San Jose State University

Software Engineering Department

Spring 2016

CMPE 296A-IoT

Class Project 1

Project Report

Submitted To:

Prof. Sang Shim

Submitted By:

Group 4

Dhruv Kalaria
Edward Hwang
James Cho
Naghma Anwar
Vivek Maheshwari

Table of Contents

1. Motivation

Save Money:

The average household spends more than $2000 a year on energy bills. Heating and cooling costs generally make up 40% to 50% of an average home's overall energy costs. With old technology, part of that cost is actually due to unnecessary waste of energy usage. Using a smart thermostat would greatly benefit the homeowner by reducing the energy cost. The homeowner would be able to save on energy costs and cut the bill. Energy Star says that a person can reduce the billing by $180 a year by simply properly setting the thermostat.

Save Energy:

This is complimentary with saving money. The smart thermostat ensures that the home remains at the temperature set by the homeowner when needed and removes any unnecessary HVAC usage. Also if the homeowner is not at home, then the thermostat will not run the heating and cooling units. This removes any wasted energy usage and saves money on the heating and cooling costs.

Provide Home Comfort:

The smart thermostat strives to provide the best possible living condition for the homeowner by ensuring that the home temperature is optimal per the homeowner's desire. An average thermostat might not be able to provide the same comfort because it requires the homeowner to constantly update it in order to match what they like. The smart thermostat can more accurately measure the indoor temperature and more efficiently raise and lower the temperature to the homeowner's optimal level. By being "smart", the thermostat can learn to adapt the homeowner's tendencies using and analyzing past data.

Easy Access:

Another large draw of smart thermostat is remote and easy access to the controller from a web app. This web app can be accessed anywhere using a mobile device and internet. Mobility allows the homeowner to make any changes and monitor the thermostat without having to always be present in the house and in front of the device. The homeowner can turn on and off the thermostat as well set the temperature from anywhere. At the same time any changes can be tracked from the web app. No matter where you are, you can just as easily cut heating and cooling costs.

2. Introduction

The Nest Learning Thermostat is an electronic, programmable, and self-learning Wi-Fi-enabled thermostat that optimizes heating and cooling of homes and businesses to conserve energy. It is based on a machine learning algorithm: for the first weeks users have to regulate the thermostat in order to provide the reference data set. One of its key feature is that the thermostat is connected to Wi-Fi and it can be controlled from your phone, tablet or laptop.

We as a team have built a working prototype that includes many of the components with Nest eco system. **THSense** is a learning thermostat which collects the Temperature and Humidity Sensing data from Arduino and sends it via RF to the Gateway (Raspberry Pi). We further perform Data Analytics on the real-time data collected in the Cloud using the ELK Stack. Also, one can control and monitor the registered thermostats (devices) remotely via the web application.

