SANTA CLARA UNIVERSITY
---Leavey School of Business

Using Raspberry Pi to Capture Environmental Factors that Affect Sleep

Capstone Independent Study Report

SNexus: Lei Xu
Tao Tang
Chao Guo
Yixin Wu
Hengxin Cui

11/15/2015

Index

1. Executive Summary

In such a fast developing society, people are eager to enjoy while working hard, thus they need to do everything efficiently and effectively. Sleep plays a vital role for people's well-being and health. Getting a required amount of quality sleep in the right time will benefit people's performance in the daytime, decrease the risk of mental and physical illness and promote the growth for children and infants.

Yet millions of people do not get enough sleep. For example, surveys conducted by the National Sleep Foundation (NSF) reveal that in 2004 about 60 percent of adults in America report having sleep problems a few nights a week or more. We have learned that many factors, such as stress, drink, illness and environment, would be responsible for a poor sleep quality. However, comparing with the other factors, it's unrealistic for people to evaluate the influence from environment by tracking its variations while sleeping.

With the rapid development of Internet of Things, we tried to solve this problem by our sleep quality measurement device (SNexus is our product name). In our project, we used the Raspberry Pi as a microcomputer with three individual sensors of light, sound, and motion, to capture and output the environmental and the sleeper's motion data. People wear the motion sensor to track the roll-over motions as sleep quality indicator. The sound and light sensors were placed on the night together with the Raspberry Pi to track the environmental variations. Then we processed the captured data by using MongoDB and Amazon Web Services EC2 to store all the meaningful data, and provided a web dashboard to interact with the user. At last, we output sleep quality feedbacks to individual person through our website, or provide anonymous processed data from a large

number of sleepers to organizations, who would be research institutes, hospitals and home appliance makers. We are confident that our feedbacks and data service could be highly valuable to help people understand and improve their sleeping quality.

2. Business Analysis

2.1 Background

The next few years are the golden time of mobile Internet, which would profoundly change our life by the application such as Internet of Things. As a result, a smart wearable device, which could generate much data by variable sensors, transport the data to cloud and provide personalized feedbacks by data analysis and visualized application would definitely have a large market and represent the development trends of technology and innovation.

Statista, a website provides statistical analysis service, estimated in 2015 that by 2018, the wearable device market value will be worth 12.6 billion U.S. dollars. (Figure 1)

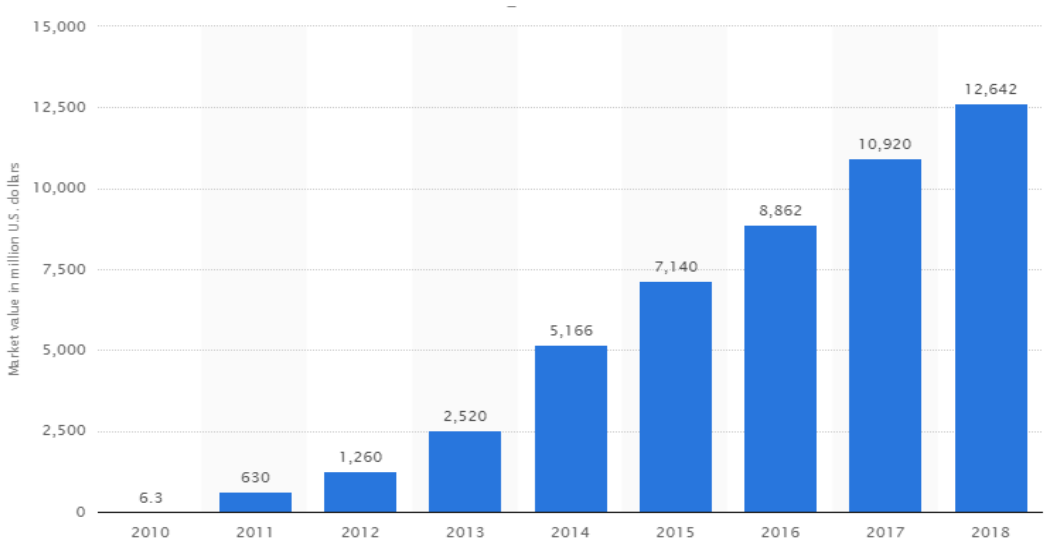


Figure 1: Statista statistical Analysis

In the field of medical wearable device, ABI Research has projected that by 2016, wearable wireless medical device sales will reach more than 100 million devices annually. Research and analysis firm GlobalData expects the market to grow in value from an estimated $0.5 billion in 2010 to over $8 billion by 2018.

Additionally, global revenue for home healthcare devices and services will rise to $12.6 billion in 2018, up from $5.7 billion in 2013, according to a report from IHS Technology. (Figure 2)

Worldwide Revenue Forecast of the Home Healthcare Device and Service Market (in Millions of US Dollars)

	2013	2014	2015	2016	2017	2018
Millions of US Dollars	5,746.5	6,127.2	6,919.0	7,995.2	9,758.0	12,560.7

Source: IHS Technology, May 2014

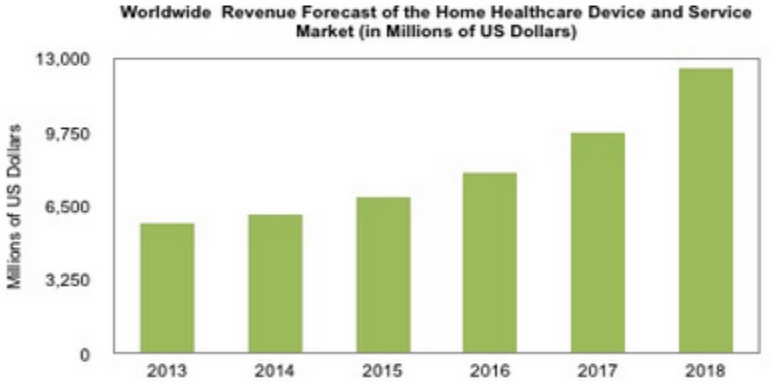


Figure 2: Home Healthcare Devices and Services Revenue Forecast

As a result, Internet of Things, especially medical and healthcare wearable device is one of the most promising products in technology area currently. This motivates us to create a smart device that could catch timely data and provide personalized feedbacks about people sleep.

2.2 Business Requirements

Good sleep is essential to everyone in this fast paced society. It could boost our moods and creativity, improve our concentration, reaction time and memory, enhance our immune systems and reduce our risk of diabetes and other health problems.

But unfortunately, a large amount of us don't have the requisite amount of sleep time. Moreover, most people could not enjoy high quality sleep although their bedtime seems to be enough. The National Institutes of Health notes that the average adult gets fewer than seven hours of sleep every night.

Based on the past research, we have learnt the right environment plays an important role for a high quality sleep. While a person can count calories consumed or miles walked without the help of a digital tracking device, it's nearly impossible to track sleep status and environment factors without assistance. Studies have shown that people do not accurately recall night waking, and that the more sleep deprived they are, the more they underestimate their impairment.

Our product consecutively catches the data about light, sound, and people's motion when they are sleeping. Therefore, it could provide feedback to people about their sleep. Consequently, it will meet the business requirements from hospitals, insurance companies, home appliances makers and any individual person.

2.3 Existing Similar Products

2.3.1 Medical grade measurement device

2.3.1.1 SleepImage

SleepImage is a Medical grade wearable measurement device, which is easy-to-use and could accurately and objectively measures sleep quality.

The SleepImage system records and uploads electrocardiogram (ECG), actigraphy, snoring, and body position data to a secure cloud-based website, where it is automatically analyzed. It then generates sleep quality reports that feature easy-to-read images and summary tables. SleepImage could use mathematical analysis to determine the synchronization between modulations of heart rate variability and respiration. Comparing with other professional medical devices, SleepImage is cost-effective and user-friendly.

2.3.2 Advanced Sleep measurement Device

2.3.2.1 Withings Aura

The Withings Aura has a pad that could place beneath a person's bedding to monitor body movement, breathing cycle and heart rate. It also has a bedside device to track the room environment, such as sound, temperature and light. It will then transfer the data to a smartphone. The Withings Aura will generate a visualized report every morning, it could wake up the person by proper color LED light and music at the right time of sleep circle.

2.3.2.2 ResMed S+ monitor

The ResMed S+ takes a different approach: The milk-carton-size contraption sits on the nightstand aimed at the user's chest as its sensors collect information not only about the movement and respiration but also the room's temperature, light and sound levels. More importantly, it is a non-contact sleep monitor system. It could provide the user personalized feedback about the sleep quality and the suggestions to improve it based on the collected data of several days.

2.3.2.3 Sense

Sense tracks the environmental data in the bedroom and disturbances such as noise and light. Sense combines these data with the user's sleep cycle information to give back a complete picture of the night about the sleep quality. It would help the user to learn how to improve the sleep quality by all the information presented.

Features:

- Smart Alarm

  Sense will wake the user when he/she is in the lightest part of sleep cycle within 30 minutes prior to the set time.

- Sleep Score

Each night Sense calculates the user's Sleep Score. This score takes into account all of the sensor data and variables that Sense monitors to generate a score for the previous night.

- Sleep Timeline

The Sleep Timeline shows the user a minute-by-minute record of the night. Any external disturbances due to light, sound and partner movements will be marked.

- User's Insights

Sense analyzes the user's data and will provide clear suggestions on what he/she can do to have a better night's sleep.

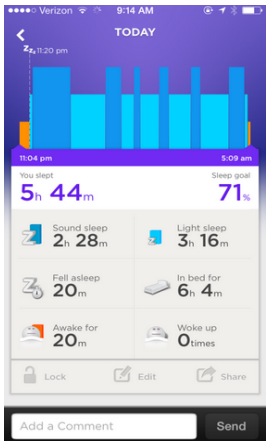
2.3.3 Simple Sleep Measurement Device

2.3.3.1 Basis Carbon Steel Edition watch

The Basis Carbon Steel's heart rate and motion sensors collect sleep data superior to other popular fitness trackers, which rely on only motion sensors.

2.3.3.2 Fitbit Flex

This tracking device is a wearable wristband. It uses motion-tracking sensors to detect the sleep status. There were only a few tiny movements that Fitbit can't pick up. The user interface summarizes the total time that the user was restless or moving in the night. The app is interactive where the user can touch the "restless" points in the chart to see the time frame of restless or awake.

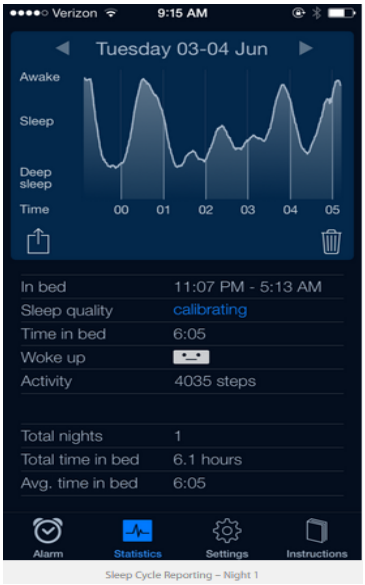


2.3.3.3 Jawbone Up

The Jawbone UP is also a wearable wristband that tracks daily activity along with sleep. UP can categorize light sleep and wake by movement measurement. For example, moderate to extensive movements will be marked as transitions from deep sleep to light sleep. All movement from slight to extensive will be recorded as light sleep. UP's user interface includes a graph that is also interactive. The user can touch one of the sleep phases recorded in the graph and it will provide details on time. This app could also display how long it takes the sleeper to fall asleep.

2.3.3.4 Sleep Cycle – IOS

Sleep Cycle is an app that acts as a bio alarm clock that tracks the user's movement using iPhone's accelerometer and then determines sleep phase. This app collects its data and then presents the information in easy-to-understand graphs. The app also functions as an alarm clock, and will wake the user by his or her phone music or any soothing alarm melodies for a gentle start to the day. In the detailed reporting, the graph makes it easy to show how long the user spent in deep sleep and the peaks of movement that brought the sleeper into different stages. This detailed reporting is clean and simple.



Sleep Cycle Reporting – Night 1

2.4 What Makes Us Unique

2.4.1 More Valuable Data Service

The most valuable cutting edge is our data service. By cooperation with hospitals and research institutions, by gathering data from our direct users, we could have a large amount of first hand data. We will provide various type of data service by processing the data in certain ways based on the detailed requirements from our customers. We could provide basic feedbacks freely to our regular uses and in-depth analysis (future work) to the primary users by charging a certain amount of money.

2.4.2 More Reliable Comparing with Simple Devices

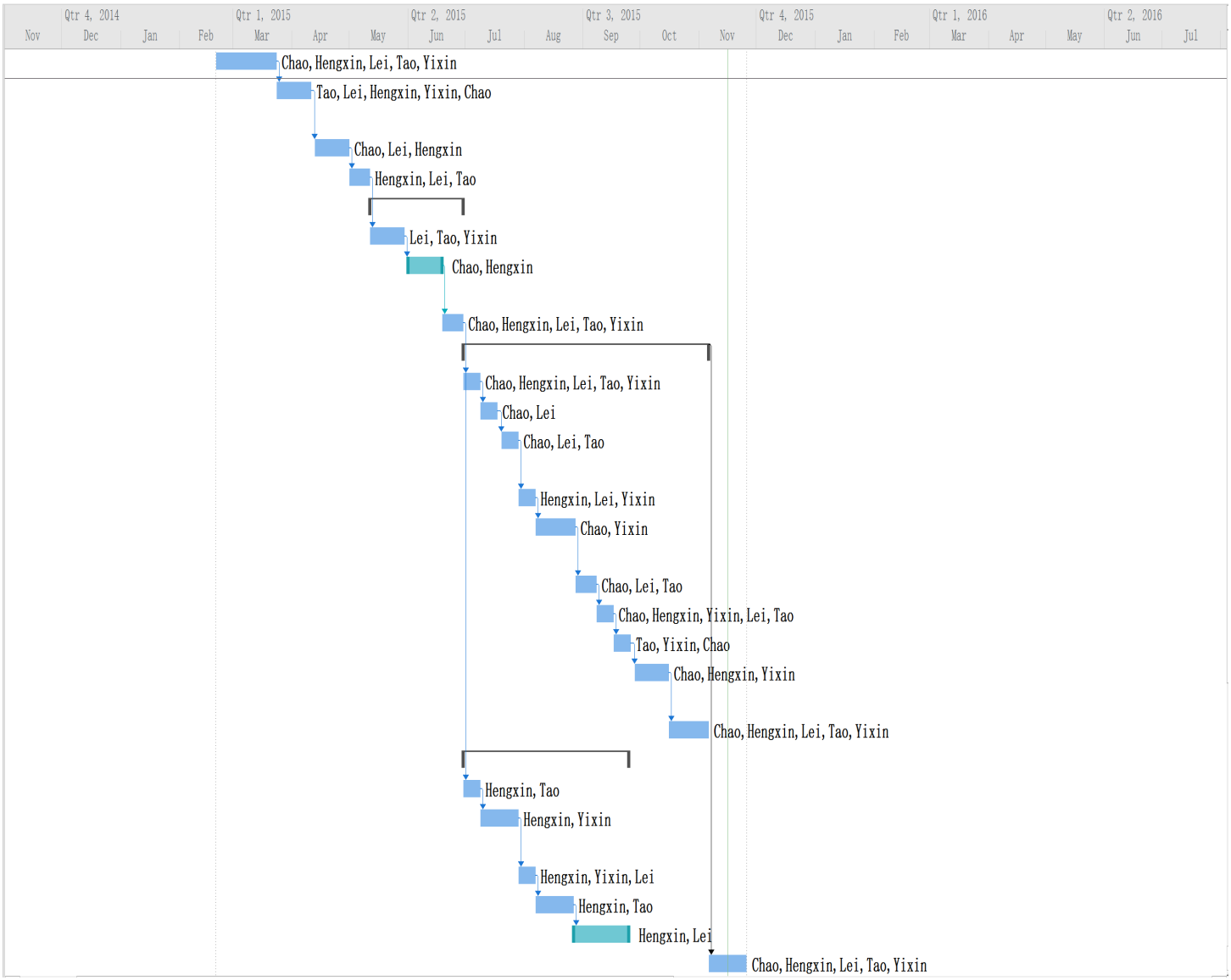
Our device could track multiple factors, such as motion, sound, and light. Based on the combination of these data, we could learn to know the person's sleep status in a more reliable and comprehensive way.

3.Project Overview

3.1 Timeline

GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Prede	Resource Names
1		Proposal & Planning	22 days	Fri 2/20/15	Mon 3/23/15		Chao,Hengxin,Lei,Tao,Yixin
2		Hardware Purchase & Research	14 days	Tue 3/24/15	Fri 4/10/15	1	Tao,Lei,Hengxin,Yixin,Chao
3		Hardware 1 Integration	14 days	Mon 4/13/15	Thu 4/30/15	2	Chao,Lei,Hengxin
4		Hardware 1 Testing	7 days	Fri 5/1/15	Mon 5/11/15	3	Hengxin,Lei,Tao
5		**Infras A (Aborted)**	**35 days**	**Tue 5/12/15**	**Mon 6/29/15**		
6		Kinesis Connection	14 days	Tue 5/12/15	Fri 5/29/15	4	Lei,Tao,Yixin
7		Kninesis and DynamoDB connection	14 days	Mon 6/1/15	Thu 6/18/15	6	Chao,Hengxin
8		Issues and solution	7 days	Fri 6/19/15	Mon 6/29/15	7	Chao,Hengxin,Lei,Tao,Yixin
9		**Infras B**	**93 days**	**Tue 6/30/15**	**Thu 11/5/15**		
10		Research	7 days	Tue 6/30/15	Wed 7/8/15	8	Chao,Hengxin,Lei,Tao,Yixin
11		MongoDB Design	7 days	Thu 7/9/15	Fri 7/17/15	10	Chao,Lei
12		MongoDB and EC2 Deployment	7 days	Mon 7/20/15	Tue 7/28/15	11	Chao,Lei,Tao
13		MongoDB Testing	7 days	Wed 7/29/15	Thu 8/6/15	12	Hengxin,Lei,Yixin
14		RESTful APIs servers design and deployment	15 days	Fri 8/7/15	Thu 8/27/15	13	Chao,Yixin
15		APIs Testing	7 days	Fri 8/28/15	Mon 9/7/15	14	Chao,Lei,Tao
16		User case Analysis	7 days	Tue 9/8/15	Wed 9/16/15	15	Chao,Hengxin,Yixin,Lei,Tao
17		UI design	7 days	Thu 9/17/15	Fri 9/25/15	16	Tao,Yixin,Chao
18		Front-end design and deployment	14 days	Mon 9/28/15	Thu 10/15/15	17	Chao,Hengxin,Yixin
19		Testing and integretion	15 days	Fri 10/16/15	Thu 11/5/15	18	Chao,Hengxin,Lei,Tao,Yixin
20		**Hardware 2**	**63 days**	**Tue 6/30/15**	**Thu 9/24/15**		
21		Hardware 2 Purchase	7 days	Tue 6/30/15	Wed 7/8/15	8	Hengxin,Tao
22		Hardware 2 (motion) Integration	14 days	Thu 7/9/15	Tue 7/28/15	21	Hengxin,Yixin
23		Hardware 2 Testing	7 days	Wed 7/29/15	Thu 8/6/15	22	Hengxin,Yixin,Lei
24		Motion Data Collection	14 days	Fri 8/7/15	Wed 8/26/15	23	Hengxin,Tao
25		Data Analysis	21 days	Thu 8/27/15	Thu 9/24/15	24	Hengxin,Lei
26		Documentation and	14 days	Fri 11/6/15	Wed 11/25/15	9	Chao,Hengxin,Lei,Tao,Yixin


Qtr 4, 2014
Qtr 1, 2015
Qtr 2, 2015
Qtr 3, 2015
Qtr 4, 2015
Qtr 1, 2016
Qtr 2, 2016
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Chao, Hengxin, Lei, Tao, Yixin
Tao, Lei, Hengxin, Yixin, Chao
Chao, Lei, Hengxin
Hengxin, Lei, Tao
Lei, Tao, Yixin
Chao, Hengxin
Chao, Hengxin, Lei, Tao, Yixin
Chao, Hengxin, Lei, Tao, Yixin
Chao, Lei
Chao, Lei, Tao
Hengxin, Lei, Yixin
Chao, Yixin
Chao, Lei, Tao
Chao, Hengxin, Yixin, Lei, Tao
Tao, Yixin, Chao
Chao, Hengxin, Yixin
Chao, Hengxin, Lei, Tao, Yixin
Hengxin, Tao
Hengxin, Yixin
Hengxin, Yixin, Lei
Hengxin, Tao
Hengxin, Lei
Chao, Hengxin, Lei, Tao, Yixin

3.1.1 Team Weekly Meeting Schedule

February 2015—July 2015:

Days of Meeting	**Hours of Meeting**	**Platform of Meeting**
Monday	4:00 pm---5: 30 pm	Lucas Hall (personal)
Tuesday	4:00 pm---5: 00 pm (Meeting with professor Haibing Lu)	Lucas 216F
Wednesday	4:00 pm---5: 30 pm	Lucas Hall (personal)
Thursday	4:00 pm---5: 30 pm	Lucas Hall (personal)
Friday	8:00 pm—10:00 pm	Google+ Hangouts (video call)
Saturday	13:00 pm—17:00 pm	Library or Lucas Hall

August 2015 – November 2015:

Days of Meeting	**Hours of Meeting**	**Platform of Meeting**
Monday	4:00 pm---5: 30 pm	Lucas Hall (personal)
Tuesday	4:00 pm---5: 00 pm	Lucas Hall
Wednesday	4:00 pm---5: 30 pm	Lucas Hall (personal)
Thursday	5:00pm---5: 30 pm (Meeting with Professor Haibing Lu)	Lucas Hall 216F
Saturday or Sunday	13:00 pm—17:00 pm	Library or Lucas Hall

3.2 Infrastructure A—Aborted Infrastructure

3.2.1 Project Module

In this Infrastructure A, the main components of this model includes hardware with sensors (sound, light and motion), network connection, real-time streaming data collection process which is stored in Amazon Web Services (AWS), database management, and a website (AWS EC2) which can show the data results. In the hardware part, we used two Raspberry Pi. One of them equipped with sound and light sensors, another one equipped with motion sensor. The two Raspberry Pi will work and upload data into database simultaneously when activated by user.

