San Jose State University

Software Engineering Department

Spring 2016

CMPE 296 – IoT

Lab 2 – Night Proximity Sensor - IoT Full Stack

Instructor

Prof. Sang Shim

Submitted By:

Vivek Maheshwari

vivek.maheshwari@sjsu.edu

SJSU ID: 010105173

Pictures of Raspberry Pi Connection / Setup

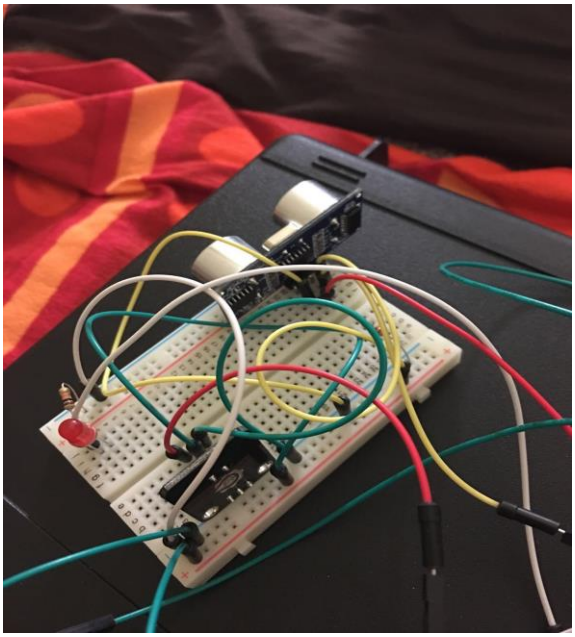

Fig 1.

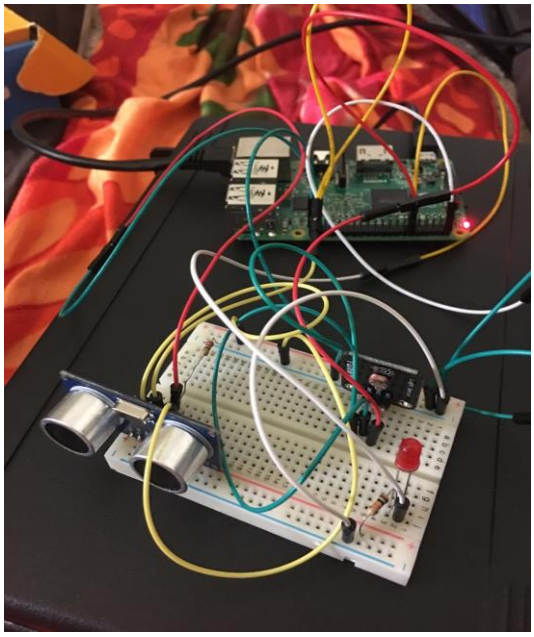

Fig 2.

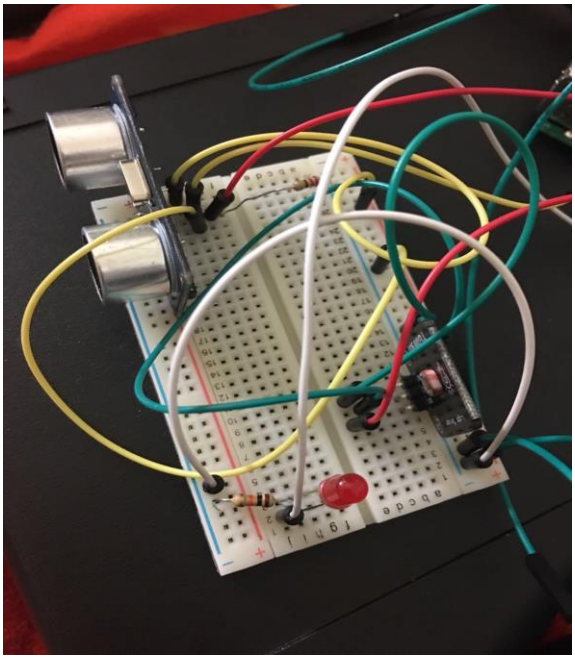

Fig 3.