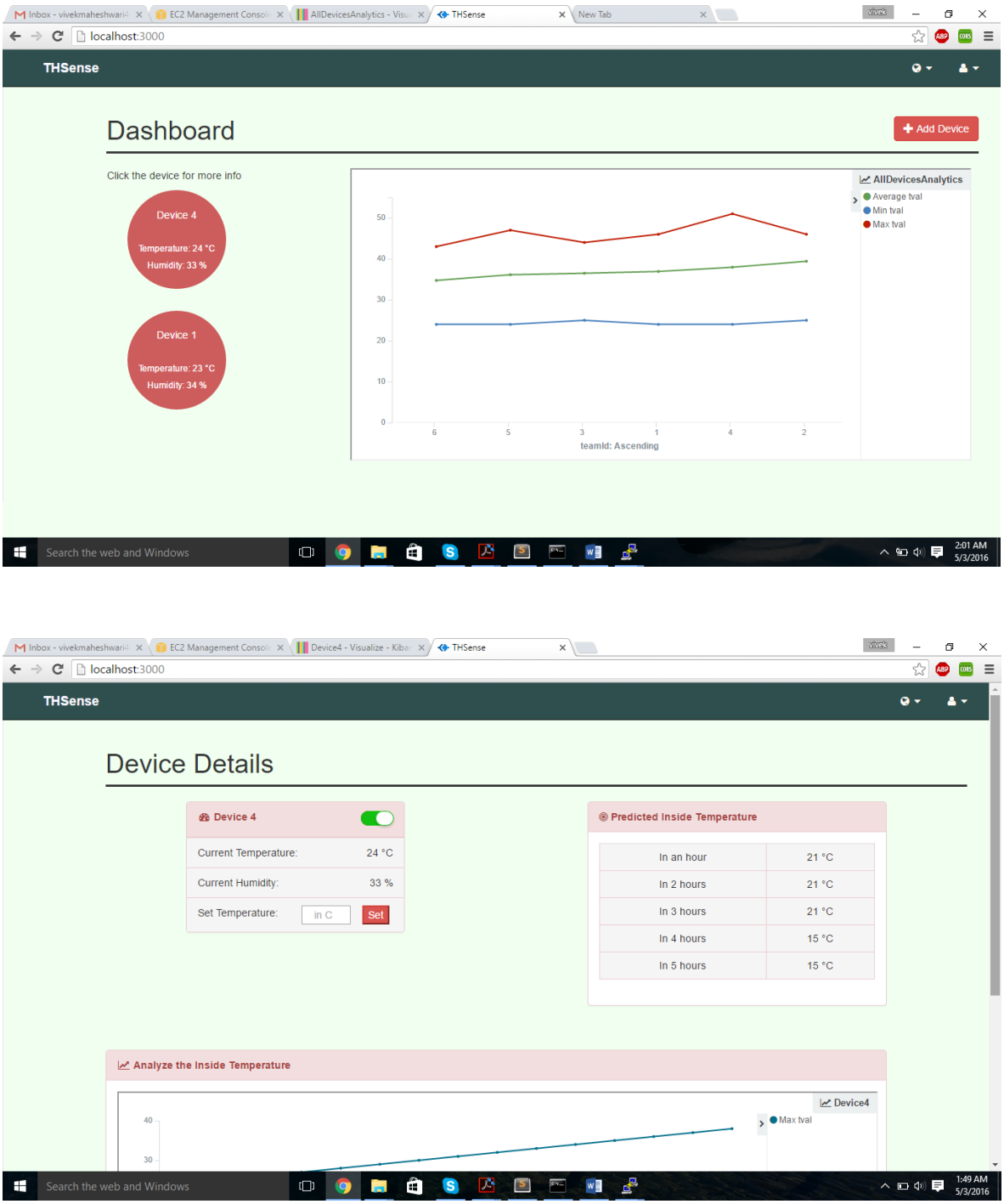


Figure 1. ThSense Web App

3. High Level Design Architecture

The figure below (Figure 2) depicts the typical project architecture and flow of the THSense thermostat.

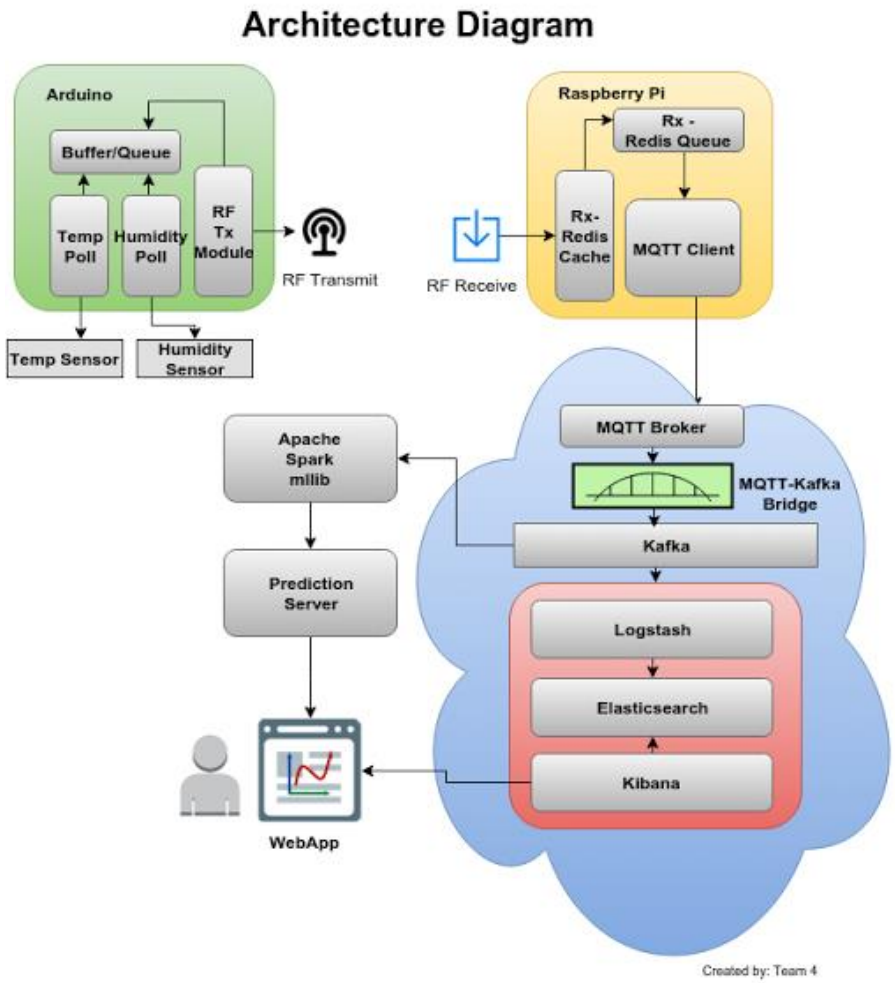


Figure 2. High Level Design Architecture

Project Architecture Description:

The above figure (Figure 2) depicts the project architecture of our project. It can be briefly described as -

Arduino – Thermostat Device:

1. The DHT11 sensor is used to collect the temperature and humidity data. It has a LCD display to show the current temperature and humidity.

2. Transmitter module is used to transmit the data via RF to the Gateway (Raspberry Pi). Transmitter should send data (5 times) only if there is any change in the current humidity or temperature.

Raspberry Pi - Gateway:

1. The gateway (Raspberry Pi) has a RF Receiver module which will receive the data from the thermostat devices and continuously put it into a Redis queue. It should omit repeated messages if same message is received more than once by maintaining a cache of previous 15 data.

2. The Paho MQTT client will publish the data to the relevant topic whenever the data is available, to the MQTT broker on the cloud.