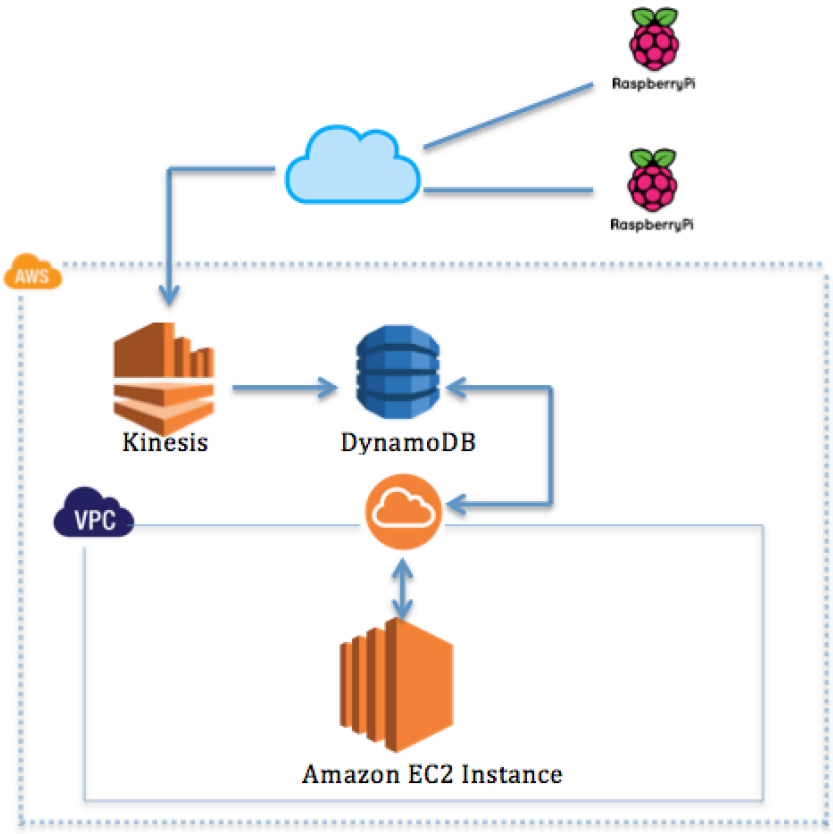


Figure 3: Infrastructure A Module

In infrastructure A, this project used Amazon Web Services Kinesis to capture the real time streaming data from raspberry pi with sensors. Amazon Kinesis is a fully managed, cloud-based service for real-time processing of large, distributed data streams. Amazon Kinesis can continuously capture and store terabytes of data per hour from hundreds of thousands of sources such as sensors, website clickstreams, financial transactions, social media feeds, IT logs, and location-tracking events. Because the Kinesis cannot store all the data over 24 hours, therefore, AWS Kinesis emit all the data and store the data to Amazon DynamoDB. Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models. Its flexible data model and reliable performance make it a great fit for Internet of Things (IoT) applications. We then proceeded to build reliable and scalable ETL (Extract, Transform and Load) processes that filter and archive Kinesis data into DynamoDB. In this process, we used Amazon Kinesis Connector Library to filter data into DynamoDB (Figure 4). It's an open source code to connect Kinesis with DynamoDB. After all data stored in DynamoDB, we planned to use Amazon Elastic Compute Cloud (Amazon EC2) to host a dynamic web dashboard for real-time data visualization, by collecting and gathering data from DynamoDB.

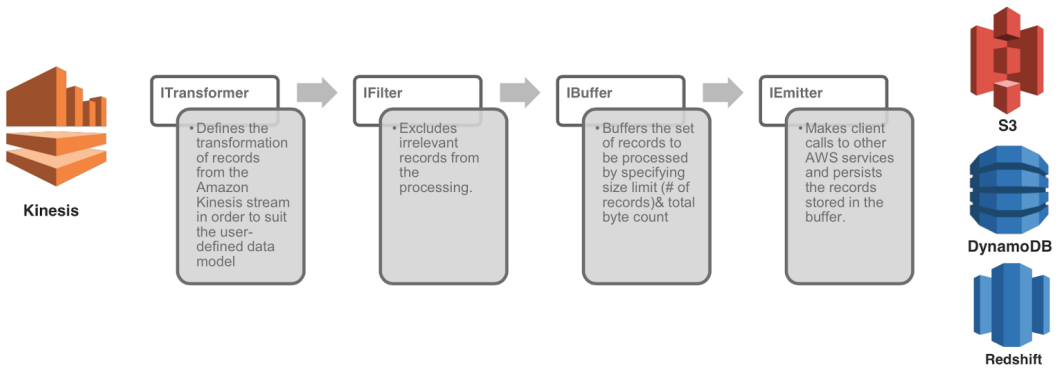


Figure 4: Amazon Kinesis Connector Library

3.2.2 Issues

3.2.2.1 Expense Issue

Although the infrastructure A used all services from Amazon Web Services, the expenses would be much more than our project budget. In this Infrastructure A module, part of the expense fee for data upload in AWS kinesis based on Shard Hour and PUT Playload Unit of streaming data. Moreover, we also need to allocate an expense for data storage, write and read fee after 25 GB of free storage. In addition, we would have to pay what we used in EC2 instance after 750 hours free tier. For example, we will use one shard, 4 records per second and each record size around 1 KB for AWS Kinesis; one EC2 instance and one EIP (Elastic IPs) associated with it; data transfer into EC2 from DynamoDB is 10 GB/Month. Therefore, according to AWS simple monthly calculator, the estimated of our project monthly bill will be $30.84. However, at the beginning of project, the purchasing expense was approximately $400 in hardware and sensors. Therefore, the remaining budget is $100 and that doesn't allow for risk if the project lasted more than 6 months.

3.2.2.2 Technical Issue

For the technical aspect, we faced a challenge in programming. For this project programming need, we chose Python as the only programming language. We completed a one-month beta test to collect and transport data from raspberry pi to AWS Kinesis (Appendix 8.1 and 8.2). However, there was a bottleneck of programming challenges in transporting data protocol into DynamoDB. The challenge was transporting data from Kinesis into DynamoDB because the programming language was only in Java by using Amazon Kinesis Connector Library. Therefore, as considering consistency and compatibility, we decided not to proceed with this infrastructure A since the Python code for Kinesis and Java code for DynamoDB have to be operated separately.

3.2.3 Solution

Based on the expense and technical issues, we developed a new infrastructure, which is infrastructure B. The only changed in infrastructure is the tools in which we used. We used MongoDB and AWS EC2 only in placed of AWS Kinesis, DynamoDB and EC2. The hardware and network connections still remain the same. With the Infrastructure B, we pay for the amount of storage we use in AWS EC2, and achieve code consistency, compatibility and simplicity by using Python.

3.3 Infrastructure B

3.3.1 Project Module

In the newly developed infrastructure B (Figure 5), the main components of this model includes hardware with sensors (sound, light and motion), network connection, real-time streaming data collection process consisting of MongoDB on Amazon Web Services EC2

host to upload, store, and a web dashboard for collected and processed data for the user. However, the hardware and network connections remain the same as infrastructure A. We used two Raspberry Pi. One of them equipped with sound and light sensors, another one equipped with motion sensor. The two Raspberry Pi worked and uploaded data into database simultaneously when activated by user. In the data collection process, we used MongoDB as the database and deployed it on AWS EC2 instance. The real-time steaming data collected and stored into MongoDB collections directly. Moreover, we used Node.js as the server and Backbone.js to create the website of the project, and then deployed them on another AWS EC2 instance. We also created the algorithm for daily data process and analysis. Compared to infrastructure A, we only pay for the amount of storage we use in AWS EC2. However, MongoDB is free to use.

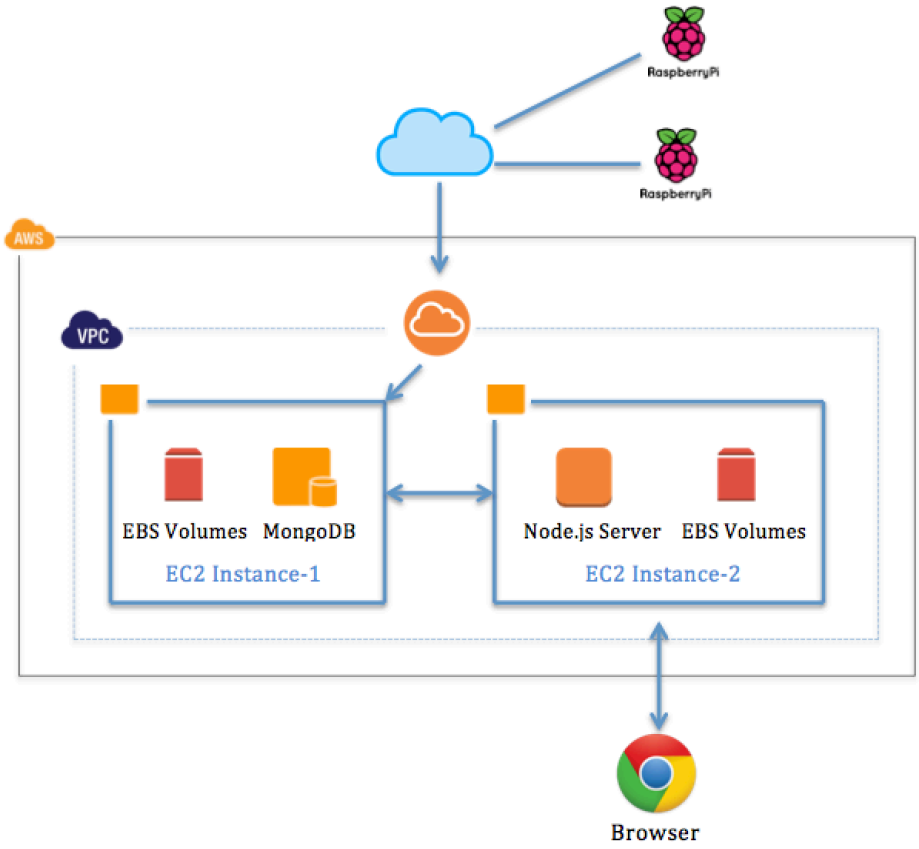


Figure 5 Infrastructure B Module

3.4 Prototype Implementation

3.4.1 Hardware

The final prototype of the project includes one Raspberry Pi with sound and light sensors (Figure 6), and another Raspberry Pi with motion sensor (Figure 7).

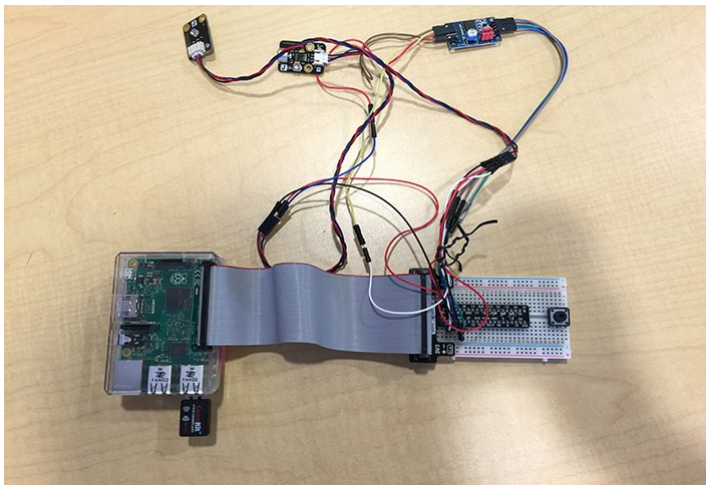


Figure 6 Prototype: Raspberry Pi with sound and light sensor

Figure 7 Prototype: Raspberry Pi with motion sensor

3.4.2 Web Application

The UL of the project is http://52.26.51.149

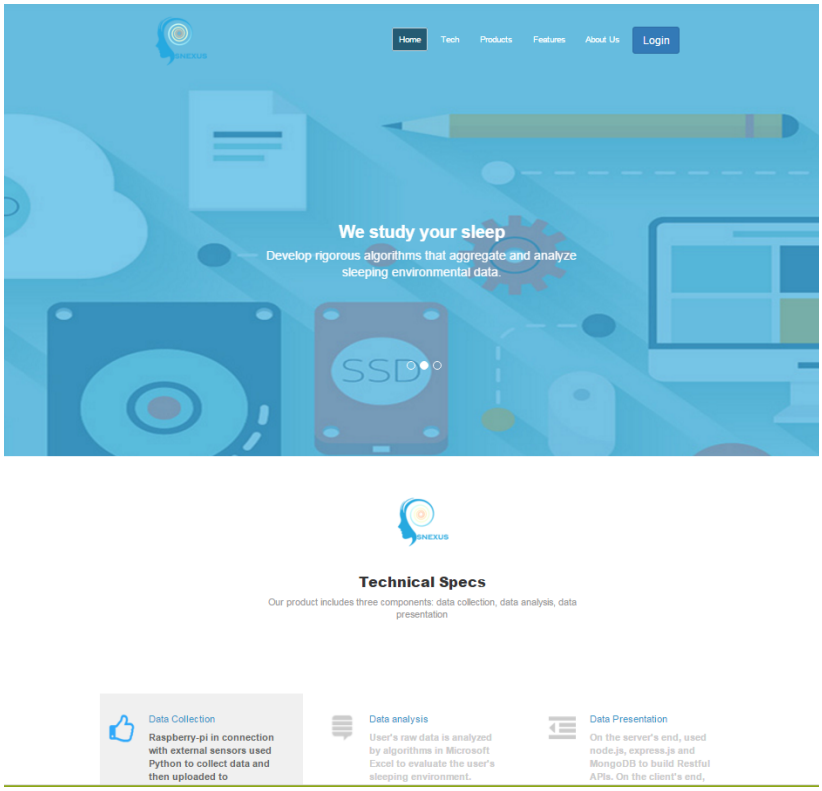


Figure 8 SNexus Main Page

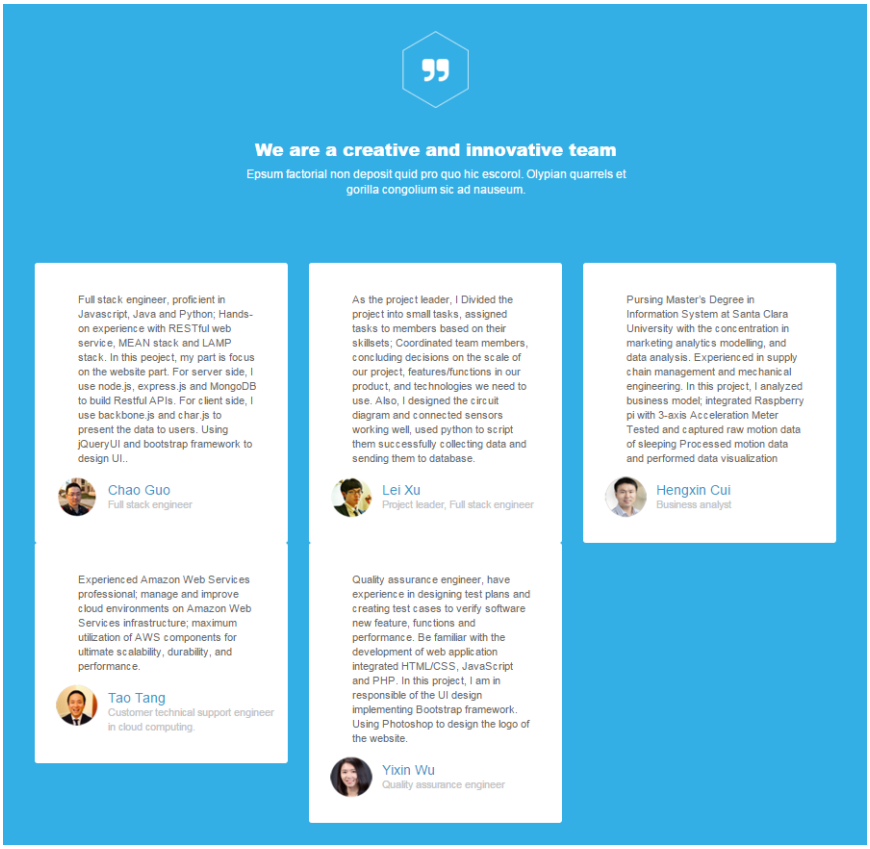


Figure 9 Team introduction page

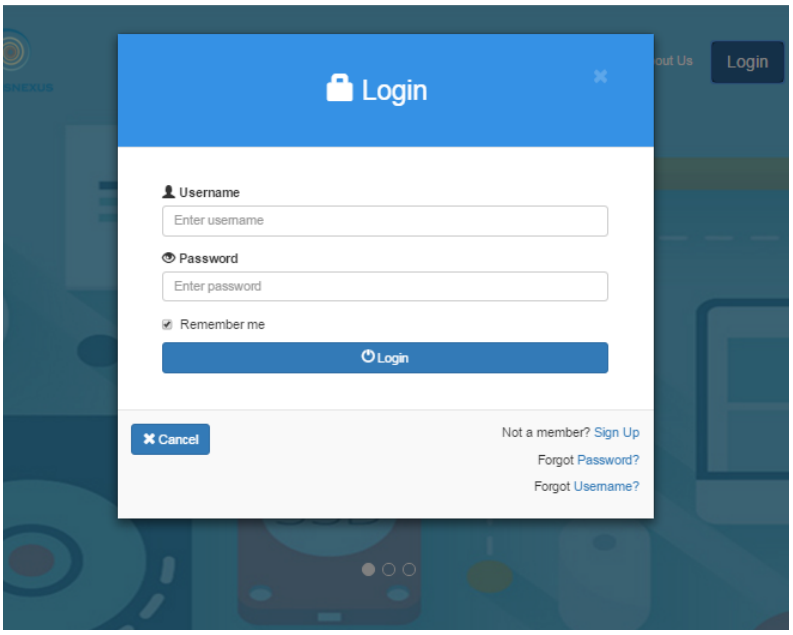


Figure 10 Login page

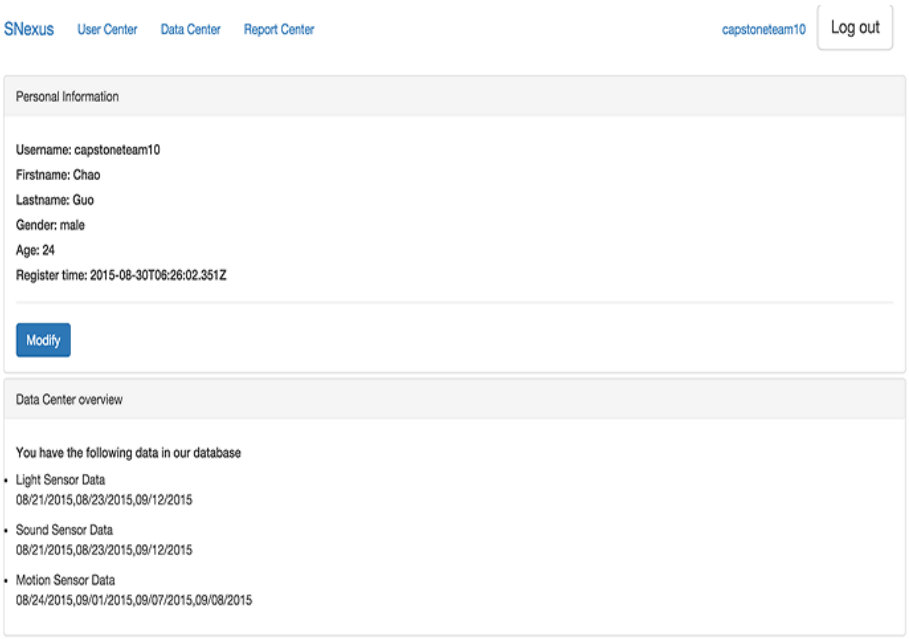


Figure 11 User Center page

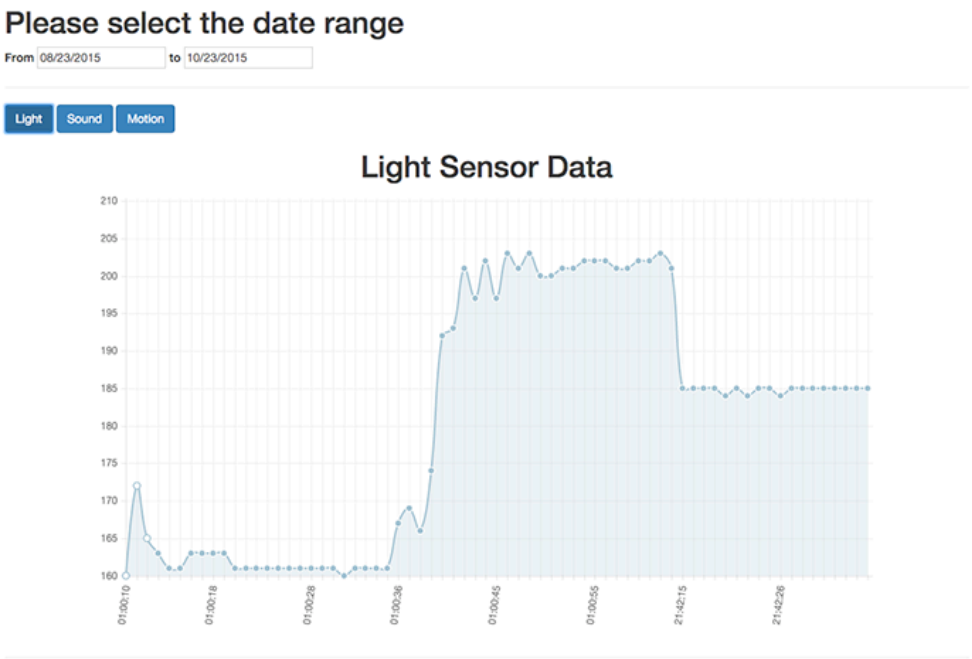


Figure 12 Data Center page

3.5 Team composition/responsibilities

Lei Xu, experienced in using Amazon Web Services, also have knowledge of electronic engineering and python programming and web developing. Mainly focus on hardware and network installation and Python programming.

Tao Tang, experienced in using Amazon Web Services and database management. Mainly focus on AWS Kinesis, DynamoBD and EC2 in the project.

Chao Guo, experienced in programming and telecommunication, such as Java and wireless connection, and Internal network settings. Mainly focus on servers' deployment and front-end programming in the project.

Hengxin Cui, experienced in operations management and mechanical engineering, mainly focus on database management, business and data analysis in the project

Yixin Wu, experienced in software engineering, mainly focus on web programming and database management.

STEP 1--Single Sensor

	Lei Xu	Tao Tang	Chao Guo	Hengxin Cui	Yixin Wu
Hardware Integration	A	B	B	B	
AWS	B	A	A		
Data processing	B	B	B	A	B
Database	B	B		A	
Web Programming	B		A	B	B

STEP 2--Multiple Sensors

	Lei Xu	Tao Tang	Chao Guo	Hengxin Cui	Yixin Wu
Hardware Integration	A	B	B	A	B
Data processing	B	A	A	A	B

4. Infrastructure B Technical Details

4.1 Hardware/Network

4.1.1 Raspberry-Pi Setup

Raspberry pi connected with a keyboard, a mouse, and a displayer connected using HDMI and a WIFI adapter. First, install Raspberian (Figure 13), which is the operating system for Raspberry Pi.

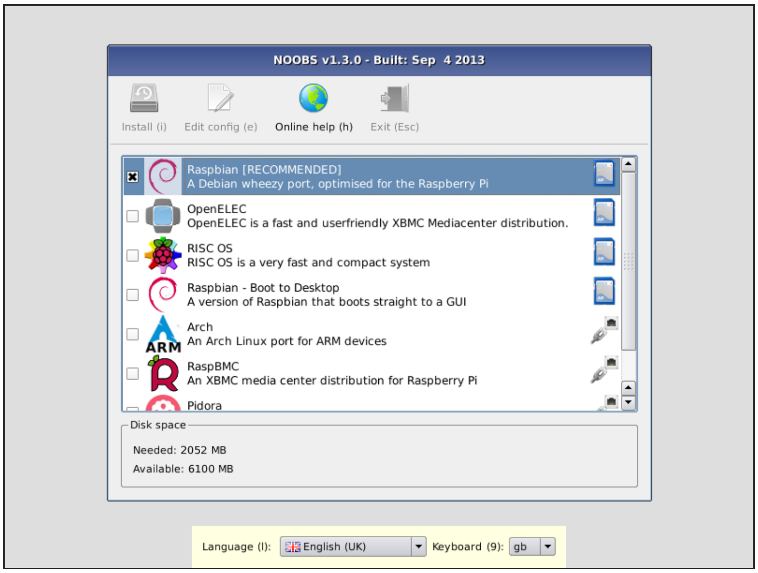


Figure 13 Raspberian installation interface

After installation was successfully, enter the default username and password, then type "startx" as the diagram showed below (Figure 14). "Startx" is the command to show the operating system's UI.

