstration:

1. The image given below is the code for Arduino and its basically measuring the light intensity of the room and changing the states of LED based upon the data generated.



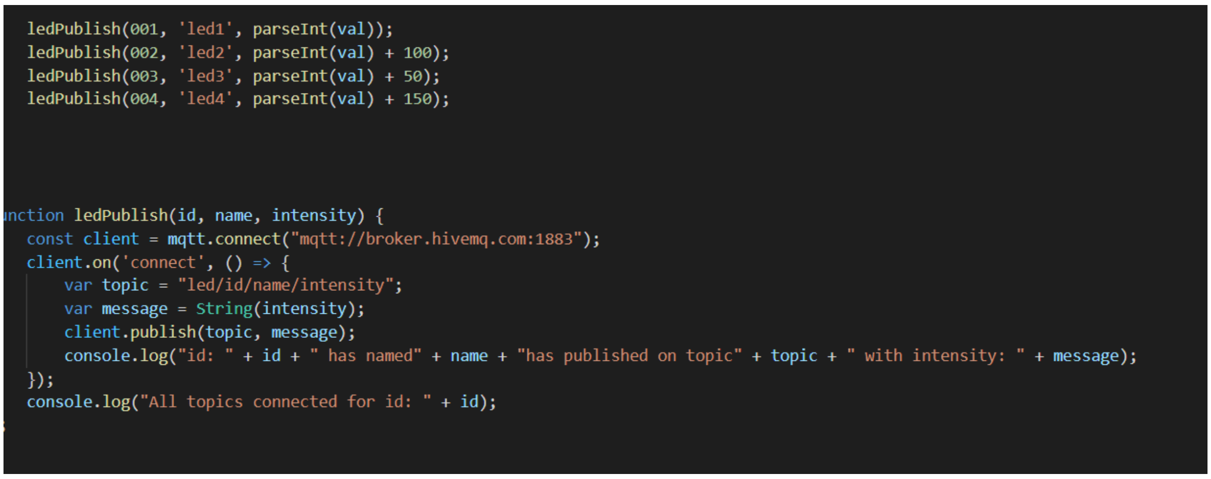
```
int light_sensor = A3;
int led = 6;

void setup() {
Serial.begin(9600); //begin Serial Communication
pinMode(led, OUTPUT);
}

void loop() {
  int light = analogRead(light_sensor); // read the raw value from light_sensor pin (A3)

  Serial.println(light); // print the light value in Serial Monitor
  if(light<50)
  {
    digitalWrite(led,HIGH);
  }
  else
  {
    digitalWrite(led,LOW);
  }

  delay(1000); // add a delay to only read and print every 1 second
}
```