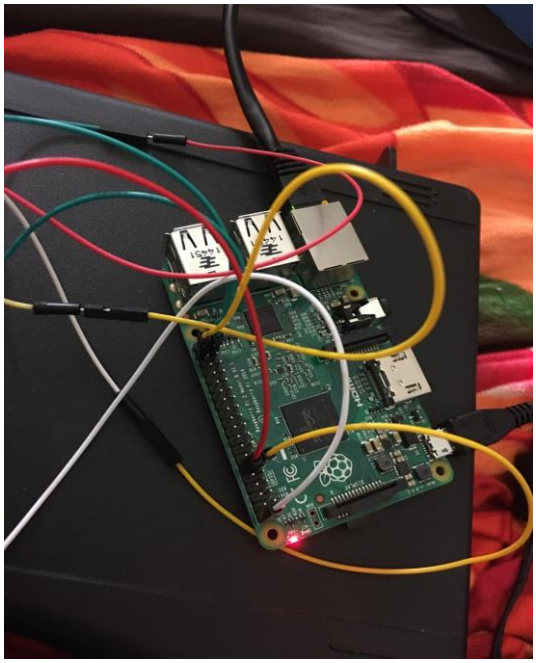

Fig 4.

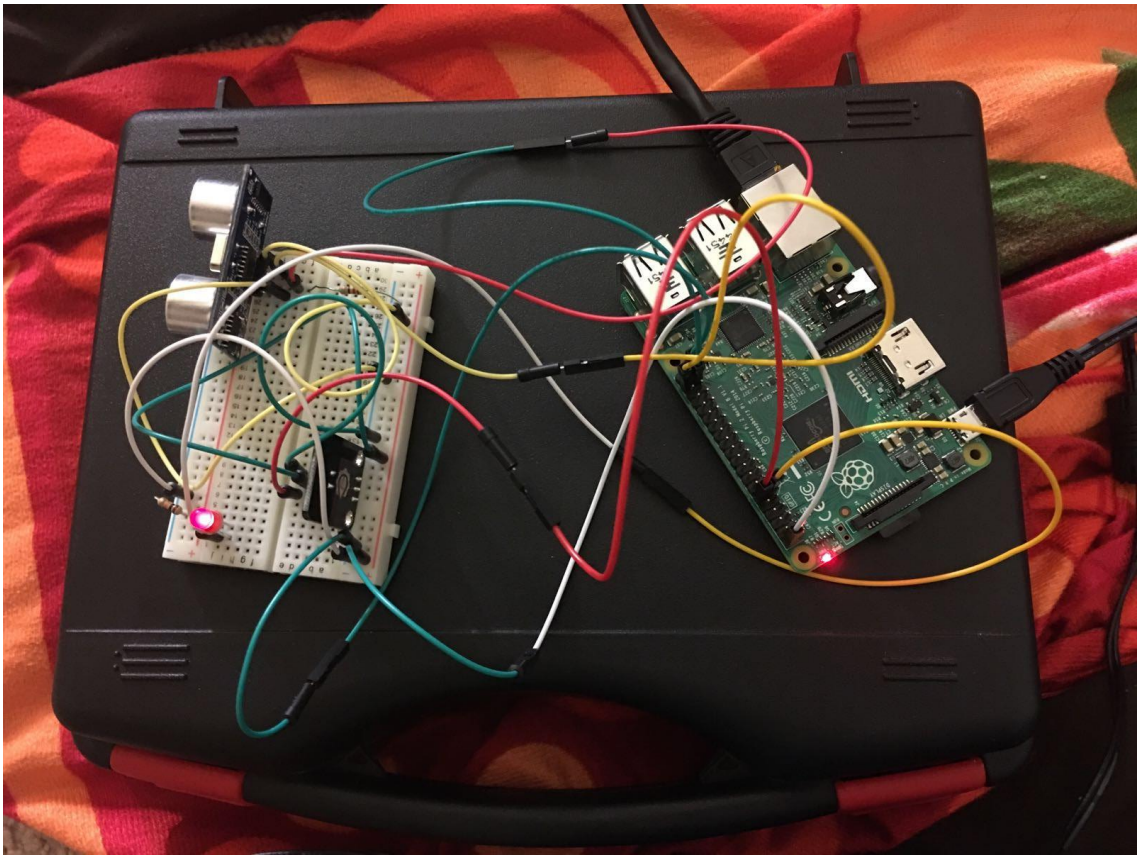

Fig 5.