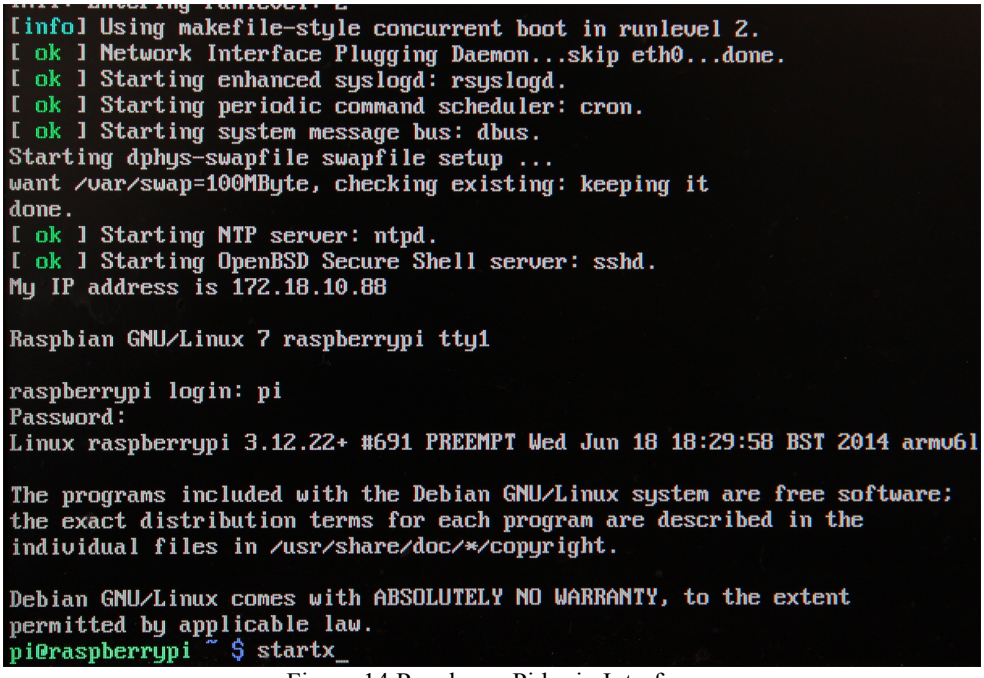


Figure 14 Raspberry Pi login Interface

The User Interface (Figure 15) as seen below is the main interface for Raspberian. As we introduced before, The Raspberry Pi is a series of credit card–sized single-board computers. With a linux-based operating system Raspberian installed, it has open menu, file system manager, and command line as seen in personal computers.

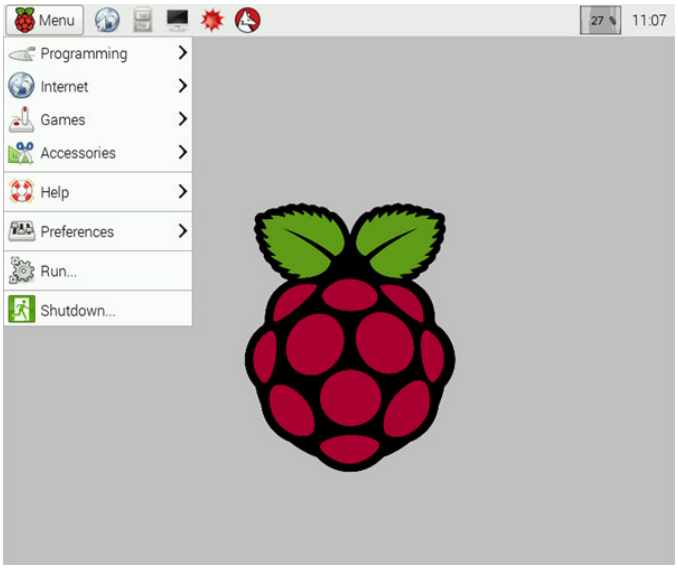


Figure 15 Raspberry Pi UI interface

Raspberry pi UI requires commands to run system update. The commands are listed below:

sudo apt-get update

sudo apt-get upgrade

sudo apt-get autoremove

Raspberry Pi was developed in UK. Therefore, we updated the time zone to Pacific Standard Time by running the command:

sudo dpkg-reconfigure locales

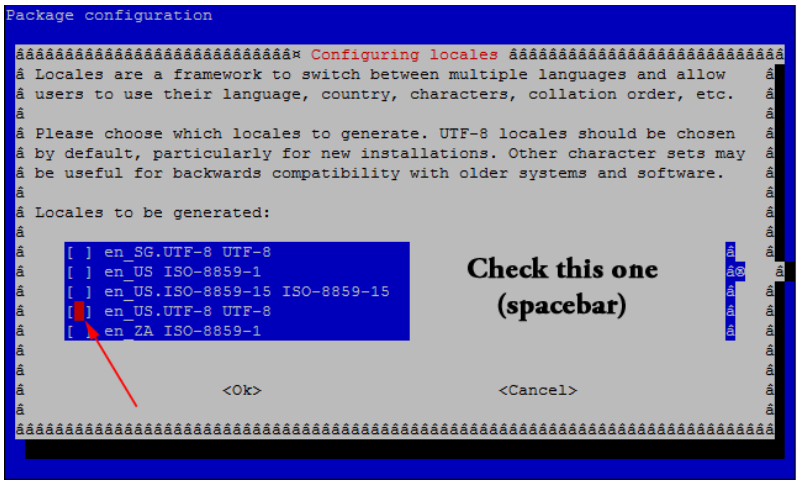


Figure 16 Setting Raspberry Pi locations

sudo dpkg-reconfigure tzdata

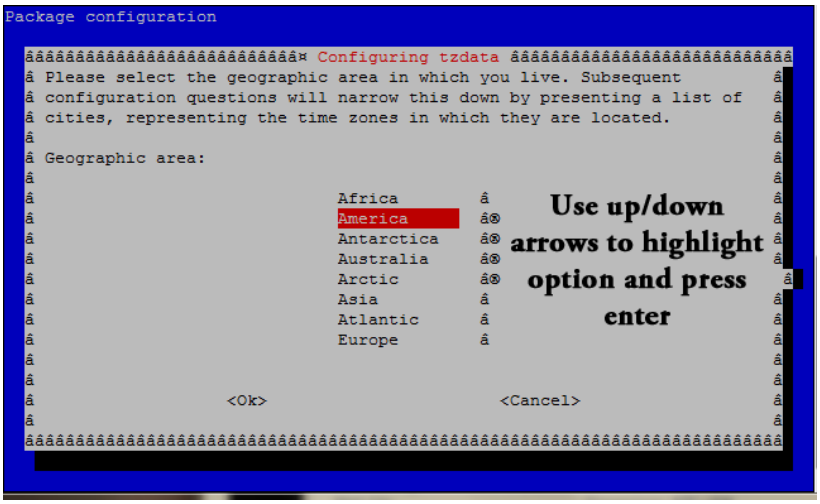


Figure 17 Setting Raspberry Pi Time Zone

4.1.2 Circuit diagram design and connection

We connected sound and light sensors into Raspberry Pi. The values that were generated by all the two sensors were analog format that can't directly be read by the Raspberry Pi. To overcome this challenge, we needed to convert the analog to digital by using PCF8591 AD/DA Converter Module. PCF8591 was connected using an I²C (Inter-Integrated Circuit). It is typically used for attaching lower-speed peripheral ICs, such as

PCF8591, to processors and microcontrollers, such as Raspberry Pi. I²C uses only two bidirectional open-drain lines, SDA (Serial Data Line) and SCL (Serial Clock Line). Typical voltages used are +5 V or +3.3 V.

The sound and light sensors (Figure 18) needed to connect to PCF8591 Ain0 and Ain1, as the sound and light signal input then PCF8591 connected its own SDA & SCL to the same pins of Raspberry Pi, converted the signal input to digital signal, which can be read by Raspberry Pi.

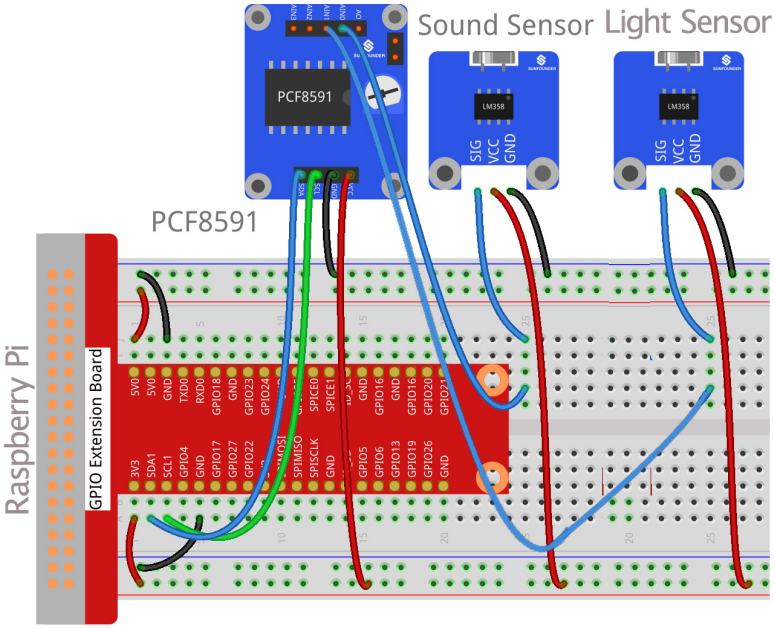


Figure 18 Raspberry Pi and peripheral ICs connection diagram using breadboard

The motion sensor, which is MPU 6050 (Figure 19), is a 3 Axis analog gyro sensors and 3 Axis Accelerometer Module. In this project, the acceleration value by 3-axis could be enough to meet our requirements. As Figure 19 showed, MPU 6050 module need to directly connect to Raspberry PI SCL/SDA pins. However, we had to use two Raspberry Pi instead of one for our project prototype because there is only one set of SCL/SDA pins in Raspberry Pi, and SCL/SDA pins set does not support parallel connection. In this

case, we could not connect both MPU 6050 module and PCF8591 AD/DA Converter module into same one Raspberry Pi. Therefore, we had to separate motion sensor, and sound and light sensors into two different Raspberry Pi. (Figure 22)

Moreover, for the motion sensor needs, we needed to weld the pins to the MPU 6050 module before connection. (Figure 20 and Figure 21)

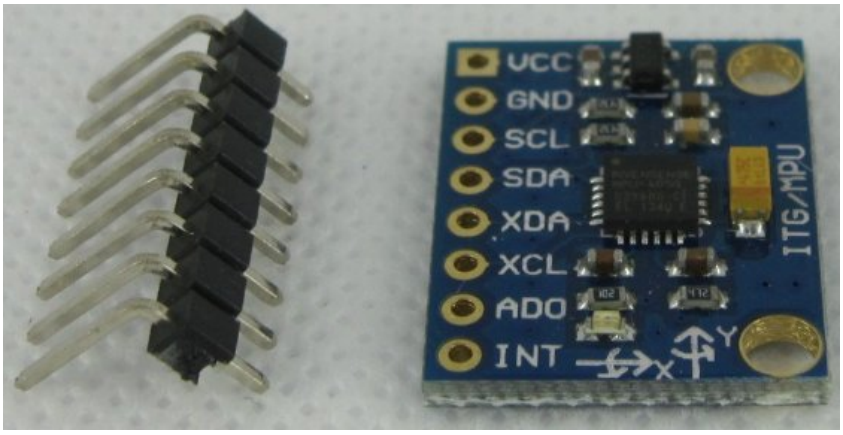


Figure 19 MPU 6050 before weld

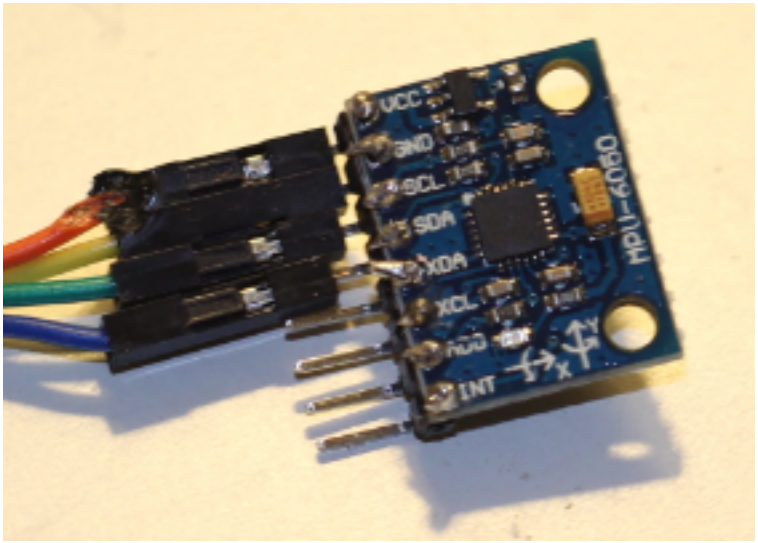


Figure 20 MPU 6050 after welded

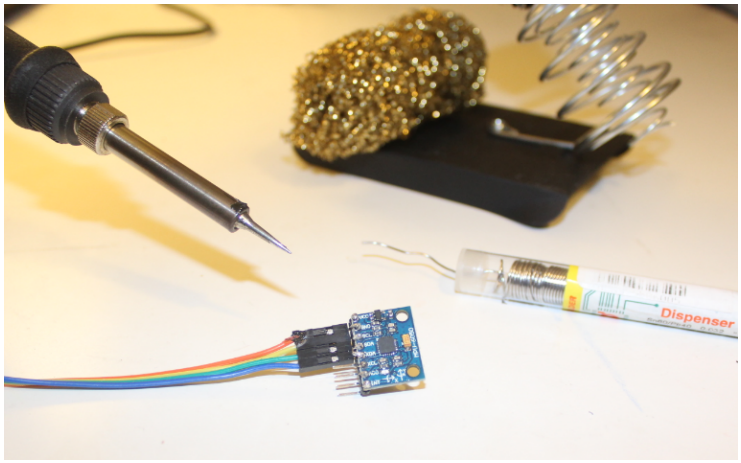


Figure 21 MPU 6050 after welded and soldering appliance

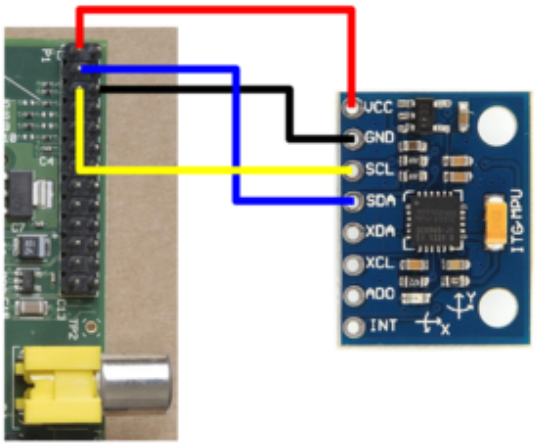


Figure 22 Raspberry Pi with MPU 6050 connection diagram

4.1.2.1 Test the connection

After connection of the sensor, we tested whether the Pi could detect it. This could be done with the following command to install the i2c tools:

sudo apt-get install i2c-tools

sudo i2cdetect -y 1

If we can see the screen showed as below (Figure 23), the connection is successful.

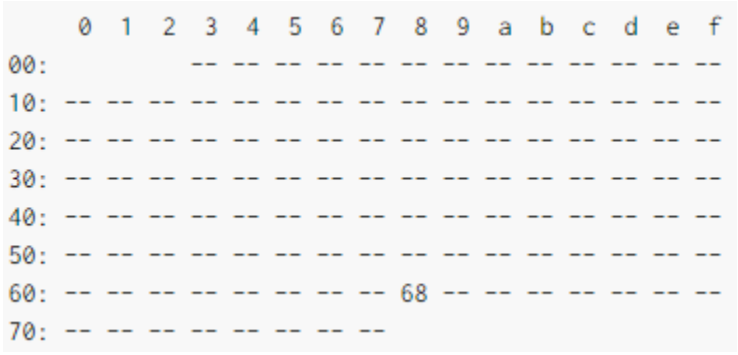


Figure 23 Connection Succeed

The MPU 6050 sensor could be detected with an address of 0x68. In this situation, if we input the below command, we will get an output of 0x68 on screen.

sudo i2cget -y 0 0x68 0x75

The above command tells Pi the motion sensor's address is 0x68 and retrieves the value in the register 0x75, which has a default value of 0x68.

4.1.3 Script: Python

Raspberry Pi requires several python libraries to be installed in order to run the python code such as pymongo, smbus, Tkinter and requests. The Smbus Python module allows SMBus access through the I²C interface on Linux hosts. The host kernel must have I²C support, I²C device interface support, and a bus adapter driver. These prerequisites are all included in a Raspberry Pi. The partial code used Smbus in our project is list below: (please note that the code below is only for sound and light sensor scripting, since the connection diagram and the output of our motion are different than sensor and light sensors.)

```
import smbus
bus = smbus.SMBus(1)
address = 0x48

def readLightSensor():
  bus.write_byte(address,0x40)
  bus.read_byte(address) # dummy read to start conversion
  temp = bus.read_byte(address)
  return temp

def runLightSensor():
  light = str(readLightSensor())
  timestamp=datetime.datetime.utcnow()
  record = str(timestamp)+":"+light
  return light

def readSoundSensor():
  bus.write_byte(address,0x41)
  bus.read_byte(address) # dummy read to start conversion
  temp = bus.read_byte(address)
  return temp

def runSoundSensor():
  sound = str(readSoundSensor())
  timestamp=datetime.datetime.utcnow()
  record = str(timestamp)+":"+sound
  return sound
```

In this example, 0x48 is the address for I²C, which was connected to PCF8591

Whole file. 0x40, 0x41and the address for the two input of PCF8591 connected to the sound sensor and light sensor separately.

bus.write_byte (address) is the function to let a certain address can access and write value to the memory. bus.read_byte (address) is the function to read the value from a certain address , then store it to memory.

Moreover, we used SD/DA converter, which is the PCF8591, to pass value from one input to memory each time. Therefore, we defined two functions to switch sound and light sensors. When switch function happens, the first value read from a new sensor probably not reliable because of the electrostatic disturbance, so we actually store from the second value. In the code example, we used "# dummy read to start conversion" to be a mark. After getting value from sensor, we recorded the timestamp and then return it. Secondly, we continued run these functions with the time interval as 0.5 seconds by using a while loop (see code below). At the end, we stored all the data that we collected into an object, and upload to database.

```
while(username!="" and status == "on"):
  lightvalue = runLightSensor()
  time.sleep(0.5)
  soundvalue = runSoundSensor()
  time.sleep(0.5)
  currenttime = roundTime(datetime.datetime.now(),roundTo=1)
  row = {
        'username':username,
        'date':currenttime.strftime("%m/%d/%Y"),
        'time':currenttime.strftime("%H:%M:%S"),
        'lightrecord':lightvalue,
        'soundrecord':soundvalue
        }
  db.insert(row)
```

The pymongo module helped us make connection to our mongoDB and upload each data recorded by sensors.

```
try:
    client = MongoClient("mongodb://52.25.49.72:27017/capstone")
    print "Database connecting successfully!"
except pymongo.errors.ConnectionFailure:
    print "Could not connect to MongoDB: %s" % e
db = client.capstone.sensor
```

The requests module helped our Raspberry Pi make http request to our server, and interacted with APIs, this made login function working well. When user input their username and password while they start to use our Sleep tracker, username and password will be send to our server and checked if there is a match (this part will be explained in the later section). The status code, which is uniform as RESTful API design role, will told us if the username and password use inputted were correct. 200 means successful, in our project means login successfully; 404 means nothing found, in our project means these is no such username; 401 means unauthenticated, which means username and password are not match.

```
def onCheck(self):
        global username
        username = self.out_name.get()
        password = self.out_pass.get()
        logininfo = {'username':username,'password':password}
        response =
 requests.post("http://52.26.51.149:3000/user/login",data=logininfo) #make http
request
        if response.status_code==200:
                box.showinfo("Information", "Login successful!”)
        if response.status_code==404:
                username=””
                box.showinfo("Information", "Login fail! There is no such user,
please create a new account in our website!”)
        if response.status_code==401:
                username=””
                box.showinfo("Infomation", "Login fail! PLease check your
password!”)
```

Last, the Tkinter module helped us made a login interface:

```
class Example(Frame):

  def __init__(self, parent):
        Frame.__init__(self, parent, background="white”)
         self.parent = parent
         self.initUI()

  def initUI(self):
         self.parent.title("SNexus – Welcome”)
         self.pack(fill=BOTH, expand=1)
         frame = Frame(self, relief=RAISED, borderwidth=3 ,
background="white”)
         frame.pack(fill=BOTH, expand=1)
         self.out_name = StringVar()
         self.out_pass = StringVar()
         photo = createImage()

         label_logo = Label(frame,image = photo)
         label_logo.image = photo
          label_logo.grid(row=1, column=2, rowspan = 3, columnspan=3)

          label1 = Label(frame, text="PLEASE
LOGIN",background="white",font=("Helvetica",20))
          label1.grid(row=4, column=2, columnspan=4)

         label2 = Label(frame, text="Your Name”,
background="white",font=("Helvetica",15))
           label2.grid(row=5, column=3,columnspan=1,pady=10)
           entryname = Entry(frame, font = "44”,
textvariable=self.out_name,width=30,justify="center”,
           highlightcolor="blue”)
           entryname.grid(row=6, column=2, columnspan=3,padx=40)

           label3 = Label(frame, text="Your
Password",background="white",font=("Helvetica",15))
           label3.grid(row=7, column=3,columnspan=1,pady=10)
           entrypass = Entry(frame, font = "44”,
textvariable=self.out_pass,width=30,justify="center”,
```

```
            highlightcolor="blue",show="*”)
            entrypass.grid(row=8, column=2,columnspan=3)

            okButton = Button(frame, text = "Submit", width = 12,
command=self.onCheck)
            okButton.grid(row=9, column=2, columnspan =3,pady=20)
            okButton3 = Button(self, text = "Stop", width = 12,
command=self.stopRunning)
            okButton3.pack(side=RIGHT)
            okButton2 = Button(self, text = "StartRecording",width = 10,
command=self.startRecording)
            okButton2.pack(side=LEFT)

  def onCheck(self):
            global username
            username = self.out_name.get()
            password = self.out_pass.get()
            logininfo = {'username':username,'password':password}
           response = requests.post("http://52.26.51.149:3000/user/login",data =
logininfo)
            if response.status_code==200:
                        box.showinfo("Information", "Login successful!”)
            if response.status_code==404:
                        username=””
                        box.showinfo("Information", "Login fail! There is no such user,
please create a new account in our website!”)
            if response.status_code==401:
                          username=””
                          box.showinfo("Infomation", "Login fail! PLease check your
password!”)

  def stopRunning(self):
            global status
            status = “off ”
            print status

  def startRecording(self):
            global status
            status = “ on ”
            print status
```

```
def main():
        root = Tk()
        root.geometry("400x500+700+300”)
        app = Example(root)
        root.mainloop()

if __name__ == ‘_main_’:
        main()
```

Above code will make the UI as showed below (Figure 24):