2. The image given below shows node.js receiving data from the microcontroller(Arduino) using serialport package.



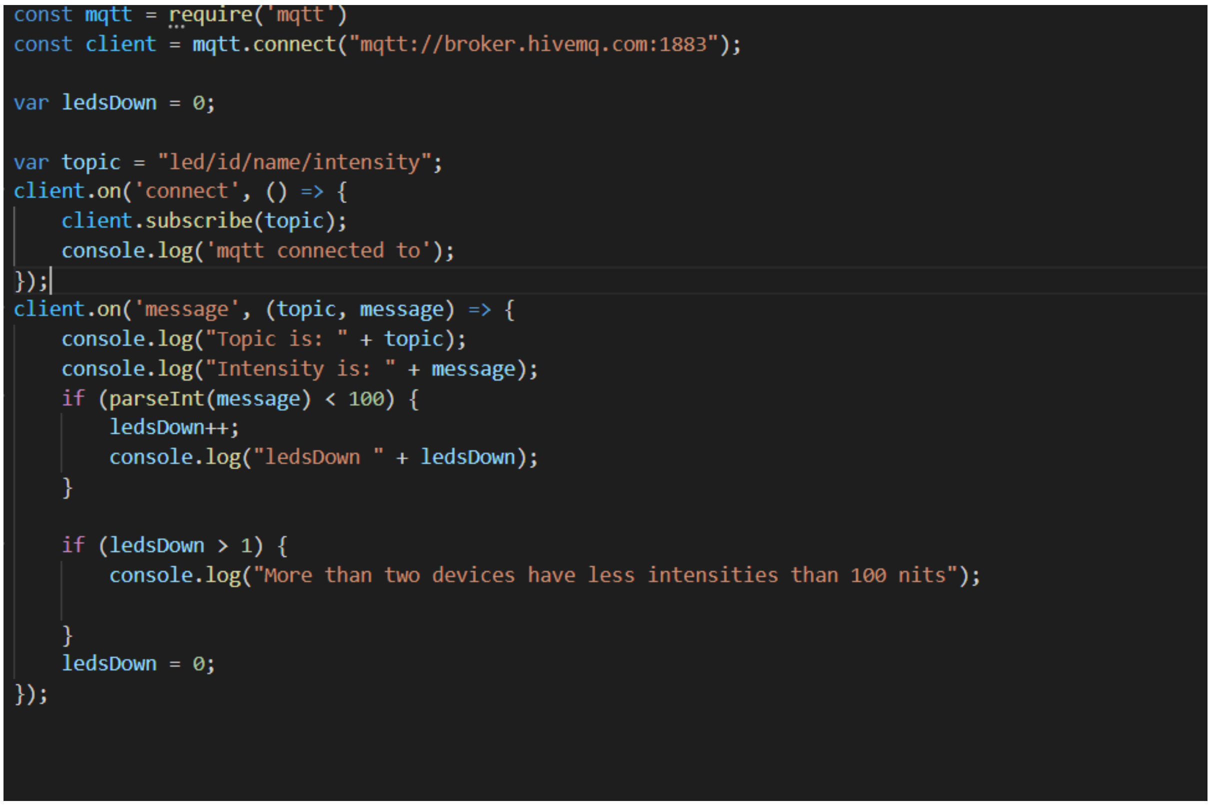
```
var express = require('express');
var app = express();
const port1 = 4000;
var temperatureData = 0;

const SerialPort = require('serialport');
const Readline = SerialPort.parsers.Readline;
const port = new SerialPort('COM7', { baudRate: 9600 });
const parser = port.pipe(new Readline({ delimiter: '\r\n' }));
parser.on('data', (temp) => {
    console.log(temp);
    temperatureData = temp;
    app.get('/', (req, res) => {
        res.send(temperatureData)
    })
});

app.listen(port1, () => {
    console.log(`Example app listening at http://localhost:${port1}`)
})
```