Instructions for Circuit Connections

The pictures above show the circuit diagram for the Raspberry Pi. I used the GPIO BCM mode for the setting up the connection. Below are the steps for the circuit connection -

1. GPIO Pin 18 – Connection to Light Sensor.

2. GPIO Pin 20 – Connection to Proximity Sensor’s TRIG.

3. GPIO Pin 26 – Connection to Proximity Sensor’s ECHO.

4. GPIO Pin 17 – Status of the sensors (both), on/off.

The pictures above show how the connections are being made up with the above mentioned GPIO pins. Also, the connection uses a 5V input voltage.

Server IP & Ports

Following are the services deployed for the assignment -

1. Mosquitto Broker -

- ✓ IP – 52.33.59.166 (AWS Instance Elastic IP)
- ✓ Port – 1883
- ✓ Websockets – 9001, 9002
- ✓ Listeners - 52.33.59.166:1883 and Websocket - 52.33.59.166:9001

2. Kafka -

- ✓ Server IP: 52.33.59.166:9092
- ✓ Zookeeper IP: 52.33.59.166:2181

3. Elasticsearch -

- ✓ IP: 52.33.59.166:9200

4. Logstash -

- ✓ IP: 52.33.59.166:9300

5. Kibana -

- ✓ IP: 52.33.59.166:5601

6. WebApp -

- ✓ IP: 52.33.59.166:3000

How to Setup and execute the code

- **Raspberry Pi**

1. Connect the circuit as described in the Instructions section and as shown in the setup pictures.
2. Connect to Raspberry Pi via SSH.
3. Install the paho mqtt dependency using the command, ‘pip install paho-mqtt’ and memcache using the command, ‘pip install memcache’.
4. The Raspberry Pi code has two files, light_sensor.py and range_sensor.py. Run both the files using the command, ‘sudo nano python light_sensor.py’ and ‘sudo nano python range_sensor.py’ in two different terminals.
5. Make sure both the files are running.
6. The above run command makes the device up and running, which can be than controlled via Web App.

- **Server**

1. Install zookeeper and then install and start kafka server using the command - './kafka-server-start.sh ../config/server.properties'.
2. Provide the Mosquitto broker IP:Port, kafka server IP:Port and zookeeper IP:Port to the bridge and then run MQTT-Kafka Broker jar as 'java –jar MqttKafkaBridge.jar'.
3. Install and Start the Elasticsearch using the command - './bin/elasticsearch'.
4. Install and Start Logstash using the command - './logstash agent –f ../logstash.conf'. logstash.conf file has the required filters, input and output setup.
5. Install and Start Kibana using the command - './bin/kibana'.

- **Web App**

1. Make sure you have the npm package manager installed.
2. Install the Serve dependency (directory server built with connect), via the command, 'npm install –g serve'.
3. Open the folder with the web app code, and make sure you could see the index.html, bower JSON file, bower components and the project code inside that directory.
4. Navigate to the folder containing WebApp code.
5. Just run Serve now with the command, 'serve' and the app is up and running on 'http:// 52.33.59.166:3000'.