Figure 24 User login panel in Raspberry Pi

Sound and light sensors full setup codes are below:

```
import json
import time
import datetime
import smbus
from pymongo import MongoClient
import Tkinter
import requests

from Tkinter import *
import tkMessageBox as box
```

```
import base64
import urllib

bus = smbus.SMBus(1)
address = 0x48

username = ""
status = "off"
def createImage():
  image = PhotoImage(file = "logo.gif")
  return image

class Example(Frame):

  def __init__(self, parent):
     Frame.__init__(self, parent, background="white")
     self.parent = parent
     self.initUI()

  def initUI(self):
        self.parent.title("SNexus -- Welcome")
        self.pack(fill=BOTH, expand=1)
        frame = Frame(self, relief=RAISED, borderwidth=3 , background="white")
        frame.pack(fill=BOTH, expand=1)
        self.out_name = StringVar()
        self.out_pass = StringVar()
        photo = createImage()

        label_logo = Label(frame,image = photo)
        label_logo.image = photo
        label_logo.grid(row=1, column=2, rowspan = 3, columnspan=3)

        label1 = Label(frame, text="PLEASE
LOGIN",background="white",font=("Helvetica",20))
        label1.grid(row=4, column=2, columnspan=4)

     label2 = Label(frame, text="Your Name",
background="white",font=("Helvetica",15))
        label2.grid(row=5, column=3,columnspan=1,pady=10)
```

```
      entryname = Entry(frame, font = "44",
textvariable=self.out_name,width=30,justify="center",
          highlightcolor="blue")
          entryname.grid(row=6, column=2, columnspan=3,padx=40)

          label3 = Label(frame, text="Your
Password",background="white",font=("Helvetica",15))
          label3.grid(row=7, column=3,columnspan=1,pady=10)
      entrypass = Entry(frame, font = "44",
textvariable=self.out_pass,width=30,justify="center",
          highlightcolor="blue",show="*")
          entrypass.grid(row=8, column=2,columnspan=3)

          okButton = Button(frame, text = "Submit", width = 12, command=self.onCheck)
          okButton.grid(row=9, column=2, columnspan =3,pady=20)
          okButton3 = Button(self, text = "Stop", width = 12, command=self.stopRunning)
          okButton3.pack(side=RIGHT)
          okButton2 = Button(self, text = "StartRecording",width = 10,
command=self.startRecording)
          okButton2.pack(side=LEFT)

  def onCheck(self):
          global username
          username = self.out_name.get()
          password = self.out_pass.get()
          logininfo = {'username':username,'password':password}
          response = requests.post("http://52.26.51.149:3000/user/login",data = logininfo)
          if response.status_code==200:
                box.showinfo("Information", "Login successful!")
          if response.status_code==404:
                username=""
                box.showinfo("Information", "Login fail! There is no such user, please
create a new account in our website!")
          if response.status_code==401:
                username=""
                box.showinfo("Infomation", "Login fail! PLease check your password!")

  def stopRunning(self):
          global status
          status = "off"
```

```
            print status

    def startRecording(self):
            global status
            status = "on"
            print status

def main():
            root = Tk()
            root.geometry("400x500+700+300")
            app = Example(root)
            root.mainloop()

if __name__ == '__main__':
    main()

if username!="" and status == "on":
    try:
        client = MongoClient("mongodb://52.25.49.72:27017/capstone")
        print "Database connecting successfully!"
    except pymongo.errors.ConnectionFailure,e:
        print "Could not connect to MongoDB: %s" % e
    db = client.capstone.sensor
    row = {}

def readLightSensor():
    bus.write_byte(address,0x40)
    bus.read_byte(address) # dummy read to start conversion
    temp = bus.read_byte(address)
    return temp


def runLightSensor():
    light = str(readLightSensor())
    timestamp=datetime.datetime.utcnow()
    record = str(timestamp)+":"+light
    #print "Light: " +record
    return light
```

```
def readSoundSensor():
   bus.write_byte(address,0x41)
   bus.read_byte(address) # dummy read to start conversion
   temp = bus.read_byte(address)
   return temp

def runSoundSensor():
   sound = str(readSoundSensor())
   timestamp=datetime.datetime.utcnow()
   record = str(timestamp)+":"+sound
   #print "Sound: " +record
   return sound

def readTHSensor():
   return bus.read_byte(0x48)

def runTHSensor():
   bus.write_byte(0x48, 0)
   last_reading = -1
   th = readTHSensor()
   timestamp=datetime.datetime.utcnow()
   record1 = str(timestamp)+":"+str(th)
   print "Temperture: " +record1

def roundTime(dt , roundTo):
   if dt == None : dt = datetime.datetime.now()
   seconds = (dt - dt.min).seconds
   rounding = (seconds +roundTo/2)
   return dt + datetime.timedelta(0, rounding-seconds,-dt.microsecond)

while(username!="" and status == "on"):
  lightvalue = runLightSensor()
  time.sleep(0.5)
  soundvalue = runSoundSensor()
  time.sleep(0.5)
  currenttime = roundTime(datetime.datetime.now(),roundTo=1)
  row = {
          'username':username,
          'date':currenttime.strftime("%m/%d/%Y"),
          'time':currenttime.strftime("%H:%M:%S"),
```

```
        'lightrecord':lightvalue,
        'soundrecord':soundvalue
        }
 print row
 db.insert(row)

client.close()
```

As the MPU-6050 motion sensor script file, since the motion's output are different than light and sound sensors, it has six output: which are three ways(x,y,z) of acceleration and gyroscope, the script code should be different than the way deal with light and sound sensors, which both output one record-- the value that represent the degree of sound and light intensity.

```
    def read_byte(adr):
        return bus.read_byte_data(address, adr)

    def read_word(adr):
        high = bus.read_byte_data(address, adr)
        low = bus.read_byte_data(address, adr+1)
        val = (high << 8) + low
        return val

    def read_word_2c(adr):
        val = read_word(adr)
        if (val >= 0x8000):
            return -((65535 - val) + 1)
        else:
            return val

while(username!="" and status == "on"):
        accel_xout = round(read_word_2c(0x3b), 2)
        accel_yout = round(read_word_2c(0x3d), 2)
        accel_zout = round(read_word_2c(0x3f), 2)
        accel_xout_scaled = round(100 * accel_xout / 16384.0 , 2)
```

```
accel_yout_scaled = round(100 * accel_yout / 16384.0 , 2)
accel_zout_scaled = round(100 * accel_zout / 16384.0 , 2)
accel_sum0 = 1*math.sqrt ((accel_xout_scaled*accel_xout_scaled) +
(accel_yout_scaled*accel_yout_scaled) + (accel_zout_scaled*accel_zout_scaled))
accel_sum = round(accel_sum0, 2)

print "accel_xout: ", accel_xout, " scaled: ", accel_xout_scaled
print "accel_yout: ", accel_yout, " scaled: ", accel_yout_scaled
print "accel_zout: ", accel_zout, "  scaled: ", accel_zout_scaled
print " accel_sum_scaled: ", accel_sum
print "\n ”

time.sleep(0.25)
```

In this example, we defined three functions first. Function read_byte(), read_word() and read_word_2c(), which read value generated by MPU-6050, and process the data into a more representative value. In our project, we only used x, y, z three axis acceleration, these three value can be read in 0x3b, 0x3d and 0x3f. Then we scaled by 16384(the device suggested value), and calculate a square root for the final use.

Not only we uploaded our motion data to our database, we also stored them locally, for the later data analysis.

```
file.write(str(str(username) + ", " + str(currenttime.strftime("%H-%M-%S")) + ",
" + str(accel_sum) + ", "+ str(accel_xout_scaled) + ", " + str(accel_yout_scaled) +
", " + str(accel_zout_scaled) + "\n"))
        # Export the data to a local txt file
file.close()
```

Motion sensor full setup codes are list below:

```
import json
import time
import datetime
import math
import smbus
```

```
from pymongo import MongoClient
import Tkinter
import requests
from Tkinter import *
import tkMessageBox as box
#Power management registers
power_mgmt_1 = 0x6b
power_mgmt_2 = 0x6c

username = “”

class Example(Frame):

def __init__(self, parent):
      Frame.__init__(self, parent, background="white")
      self.parent = parent
      self.initUI()

def initUI(self):
          self.parent.title("SNexus -- Welcome")
          self.pack(fill=BOTH, expand=1)
          frame = Frame(self, relief=RAISED, borderwidth=3 , background="white")
          frame.pack(fill=BOTH, expand=1)
          self.out_name = StringVar()
          self.out_pass = StringVar()
          photo = createImage()

          label_logo = Label(frame,image = photo)
          label_logo.image = photo
          label_logo.grid(row=1, column=2, rowspan = 3, columnspan=3)

          label1 = Label(frame, text="PLEASE
LOGIN",background="white",font=("Helvetica",20))
          label1.grid(row=4, column=2, columnspan=4)

            label2 = Label(frame, text="Your Name",
background="white",font=("Helvetica",15))
          label2.grid(row=5, column=3,columnspan=1,pady=10)
          entryname = Entry(frame, font = "44",
textvariable=self.out_name,width=30,justify="center",
```

```
        highlightcolor="blue")
        entryname.grid(row=6, column=2, columnspan=3,padx=40)

        label3 = Label(frame, text="Your
Password",background="white",font=("Helvetica",15))
        label3.grid(row=7, column=3,columnspan=1,pady=10)
        entrypass = Entry(frame, font = "44"",
textvariable=self.out_pass,width=30,justify="center",
        highlightcolor="blue",show="*")
        entrypass.grid(row=8, column=2,columnspan=3)

        okButton = Button(frame, text = "Submit", width = 12, command=self.onCheck)
        okButton.grid(row=9, column=2, columnspan =3,pady=20)
        okButton3 = Button(self, text = "Stop", width = 12, command=self.stopRunning)
        okButton3.pack(side=RIGHT)
        okButton2 = Button(self, text = "StartRecording",width = 10,
command=self.startRecording)
        okButton2.pack(side=LEFT)

def onCheck(self):
        global username
        username = self.out_name.get()
        password = self.out_pass.get()
        logininfo = {'username':username,'password':password}
        response = requests.post("http://52.26.51.149:3000/user/login",data = logininfo)
        if response.status_code==200:
                box.showinfo("Information", "Login successful! ")
        if response.status_code==404:
                username=""
                box.showinfo("Information", "Login fail! There is no such user, please
create a new account in our website! ")
        if response.status_code==401:
                username=""
                box.showinfo("Infomation", "Login fail! PLease check your password! ")

def stopRunning(self):
        global status
        status = "off "
        print status
```

```
def startRecording(self):
        global status
        status = "on"
        print status

def main():
        root = Tk()
        root.geometry("400x500+700+300")
        app = Example(root)
        root.mainloop()

if __name__ == '__main__':
        main()

if username!="" and status == "on":
        try:
          client = MongoClient("mongodb://52.25.49.72:27017/capstone")
          print "successfully! "
        except pymongo.errors.ConnectionFailure,e:
          print "Could not connect to MongoDB: %s" % e

        db = client.capstone.sensor
        #username = "team10"
        row = {}

def read_byte(adr):
    return bus.read_byte_data(address, adr)

def read_word(adr):
      high = bus.read_byte_data(address, adr)
      low = bus.read_byte_data(address, adr+1)
      val = (high << 8) + low
      return val

def read_word_2c(adr):
      val = read_word(adr)
      if (val >= 0x8000):
            return -((65535 - val) + 1)
      else:
            return val
```

```
bus = smbus.SMBus(1)
address = 0x68 # This is the address value read via the i2cdetect command

# Now wake the 6050 up as it starts in sleep mode
bus.write_byte_data(address, power_mgmt_1, 0)

def roundTime(dt,roundTo):
        if dt == None : dt = datetime.datetime.now()
        seconds = (dt - dt.min).seconds
        rounding = (seconds +roundTo/2)
        return dt + datetime.timedelta(0, rounding-seconds,-dt.microsecond)

currenttime = roundTime(datetime.datetime.now(),roundTo=1)
file = open("motion" +currenttime.strftime("%m%d%y") + currenttime.strftime("-
%H%M%S") +".txt", "w")

#Capture the three axis acceleration value per 0.25s
while(username!="" and status == "on"):
          accel_xout = round(read_word_2c(0x3b), 2)
          accel_yout = round(read_word_2c(0x3d), 2)
          accel_zout = round(read_word_2c(0x3f), 2)
          accel_xout_scaled = round(100 * accel_xout / 16384.0 , 2)
          accel_yout_scaled = round(100 * accel_yout / 16384.0 , 2)
          accel_zout_scaled = round(100 * accel_zout / 16384.0 , 2)
          accel_sum0 = 1*math.sqrt ((accel_xout_scaled*accel_xout_scaled) +
          (accel_yout_scaled*accel_yout_scaled) + (accel_zout_scaled*accel_zout_scaled))
          accel_sum = round(accel_sum0, 2)
          print "accel_xout: ", accel_xout, " scaled: ", accel_xout_scaled
          print "accel_yout: ", accel_yout, " scaled: ", accel_yout_scaled
          print "accel_zout: ", accel_zout, " scaled: ",  accel_zout_scaled
          print " accel_sum_scaled: ", accel_sum
          print "\n"

          time.sleep(0.25)

          row = {
                    'username':username,
                    'date':currenttime.strftime("%m/%d/%Y"),
          'time':currenttime.strftime("%H:%M:%S"),
                    'motionrecord':accel_sum
```

```
        }
        print row
          db.insert(row)          # insert the data to MongoDB

        file.write(str(str(username) + ", " + str(currenttime.strftime("%H-%M-%S")) + ",
" + str(accel_sum) + ", "+ str(accel_xout_scaled) + ", " + str(accel_yout_scaled) + ", "  +
str(accel_zout_scaled) + "\n"))
        # Export the data to a local txt file
file.close()
client.close()
```

4.2 Servers deployment

4.2.1 Infrastructure Platform: Amazon EC2

The project chose to deploy MongoDB, Restful APIs and website application on Amazon Web Services EC2. The steps that to build EC2 instance are list below:

4.2.1.1 Choose an Amazon Machine Image (AMI). We used Amazon Linux AMI (HVM), SSD Volume Type in the project. (Figure 25)

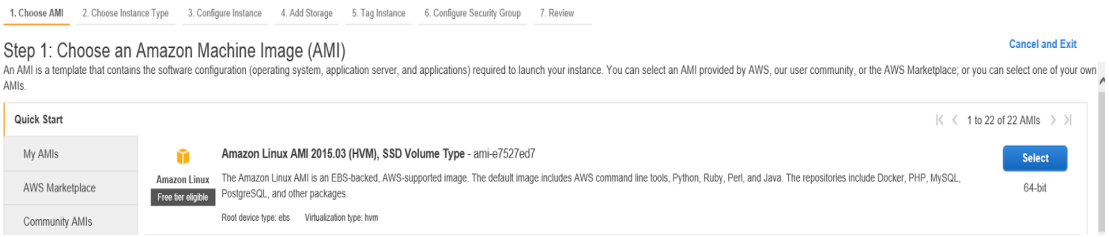


Figure 25

4.2.1.2 Choose the Instance Type. Here we used t2.micro (Figure 26)

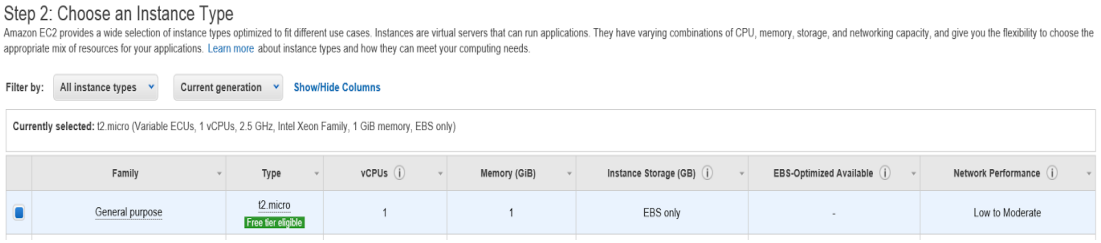


Figure 26

4.2.1.3 Configure Instance Details as list below. First, we chose the network and subnet of VPC that you want the instance deployed to. As for the project, we used default VPC and subnet setting. Moreover, we assigned a public IP address by choosing " Auto-assign Public IP" to let the instance has the ability to access Internet. (Figure 27)

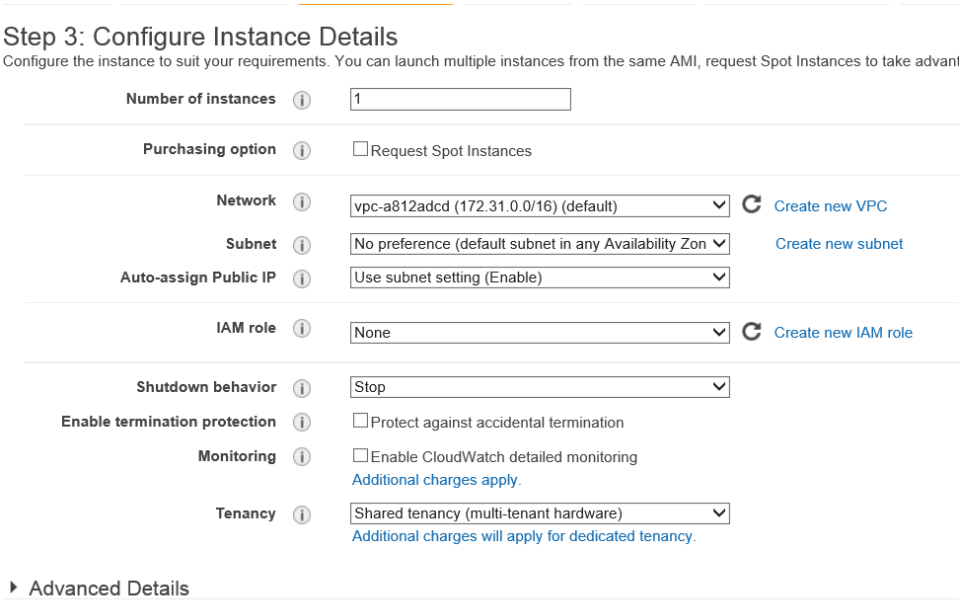


Figure 27

4.2.1.4 Add Storage. We used default setting for EBS storage. (Figure 28)

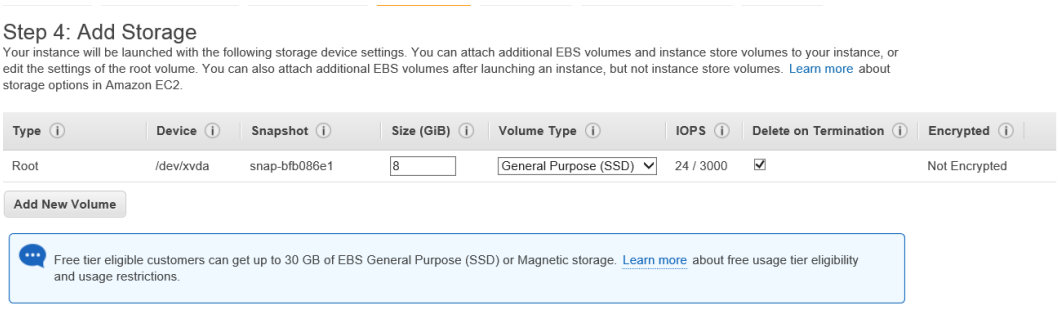


Figure 28

4.2.1.5 Tag Instance. We can manage Instances after we build all instances by tagging instance. (Figure 29)

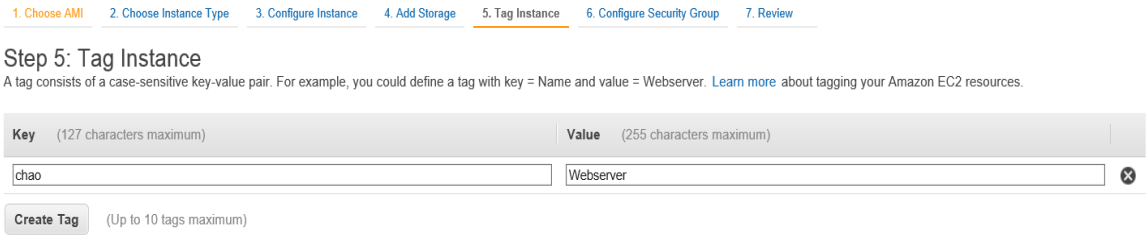


Figure 29

4.2.1.6 Configure Security Group. By created a new Security group to allow EC2 instance has the ability to access to Internet through HTTP and HTTPS, and also allow developers or administrators to perform SSH under specific IP address (My IP). (Figure 30)

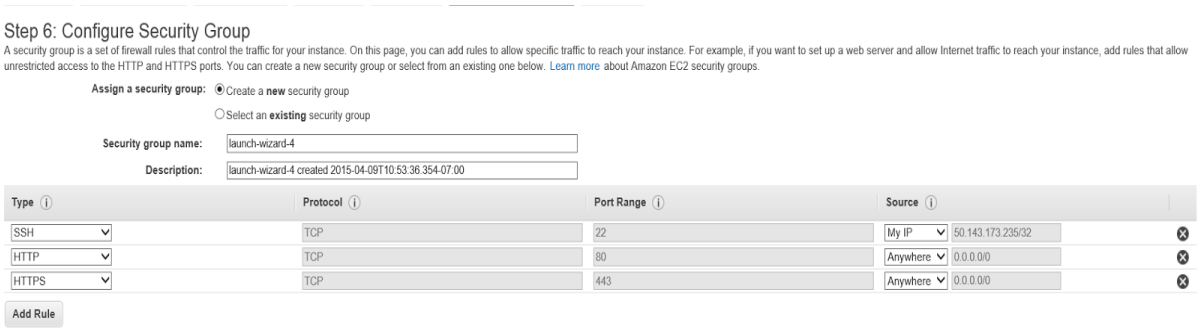


Figure 30

4.2.1.7 Review Instance Launch (Figure 31)

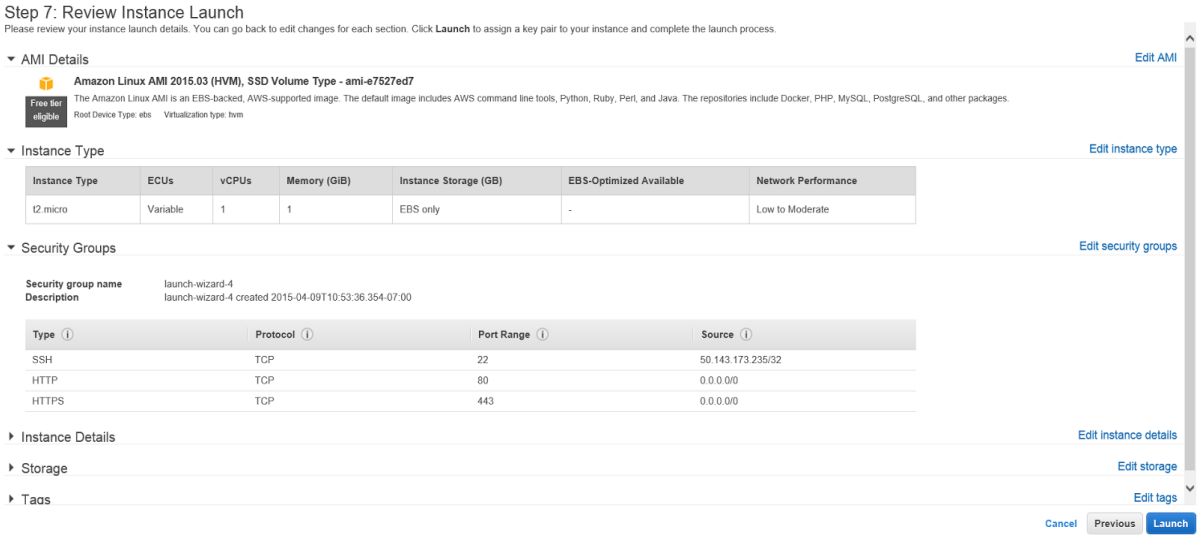


Figure 31

4.2.1.8 Connect to Instance. For Mac OS X or Linux user, choose “A standalone SSH client”(Figure 32) to connect instance through your local computer. For Windows user, choose “A Java SSH Client directly from browser (Java required)”. (Figure 33)

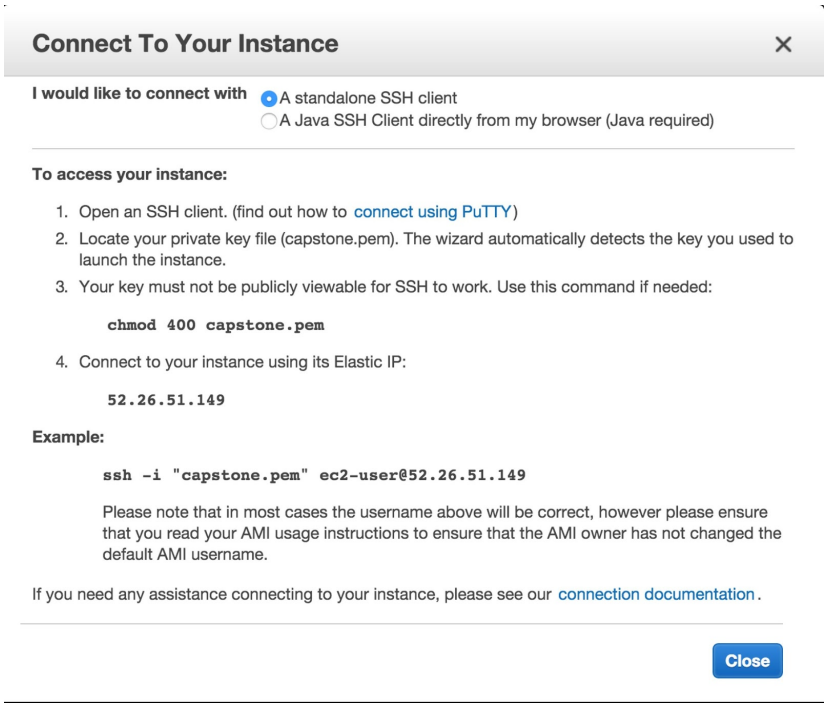


Figure 32

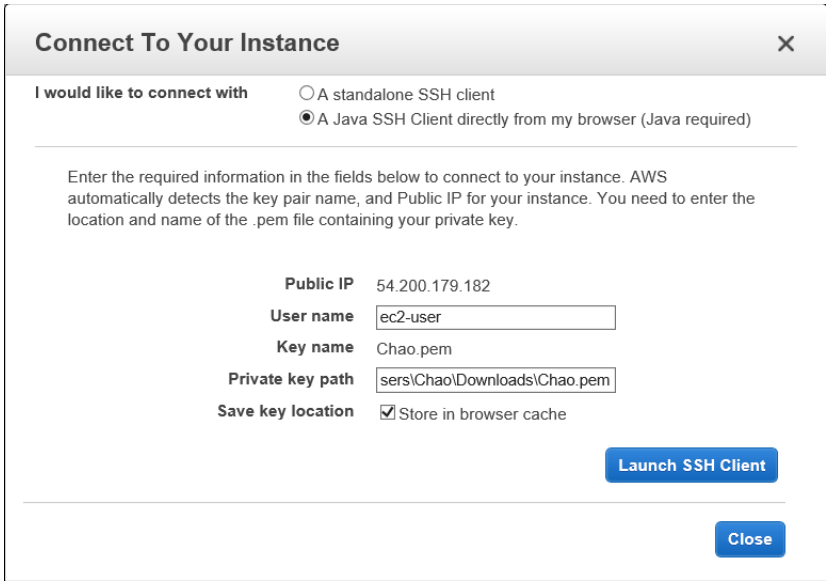


Figure 33

4.2.2 Database: MongoDB

We used MongoDB as the backend database, and stored all users' profile and sensors data into it. In the project, we deployed MongoDB on an EC2 instance, and then exposed the IP port to world.