3. The image given below shows the MQTT which is creating four LEDs based upon the data received from Arduino and publishing it upon the Topic: “led/id/name/intensity”.



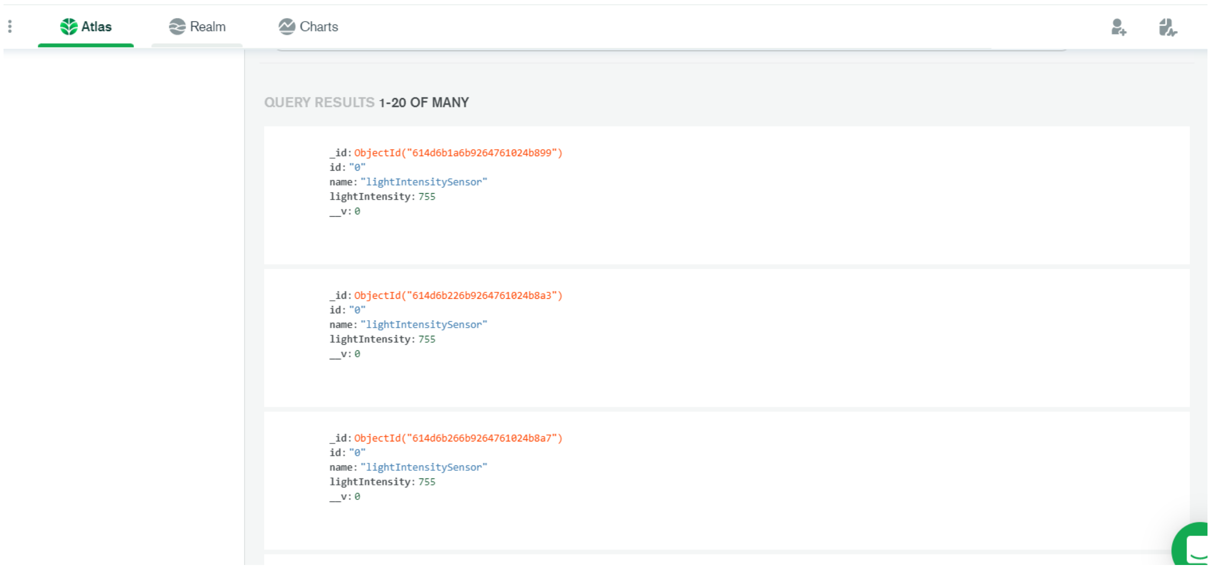
```
  ledPublish(001, 'led1', parseInt(val));
  ledPublish(002, 'led2', parseInt(val) + 100);
  ledPublish(003, 'led3', parseInt(val) + 50);
  ledPublish(004, 'led4', parseInt(val) + 150);



nction ledPublish(id, name, intensity) {
  const client = mqtt.connect("mqtt://broker.hivemq.com:1883");
  client.on('connect', () => {
      var topic = "led/id/name/intensity";
      var message = String(intensity);
      client.publish(topic, message);
      console.log("id: " + id + " has named" + name + "has published on topic" + topic + " with intensity: " + message);