Screenshots of the Web application

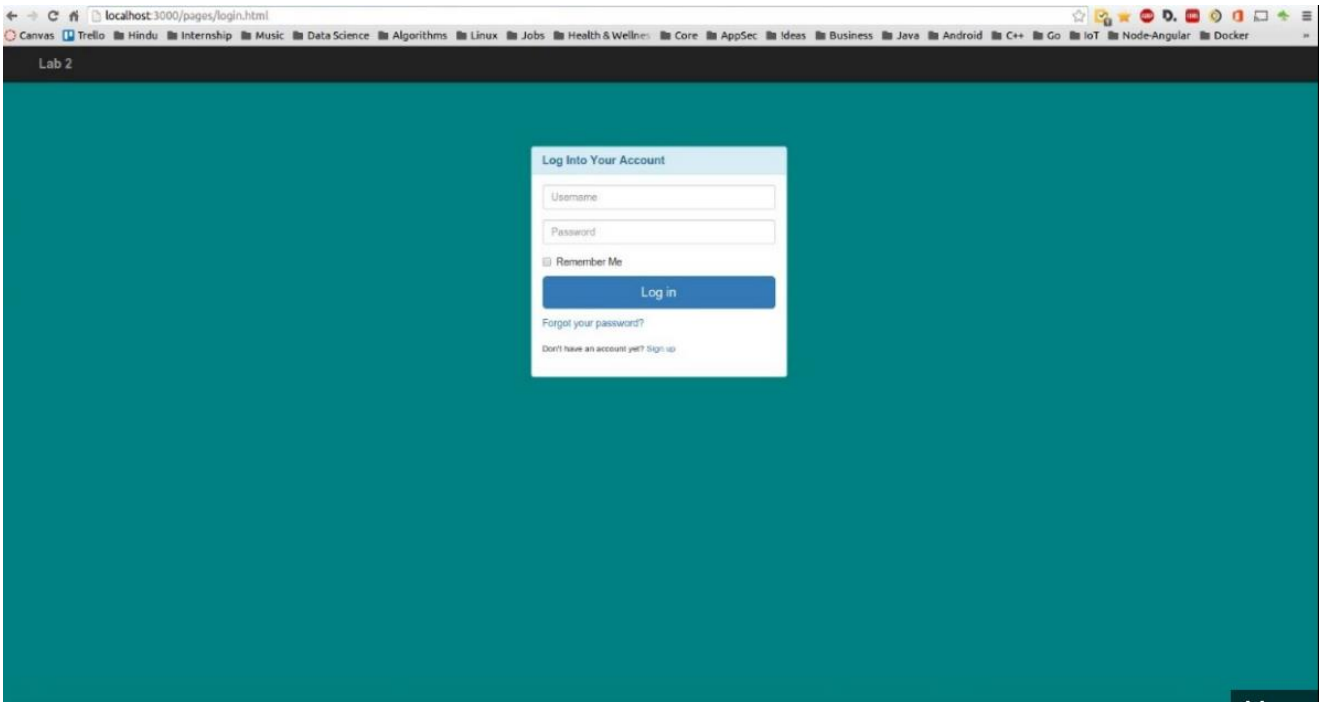

Fig 6. Login Page

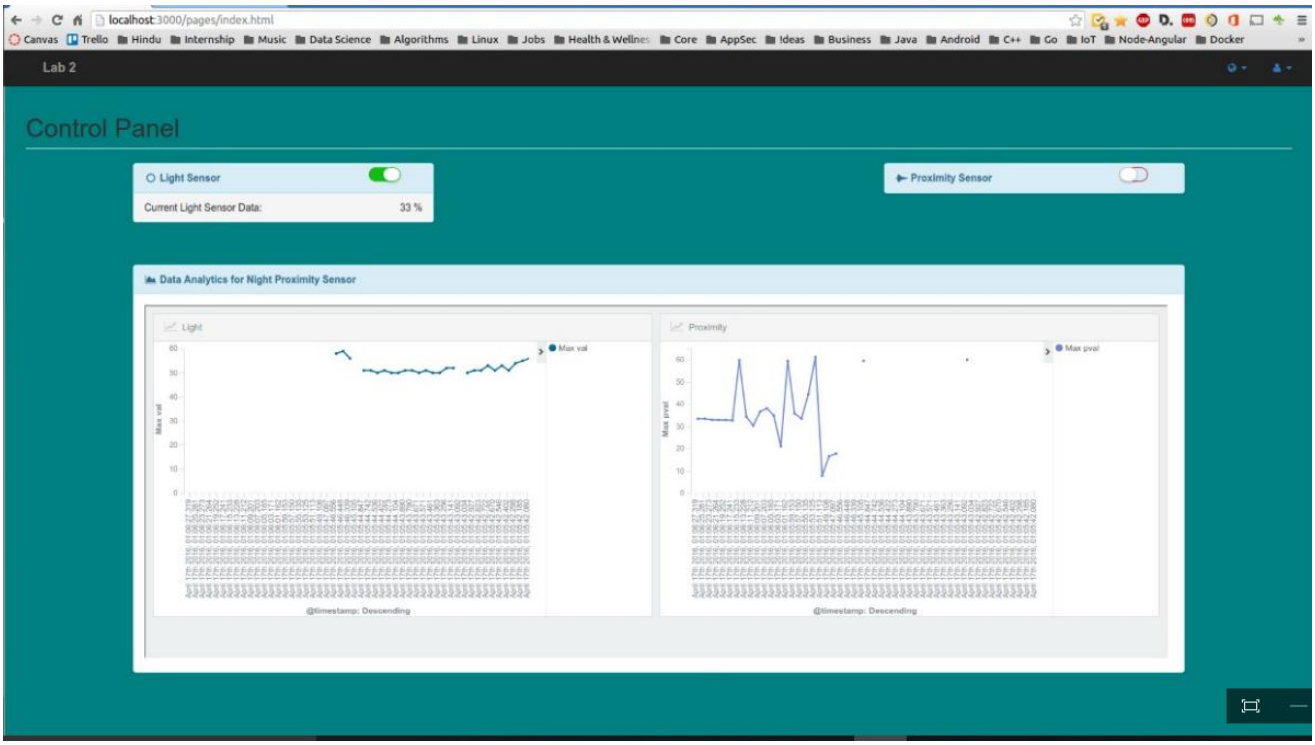