4.2.2.1 Install MongoDB

Configure the package management system (yum):

Create a /etc/yum.repos.d/mongodb-org-3.0.repo file so that you can install MongoDB directly, using yum.

For the latest stable release of MongoDB

Use the following repository file:

```
[mongodb-org-3.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/3.0/x86_64/
gpgcheck=0
enabled=1
```

Install the MongoDB packages and associated tools:

When you install the packages, you choose whether to install the current release or a previous one. This step provides the commands for both.

To install the latest stable version of MongoDB, issue the following command:

sudo yum install -y mongodb-org

4.2.2.2 Expose MongoDB to world

Start MongoDB:

You can start the mongod process by issuing the following command, however, we allowed all the remote access to our data for the convenience:

sudo service mongod start

4.2.3 Node.js and Express

In the project, we used Node.js and Express framework to build the RESTful APIs. The below listed steps explained how to build RESTful APIs.

The project used express.js and node.js with Mongo DB to build the whole back-end AP. Therefore, the project used MongoClient library to communicate with MongoDB. Moreover, the project used querystring and body-parser library to handle the requests from outside. As considering the project needed some sample APIs, we write the whole codes in a single app.js. The code is list below:

```
var express = require('express');
var MongoClient = require('mongodb').MongoClient;
var app = express();
var qs = require('querystring');
var cors = require('cors');
var bodyParser = require('body-parser');
```

4.2.3.1 Connecting to MongoDB and password encrypt function

We set a group variable for database connection. As considering security, the project included part code to encrypt user's passwords.

```
var url = 'mongodb://52.25.49.72:27017/capstone';
var crypto = require('crypto'),
       algorithm = 'aes-256-ctr',
       password = 'd6F3Efeq';
function encrypt(text){
       var cipher = crypto.createCipher(algorithm,password)
       var crypted = cipher.update(text,'utf8','hex')
       crypted += cipher.final('hex');
       return crypted;
}
function decrypt(text){
       var decipher = crypto.createDecipher(algorithm,password)
       var dec = decipher.update(text,'hex','utf8')
       dec += decipher.final('utf8');
       return dec;
}
```

4.2.3.2 Handling the User login requests

This API handles user login requests. First, users put their username and password into it. After system automatically does a query to the database, it will return code 200 with username if the inputs match the username and password. Or it will return code 401 "Password is not correct". Otherwise, it will return code 404 "Username does not exist" if the system could not find the username in the database.

```
app.post('/user/login',function(request,response){
    if (request.method == 'POST') {
        var body = '';
        request.on('data', function (data) {
            body += data;

            if (body.length > 1e6)
                request.connection.destroy();
        });
        request.on('end', function () {
            var post = qs.parse(body);

        });
    }
    var username = request.body['username'];
    username = username.toLowerCase();
```

```
    var password = encrypt(request.body['password']);
    MongoClient.connect(url,function(err,db){
        if(!err){
            console.log("We are connected");
        }
        var collection = db.collection('user');

collection.find({username:username}).toArray(function(err,data){
            if(data.length==0)
            {
                response.status(404).send({"Message":"Username not
exists"});
                db.close();
            }
            else
            {
                if(password == data[0]["password"])
                {

response.status(200).send({username:data[0]["username"]});
                    db.close();
                }
                else
                {
                    response.status(401).send({"Message":"Password is
not correct"});
                    db.close();
                }
            }
        });
    });
});
```

4.2.3.3 Handling the User register requests

This API handles user register requests. After users submit their register information form, it will return code 200 with username if register succeeds. Otherwise, it will return code 201 “Username exists” if resisted username exists in the database.

```
var username = request.body['username'];
    username = username.toLowerCase();
    var password = encrypt(request.body['password']);
    var email = request.body['email'];
    var toDate = new Date();

    MongoClient.connect(url, function(err, db) {
        if(!err){
            console.log("We are connected");
        }
        var collection = db.collection('user');
```

```
collection.find({username:username}).toArray(function(err,data){
            if(data.length!=0)
            {
                response.status(201).send({"Message":"Username
exists!"});
                db.close();
            }
            else
            {

collection.insert({username:username,password:password,email:email,firs
tname:"",lastname:"",
                    gender:"",age:"",date:toDate});
                    response.status(200).send({"username":username});
                    db.close();
            }
        })
    });
```

4.2.3.4 Handling the get user profile requests

This API displays all users' profile information to users.

```
app.get('/user/profile',function(request,response){
    var username = request.query.username;
    username = username.toLowerCase();

    MongoClient.connect(url,function(err,db){
        if(!err){
            console.log("We are connected");
        }
        var collection = db.collection('user');

collection.find({username:username},{_id:0,password:0}).toArray(functio
n(err,data){
            response.status(200).send(data);
            db.close();
        });
    });

});
```

4.2.3.5 Handling the modify user profile requests

This API accepts users to update their information.

```
app.post('/user/profile',function(request,response){
    if (request.method == 'POST') {
        var body = '';
        request.on('data', function (data) {
            body += data;
            if (body.length > 1e6)
                request.connection.destroy();
        });
        request.on('end', function () {
            var post = qs.parse(body);
        });
    }
    var username = request.body['username'];
    username = username.toLowerCase();
    var firstname = request.body['firstname'];
    var lastname = request.body['lastname'];
    var gender = request.body['gender'];
    var age = request.body['age'];
    MongoClient.connect(url,function(err,db){
        if(!err){
            console.log("We are connected");
        }
        var collection = db.collection('user');

collection.find({username:username}).toArray(function(err,data){
            collection.update({username:username},{
                username:username,password:data[0]["password"],

firstname:firstname,lastname:lastname,date:data[0]["date"],
                gender:gender,age:age
            });
            response.status(200).send({"Message":"Profile update
success"});
            db.close();
        })
    });
});
```

4.2.3.6 Handling the reset password requests

This API handles users' requests to reset the password. It will return code 200 "Password has been updated " if old password has been validated and updated to new password. Otherwise, it will return code 401 "Old password is not correct".

```
app.post('/user/resetPassword',function(request,response){
    if (request.method == 'POST') {
        var body = '';
        request.on('data', function (data) {
            body += data;
            if (body.length > 1e6)
                request.connection.destroy();
        });
        request.on('end', function () {
            var post = qs.parse(body);
        });
    }
    var username = request.body['username'];
    username = username.toLowerCase();
    var oldPassword = encrypt(request.body['oldPassword']);
    var newPassword = encrypt(request.body['newPassword']);
    MongoClient.connect(url,function(err,db){
        if(!err){
            console.log("We are connected");
        }
        var collection = db.collection('user');

collection.find({username:username}).toArray(function(err,data){
            if(oldPassword == data[0]["password"])
            {

collection.update({username:username},{username:username,password:newPa
ssword,

firstname:data[0]["firstname"],lastname:data[0]["lastname"],date:data[0
]["date"],
                    gender:data[0]["gender"],age:data[0]["age"]
                });
                response.status(200).send({"Message":"Password has been
updated"});
                db.close();
            }
            else
            {
                response.status(401).send({"Message":"Old password is
not correct"});
                db.close();
            }
        })
    })
});
```

4.2.3.7 Handling the get user overview requests

This API allows users to select data to view by range of dates. It will return code 200 with users' sleeping data if database matches and finds the user's information. Otherwise, it will return code 400 "No data" if no data information stored related to the dates of the users.

```
app.get('/user/overview',function(request,response){
    var username = request.query.username;
    var sensor = request.query.sensor;
    username = username.toLowerCase();
    sensor = sensor.toLowerCase();
    if(sensor == "motionrecord"){
        MongoClient.connect(url,function(err,db){
            if(!err){
                console.log("We are connected");
            }
            function createObject(propName, propValue){
                this[propName] = propValue;
              }
            var soundQuery = new createObject(sensor,{$exists:true});
            soundQuery.username=username;

            var collection = db.collection('motiondatas');
            var uniqueDate = new Array();


collection.find(soundQuery,{_id:0,time:0}).toArray(function(err,data){
                if(data.length==0)
                {
                    response.status(404).send({"Message":"No data!"});
                    db.close();
                }
                else
                {
                    var date = new Array();
                    for(i=0;i<data.length;i++){
                        date.push(data[i]['date'])
                    }

                    uniqueDate = date.filter(function(elem, pos) {
                        return date.indexOf(elem) == pos;
                    });

                    response.status(200).send(uniqueDate);
```

```
                    db.close();
                }
            });
        });
    }
    else{
        MongoClient.connect(url,function(err,db){
            if(!err){
                console.log("We are connected");
            }

            function createObject(propName, propValue){
                this[propName] = propValue;
              }
            var soundQuery = new createObject(sensor,{$exists:true});
            soundQuery.username=username;

            var collection = db.collection('sensor');
            var uniqueDate = new Array();


collection.find(soundQuery,{_id:0,time:0}).toArray(function(err,data){
                if(data.length==0)
                {
                    response.status(404).send({"Message":"No data!"});
                    db.close();
                }
                else
                {
                    var date = new Array();
                    for(i=0;i<data.length;i++){
                        date.push(data[i]['date'])
                    }

                    uniqueDate = date.filter(function(elem, pos) {
                        return date.indexOf(elem) == pos;
                    });

                    response.status(200).send(uniqueDate);
                    db.close();
                }
            });
        });
    }
});
```

4.2.3.8 Handling the get user sensor data requests

This API gets the user's sleeping data from database, and returns to user.

```
app.get('/user/sensorData',function(request,response){
    var username = request.query.username;
    username = username.toLowerCase();
    var sensor = request.query.sensor;
    if(sensor == "motionrecord"){
        MongoClient.connect(url, function(err, db) {
              if(!err) {
                console.log("We are connected");
              }
              function createObject(propName, propValue){
                this[propName] = propValue;
              }
              var queryObject = new
createObject(sensor,{$exists:true});
              queryObject.username=username;

              var collection = db.collection('motiondatas');
              collection.find(queryObject).toArray(function(err,docs){
                    if(docs.length==0)
                    {
                        response.status(204).send("No data");
                        db.close();
                    }
                    else
                    {
                        response.status(200).send(docs);
                        db.close();
                    }
                });
        });
    }
    else{
        MongoClient.connect(url, function(err, db) {
              if(!err) {
                console.log("We are connected");
              }
              function createObject(propName, propValue){
                this[propName] = propValue;
              }
              var queryObject = new
createObject(sensor,{$exists:true});
              queryObject.username=username;
              var collection = db.collection('sensor');
              collection.find(queryObject).toArray(function(err,docs){
                    if(docs.length==0)
```

```
                    {
                        response.status(204).send("No data");
                        db.close();
                    }
                    else
                    {
                        response.status(200).send(docs);
                        db.close();
                    }
                });
            });
        }
});
```

4.2.4 APIs documents

4.2.4.1 URL

The URL of API server is http://52.26.51.149:3000/

4.2.4.2 General API structure

URL	GET	POST	PUT	DELETE
user/sensorData	Parameters: username, sensor Response: (200) A JSON object includes a list of sensor data (204)Not found	none	none	none
user/login	Parameters: username, password Response: (200) username (401)" password is not correct" (404)" Username nor exists"			
user/register	Parameters: username, password, email Response: (200) username (201) "Username exists!"			

user/resetPassword	Parameters: username, oldPassword, newPassword Response: (200)" Password has been updated (401) "Old password is not correct"			
user/overview	Parameters: username, sensor Response: (200)A JSON object includes the date. (404)Not date			
user/profile	Parameters: username Response: (200)A JSON object includes user's profile data	Parameters: username, firstname, lastname, gender, age Response: (200) 'Profile update success'		

4.2.4.3 API examples

- "/"

Method: GET

Root url, it will return "Hello world testing".

- **user/sensorData**

Method: GET

The url takes two parameters: username and sensor. Right now, the sensor only takes "sensorrecord, motionrecord, lightrecord" as parameters. It will return with a JSON object includes a list of query data and status code 200 if running succeed. Otherwise, it will return status code "204 not found". (Figure 34)

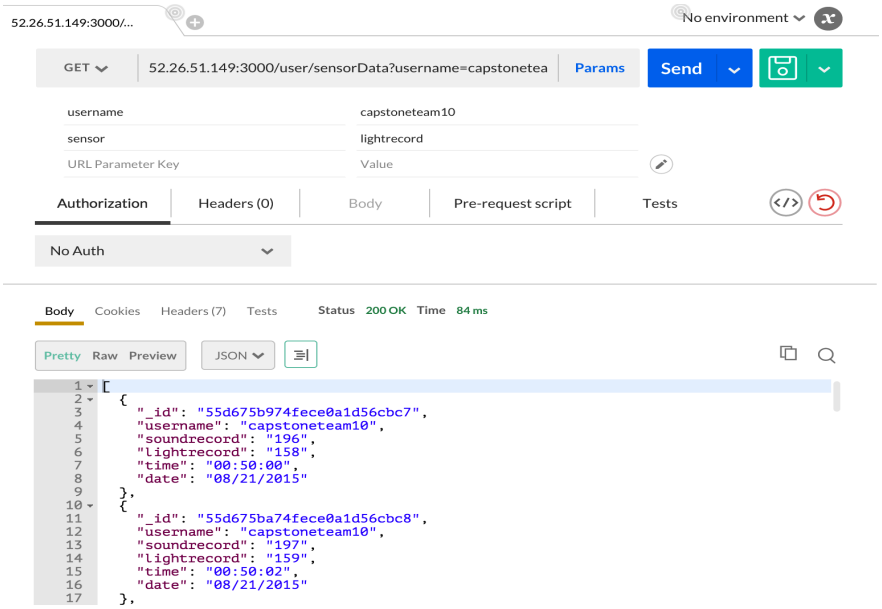


Figure 34 GET

- **user/login**

Method: POST

The url takes two parameters: username and password. It will return status code 200 with username if running succeed; return status code 401 "Password is not correct" if password does not match; return status code 404 "Username does not exist" if username does not exist. (Figure 35)

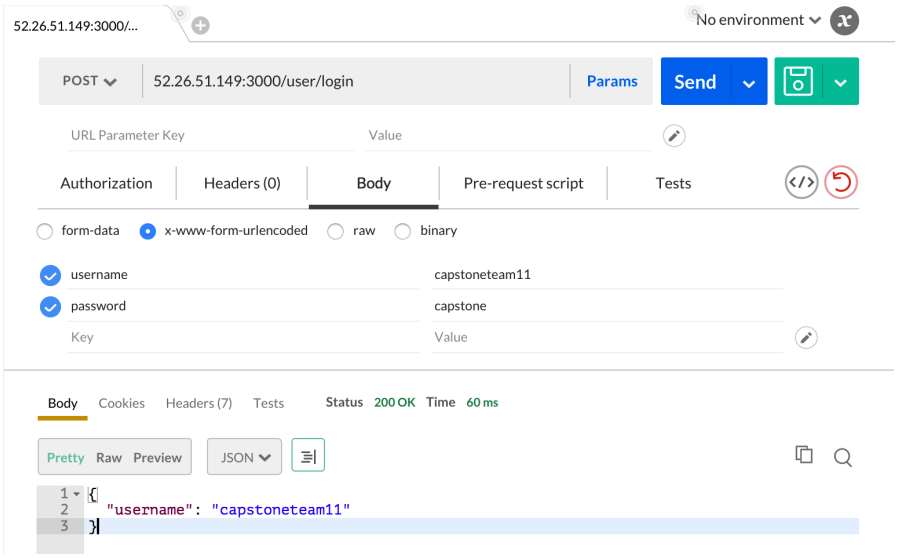


Figure 35 POST (Login)

- **user/register**

Method: POST

The url takes three parameters: username, password and email. It will return status code 200 with username if running succeed. Otherwise, it will return status code 20 with "Username exists!" if username exists in database. (Figure 36)

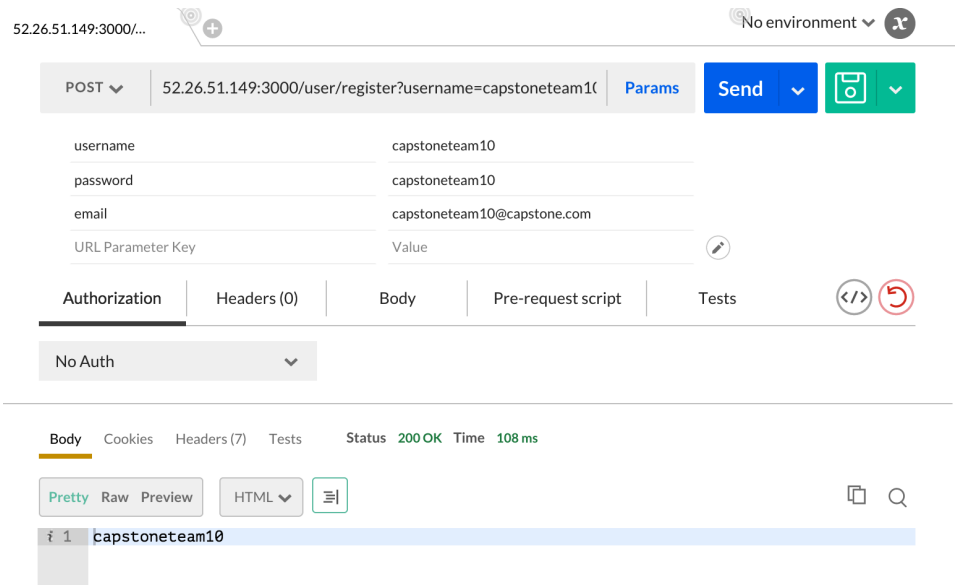


Figure 36 POST (Register)

- **user/resetPassword**

Method: POST

The url takes three parameters: username, oldPassword, newPassword. It will return status code 200 "Password has been updated" if running succeed. Otherwise, it will return status code 401 "Old password does not match" if old Password is incorrect. (Figure 37)

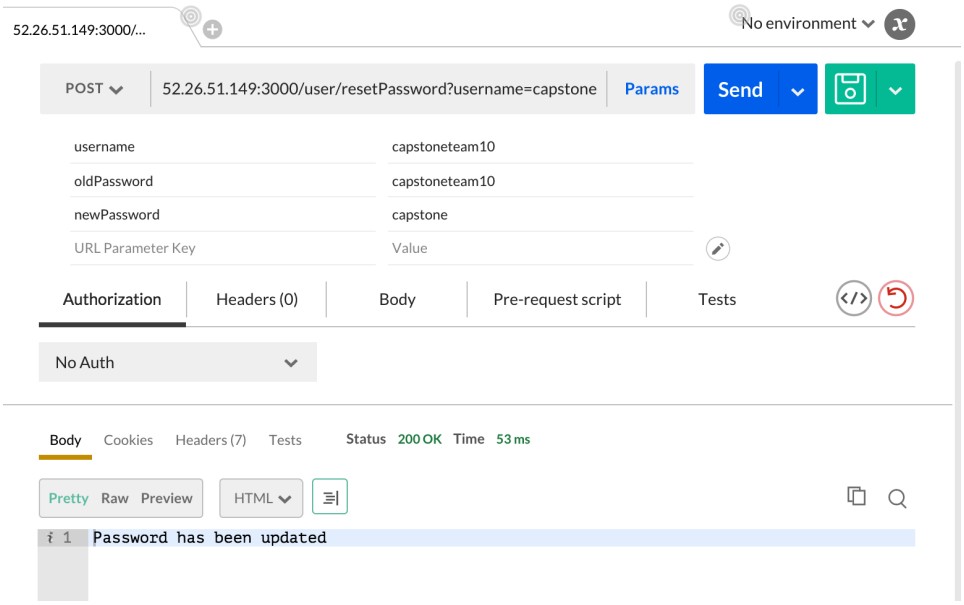


Figure 37 POST (resetPassword)

- **user/profile**

Method: POST

The url takes five parameters: username, firstname, lastname, age and gender. It will return status code 200 “Profile updates succeed” if running succeed. (Figure 38)

