University of British Columbia

CPEN 291 2016W2

Project 2

Conceptual Home Automation and Security System

Lab section: **L2A** Group #: **G5**

Group's Lab-Bench #s: **A-5A** and **A-5B**

Name	Student No.	Contribution percentage
Yu Chen	11691152	1/6
Benjamin Hong	38307154	1/6
Yuxiang Huang	14605159	1/6
Yuhao Huang	55562152	1/6
Ziqiao Lin	10668168	1/6
Hanyu Wu	36434158	1/6

Name	Contributions
Yu Chen	1. Virtual machine setup and installed apache server 2. NodeJS backend logic design 3. Designed GUI using html, CSS, javascript and chat room
Benjamin Hong	1. Arduino Security mode logic/algorithm, consolidation of most input and output components code. 2. Handling Bluetooth and Serial communication, Android App SocketIO 3. Fritzing drawings
Yuxiang Huang	1. Voice Command using Alexa: Wrote python code to communicate with Arduino, installed Alexa program on PI, wrote our custom Alexa code on Amazon website, did testing and improvements. 2. Arduino Servo motor code
Yuhao Huang	1. Setting up the virtual machine and designing main webpage 2. Designing the Arduino Circuit and the Demo House
Ziqiao Lin	1. Android App Java Code and designed GUI, implemented Bluetooth, Chat Room communication using SocketIO, etc 2. Designing the Arduino Circuit and the Demo House
Hanyu Wu	1. Help with HTML GUI design, made adjustments and fixed bugs 2. Raspberry Pi camera and video streaming 3. System Diagrams

B. Introduction and motivations

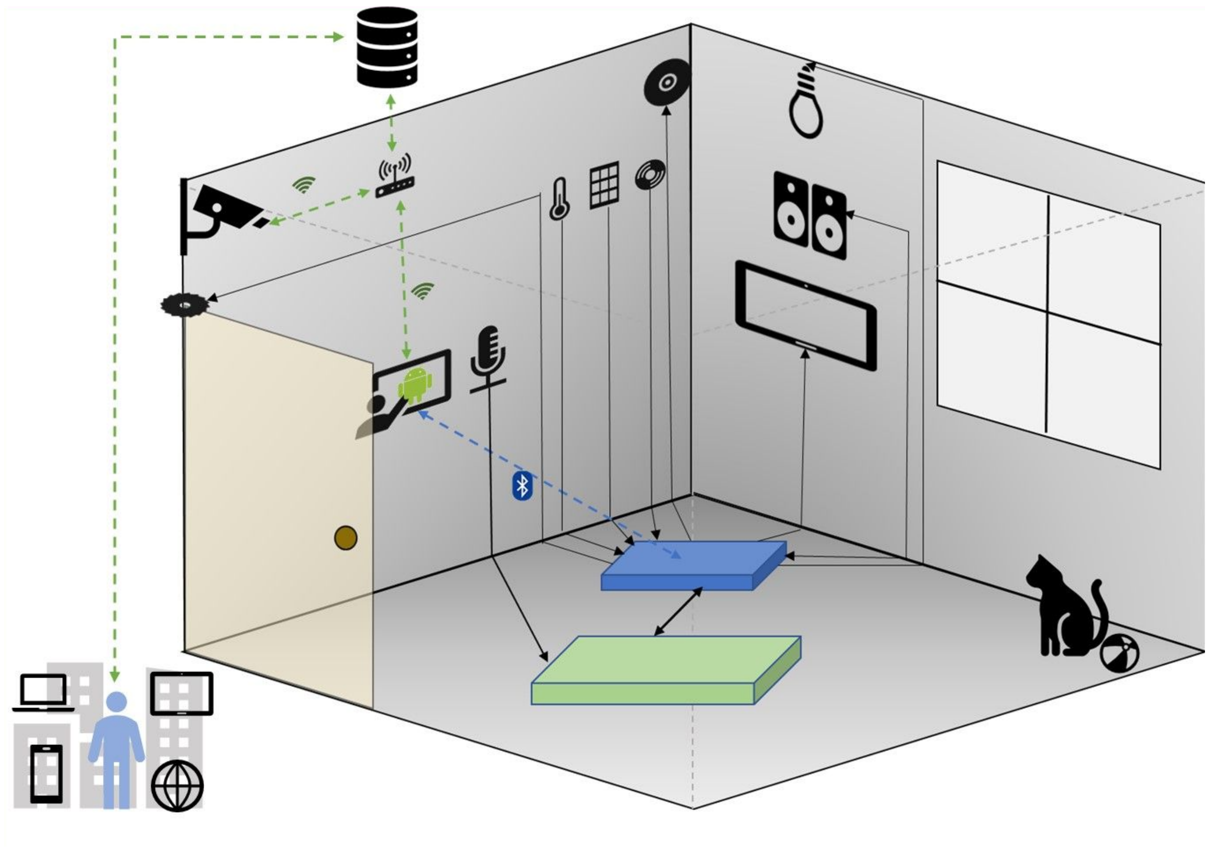
This project is about building a Conceptual Home Automation and Security System. We initially came up with this idea since we saw similar video tutorials on Youtube. We selected the topic since it includes enough work for both software and hardware part, and it is highly related to mobile app and web development technology, and also since we always expand its functionalities without losing our initial focus, to make our system smarter and able to do more things. Our initial design objective is to enable user interaction with Arduino and multiple sensors on three different interfaces: Website, Android App, and Voice Control; however, as we were making progress, our objective has changed, since we kept learning new methods and technology and we found that it some additional functionalities were much easier to implement than we initially thought, for example, adding live video surveillance or a chatroom to our website, and enable our smart system to switch between different modes in certain circumstances to improve the security of one's home, without losing human computer interaction.