Cloud:

1. MQTT broker will get the data and publish to the Kafka Publisher.

2. We have used Kafka to scale the MQTT Broker. The design is flexible to connect any number of MQTT brokers to Kafka.

3. An MQTT bridge is required to publish the data from MQTT broker to Kafka. A bridge is a small program which will subscribe to topics by MQTT broker, extract the data and publish to Kafka.

4. Input, Output and Filters to Elastic search is defined in Logstash. It will limit the data which is going into ES.

Web App:

1. It has Google OAuth Authentication for login and sign up purposes.

2. It shows the dashboard with all the registered devices for the user and analytics for those devices.

3. The web app gives device control features and also shows the predicted inside temperature for the next 5 hours based on the ML algorithm using Spark.

4. We have used Angular JS to build the web app and Serve (a Directory Server) to host the application.

4. Technology Stack

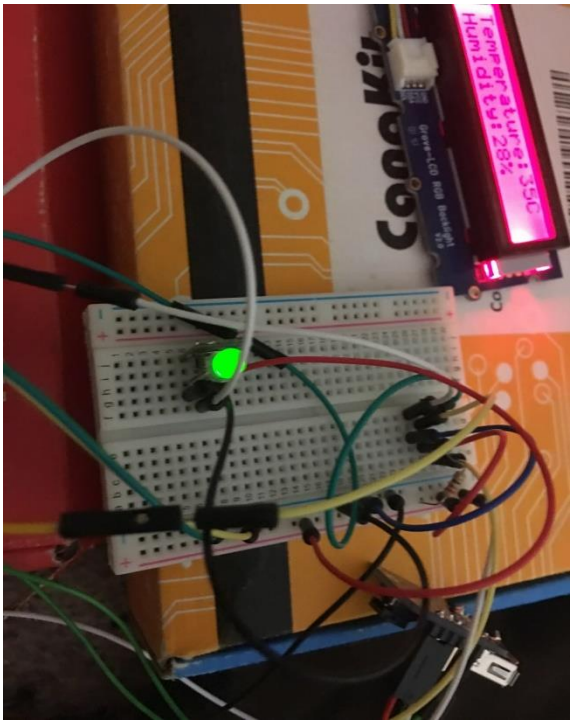
Hardware

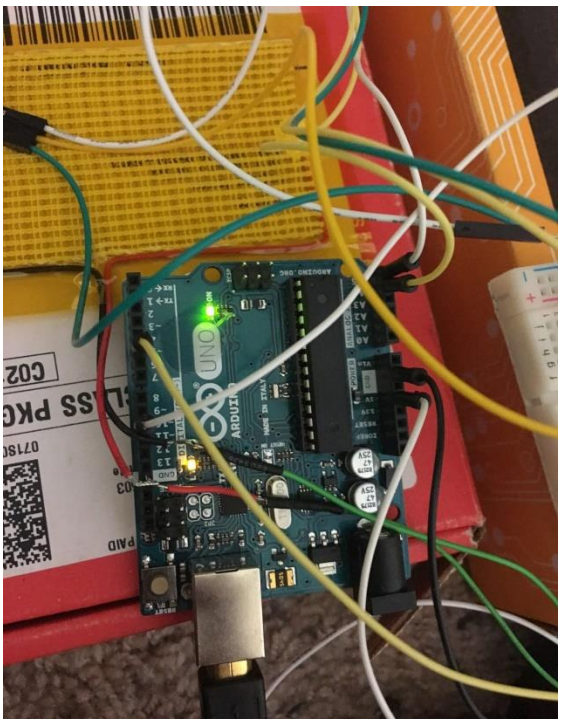
- ✓ Arduino Uno
- ✓ Raspberry Pi
- ✓ RF Transmitter and Receiver
- ✓ DHT Sensor
- ✓ Heating Pad
- ✓ LCD Display

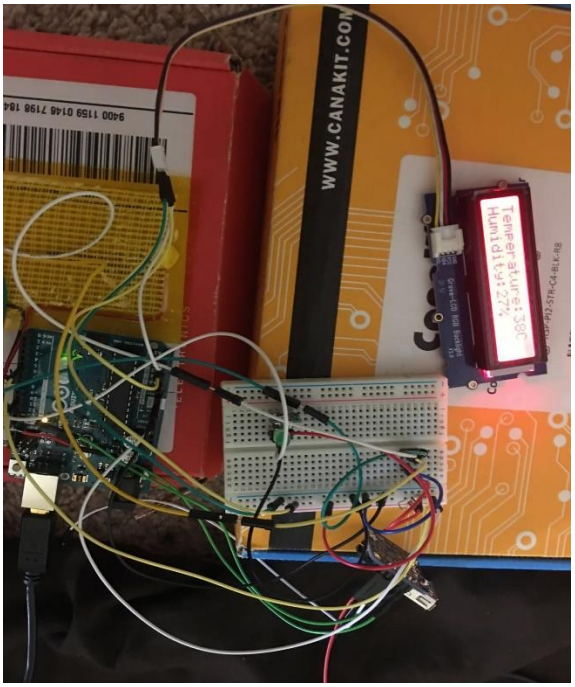
Software/Cloud

- ✓ Redis
- ✓ Mosquitto Broker
- ✓ Apache Kafka
- ✓ ELK Stack
- ✓ Apache Spark
- ✓ Web App using AngularJS