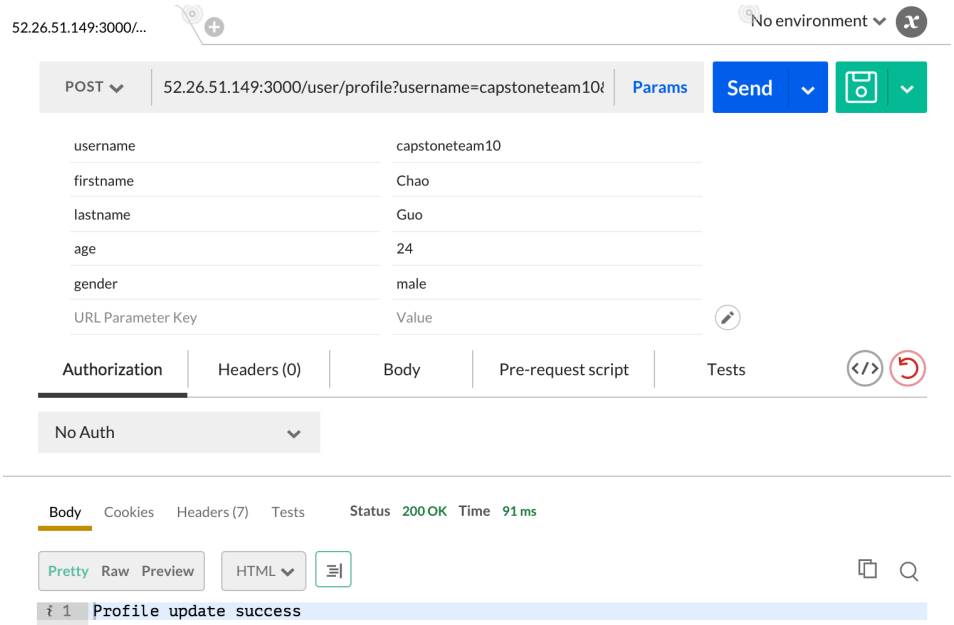


Figure 38 POST (profile)

- **user/profile**

Method: GET

The url takes one parameter: username. It will return status code 200 and a JSON object that includes user's profile data if running succeed. (Figure 39)

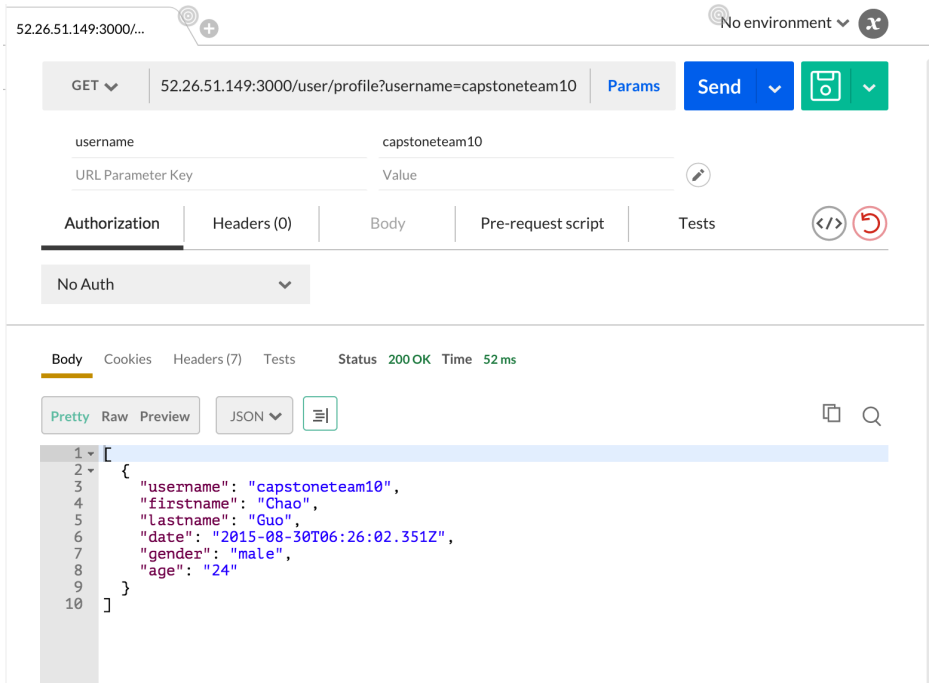


Figure 39 GET (profile)

4.3 Front-end

For the front-end, we used MVC framework Backbone.js to build website, Chart.js to display the data, Bootstrap and JqueryUI to design UI.

4.3.1 Folder structure

The folder structure used Backbone (Figure 40)

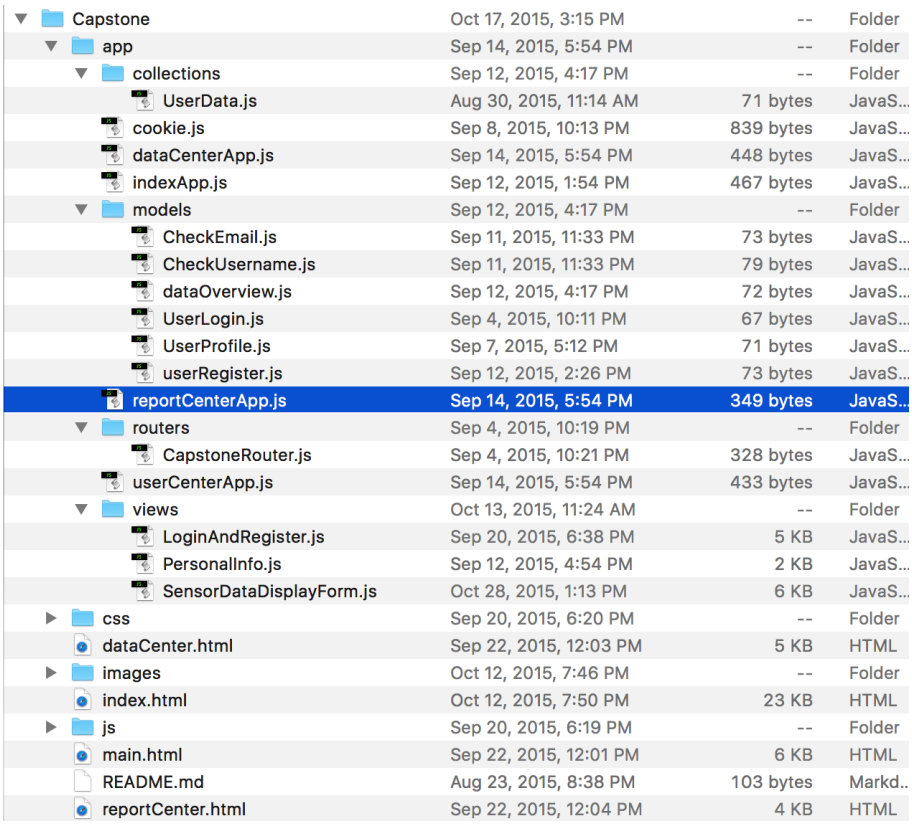


Figure 40 Backbone Structure

4.3.2 Website

The URL of the website is http://52.26.51.149

First, we built a login template by using Backbone and calling underscore template library. The code list below:

```
<script type="text/template" id="login-register-template">
    <div class="banner" id="home">
        <div class="container">
            <div class="header">
            <div class="logo">
                <a href="index.html"><img src="images/logo.png"
alt=""></a>
            </div>
            <div class="navigation">
                 <span class="menu"></span>
                    <ul class="navig">
                        <li><a href="index.html" class="scroll
active">Home</a></li>
                        <li><a href="#about"
class="scroll">Tech</a></li>
```

```
                        <li><a href="#products"
class="scroll">Products</a></li>
                        <li><a href="#pricing"
class="scroll">Features</a></li>
                        <li><a href="#testimonials"
class="scroll">About Us</a></li>
                        <li id="loginBtn"><button type="button"
class="btn btn-primary btn-lg">Login</button></li>
                        <li id="loginDiv"><a href="main.html"
id="displayUsername"><span class="glyphicon glyphicon-
user"></span></a></li>
        <li id="loginDiv"><button type="button" class="btn btn-default
btn-lg" id="logOutBtn">Log out</button></li>
                    </ul>
            </div>
                <div class="clearfix"></div>
            </div>
            <div class="banner-btm">
                <div  id="top" class="callbacks_container">
                    <ul class="rslides" id="slider4">
                        <li>
                            <div class="banner-bottom">
                                <h1>We learn your sleep</h1>
                                <p>Develop an IoT cloud analytics
product that collects real-time streaming data.</p>
                            </div>
                        </li>
                        <li>
                            <div class="banner-bottom">
                                <h1>We study your sleep</h1>
                                <p>Develop rigorous algorithms that
aggregate and analyze sleeping environmental data.</p>
                            </div>
                        </li>
                        <li>
                            <div class="banner-bottom">
                                <h1>We present your sleep</h1>
                                <p>Develop an user-friendly applictaion
that visualizes processed environmental data. </p>
                            </div>
                        </li>
                    </ul>
                </div>
              <div class="clearfix"> </div>
            </div>
            </div>
        </div>
<!-- script-for-login modal -->
        <div class="modal fade" id="myModal" role="dialog">
    <div class="modal-dialog">
      <div class="modal-content" id="loginForm">
        <div class="modal-header" style="padding:35px 50px;">
          <button type="button" class="close" data-
dismiss="modal">&times;</button>
          <h4><span class="glyphicon glyphicon-lock"></span> Login</h4>
        </div>
        <div class="modal-body" style="padding:40px 50px;">
```

```
          <div role="form">
            <div class="form-group">
              <label for="usrname"><span class="glyphicon glyphicon-
user"></span> Username</label>
              <input type="text" class="form-control" id="usrname"
placeholder="Enter username">
            </div>
            <div class="form-group">
              <label for="psw"><span class="glyphicon glyphicon-eye-
open"></span> Password</label>
              <input type="password" class="form-control" id="psw"
placeholder="Enter password">
            </div>
            <div class="checkbox">
              <label><input type="checkbox" value="" checked>Remember
me</label>
            </div>
              <button class="btn btn-primary btn-block" id =
"submitForLogin"><span class="glyphicon glyphicon-off"></span>
Login</button>
          </div>
        </div>
        <div class="modal-footer">
          <button type="submit" class="btn btn-primary btn-default
pull-left" data-dismiss="modal"><span class="glyphicon glyphicon-
remove"></span> Cancel</button>
          <p>Not a member? <a href="#" id = "signUp">Sign Up</a></p>
          <p>Forgot <a href="#">Password?</a></p>
          <p>Forgot <a href="#">Username?</a></p>
        </div>
      </div>

      <div class="modal-content" id="registerForm">
        <div class="modal-header" style="padding:35px 50px;">
          <button type="button" class="close" data-
dismiss="modal">&times;</button>
          <h4><span class="glyphicon glyphicon-lock"></span>
Register</h4>
        </div>
        <div class="modal-body" style="padding:40px 50px;">
          <div role="form">
            <div class="form-group">
              <label for="usrname"><span class="glyphicon glyphicon-
user"></span> Username</label>
              <input type="text" class="form-control" id="usrname"
placeholder="Enter username"
              title="Username can only be numbers, characters, -, _. No
space please ! .">
              <span id = "usernameError"></span>
            </div>
            <div class="form-group">
              <label for="psw"><span class="glyphicon glyphicon-eye-
open"></span> Password</label>
              <input type="password" class="form-control" id="psw"
placeholder="Enter password"
              title="Enter a combination of at least six numbers,
letters and punctuation marks!">
```

```
            <span id = "pswError"></span>
          </div>
          <div class="form-group">
            <label for="psw"><span class="glyphicon glyphicon-eye-
open"></span> Re-Password</label>
            <input type="password" class="form-control" id="rePsw"
placeholder="Enter password">
            <span id = "rePswError"></span>
          </div>
          <div class="form-group">
            <label for="email"><span class="glyphicon glyphicon-
envelope"></span> E-mail</label>
            <input type="email" class="form-control" id="email"
placeholder="Enter E-mail"
            title="Please input a validate E-mail! E-mail is used for
find back password and username">
            <span id = "emailError"></span>
          </div>
            <button class="btn btn-primary btn-block" id =
"submitForRegister" ><span class="glyphicon glyphicon-off" ></span>
Register</button>
          </div>
        </div>
        <div class="modal-footer">
          <button type="submit" class="btn btn-primary btn-default
pull-left" data-dismiss="modal"><span class="glyphicon glyphicon-
remove"></span> Cancel</button>
          <p>Already a member? <a href="#" id ="login">Login</a></p>
        </div>
      </div>
    </div>
  </div>
</script>
```

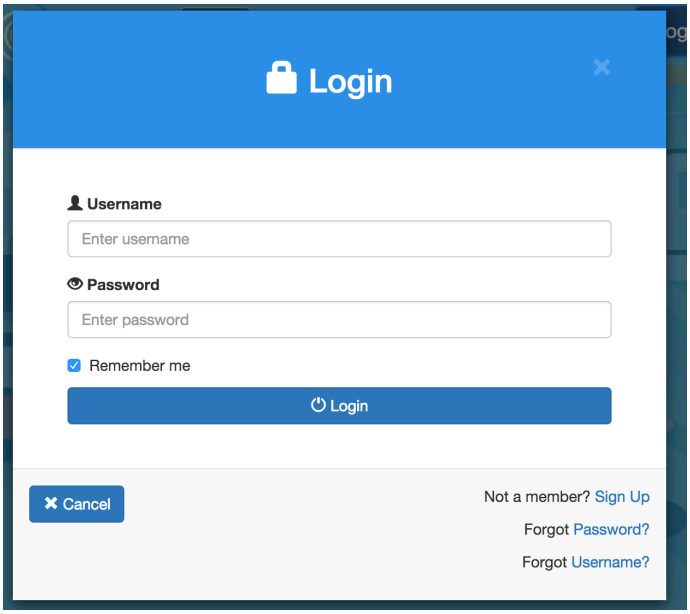


Figure 41 Login

Second, Backbone handles the whole process after user fills out the information (Figure 41) and submits to server. The code lists below:

```
var LoginAndRegister = Backbone.View.extend({
    el:"#loginAndRegisterForm",
    events:{
        'click #signUp':'showSignUpForm',
        'click #login':'showLogin',
        'click #submitForLogin':'submitForLogin',
        'click #logOutBtn':'logOut',
        'click #submitForRegister':'submitForRegister',
        'focusout #registerForm #usrname':'validateUsername',
        'focusout #registerForm #psw':'validatePsw',
        'focusout #registerForm #rePsw':'validateRePsw',
        'focusout #registerForm #email':'validateEmail',
        'keypress #registerForm #usrname':'validateUsername',
        'keypress #registerForm #psw':'validatePsw',
        'keypress #registerForm #rePsw':'validateRePsw',
        'keypress #registerForm #email':'validateEmail'
    },
    showSignUpForm:function(ev){
        $('#loginForm').hide();
        $('#registerForm').show();

    },
    showLogin:function(ev){
        $('#loginForm').show();
        $('#registerForm').hide();
    },
    submitForLogin:function(ev){

        var userLogin = new UserLogin();

        var userDetails = {
            username: $('#usrname').val(),
            password: $('#psw').val(),
        };

        userLogin.save(userDetails,{
            success:function(model,response){
                setCookie("username", response.username, 0.05);
                window.location.replace("main.html");
            },
            error:function(model,response){
                alert(response.responseJSON.Message);
            }
        });

    },
    logOut:function(){
        document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00
UTC";
        this.render();
    },
```

```
    validateUsername:function(){
        var username = $('#registerForm #usrname').val();

        if(username.length<=5){
            $('#usernameError').html("Username length should not be
less than 5 !");
            $('#usernameError').css("color","red");
            cname = '';
        }
        else if(!username.match("^[a-zA-Z0-9_.-]*$")){
            $('#usernameError').html("Username format is not
correct!");
            $('#usernameError').css("color","red");
            cname = '';
        }
        else{
            var checkUsername = new CheckUsername();
            checkUsername.save({username:username},{
                success:function(){
                    cname = 'yes';
                    $('#usernameError').html("Your username is ready
for register!");
                    $('#usernameError').css("color","green");
                    chkreg();
                },
                error:function(model,response){
                    cname = '';

$('#usernameError').html(response.responseJSON.Message);
                    $('#usernameError').css("color","red");
                }
            });
        }
        chkreg();
    },

    validatePsw:function(){
        var psw = $("#registerForm #psw").val();
        if(psw.length<= 5){
            $('#pswError').html("Password length should not be less
than 6 !");
            $('#pswError').css("color","red");
            cpwd1 = '';
        }
        else{
            $('#pswError').html("Good!");
            $('#pswError').css("color","green");
            cpwd1 = 'yes';
            chkreg();
        }
        chkreg();

    },
    validateRePsw:function(){
        var rePsw = $('#registerForm #rePsw').val();
        var psw = $("#registerForm #psw").val();
        if(psw == rePsw){
```

```
            $('#rePswError').html("Good!");
            $('#rePswError').css("color","green");
            cpwd2 = 'yes';
            chkreg();
        }
        else{
            $('#rePswError').html("Password is not same!");
            $('#rePswError').css("color","red");
            cpwd2 = '';
        }
        chkreg();

    },
    validateEmail:function(){
        var email = $("#registerForm #email").val();
        var re =
/^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0
-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-
zA-Z]{2,}))$/;
        if(!re.test(email))
        {
            $('#emailError').html("Please input a correct format E-mail
!");
            $('#emailError').css("color","red");
            cemail = '';

        }
        else {
            var checkEmail = new CheckEmail();
            checkEmail.save({email:email},{
                success:function(){
                    cemail = 'yes';
                    $('#emailError').html("Your E-mail is ready for
register!");
                    $('#emailError').css("color","green");
                    chkreg();
                },
                error:function(model,response){
                    cemail = '';

$('#emailError').html(response.responseJSON.Message);
                    $('#emailError').css("color","red");
                }
            });
        }
        chkreg();
    },
    submitForRegister:function(){

        var userRegister = new UserRegister();
        var userDetails = {
            username: $('#registerForm #usrname').val(),
            password: $('#registerForm #psw').val(),
            email: $('#registerForm #email').val()
        }
        userRegister.save(userDetails,{
            success:function(model,response){
```

```
                setCookie("username", response.username, 0.05);
                location.reload();
            },
            error:function(model,response){
                alert(response.responseJSON.Message);
            }
        });
    },
     render:function(){
        var template = _.template($('#login-register-
template').html());
        $("#loginAndRegisterForm").html(template());
        $('#loginForm').show();
        $('#registerForm').hide();
        $(document).ready(function(){
            $("#loginBtn").click(function(){
                $("#myModal").modal();
            });
        });
        $(function() {
            var tooltips = $( "[title]" ).tooltip({
              position: {
                my: "left top",
                at: "right+5 top-5"
              }
            });
            $( "<button>" )
              .text( "Show help" )
              .button()
              .click(function() {
                tooltips.tooltip( "open" );
              })
              .insertAfter( "form" );
         });

        $('#submitForRegister').attr("disabled", "disabled");
        var user = getCookie("username");
        if (user != "")
        {
            $("#loginBtn").hide();
            $("#loginDiv").show();
            $("#displayUsername").html(user);
        }
        else
        {
            $("#loginBtn").show();
            $("#loginDiv").hide();
            $("#logOutBtn").hide();
        }
    }
});
```

At the end, it will return to main page with succeed status. (Figure 42)

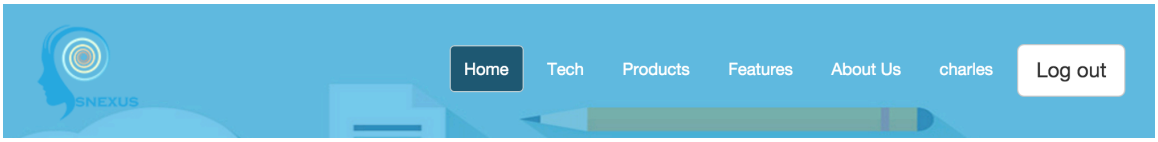


Figure 42 Login Succeed

4.3.3 The User Center page

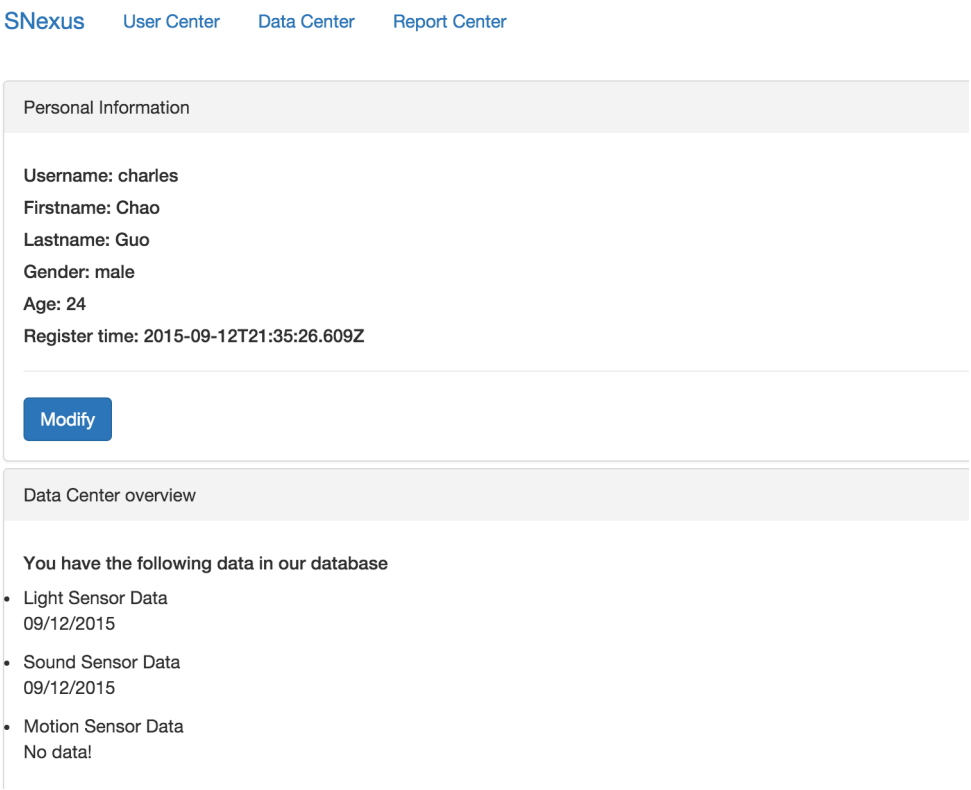


Figure 43 User Center

This page built with Backbone, and the underscore template codes are list below:

```
<script type="text/template" id="userCenterPersonalInfo-template">
<div class="container">
  <div class="panel-group">
    <div class="panel panel-default">
      <div class="panel-heading">Personal Information</div>
      <div class="panel-body">
          <div id="personalInfo">
              <h5>Username: <%= username %></h5>
              <h5>Firstname: <%= firstname %></h5>
```

```
                <h5>Lastname: <%= lastname %></h5>
                <h5>Gender: <%= gender %></h5>
                <h5>Age: <%= age %></h5>
                <h5>Register time: <%= date %></h5>
            <hr />
            <button type="button" class="btn btn-primary" id
='modifyPersonalInfo' >Modify</button>
            </div>

            <div id="personalInforModify" style="display:none" >
            <label for="Firstname">Firstname:</label>
            <input type="text" id="firstname" placeholder="Enter
Firstname" value = <%= firstname %> >
            <br>
            <label for="Lasttname">Lastname:</label>
            <input type="text" id="lastname" placeholder="Enter Lastname"
value = <%= lastname %> >
            <br>
            <label for="Gender">Gender:</label>
            <input type="radio" name="sex" class = "gender" value="male"
checked> Male
            <input type="radio" name="sex" class = "gender"
value="female" > Female
            <br>
            <label for="Age">Age:</label>
            <input type="number" id="age" placeholder="Enter Age" value =
<%= age %> >
            <hr />
            <button type="button" class="btn btn-primary" id
='cancelModifyPersonalInfo' >Cancel</button>
            <button type="button" class="btn btn-primary" id
='saveModifyPersonalInfo' >Save</button>
            </div>

      </div>
    </div>
    <div class="panel panel-default">
      <div class="panel-heading">Data Center overview</div>
      <div class="panel-body">
      <h5>You have the following data in our database</h5>
      <ul>
      <li>Light Sensor Data</li>
      <p id = "lightOverview"></p>
      <li>Sound Sensor Data</li>
      <p id = "soundOverview"></p>
      <li>Motion Sensor Data</li>
      <p id = "motionOverview"></p>
      </ul>
      </div>
```

```
    </div>
    <div class="panel panel-default">
      <div class="panel-heading">Report overview</div>
      <div class="panel-body">To be continue...</div>
    </div>
  </div>
</div>
</script>
```