  });
  console.log("All topics connected for id: " + id);
```

4. MQTT will subscribe to the topic and check it more than two sensors are showing faulty data and create alarm if there is any fault with the system.



```
const mqtt = require('mqtt')
const client = mqtt.connect("mqtt://broker.hivemq.com:1883");

var ledsDown = 0;

var topic = "led/id/name/intensity";
client.on('connect', () => {
    client.subscribe(topic);
    console.log('mqtt connected to');
});
client.on('message', (topic, message) => {
    console.log("Topic is: " + topic);
    console.log("Intensity is: " + message);
    if (parseInt(message) < 100) {
        ledsDown++;
        console.log("ledsDown " + ledsDown);
    }

    if (ledsDown > 1) {
        console.log("More than two devices have less intensities than 100 nits");

    }
    ledsDown = 0;
});
```

5. Data stored on MongDB.



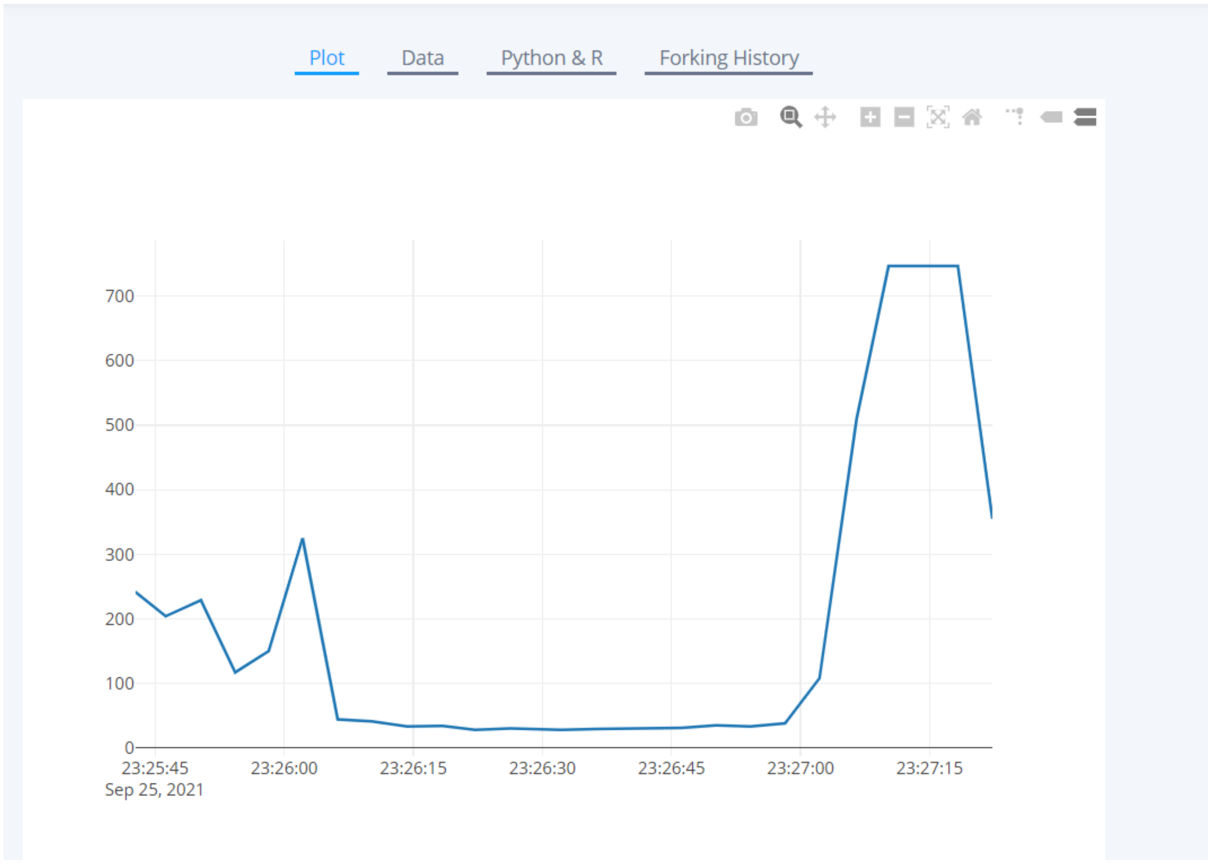
QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("614d6b1a6b9264761024b899")
id: "0"
name: "lightIntensitySensor"
lightIntensity: 755
__v: 0
```

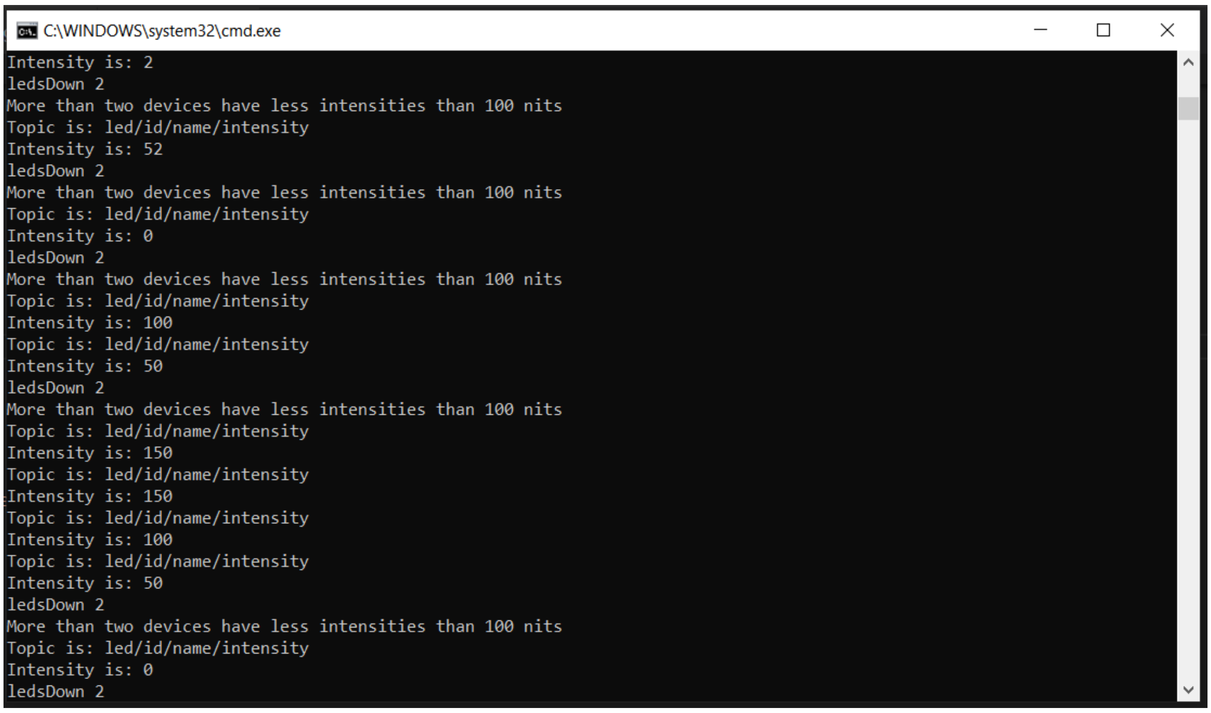
```
_id: ObjectId("614d6b226b9264761024b8a3")
id: "0"
name: "lightIntensitySensor"
lightIntensity: 755
__v: 0
```

```
_id: ObjectId("614d6b266b9264761024b8a7")
id: "0"
name: "lightIntensitySensor"
lightIntensity: 755
__v: 0
```

6. Graph plotted on Plot.ly.

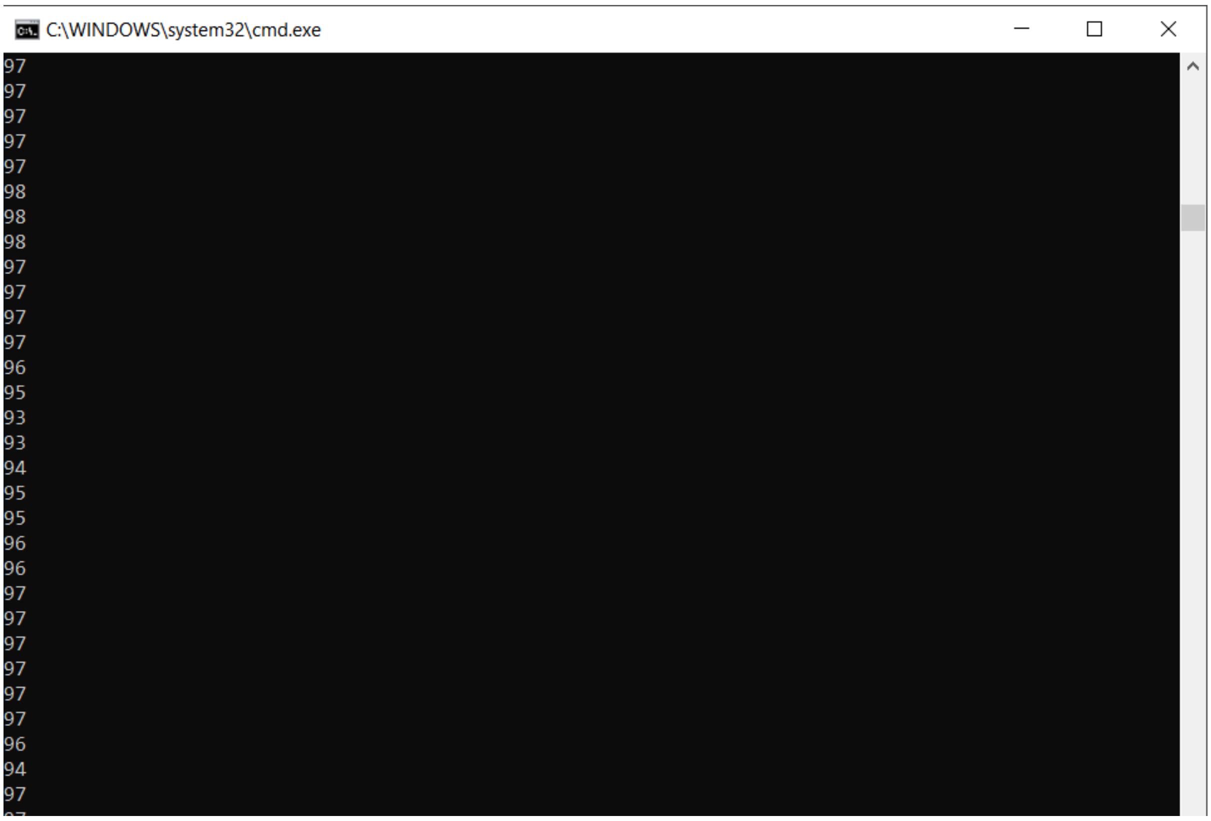