5. Pictures of Device and Gateway Connection/Setup







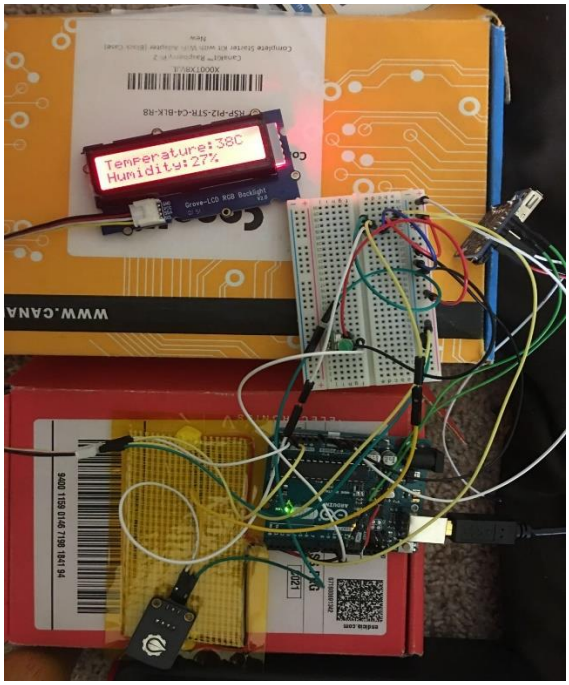


Figure 3. Device Connections

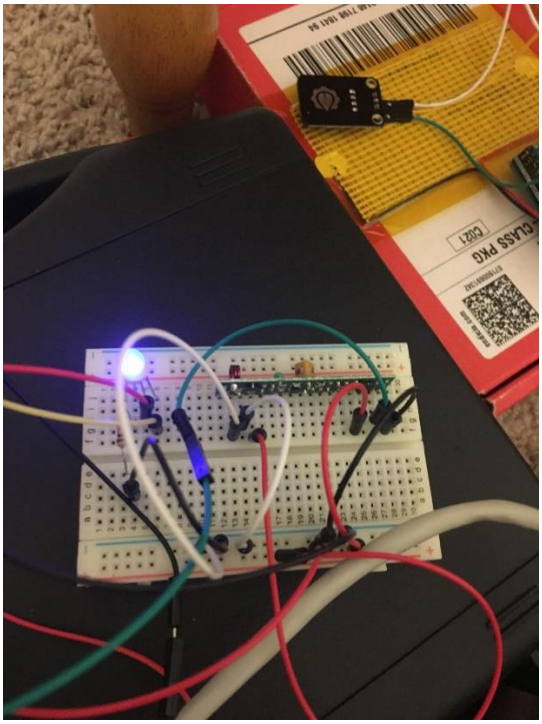


Figure 4. Gateway Connections

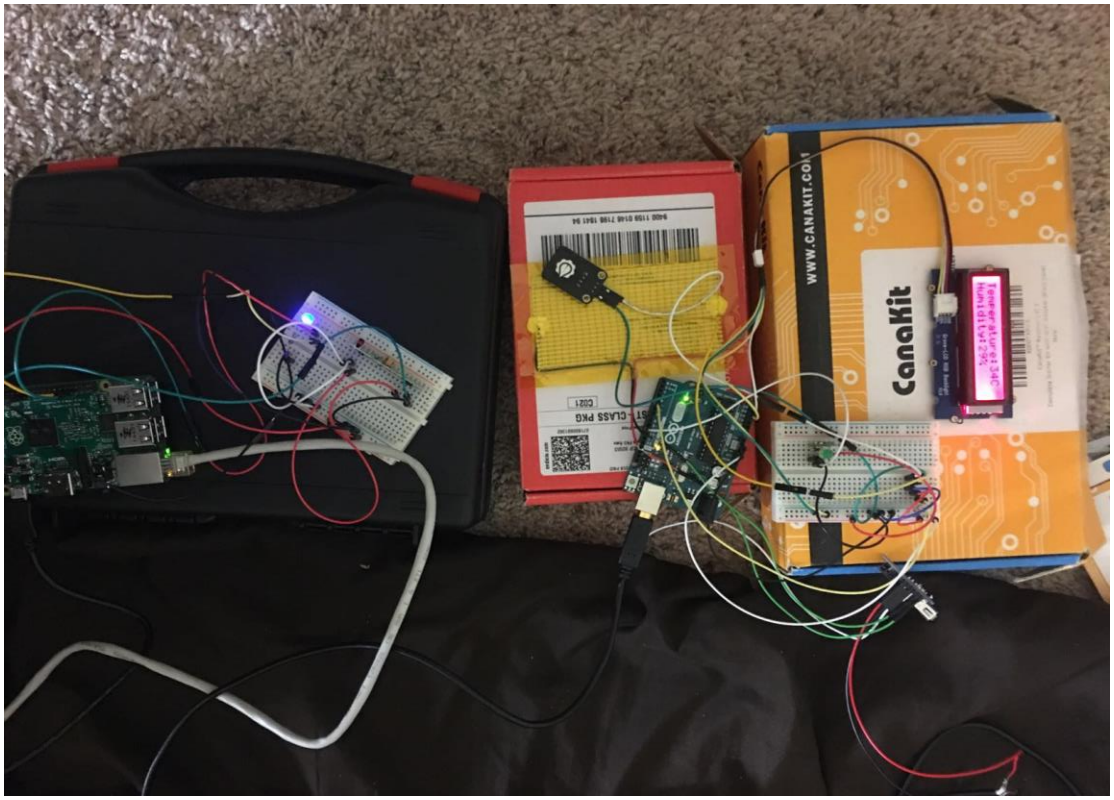


Figure 5. Entire Setup (with Heat pad and LCD)

6. Instructions for Circuit Connections

Below are the steps for the circuit connections -

Arduino – Thermostat Device

1. Pin 5 – Connection to DHT11 Sensor.
2. Pin 10 – Connection to RF Transmitter.
3. Connect the LCD Display as shown in the images above.