Next, Backbone view codes, these codes render the data from server and display them to DOM.

```
var PersonalInfoView = Backbone.View.extend({
    el:"#personal",
    events:{
        'click #modifyPersonalInfo':'modifyPersonalInfo',
        'click #cancelModifyPersonalInfo':'cancelModifyPersonalInfo',
        'click #saveModifyPersonalInfo':'saveModifyPersonalInfo',
    },
    modifyPersonalInfo:function(){
        $('#personalInfo').hide();
        $('#personalInforModify').show();
    },
    cancelModifyPersonalInfo:function(){
        this.render();
    },
    saveModifyPersonalInfo:function(ev){
        var user = getCookie("username");
        var userProfile = new UserProfile();
        var userProfileDetails = {
            username: user,
            firstname: $('#firstname').val(),
            lastname: $('#lastname').val(),
            gender: $("input[type='radio'].gender:checked").val(),
            age: $('#age').val(),
        };
        userProfile.save(userProfileDetails,{
            success:function(){
                var personalInfoView = new PersonalInfoView();
                personalInfoView.render();
            },
        })
    },


    render:function(){
```

```
        var template = _.template($('#userCenterPersonalInfo-
template').html());
        var user = getCookie("username");
        var userProfile = new UserProfile();
        userProfile.fetch({data:$.param({username: user}),
            success: function(userProfile){

$("#personal").html(template({username:userProfile.attributes[0].userna
me,firstname:userProfile.attributes[0].firstname,

lastname:userProfile.attributes[0].lastname,gender:userProfile.attribut
es[0].gender,age:userProfile.attributes[0].age,date:
userProfile.attributes[0].date}));
            },
        });
        var dataOverview = new DataOverview();
        dataOverview.fetch({data:$.param({username:
user,sensor:"lightrecord"}),
            success:function(model, response){
                $('#lightOverview').html(response.toString());

            },
            error:function(model, response){
                $('#lightOverview').html(response.responseText);
            },
        });
        dataOverview.fetch({data:$.param({username:
user,sensor:"soundrecord"}),
            success:function(model, response){
                $('#soundOverview').html(response.toString());

            },
            error:function(model, response){
                $('#soundOverview').html(response.responseText);
            },
        });
        dataOverview.fetch({data:$.param({username:
user,sensor:"motionrecord"}),
            success:function(model, response){
                $('#motionOverview').html(response.toString());

            },
            error:function(model, response){
                $('#motionOverview').html(response.responseText);
            },
        });
    },
})
```

4.3.4 Data Center page

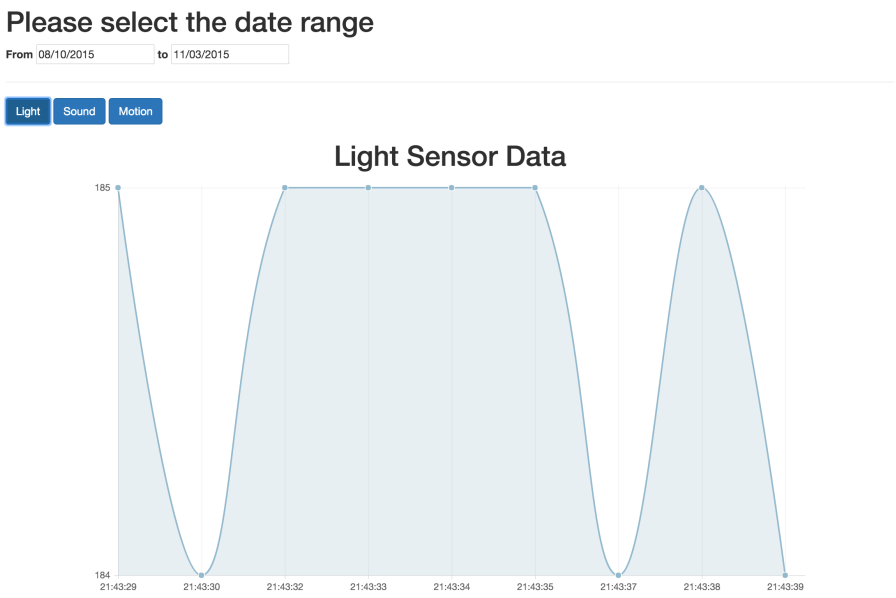


Figure 44 Data Center

This page (Figure 44) built with Backbone, and used chart.js to render the data to charts.

First, by using the underscore template codes:

```
<script type="text/template" id="searchForm-template">
<h1>Please select the date range</h1>
    <div class="form-group">
          <label for="from">From</label>
          <input type="text" id="from" name="from">
          <label for="to">to</label>
          <input type="text" id="to" name="to">
          <hr />
          <button type="button" class="btn btn-primary" id
='submitForLightData' data-sensor="lightrecord" >Light</button>
          <button type="button" class="btn btn-primary" id
='submitForSoundData' data-sensor="soundrecord" >Sound</button>
          <button type="button" class="btn btn-primary" id
='submitForMotionData' data-sensor="motionrecord" >Motion</button>
    </div>
    <div id= "lightChartDiv" style="display:none">
    <h1>Light Sensor Data</h1>
        <canvas id="lightChart" height="350" width="600"></canvas>
    </div>
    <hr />
    <div id= "soundChartDiv" style="display:none">
    <h1>Sound Sensor Data</h1>
```

```
        <canvas id="soundChart" height="450" width="600"></canvas>
    </div>
    <hr />
    <div id= "motionChartDiv" style="display:none">
    <h1>Motion Sensor Data</h1>
        <canvas id="motionChart" height="450" width="600"></canvas>
    </div>
</div>
</script>
```

Next, the Backbone views codes, calls the chart.js library, and renders all the data to charts.

```
var SensorDataDisplayForm = Backbone.View.extend({
    el:"#display-data",
    events:{
        'click #submitForLightData':'displayLightData',
        'click #submitForSoundData':'displaySoundData',
        'click #submitForMotionData':'displayMotionData',
    },
    displayLightData:function(ev){
        var startDate = new Date($('#from').val());
        var endDate = new Date($('#to').val());
        var username = getCookie("username");
        var time = new Array();;
        var light = new Array();
        var sensor = $(ev.target).data('sensor');
        var userData = new UserData();
         userData.fetch({
            data: $.param({username:username,sensor:sensor}),
            success: function (userData) {
                for (i = 0; i<userData.models.length; i++)
                {
                    var dataDate = new
Date(userData.models[i].attributes.date);
                    if(startDate <= dataDate && dataDate <= endDate)
                    {
                        time.push(userData.models[i].attributes.time);

light.push(userData.models[i].attributes.lightrecord);
                    }
                }
                if(time.length == 0)
                {
                    $( "#errDateDialog" ).dialog();
                    return;
                }
                if(time.length>10)
                {
                    var j = 0;
                    for(i=0;i<=time.length;i++)
```

```
                    {
                        if(i!=Math.floor(time.length*(j/8)))
                        {
                            time[i] = "";
                        }
                        else
                        {
                            j++;
                        }
                    }
                }

                var lightChartData = {
                    labels : time,
                    datasets : [
                        {
                            label: "light Data",
                            fillColor : "rgba(151,187,205,0.2)",
                            strokeColor : "rgba(151,187,205,1)",
                            pointColor : "rgba(151,187,205,1)",
                            pointStrokeColor : "#fff",
                            pointHighlightFill : "#fff",
                            pointHighlightStroke :
"rgba(151,187,205,1)",
                            data : light
                        }
                    ]
                }

                $('#lightChartDiv').show();

                var ctx =
document.getElementById("lightChart").getContext("2d");
                window.myLine = new Chart(ctx).Line(lightChartData, {
                    responsive: true
                });
            }
        });
    },
    displaySoundData:function(ev){
        var startDate = new Date($('#from').val());
        var endDate = new Date($('#to').val());
        var username = getCookie("username");
        var time = new Array();;
        var sound = new Array();
        var sensor = $(ev.target).data('sensor');
        var userData = new UserData();
         userData.fetch({
            data: $.param({username:username,sensor:sensor}),
            success: function (userData) {
                for (i = 0; i<userData.models.length; i++)
                {
                    var dataDate = new
Date(userData.models[i].attributes.date);
                    if(startDate <= dataDate && dataDate <= endDate)
                    {
                        time.push(userData.models[i].attributes.time);
```

```
sound.push(userData.models[i].attributes.soundrecord);
                    }
                }
                if(time.length==0)
                {
                    $( "#errDateDialog" ).dialog();
                    return;
                }

                if(time.length>10)
                {
                    var j = 0;
                    for(i=0;i<=time.length;i++)
                    {
                        if(i!=Math.floor(time.length*(j/10)))
                        {
                            time[i] = "";
                        }
                        else
                        {
                            j++;
                        }
                    }
                }
                var soundChartData = {
                    labels : time,
                    datasets : [
                        {
                            label: "light Data",
                            fillColor : "rgba(151,187,205,0.2)",
                            strokeColor : "rgba(151,187,205,1)",
                            pointColor : "rgba(151,187,205,1)",
                            pointStrokeColor : "#fff",
                            pointHighlightFill : "#fff",
                            pointHighlightStroke :
"rgba(151,187,205,1)",
                            data : sound
                        }
                    ]
                }
                $('#soundChartDiv').show();
                var ctx =
document.getElementById("soundChart").getContext("2d");
                window.myLine = new Chart(ctx).Line(soundChartData, {
                    responsive: true
                });
            }
        });
    },
    displayMotionData:function(ev){
        var startDate = new Date($('#from').val());
        var endDate = new Date($('#to').val());
        var username = getCookie("username");
        var time = new Array();;
        var motion = new Array();
        var sensor = $(ev.target).data('sensor');
```

```
        var userData = new UserData();
         userData.fetch({
            data: $.param({username:username,sensor:sensor}),
            success: function (userData) {
                for (i = 0; i<userData.models.length; i++)
                {
                    var dataDate = new
Date(userData.models[i].attributes.date);
                    if(startDate <= dataDate && dataDate <= endDate)
                    {
                        time.push(userData.models[i].attributes.time);

motion.push(userData.models[i].attributes.motionrecord);
                    }
                }
                if(time.length==0)
                {
                    $( "#errDateDialog" ).dialog();
                    return;
                }

                if(time.length>10)
                {
                    var j = 0;
                    for(i=0;i<=time.length;i++)
                    {
                        if(i!=Math.floor(time.length*(j/10)))
                        {
                            time[i] = "";
                        }
                        else
                        {
                            j++;
                        }
                    }
                }
                var motionChartData = {
                    labels : time,
                    datasets : [
                        {
                            label: "light Data",
                            fillColor : "rgba(151,187,205,0.2)",
                            strokeColor : "rgba(151,187,205,1)",
                            pointColor : "rgba(151,187,205,1)",
                            pointStrokeColor : "#fff",
                            pointHighlightFill : "#fff",
                            pointHighlightStroke :
"rgba(151,187,205,1)",
                            data : motion
                        }
                    ]
                }
                $('#motionChartDiv').show();
                var ctx =
document.getElementById("motionChart").getContext("2d");
                window.myLine = new Chart(ctx).Line(motionChartData, {
                    responsive: true
```

```
                });
            }
        });
    },
    render:function(){
        var template = _.template($('#searchForm-template').html());
        $("#display-data").html(template());

        $(function() {
            $( "#from" ).datepicker({
              defaultDate: "+1w",
              changeMonth: true,
              numberOfMonths: 3,
              onClose: function( selectedDate ) {
                $( "#to" ).datepicker( "option", "minDate",
selectedDate );
              }
            });
            $( "#to" ).datepicker({
              defaultDate: "+1w",
              changeMonth: true,
              numberOfMonths: 3,
              onClose: function( selectedDate ) {
                $( "#from" ).datepicker( "option", "maxDate",
selectedDate );
              }
            });
        });
    },
});
```

4.4 Website Structure Summary

- A website to show data for users
- The MEAN Stack, for MongoDB, ExpressJS, AngularJS and Node.js
- The “MEBN” Stack, which B represents Backbone.js
- Server side, node.js and express.js framework with mongoDB to build RESTful APIs
- Client side, backbone.js framework, chart.js and bootstrap
- Deployed on Amazon Web Services

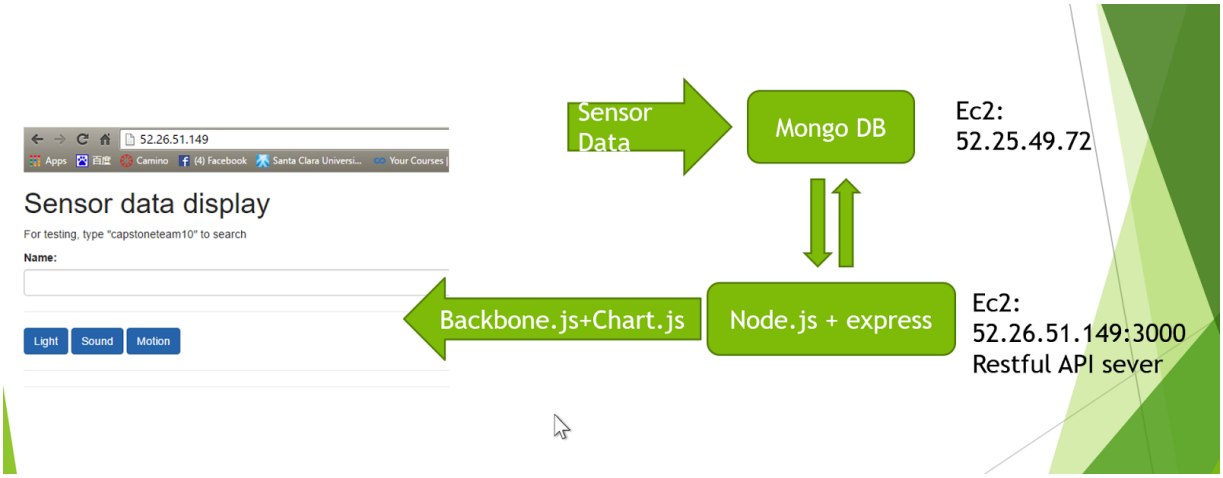


Figure 45 Website Structure

4.5 Data analysis

Based on the motion data captured by 3-axis accelerator MPU-6050, we developed some analysis to form a general understanding of our sleep quality in one night.

In the project, we used motion level as the indicator of sleep quality. The more obvious of the movement, the worse of the sleep quality. To use this principle, we have two assumptions as follows:

- People will have less movement while in deep sleep and will have more movement in light sleep.
- Either of the wrists will have the same intensity of movement as the body.

Here, we used the motion data from one night as an example to develop the analysis.

4.5.1 In the first part, we calculated the accumulated deep sleep time.

4.5.1.1 Create one table in SQL Server and Import the motion data to a table

```
create table sleep_motion_original  -- Create table
(username     nvarchar(30),
motiontime   nvarchar(20),
motionaction FLOAT,
```

```
x_motion    float,
y_motion    float,
z_motion    float)

BULK INSERT sleep_motion_original  --Import motion data
FROM 'D:\academic\Capstone\Instruction\Data\Lumira_data\test.csv'
WITH(
 -- firstrow = 2,
        FIELDTERMINATOR = ',',
        ROWTERMINATOR = '0x0a'
)
```

4.5.1.2 Process the raw data to make them meaningful and meet the requirements for statistical analysis.

In the project, we processed the data by calculating the incremental value between rows. Why do we need to use incremental data? That's because if we want to use statistical methods for analysis, we have to ensure the data distribution is normalized. However, if we plot the raw data (data from one hour), it's obvious that there are some steps. That's because once we turn around on the bed, the sensor will work at a new posture and the acceleration data will fluctuate at a different level. (Figure 46)

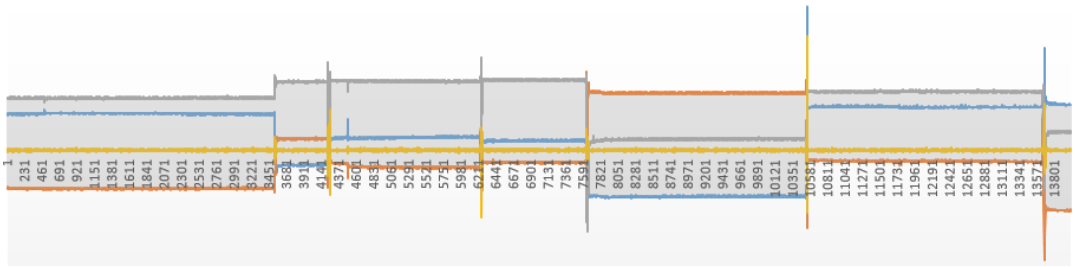


Figure 46 data fluctuated in different levels

To solve this problem, we developed the incremental acceleration data of each axis and plot them (data from the same one hour) in the below chart Obviously. (Figure 47)

Figure 47 incremental acceleration data

```
create table dbo.sleep_motion_t
(username     nvarchar(30),
motiontime   nvarchar(20),
motionaction FLOAT
)

declare @username     nvarchar(30)
declare @motiontime   nvarchar(20)
declare @first_motion float
declare @motionaction float

declare my_cur Cursor for
select username,motiontime,motionaction from dbo.sleep_motion_original

open my_cur
fetch next from my_cur into @username,@motiontime,@first_motion

while @@fetch_status=0
begin

 Fetch Next From my_cur Into @username,@motiontime,@motionaction

  PRINT  @first_motion

  print @motionaction

  insert into sleep_motion_t Values(@username,@motiontime,@motionaction-
@first_motion)

  PRINT ':----'
  set  @first_motion= @motionaction
end
Close my_cur
Deallocate my_cur
```

4.5.1.3 Simplify the data by each minute

Since our sensor will capture the acceleration data 4 times per second, which means it will create 14,000+ rows of data per hour. To make the analysis easy to read, we used one value per minute to indicate the movement level.

In our model, the acceleration data are incremental values; simply calculating the sum of the data will be non-sense. As a result, we used the sum of the absolute value to indicate the movement per minute. The SQL codes are as follows:

```
create view sleep_motion_v as
select convert(time,(CASE CHARINDEX(':',motiontime)
                    WHEN 2 THEN SUBSTRING(motiontime,1,Len(motiontime)-
(CHARINDEX(':',motiontime)+1))
                    ELSE SUBSTRING(motiontime,1,Len(motiontime)-
CHARINDEX(':',motiontime))
                END),103) as motionactionmin ,
 round(sum(abs(motionaction)),2) as motionaction
from dbo.sleep_motion_t
group by convert(time,(CASE CHARINDEX(':',motiontime)
                    WHEN 2 THEN SUBSTRING(motiontime,1,Len(motiontime)-
(CHARINDEX(':',motiontime)+1))
                    ELSE SUBSTRING(motiontime,1,Len(motiontime)-
CHARINDEX(':',motiontime))
                END),103)
```

Based on the calculation results, we plot the data for the whole night (GMT time) as below (Figure 48):

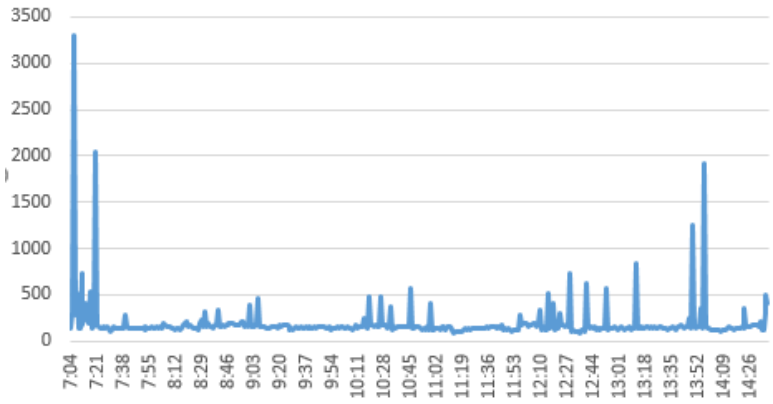


Figure 48

4.5.1.4 Find out the roll-over moments and the time

To calculate the accumulated deep sleep time based on the model we cited *[Roll-over Detection and Sleep Quality Measurement using a Wearable Sensor]*, we need to determine the deep sleep status based on the movement value.

We used a detailed term "roll-over" here to replace the general term "motion" we used before. Roll-over is defined as unconscious motions during sleep such as rotational body movements. In the project, we defined roll-over as a series of trunk movements from static state to static state during sleep and these motions can be effectively detected by motion sensor. After several experiments, in this project we set 350/min as the threshold value, which means if the motion value exceeds 350 per min, in that minute the motion is obvious enough to be considered as roll-over.

The below SQL code selected the time and motion value if the value is big enough to be considered as roll-over.

```
create table sleep_motion_a
(
motiontime   time,
motionaction FLOAT)

USE [Capstone]
declare @motiontime time
declare @motionaction float

declare my_cur Cursor for
select * from sleep_motion_v order by 1

open my_cur
fetch next from my_cur into @motiontime,@motionaction
while @@fetch_status=0
begin
 if @motionaction>= 350

 begin
```

```
 insert into dbo.sleep_motion_a values(@motiontime,@motionaction)
end

Fetch Next From my_cur Into @motiontime,@motionaction
End
Close my_cur
Deallocate my_cur
```

4.5.1.5 Calculate Deep Sleep Time

We classified sleep into two stages, deep sleep and light sleep, based on the frequency of roll-overs, which increased during light sleep and decreased during deep sleep. We have recorded the time (accurate to minute) when roll-overs were detected by sensors. The intervals between roll-overs were calculated and compared with the threshold shown in equation below. Intervals exceeding the threshold were classified into deep sleep, and intervals below the thresholds were classified into light sleep. According to the scientific paper, we used the threshold as: T4 = 20.

```
USE [Capstone]

declare @firstmotiontime time
declare @motiontime time
declare @motionaction float
declare @total_time  float = 0

declare my_cur Cursor for
select * from sleep_motion_a order by 1

open my_cur
fetch next from my_cur into @firstmotiontime,@motionaction
while @@fetch_status=0
begin
 Fetch Next From my_cur Into @motiontime,@motionaction

  PRINT  @firstmotiontime

  print @motiontime
  print datediff(mi,@firstmotiontime,@motiontime)
  if datediff(mi,@firstmotiontime,@motiontime) >=20
```

```
 begin
  set @total_time = @total_time+datediff(mi,@firstmotiontime,@motiontime)

 end

 PRINT @total_time

 PRINT ':----'
 set  @firstmotiontime= @motiontime
end
Close my_cur
Deallocate my_cur
```

After running the above codes, we can get the accumulated deep sleep time length as 327 minutes (Figure 49). Since the whole sleep lasted for 454 minutes, the deep sleep rate is 72%.