7. Alert is created when at least two sensors show less than 100 units of light intensity.


C:\WINDOWS\system32\cmd.exe

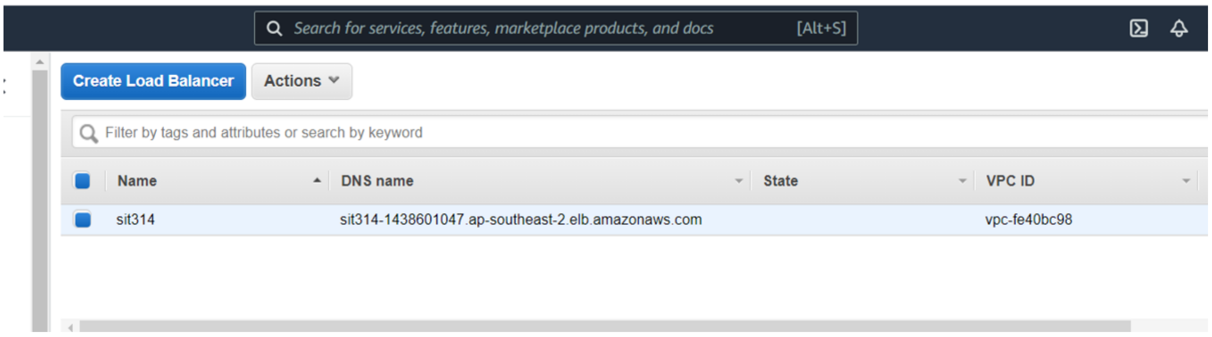
```
Intensity is: 2
ledsDown 2
More than two devices have less intensities than 100 nits
Topic is: led/id/name/intensity
Intensity is: 52
ledsDown 2
More than two devices have less intensities than 100 nits
Topic is: led/id/name/intensity
Intensity is: 0
ledsDown 2
More than two devices have less intensities than 100 nits
Topic is: led/id/name/intensity
Intensity is: 100
Topic is: led/id/name/intensity
Intensity is: 50
ledsDown 2
More than two devices have less intensities than 100 nits
Topic is: led/id/name/intensity
Intensity is: 150
Topic is: led/id/name/intensity
Intensity is: 150
Topic is: led/id/name/intensity
Intensity is: 100
Topic is: led/id/name/intensity
Intensity is: 50
ledsDown 2
More than two devices have less intensities than 100 nits
Topic is: led/id/name/intensity
Intensity is: 0
ledsDown 2
```

8. Data from serialport:

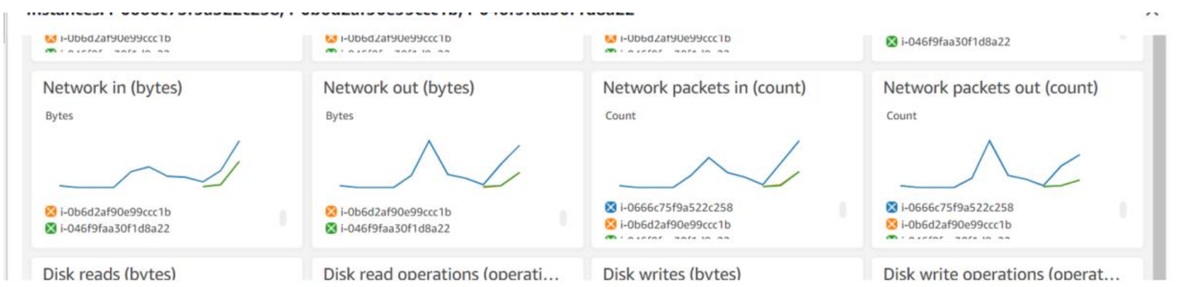

C:\WINDOWS\system32\cmd.exe

```
97
97
97
97
97
98
98
98
97
97
97
97
96
95
93
93
94
95
95
96
96
97
97
97
97
97
97
96
94
97
```

9. AWS and Load Balancer:


Search for services, features, marketplace products, and docs
[Alt+S]
Create Load Balancer
Actions
Filter by tags and attributes or search by keyword

Name	DNS name	State	VPC ID
sit314	sit314-1438601047.ap-southeast-2.elb.amazonaws.com		vpc-fe40bc98



Link for GitHub: https://github.com/ivij/SIT314-project

Bibliography:

Sikder, A.K. et al., Iot-enabled smart lighting systems for smart cities. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/abstract/document/8301744 [Accessed September 26, 2021].