Fig 7. Home Page (Light Sensor On and Proximity Sensor Off)

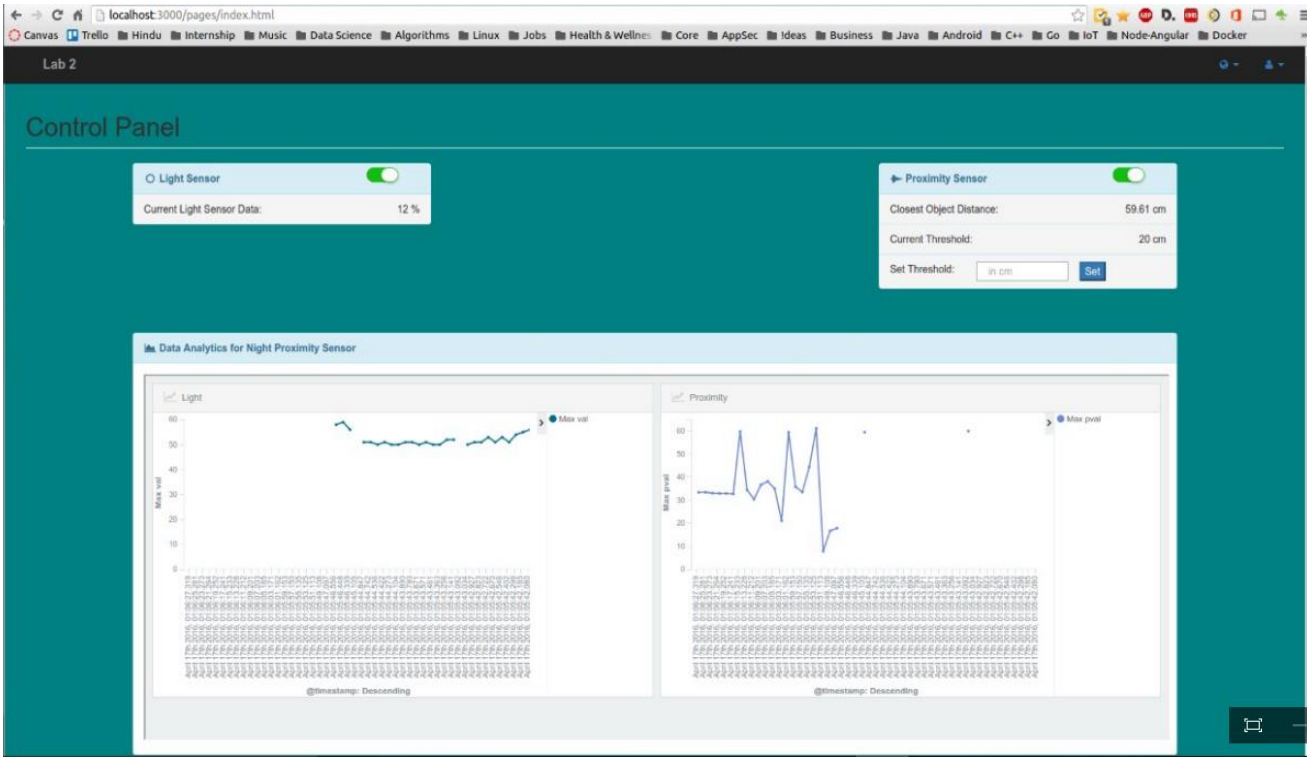

Fig 8. Home Page (Light and Proximity Sensors On)