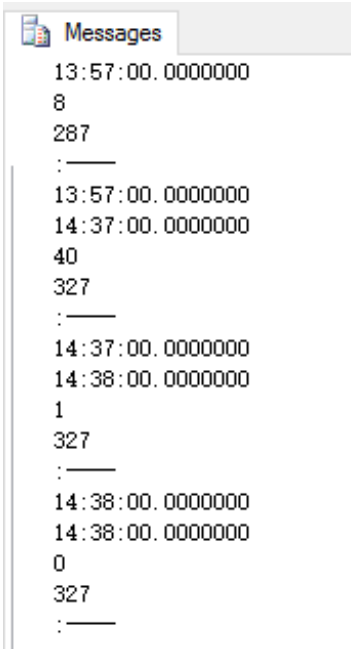


Figure 49 Deep sleep time lengths

4.5.2 Compared Sleep Quality by hour

The methodology is as below:

a. Transform the raw data to make them meaningful and normally distributed. Similarly as before, we'd like to use the incremental value to indicate the movement's intensity among different timing points.

b. Calculate the average value and standard deviation of the dataset.

c. Set the scope of normal movements.

d. Count the number of outliers, which means the movement value exceeds the scope we set.(Figure 50)

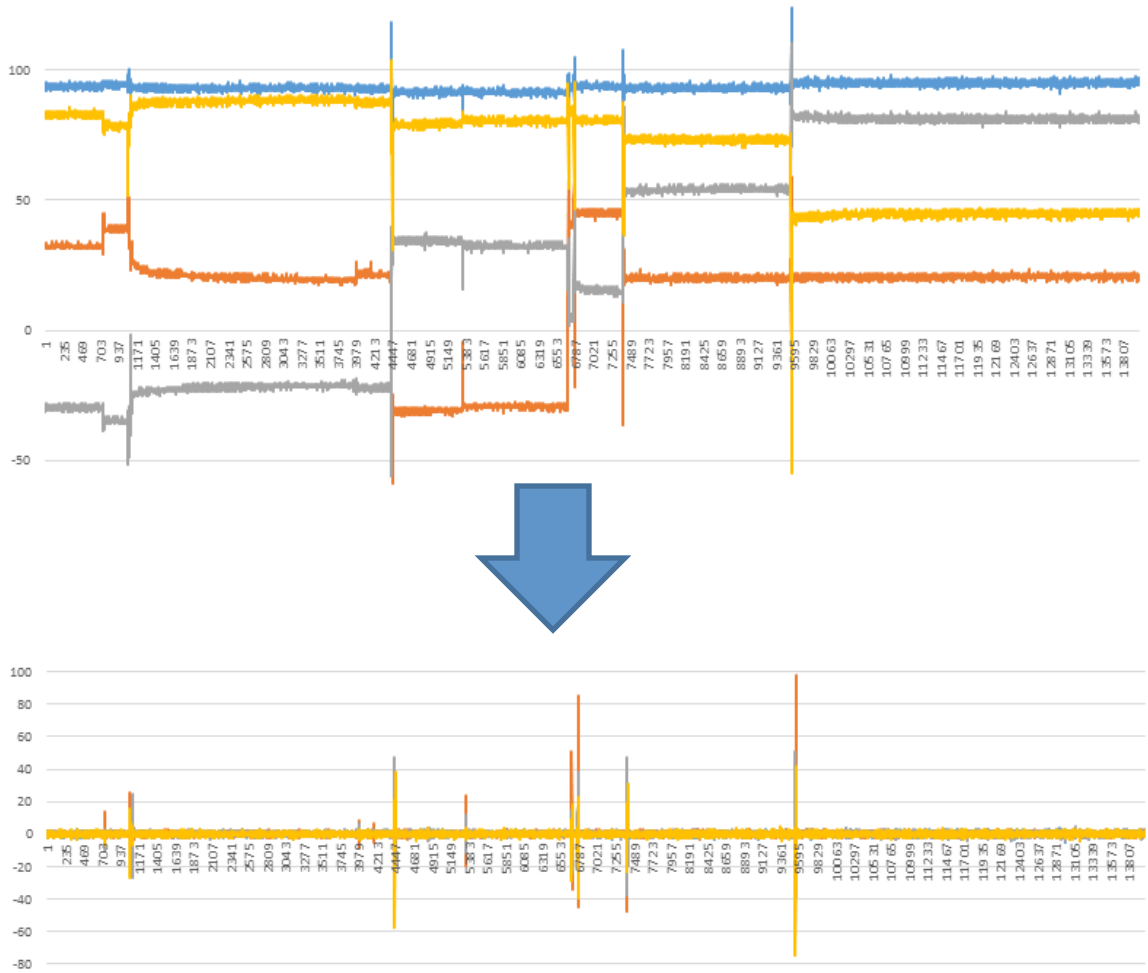


Figure 50

e. Compare the number of outliers by hour (Figure 51). To confirm if we can use the comprehensive acceleration value, we plotted the number of outliers by hour from comprehensive acceleration value as CNT-1S and the ones from three axis as CNT-1-X, CNT-1-Y and CNT-1-Z, respectively. From the below table, we learned that the trends of the curve from different dimensions are similar. Consequently, to simplify the analysis, we only use the comprehensive acceleration value in this project.

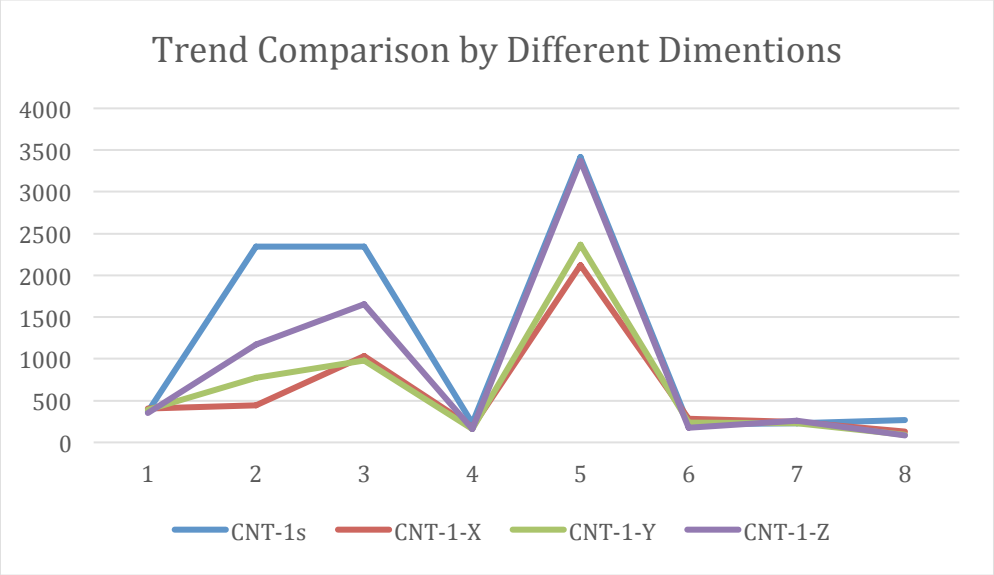


Figure 51

To realize the above function, we used the code as below:

```
var express = require('express');
var router = express.Router();
var mongoose = require('mongoose');
var MongoClient = require('mongodb').MongoClient;
```

```
//create a new collection  ===>  motiondatas   , which is userd to store new motion data.
var motion_data_model=mongoose.model('motiondata',{
  username:String,
  date: {
    type:String,
  },
  time:{
    type:String,
  },
  motionrecord:{
    type:Number,
  }
});

var url = 'mongodb://52.25.49.72:27017/capstone';

// connect the source data (before processed)
MongoClient.connect(url, function(err, db) {
  if(!err) {
    console.log("We are connected");
  }

  function createObject(propName, propValue){
    this[propName] = propValue;
  }

//time=========================start=======================
  var currentDate = new Date();
  var previousDate = currentDate - 86400000;
  function convertDate(inputFormat) {
    function pad(s) { return (s < 10) ? '0' + s : s; }
```

```
  var d = new Date(inputFormat);
  return [pad(d.getMonth()+1),pad(d.getDate()), d.getFullYear()].join('/');
 }

 var nextday=convertDate(currentDate);
 var thisday=convertDate(previousDate);
 console.log(nextday);
 console.log(thisday);
//time=========================end======================
 var user_collection = db.collection('user');
 var sensor_collection = db.collection('sensor');

 mongoose.connect(url,function(err){
  //search all the usernames
  user_collection.find({},function(err,users){
   users.forEach(function(ele){
    console.log("username", ele.username);

    // execute the two sleep data for nextday and thisday !!!!!!!!!!!! important
    process_nextday_data.call(this,ele.username,nextday);
    process_thisday_data.call(this,ele.username,thisday);
    // !!!!!!!!!!!! important

    function process_nextday_data(username,nextday){
     var queryObject_nextday = new createObject("motionrecord",{$exists:true});
     queryObject_nextday.username=username;
     queryObject_nextday.date=nextday;
     sensor_collection.find(queryObject_nextday).toArray(function(err,docs){
      console.log("next query",queryObject_nextday);
      if(docs.length==0)
      {
```

```
  console.log("next day No data");
}
else
{
  var nextday_data = docs.filter(function(element){
    var hour = element.time.split(':')[0];
    return hour < '11';   //will collect time before 11:00
  });
  console.log("the first data of nextday",nextday_data[0]);

  var hourlist = ["01","02","03","04","05","06","07","08","09","10"];
  var dataobj  ={
    username:queryObject_nextday.username,
    date:nextday,
  }

  for(var i =0; i<10;i++) {
    var count = 0;
    var eachhour_data = nextday_data.filter(function(element){
      var hour = element.time.split(':')[0];
      return hour == hourlist[i];
    });
    if(eachhour_data.length==0){
      dataobj.time=hourlist[i];
      dataobj.motionrecord=0;
      (new motion_data_model(dataobj)).save();
    }else{
      var eachhour_data_afterprocess = [];
      eachhour_data.reduce(function (first, second) {
        eachhour_data_afterprocess.push(second.motionrecord - first.motionrecord);
        return second;
```

```
            });
            var avg_stdDev = standardDeviation(eachhour_data_afterprocess);
            console.log("next day",avg_stdDev);

            eachhour_data_afterprocess.forEach(function(element){
              if(element > avg_stdDev[0] + avg_stdDev[1] || element < avg_stdDev[0] -
avg_stdDev[1]){
                count++
              };
            });
            dataobj.time=hourlist[i];
            dataobj.motionrecord=count;
            (new motion_data_model(dataobj)).save();
          }
        };
      };
    });
  }
  function process_thisday_data(username,thisday){
    var queryObject_thisday = new createObject("motionrecord",{$exists:true});
    queryObject_thisday.username=username;
    queryObject_thisday.date=thisday;
    sensor_collection.find(queryObject_thisday).toArray(function(err,docs){
      console.log("this query",queryObject_thisday);
      if(docs.length==0)
      {
        console.log("this day No data");
      }
      else
      {
        var thisday_data = docs.filter(function(element){
```

```
  var hour = element.time.split(':')[0];

  return hour > '19';    //will collect time after 20:00
});
console.log("the first data of thisday",thisday_data[0]);

var hourlist = ["20","21","22","23","24"];
var dataobj  ={
  username:queryObject_thisday.username,
  date:thisday,
}

for(var i =0; i<5;i++) {
  var count = 0;
  var eachhour_data = thisday_data.filter(function(element){
    var hour = element.time.split(':')[0];
    return hour == hourlist[i];
  });
  if(eachhour_data.length==0){
    dataobj.time=hourlist[i];
    dataobj.motionrecord=0;
    (new motion_data_model(dataobj)).save();
  }else{
    var eachhour_data_afterprocess = [];
    eachhour_data.reduce(function (first, second) {
      eachhour_data_afterprocess.push(second.motionrecord - first.motionrecord);
      return second;
    });
    var avg_stdDev = standardDeviation(eachhour_data_afterprocess);
    console.log("this day",avg_stdDev);
```

```
            eachhour_data_afterprocess.forEach(function(element){
              if(element > avg_stdDev[0] + avg_stdDev[1] || element < avg_stdDev[0] -
avg_stdDev[1]){
                count++
              };
            });
            dataobj.time=hourlist[i];
            dataobj.motionrecord=count;
            (new motion_data_model(dataobj)).save();
          }
        };
      }
    });
  }
  // all function end here ---------------------------------
  })
 })
});

function standardDeviation(values){
  var avg = average(values);
  var squareDiffs = values.map(function(value){
    var diff = value - avg;
    var sqrDiff = diff * diff;
    return sqrDiff;
  });

  var avgSquareDiff = average(squareDiffs);

  var stdDev = Math.sqrt(avgSquareDiff);
```

```
  return [avg,stdDev];
};

function average(data){
  var sum = data.reduce(function(sum, value){
    return sum + value;
  }, 0);

  var avg = sum / data.length;
  return avg;
}
module.exports = router;
```

To visualize the comparison of obvious movements by hour, we developed bouncing balls with different colors on the platform of SAP Lumira. (Figure 52)

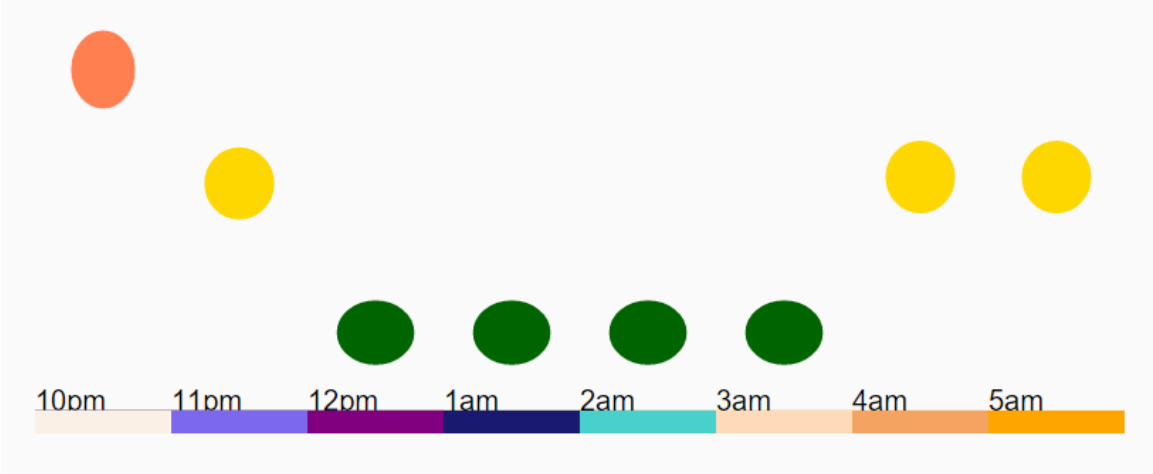


Figure 52 SAP Lumira visualization chart

5. What we learnt from this project

This project is a combination of technology in hardware, web development and data analysis fields. For each field, there were a lot of new things we need to explore. We got difficulties in the exploration process, but every time we overcome the difficulty, we got harvest from it. In this process, we also made good integration of the things we learnt from MSIS program with new knowledge. This project has great value for each of our team member.

5.1 Implementation of past knowledge

This project provided us with the opportunity to work through the whole SDLC of a product. Based on the process modules we learnt from course “MSIS2602 Information Systems Analysis and Design”, we chose system prototyping process model to build our product by iterating the development prototype until all features meet the requirement. Moreover, to achieve systemic project management on the whole SDLC, we incorporated the development process with knowledge learnt from course “MSIS2606 Software Project Management”. We used Microsoft Project to build our project plan and timeline. We conducted weekly scrum meeting for the team members to exchange their idea, report their progress and divide task for the next step.

The course “MSIS696 Business of Cloud Computing” provided us with the cognition of some common services of AWS such as EC2, Kinesis, DynamoDB and S3. Our initial plan of this project was to build a data transfer channel through Rasperry Pi, AWS Kinesis and DynamoDB. But due to the low feasibility, we aborted this plan. But we still implement AWS EC2 in our project to build our web server and database on it.

We have learnt many pragmatic skills and gained hands-on experience from course "MSIS696 Web Programming". The knowledge of HTML, CSS, JavaScript and interaction between front end and back end of a website we got from this class has built solid foundation for the development of our web application.

5.2 New things we learnt

From the beginning, we learnt how to design the circuit to connect our hardware components including Rasperry Pi, DAC and different sensors. As the operate system of Raspberry Pi is based on Linux, we need to use Linux command to manipulate the device. Through this process, we learnt many useful Linux command and started to write python script in Linux environment. Python is also a brand new programming language for us, but it is involved in many crucial processes for our project. Through data collection to data transfer, we wrote python scripts to make setting for the device and connect it with the mongo DB server located on AWS EC2. Rather than relational database we used before, we chose NoSQL database —— mongo DB as our data storage. It use JSON file as its data storage form, which is totally different from the traditional relational database we learnt past using table to store data. After studying on our own, we built our website using Node.JS and Backbone.JS, which are the most popular JavaScript frameworks currently. And we integrated our backend into RESTful API to achieve seamless joint with our front end, which is developed using bootstrap framework. For sleep analysis part, we leant new statistic and business intelligence knowledge to built our analysis model.

6. Future Enhancements

Currently, our prototype has demonstrated its functions to collect and show the sleep-related data and make some analysis about the sleep quality. Nonetheless, due to the limit of time and capital, we still have certain distance to our ultimate goal and need to perfect the functions of this prototype in several aspects.

6.1 Hardware

As shown above, the prototype now is a connection of different sensors with the mainframe of Raspberry Pi using bare wire. At this time, we use this device to collect the motion data when sleeping by fixing it on the arm with tapes. Obviously, the current status of the device can't be regard as a real product.

As our original intention is to develop a wearable device, people must comfortably wear the device during sleep. Moreover, a wearable device must have appealing appearance to attract customers. To fulfill these requirements, we need to rebuild the circuit and use smarter and micro sensors to take place of the current ones. We also need to implement mature industrial design to integrate the product with powerful function and pretty appearance.

The existing wearable devices on the market such as Fitbit, Jawbone and FuelBand are equipped with the capability to enable interaction between hardware and web application. Users can set their preference on the device through web application. Mutually, the device can collect data needed by the back end of web application. At this time, our prototype only demonstrated unidirectional information transferedd from hardware to

web application. So we want to achieve mutual information stream through hardware and software in the future.

6.2 Web application

Our web application enables users to find out their mothion data and environment data. Users can define the specific date or date range they would like to see their sleep data. But our application is not able to let the users define the specific time range within one day to check their data. As the python script used to collect data is running on Rasperry Pi side and our web application can't control the hardware at this time, we don't want to let the user manually run the script on Raspberry Pi by themselves. So we used crontab to set automatic run for the python script to collect data on a fixed time period everyday, which is 10pm at night to 10am the next day no matter the user is sleeping or not. This is kind of not user friendly, so we want to make enhancement in the future to let the device start collect data only when it detects the user is getting into sleep status and the user can define specific time on the web application to check their sleep quality and the environment condition.

At this time, Uses can easily see the fluctuation of data through the line chart displayed on our website. But only seeing these fluctuations will not give people a direct result about how their sleep quality is. Our report center is under development currently. The new page of our website will provide users with detailed analysis results about their sleep.

6.3 Data Analysis

We have built our own analysis model to determine if a user is under deep sleep based on the motion data we collected. However, our final goal of this product is to help people find out how their sleep quality is in relation with environmental factors. The analysis of this relation will make people get to know the source of their sleep problem and make according solution. So although this analysis is a great challenge to us, it will also bring us great value on the produce. Therefore, we will continue make effort on this product to build rigorous analysis model to solve this problem in the future.

7. Appendix

7.1 Python for putting data into AWS Kinesis (PutRecord)

```
import boto
import json
import time
import datetime
import boto.kinesis
from boto import kinesis
from boto.kinesis.exceptions import ResourceNotFoundException
from random import randint
from smbus import SMBus

bus = SMBus(1)
print("Read the A/D and put record Kinesis")
print("Ctrl C to stop")
bus.write_byte(0x48, 0)
last_reading =-1

ACCESS_KEY="AKIAIMV6XL4QVZA5GB2Q"
SECRET_KEY="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"
region_name="us-west-2"

kinesis =boto.kinesis.connect_to_region(region_name,aws_access_key_id =
ACCESS_KEY,aws_secret_access_key = SECRET_KEY)
streamName="sound"
partitionKey="IoTExample"
shardCount=1
global stream

def readSoundSensor():
    return bus.read_byte(0x48)

def runController():
  sound=readSoundSensor()
  timestamp=datetime.datetime.utcnow()
  record=str(timestamp)+":"+str(sound)
  print "Putting record in stream:"+record
  response=kinesis.put_record(stream_name=streamName,data=record,partition_key=partitionKe
y)
  print ("-=put seqNum:",response['SequenceNumber'])

def get_or_create_stream(stream_name, shard_count):
  stream=None
  try:
     stream=kinesis.describe_stream(streamName)
     print(json.dumps(stream,sort_keys=True,indent=2,separators=(',',':')))
  except ResourceNotFoundException as enfe:
     while(stream is None) or (stream['StreamStatus'] is not 'ACTIVE'):
```

```
        print('Could not find ACTIVE stream:0 trying to create.'.format(stream_name))
        stream=kinesis.create_stream(stream_name,shard_count)
        time.sleep(0.5)
  return stream

def setupController():
  global stream
  stream=get_or_create_stream(streamName,shardCount)

setupController()
while True:
  runController()
  time.sleep(1)
```

7.2 Python for getting data from AWS Kinesis (GetRecords)

```
import boto

import json

import time

import datetime

import boto.kinesis

from boto.kinesis.exceptions import ResourceNotFoundException

from boto.kinesis.exceptions import ProvisionedThroughputExceededException


ACCESS_KEY="AKIAIMV6XL4QVZA5GB2Q"

SECRET_KEY="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"

region_name="us-west-2"

kinesis =boto.kinesis.connect_to_region(region_name,aws_access_key_id =
ACCESS_KEY,aws_secret_access_key = SECRET_KEY)
streamName="sound"
partitionKey="IoTExample"
shardCount=1
iterator_type='LATEST'

stream=kinesis.describe_stream(streamName)
print(json.dumps(stream,sort_keys=True,indent=2,separators=(',',':')))
shards=stream['StreamDescription']['Shards']
print('# Shard Count:', len(shards))
```

```
def processRecords(records):
  for record in records:
     text=record['Data'].lower()
     print 'Processing record with data:' + text

i=0
response=kinesis.get_shard_iterator(streamName,shards[0]['ShardId'],
'TRIM_HORIZON',starting_sequence_number=None)
next_iterator=response['ShardIterator']
print('Gettinng next records using iterator:', next_iterator)
while i<4000:
  try:
    response=kinesis.get_records(next_iterator,limit=1)
        #print response
    if len(response['Records'])>0:
        #print 'Number of records fetched:' + str(len(response['Records']))
        processRecords(response['Records'])

    next_iterator=response['NextShardIterator']
    time.sleep(1)
    i=i+1

  except ProvisionedThroughputExceededException as ptee:
    print(ptee.message)
    time.sleep(5)
```

8. Reference

1. http://www.raspberrypi.org/recantha-tricorder/

2. http://scn.sap.com/community/developer-center/cloud-platform/blog/2014/10/15/raspberrypi-on-sap-hcp-blog-series-part-1-setting-up-your-raspberrypi

3. http://www.thingworx.com/

4. http://blog.buildinginternetofthings.com/2012/10/13/data-mining-and-the-iot/

5. http://www.recantha.co.uk/blog/?p=2663

6. http://server.zdnet.com.cn/server/2014/0520/3021130.shtml

7. http://www.bioon.com/health/menopause/84936.shtml

8. http://blog.sachleen.com/posts/1353217318/body-movement-during-sleep/

9. http://www.fritz-hut.com/2012/08/31/arduinopi-web-based-controller-for-arduino/

10. http://www.sleepimage.com/

11. http://www.withings.com/us/withings-aura.html

12. https://sleep.mysplus.com/

13. http://www.nhlbi.nih.gov/health/health-topics/topics/sdd/why

14. http://www.apa.org/topics/sleep/why.aspx

15. Xiangdong Tang,Environmental Factor's Influence on Sleep,[J] Guangdong Medical Journal, Jan.2007, Vol.28, No.1

16. Mary A. Carskadon, William C. Dement, Normal Human Sleep, Principles and practice of sleep medicine, 5th edition, (pp. 16-26).

17. Alain Muzet, Environmental noise, sleep and health, FORENAP, BP 27, Rouffach, France

18. Miwa, H.,Sasahara, S.-i., Matsui, T. Roll-over Detection and Sleep Quality Measurement using a Wearable Sensor. Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, (pp. 1507 - 1510)