Features and Expected Functionalities:

- Live video surveillance
- Controlling doors, lights, and alarms remotely on website data page, android app, or using voice control
- Getting sensor readings (ultrasonic sensor, temperature sensor, humidity sensor, and photovoltaic sensor) remotely on website data page, android app, or using voice control
- If set to auto mode, then whenever there is an intruder, the alarm and lights will be turned on while the door will be closed
- Chatting using the chat room on the website
- Using voice control to open any website that has a url
- Using voice control to receive live news streaming and listen to music (those are built in Alexa functionalities)

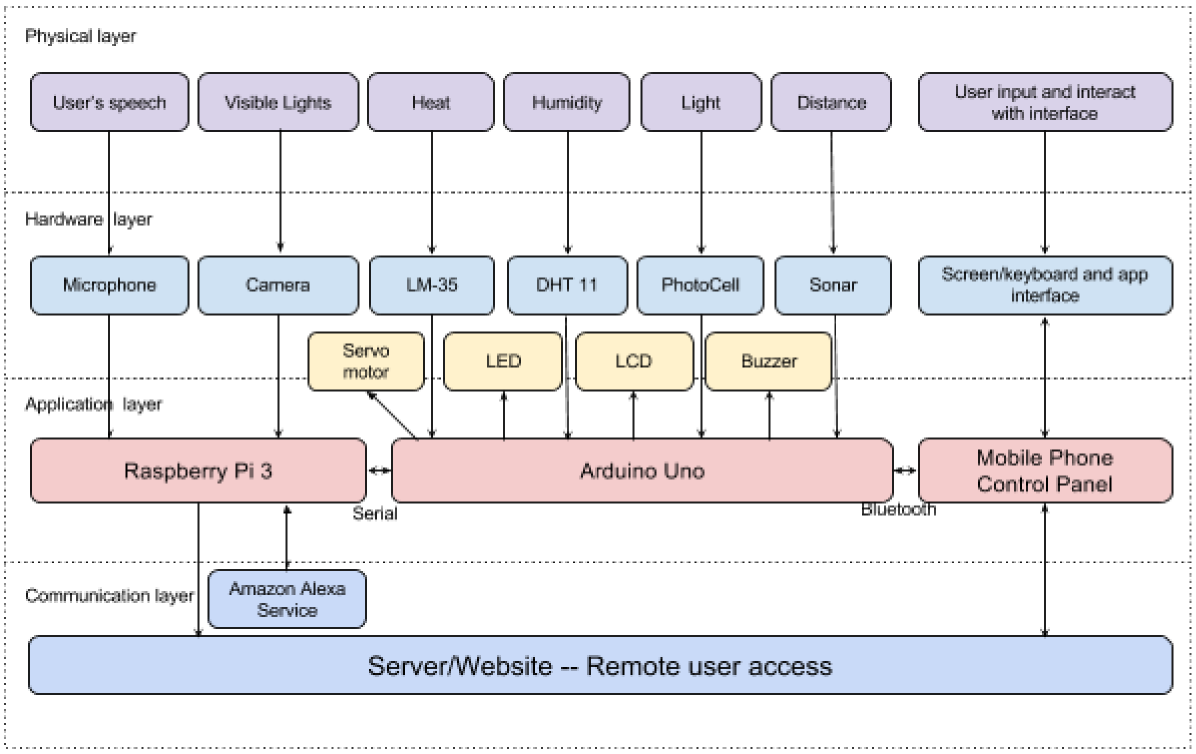
C. System diagram

Real life implementation:

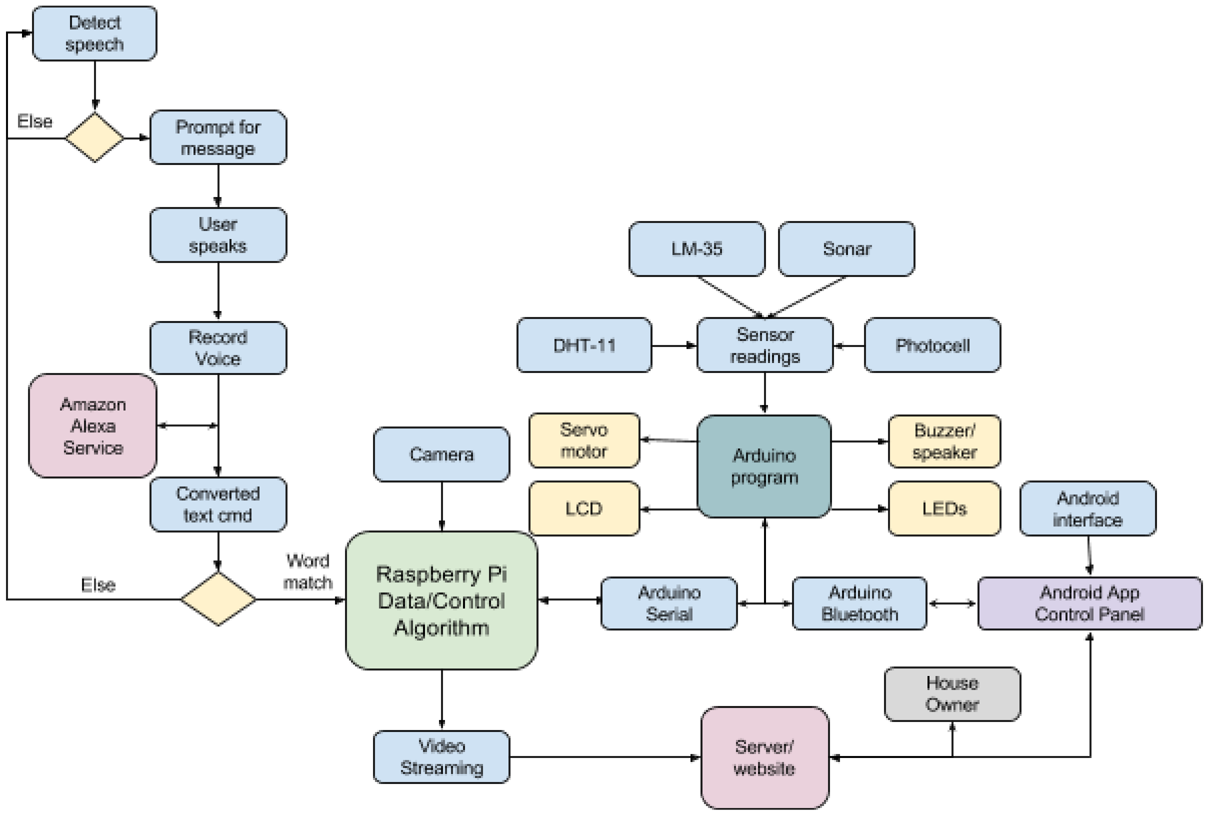


Symbol	Meaning	Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
🟦	Arduino Board	📡	Household Router	◉	Buzzer as alarm	💡	LED lights
🟩	Raspberry Pi Board (Alexa)	📱	Android app Control Panel	🔊	Sonar	⚙	Servo motor Door control
ᛒ	Bluetooth® Connection	🗄	Online Server	💿	Photocell	▭	LCD display for messages
📶	Wi-Fi Connection	📹	Raspberry Pi Camera	🌡	LM-35 Temp. sensor	🐈 ⚽	A cat, and his favorite ball
🎤	Microphone for Alexa	🧍	House owner	▦	DHT-11 Humidity	💻 📱	Devices with network

Hardware diagram:



Software diagram:



D. Project Description

Overall Description

This project focuses on home automation via voice control and the internet as well as home security functions such as motion sensor, etc. We presented the functionalities by creating a miniature demo house.

In the demo house, we have humidity, temperature, light, ultrasonic sensor for input components and LCD display, Piezo buzzer, Servo motor (for the door) for output components. All these components are hooked up to the Arduino Uno except the Pi Camera which is hooked up to a Raspberry Pi. These are all inside the house. There is also another Raspberry Pi for voice control using Alexa outside the demo house.

The input components are updated and can be viewed on the Android app, the website and can be told by Alexa (voice control AI) if the user asks for the input values. Likewise, the output components can be controlled via the website, app and Alexa if the user asks Alexa to for example, open/close the door, turn the alarm on/off. This is the home automation via voice control (Alexa) and the internet (website/app) that we have worked creating.

Regarding home security, the house has a functionality called “security mode”. This can be controlled via Alexa, website and app. Once this is on, the ultrasonic sensor will keep monitoring the distance in the house from its current position and the wall that it faces. We set that distance between it and the wall as the threshold and if the distance shortens, we know that there is an intruder inside the house; then the door closes, alarm rings and the red LED flashes, letting the user know that there is an intruder inside the house. This trigger and the security mode can be turned off via the website, app and Alexa.

Furthermore, the user can always monitor outside (and inside if we had two pi cameras) the house via the PI Camera over the internet using either the website or the android app.

For houses with multiple members, we also implemented a chat room for the members using the app so that they can communicate through the home automation/security app

Raspberry Pi Voice Control

The voice control functionality is implemented to make our home automation/security system more user-friendly, as well as achieve better human computer interaction. The hardware module of the voice control functionality is composed of an Arduino, a Raspberry Pi, a webcam and a speaker. The software module consists of Arduino code, Python code, and Amazon Alexa Voice Service. The Alexa Voice Service program was downloaded to Raspberry Pi 3 model B as a program operating on Raspbian Jessie operating system.