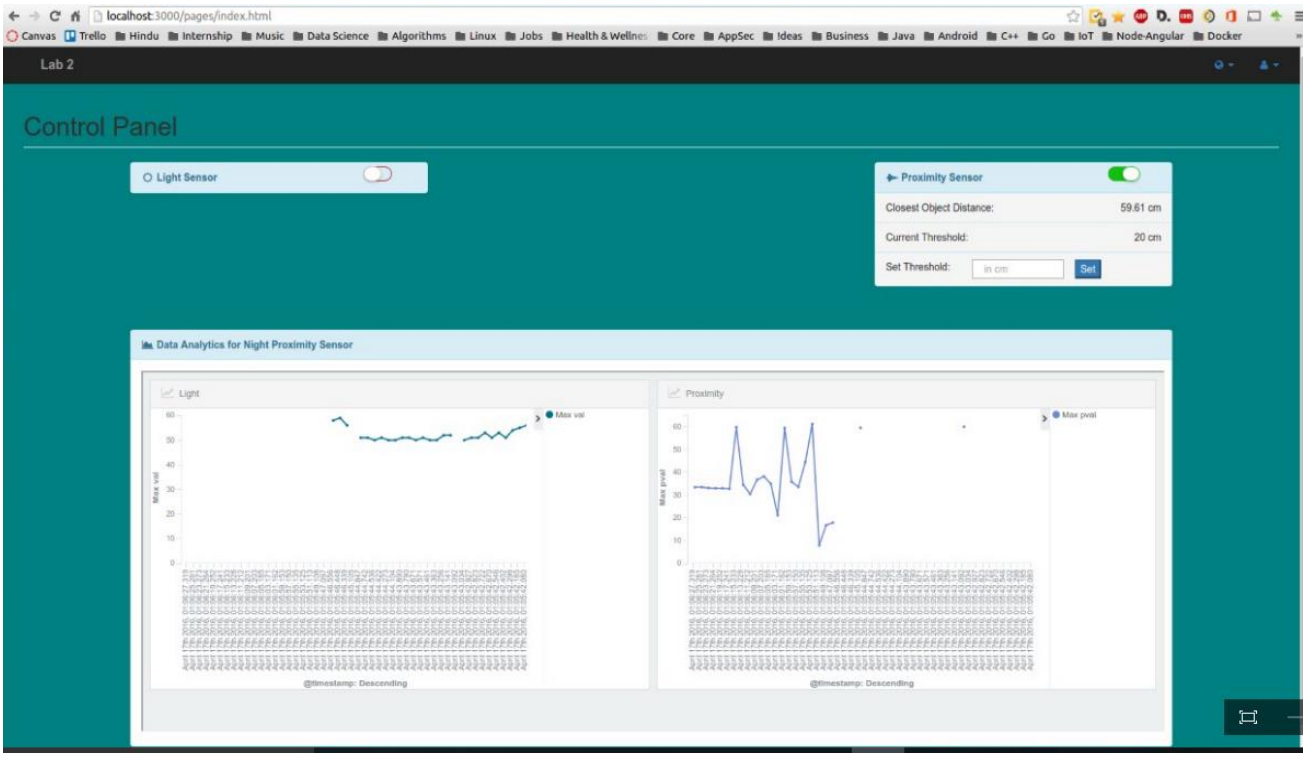

Fig 9. Home Page (Light Sensor Off and Proximity Sensor On)