Raspberry Pi - Gateway

1. GPIO Pin 2 - RF Receiver Codes Receive
2. GPIO Pin 6 - HVAC/Device On (Green) using RGB LED
3. GPIO Pin 5 - HVAC/Device Off (Blue) using RGB LED

7. Brief Description about Device and Gateway

Arduino - Thermostat

- ✓ The DHT11 sensor is used to collect the temperature and humidity values.
- ✓ The LCD Display shows the current temperature and current humidity.
- ✓ Connect an RF Transmitter to send the data to the gateway.
- ✓ Using the RCSwitch library to transmit data using RF Transmitter.
- ✓ The RF Transmitter sends data only if there is any change in the current humidity or temperature.
- ✓ It Send the same message at least 5 times to guarantee that the data is received by the gateway.
- ✓ Appropriate checks on the data interface have been done on the sensor transmitter side.

Raspberry Pi - Gateway

- ✓ The gateway receives the data from multiple devices.
- ✓ Connect an RF Transmitter to receive the data from the devices.
- ✓ Using the RCSwitchReceiver library to receive the data.
- ✓ The gateway omits the repeated messages, if the same message is received more than once.
- ✓ To make sure that while receiving data from multiple devices, the actual data doesn't gets lost, we used **Redis as a Cache** which stores 15 entries of the data it is receiving.

- ✓ The cache pushes this data to a queue than, which helps make sure that no data is lost in the process. We used **Redis as a Queue** for this purpose.
- ✓ Further, the Mosquitto client will get the data from queue and publish to the relevant topic to the MQTT broker on the cloud.

8. Machine learning using Spark

8.1 Dataset - Capture Temperature (with timestamps)

The DHT11 sensor is used to measure the temperature and humidity inside the home every 30 seconds. We collected the following data fields to build our required dataset –

1. Date and Time – Timestamp
2. Outside Temperature (in °C) – Temperature outside the home, using Open Weather API.
3. Outside Humidity (in %) – Humidity outside the home, using Open Weather API.
4. Inside Temperature (in °C) – Temperature inside the home.
5. Inside Humidity (in %) – Humidity inside the home.

- ✓ The above mentioned data was collected from 04/22/2016 22:42 to 04/24/2016 12:35 and was stored onto a csv file.

- ✓ Since we didn't had much of data to collect and we have to achieve better accuracy, we divided the complete dataset into 4 different dataset based on the timestamp (hours) i.e.

  - 0:00 – 5:59 (Dataset 1)
  - 6:00 – 11:59 (Dataset 2)
  - 12:00 – 17:59 (Dataset 3)
  - 18:00 – 23:59 (Dataset 4)

8.2 Algorithm to build the Home Thermal Model

Linear Regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X.

- ➢ **For Modelling Temperature**

  - ✓ Independent Variable - Outside Temperature
  - ✓ Dependent Variable - Inside Temperature
  - ✓ Vector Form - $y = X\beta + \varepsilon$, where
    β = parameter vector or regression coefficient.
    ε = error term.

Why Linear Regression?

If the goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.

Thus, it well suits to our requirement and performs good enough to predict the temperature inside the home, given the outside temperature and humidity values.

8.3 Home Thermal Model and Prediction Component

As discussed, the dataset is divided into 4 different datasets, hence there are 4 different Temperature models evaluated for the specific datasets.

Although the home thermal model predicts the future temperatures inside a home based on the weather, current state of the home, and HVAC state, for our case, the inside temperature depends upon following -

- ✓ Weather (Outside Temperature)
- ✓ Time, in slots of 6 hours (discussed above)

The table below (Table 1) shows the accuracy of the Temperature Models.

S.No.	Model	Mean Square Error for Temperature
1.	Model 1	0.74
2.	Model 2	8.89
3.	Model 3	0.79
4.	Model 4	5.96

Table 1. Accuracy of the Application

Prediction Component

The table below (Table 2) shows the predicted inside temperature and inside humidity based on the given input and shows the model selected for the prediction -

S.No.	Input Time (YYYY-MM-DD HH:MM:SS+00)	Selected Model	Forecast Outside	Predicted Inside
			Temperature (in °C)	
1.	2016-04-27 03:00:00+00	Model 1	11.95	21.82
2.	2016-04-27 09:00:00+00	Model 2	9.27	15.02
3.	2016-04-28 15:00:00+00	Model 3	9.97	11.64
4.	2016-04-29 21:00:00+00	Model 4	22.81	34.63

Table 2. Prediction Component based on the given Input

8.4 How to run the source code for Home Thermal Model

The source code zip contains the code for Home Thermal Model. It has following python files –

- ✓ linear_reg_temperature_model - Evaluating the Temperature Model on training data.
- ✓ prediction_component - Component to predict temperature and humidity.