Walia, N.K., Kalra, P. & Mehrotra, D., 2016. An iot by information retrieval approach: Smart lights controlled using wifi. *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*.

Anon, 2020. Industrial led lighting solutions for your warehouse or factory space. *Power Save - Energy Solutions*. Available at: https://powersave.com.au/industrial-solutions/ [Accessed September 26, 2021].

Piunno, A., Building intelligent environments with led lighting and the industrial internet of things. *Current*. Available at: https://www.gecurrent.com/inspiration/building-intelligent-environments-with-LED-and-IoT [Accessed September 26, 2021].

Anon, 2021. Functional vs Non-Functional Requirements: The Definitive Guide. *QRA Corp*. Available at: https://qracorp.com/functional-vs-non-functional-requirements/ [Accessed September 26, 2021].

Anon, 2021. IoT security issues IN 2021: A business perspective. *Thales Group*. Available at: https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/magazine/internet-threats [Accessed September 26, 2021].

Maxim, L., 2016. Divide and Conquer: High scalability with MONGODB sharding. *dzone.com*. Available at: https://dzone.com/articles/divide-and-conquer-high-scalability-with-mongodb-t#:~:text=A%20system%20that%20scales%20well,with%20large%20sets%20of%20data. [Accessed September 26, 2021].

Marking Grid - Report

Criteria	Pass	Credit	Distinction	High Distinction	Grade Awarded
Problem Description and aims of the system	Problem is not well defined, or description lacks significant detail needed to understand problem features or why it is a problem	Problem description lacks detail and demonstrates insufficient understanding of problem features	Description conveys major features of problem and relates relevant dimensions that make the problem hard to solve	Description presents cohesive and thorough understanding of the problem and why it is difficult to solve given current knowledge and capabilities in computing and/or engineering	
Requirements of the system	List of unstructured system requirements	Functional and none-functional (quality) requirements listed. Good attempts relate these to high-level system requirements.	Ranked requirements based on an estimation of effort or time needed.	Relationships between requirements clear including risks of not completing features.	
System Design	Overall design and block diagram	Convincing argument for decisions made	Well referenced to commercial and academic systems	Innovative system design, fully referenced and justified.	
Data Formats	XML or JSON	Well described and justified argument for data format.	Use of JSON or XML schemas. Fully annotated data descriptions.	All messages in the system using well defined and run-time checked data formats.	
Scalability	General description of how the system performance scales	Consideration of scalability issues beyond performance	Fully referenced academic literature of scalability issues in IoT applications	Clear well-defined consideration of scalability in all aspects of system design.	
	Evidence of work completed.	Clear demonstration of scalability	Clear and appropriate	Innovative demonstration of all features. This could be a	

Demonstration		features.	demonstration of all aspects of the operation and scalability of the system	video, interactive powerpoints etc.	
Overall:					