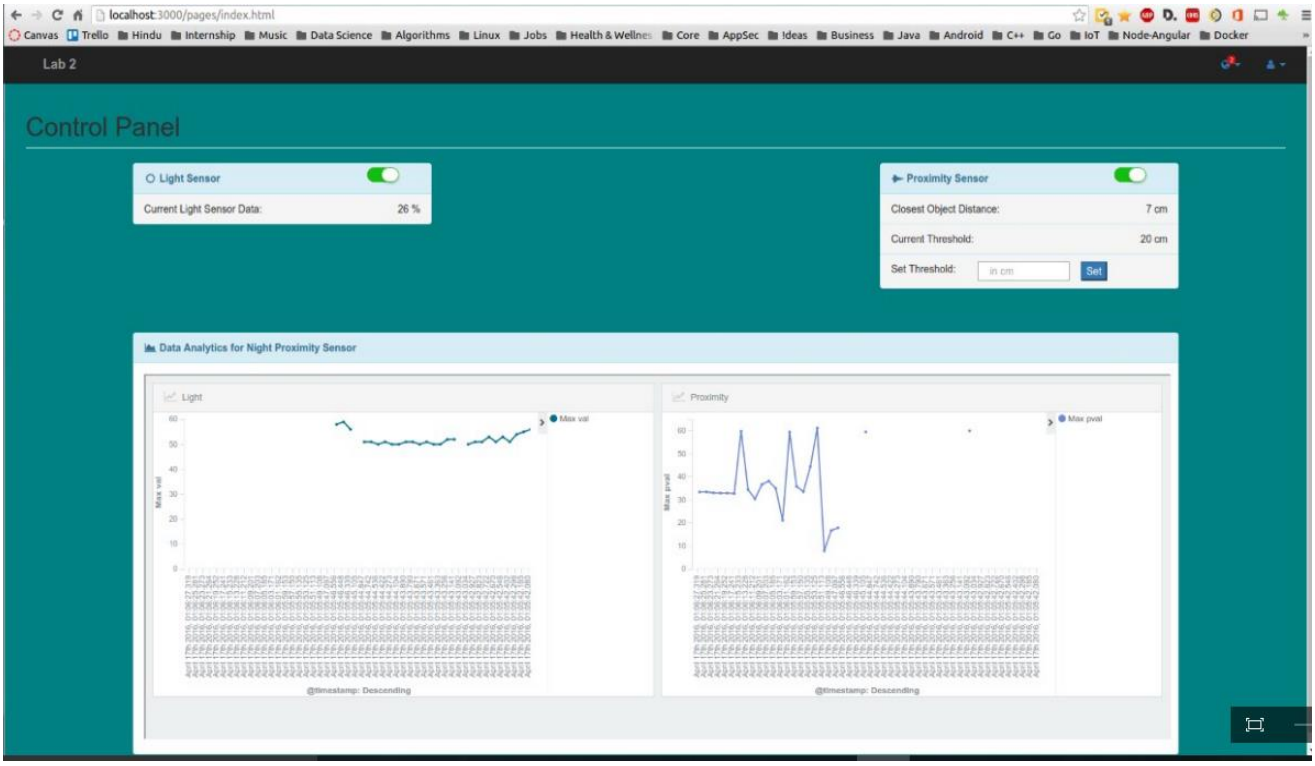

Fig 10. Home Page (Both sensors on with Notifications count)