Other than these files, the folder contains 4 data log files which represents 4 different datasets based on the suggested approach.

Build the Temperature Model

Navigate to the folder directory and run the following commands to build the model -

1. python linear_reg_temperature_model.py

These commands on running successfully creates 4 models each for temperature and humidity component. The folder names are -

- model# - Temperature Models, where # = 1, 2, 3, 4.

Prediction Component

- ✓ The file "prediction_temperature.py" contains the business service for prediction of the inside temperature based on the model build above.

- ✓ **Input** - The input to the component can be given in this file. Although, the input time format is YYYY-MM-DD HH:MM:SS+00.

- ✓ **Predict** - The service can be used to predict the temperature for the next 5 hours from the current time.

- ✓ **REST API** – The predicted values are available using the following REST API –
    - http://<ip>:<port>/gettemp

9. How to Setup and execute the code

- ➢ **Arduino - Device**

1. Connect the circuit as described in the Instructions section and as shown in the setup pictures.
2. Include the “dht.h” file to enable the DHT11 sensor to collect data.
3. Include the RCSwitch, Wire and RGB_LCD libraries as well.
4. Upload the file “DHT11” onto the Arduino and it starts transmitting the data.

- ➢ **Raspberry Pi - Gateway**

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 also install the Pi switch, Redis Queue and Redis Cache.
4. The Raspberry Pi code has two files, receiver.py which enables the RF Receiver and thermostat.py which pushes the data to cloud. Run both the files using the command, ‘sudo python receiver.py’ and ‘sudo python thermostat.py’.
5. Make sure both the files are running.

- ➢ **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'.