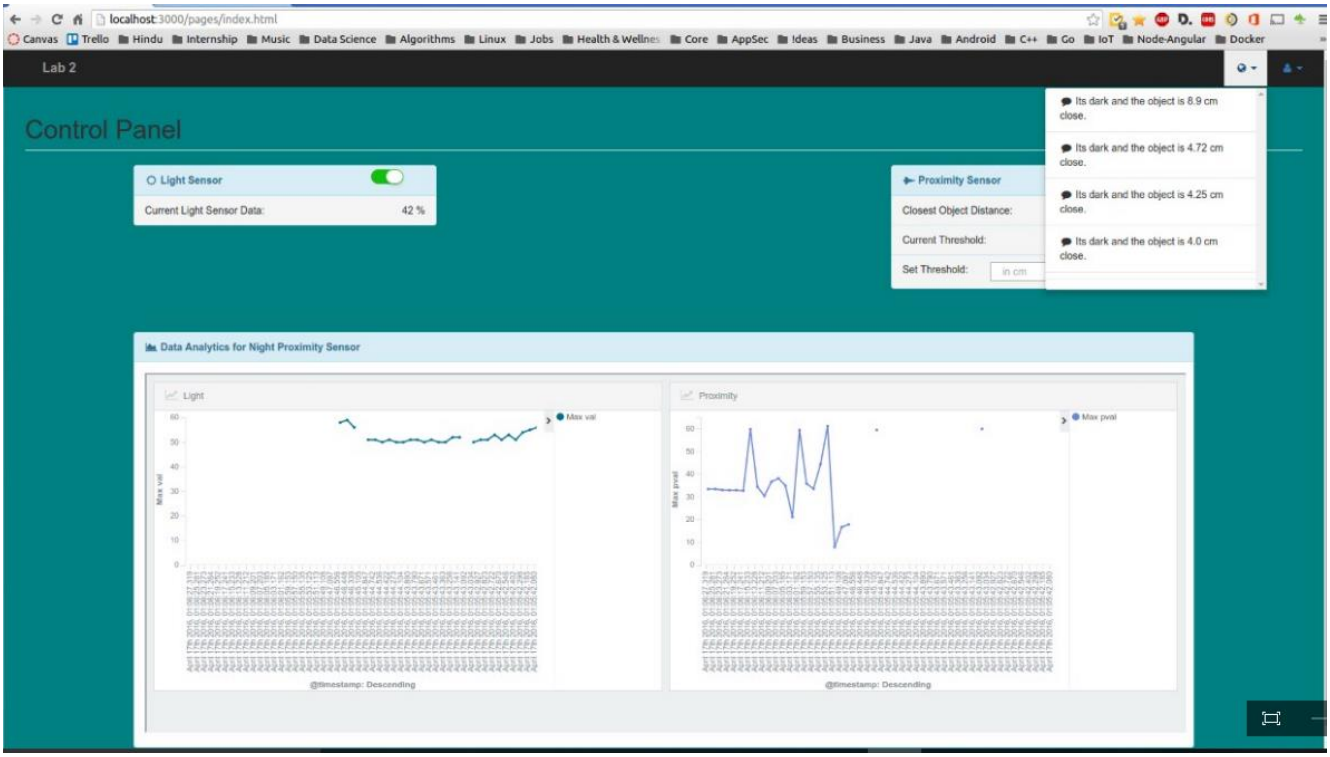

Fig 11. Home Page (Both sensors on and Notification Panel, showing the notifications with the object distance)

Part 2: Answer the following questions (5 pts):

Q1. State Pros and Cons of using Kafka in your IoT infrastructure.

Ans. Apache Kafka is an open-source message broker and provides a unified, high-throughput, low-latency platform for handling real-time data feeds. Below are the Pros and Cons of using Kafka in my IoT infrastructure –

Pros of Using Kafka -

1. The most important reason of using Apache Kafka is high-throughput, so that the infrastructure could support real-time data feed.

2. Also, Kafka provides a fault tolerant stream of data to the Logstash consumer. Now, Mosquitto isn't scalable so, when the velocity of data increases, Kafka can accommodate multiple mosquitto broker streams.

3. Another reason for using Kafka in the current IoT infrastructure is that Logstash doesn't support MQTT Protocol currently, so we used Kafka which converts the MQTT to HTTP.

Cons of Using Kafka -

1. Kafka's server utilizes large memory footprint on the server which makes resource utilization very high.

2. Also, as Kafka doesn't support MQTT by default, an additional Mosquitto-Kafka bridge is required to provide input from Mosquitto to Kafka which utilizes additional resources.

3. Kafka somewhere also acts as a bottleneck on non HA configuration, as if Kafka fails than the whole pipeline will fail.

Q2. State reasons why you would not use Apache Flume in your IoT Infrastructure. Discuss situations where you can use Flume in an IoT Infrastructure.

Ans. In the current IoT Infrastructure, the reasons for not using Flume can be -

1. Apache Flume doesn't provide velocity control to the consumers as it is a PUSH based model, so it pushes data to the Sink (Targeted Storage). Thus when the velocity of data is high, the consumers may drop some packets. On the other hand, Kafka is a good option as consumers can PULL data of it.

2. Next, Apache Flume does not provide broker overflowing while Apache Kafka provides it by storing the data on the disk.

3. Also, Flume does not provide replication of the data/events. So, if any of the node fails then we might lose all the events.

The situations where Apache Flume can be used in an IoT Infrastructure can be -

1. If Mosquitto can directly publish the data into Apache Flume, so that the data is stored inside HDFS.

2. In circumstances where high velocity data can be stored in Sinks like Cassandra or HDFS and further batch processing can be done on it using Hadoop.

3. Flume has the ability to use multiple sink to store data, so we can use Kafka as one of the sink for Flume and get the work done.