Server IP and 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

7. Spark -

- ✓ IP: 50.18.94.136:9999

10. Screenshots of Web Application

- **Login Page**

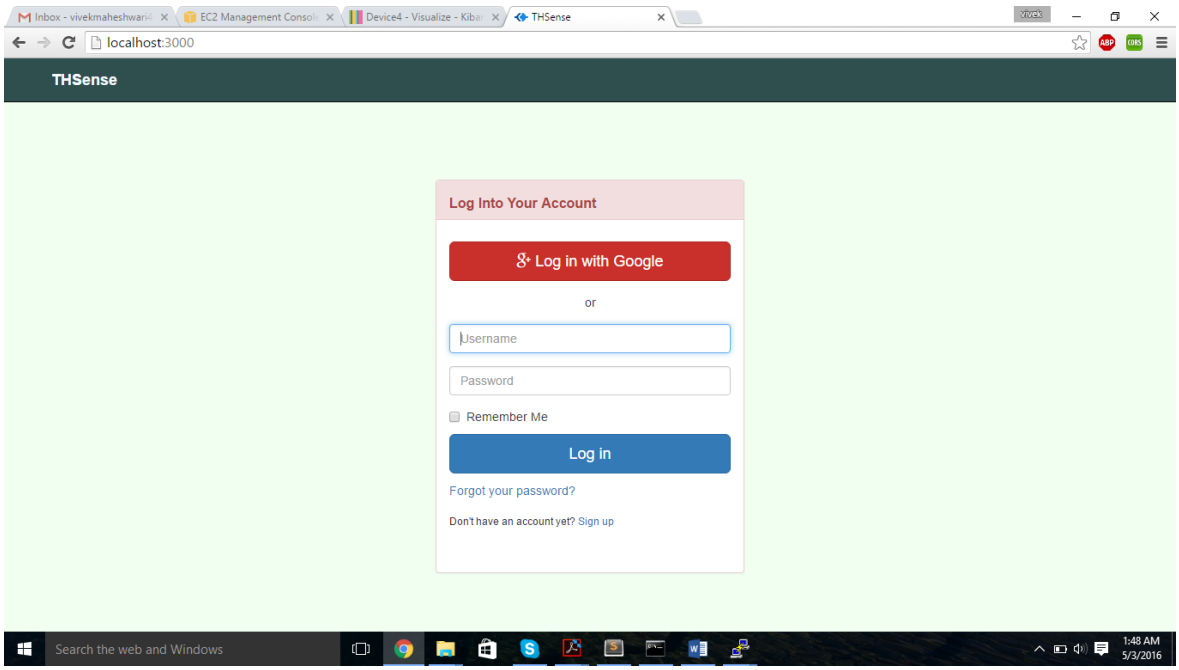


Figure 6. Login Page – THSense App

- **Dashboard**

Figure 7. Dashboard – THSense App

➢ Add Device

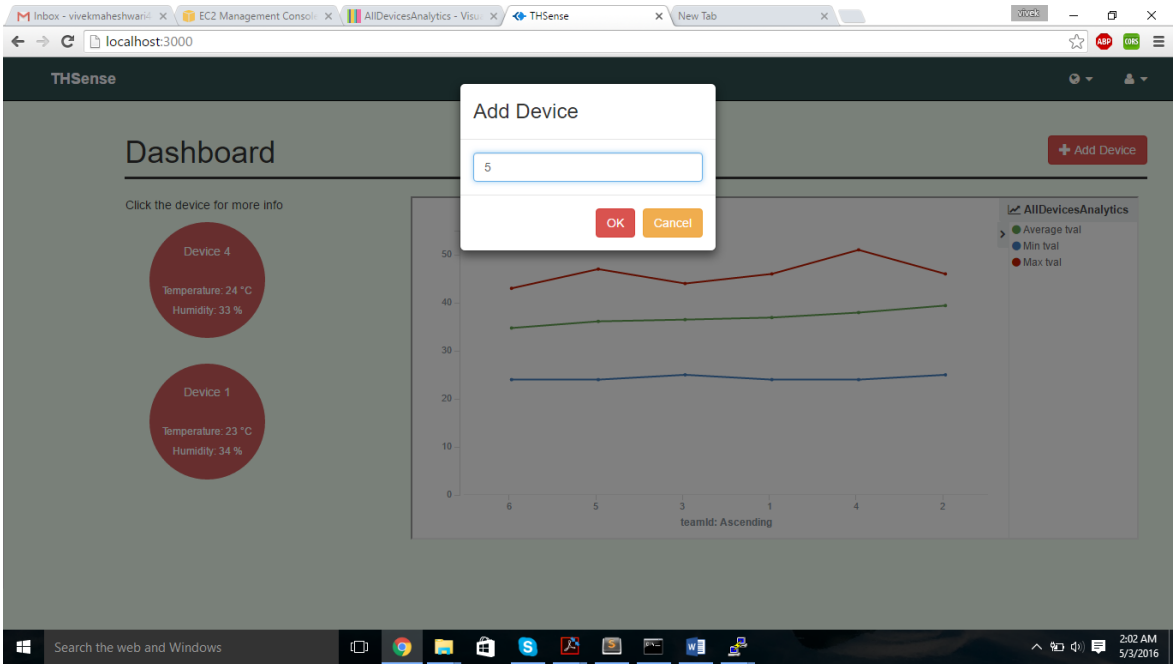


Figure 8. Add Device – THSense App

Figure 9. Device Added to Dashboard – THSense App

➢ Device Details

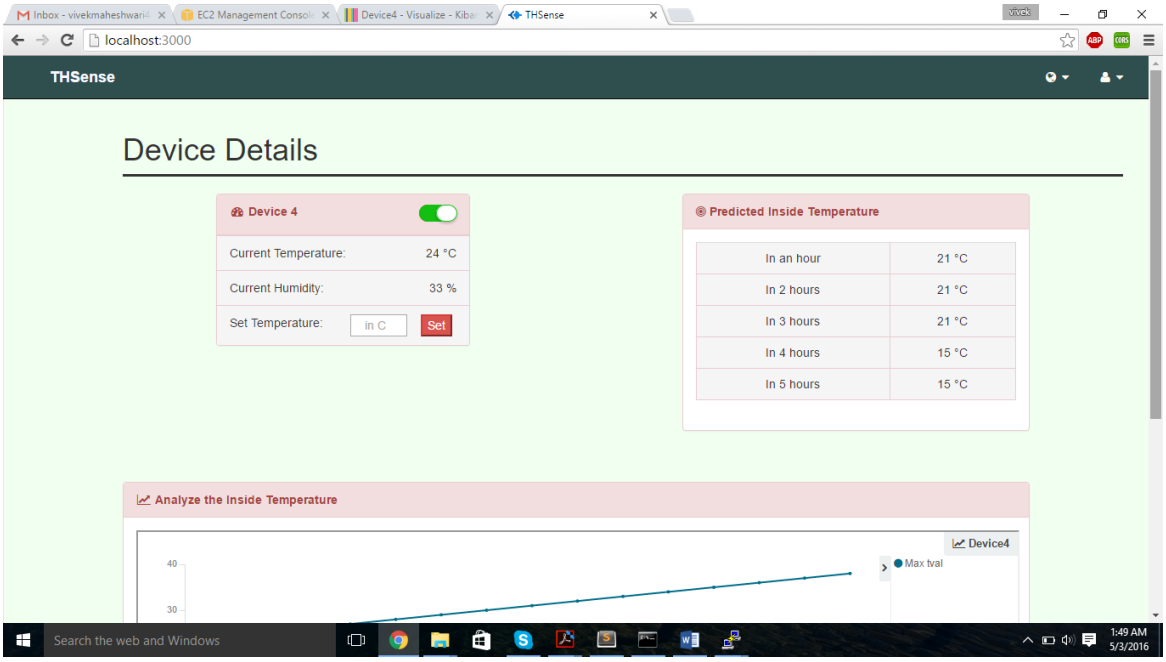


Figure 10. Device Details – THSense App

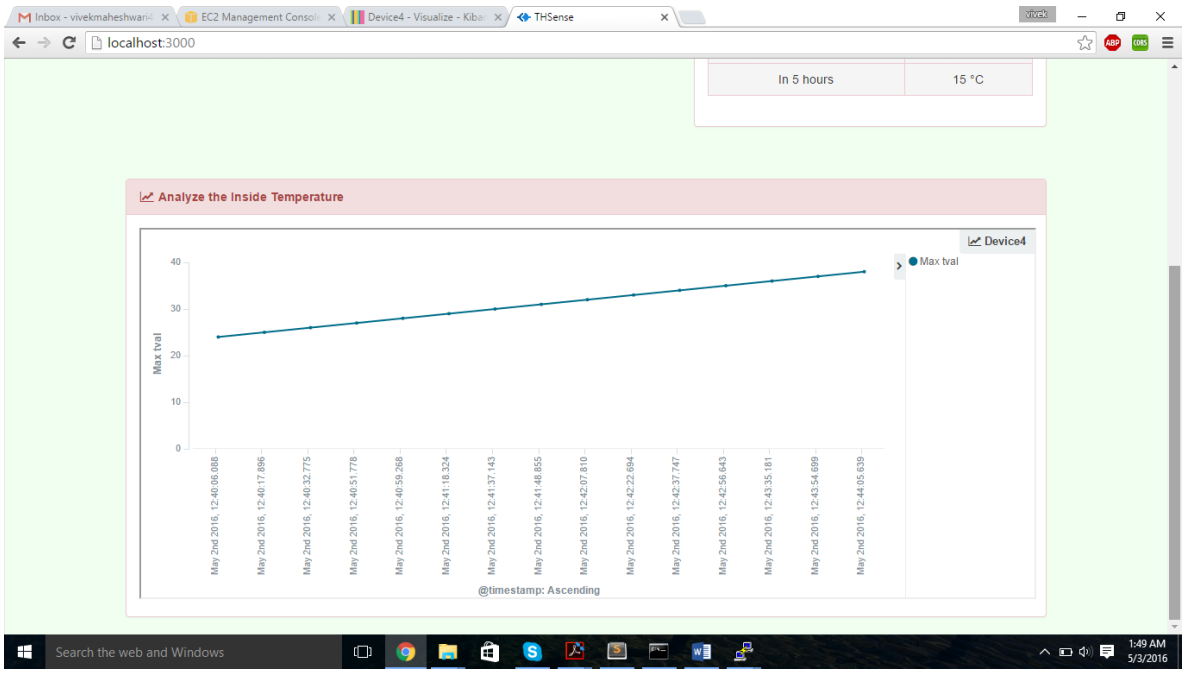


Figure 11. Device Details (Temperature Visualization) – THSense App

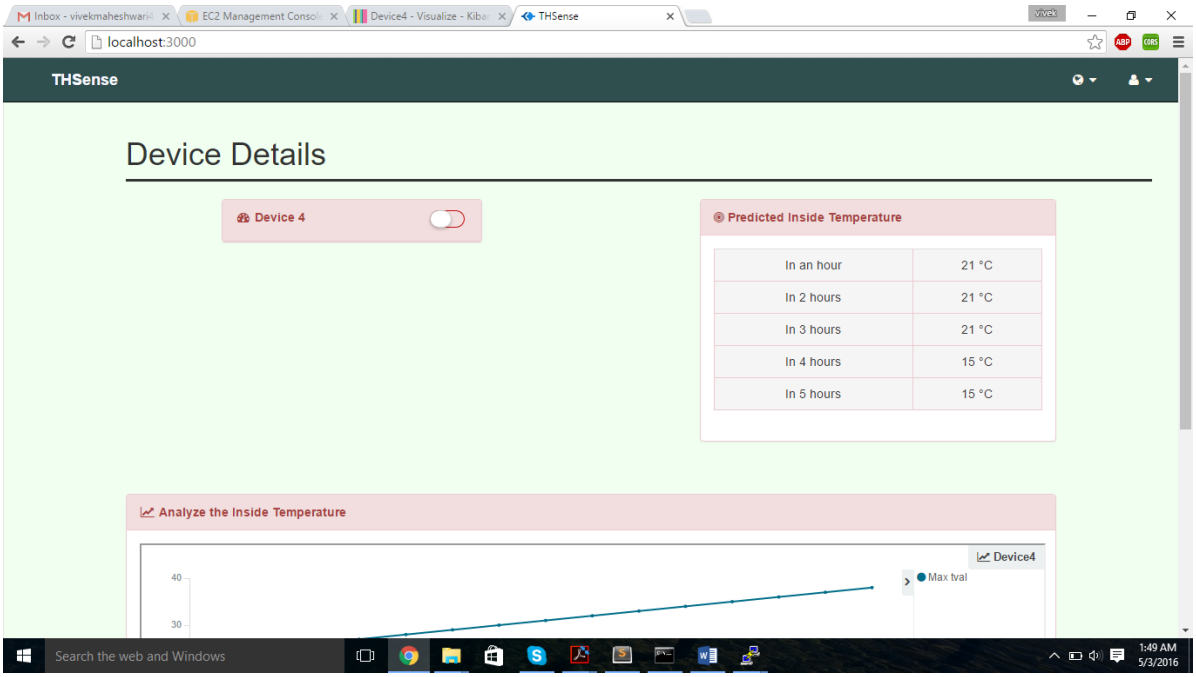


Figure 12. Device Details (Device Turned Off) – THSense App

11. Lessons Learned and Possible Future Work

Lessons Learned:

We implemented the Nest learning thermostat build to collect the temperature and humidity at the minimum. The thermostat is connected to cloud, so that certain data analysis and temperature prediction component can be applied on it using the ELK Stack and Spark. We tried to utilize all the concepts taught to us in the class by the professor.

Possible Future Work:

- ✓ Control the HVAC System based on the thermostat and the Home Thermal Model.
- ✓ Make the thermostat device more intelligent by implementing more sophisticated ML Algorithms.
- ✓ Add more features like Auto-Away, Time-to-temperature, Early-On and more.
- ✓ Integrate the device with more IoT products like Smart Cam or Nest Protect.
- ✓ Provide mobile notifications like Safety alerts or filter reminders to the user.
- ✓ Having built a handy tablet application for convenience of users.