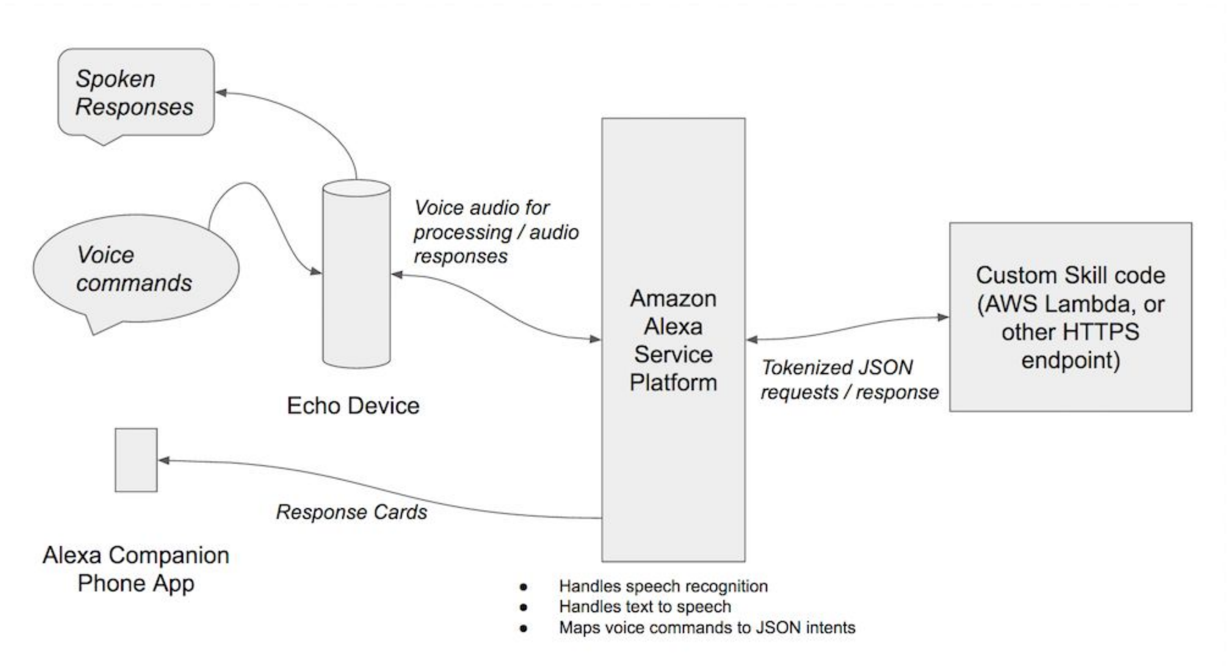
The voice control function allows the user to directly speak to the Pi to control appliance at home, receive data from sensors, as well as open websites, live stream news, listen to music, etc. We connect Arduino to Raspberry Pi using a USB cable to enable serial communication between Arduino and the Pi.

The core part of implementing this functionality is using Alexa Voice Service (AVS) of Amazon. Alexa Voice Service is an intelligent voice recognition and natural language understanding service developed by Amazon. The reason why we finally chose to use AVS is that it allows developers to write their own customised functions while taking advantage of the excellent voice recognition function.

To implement one's own customized functions, the developer first needs to build an interaction model on Amazon Developer Console; more specifically, determine an activation word, and write an intent schema and sample utterances. The activation word is a keyword expected to be said by the user whenever he/she wants to use the function ("Computer" in our case). A intent schema is basically a list of functions that need implementing; it contains many intents, and each function name corresponds to a unique intent. For example, in our project, we used 20 intents for 20 different functions, such as turning on the lights and getting sensor readings. Sample utterances are the words user needs to say in order to trigger one intent/function. One intent can have multiple sample utterances, and ideally the user can trigger the intent as long as the recorded voice contains one of the sample utterances, regardless of other words said by the user.

After building the interaction model, the developer then needs to actually write the customized function code on a computer. There are two ways to do this, the first one is using AWS Lambda function, which directly stores the written Node.js code on Amazon Web Server (AWS); and the second one is using HTTPS to send data to AWS from the local machine. Although the first one is more convenient to use since it does not need external libraries and software to be implemented, we chose the second option since it allows us to interact with the data received by the local machine (Raspberry Pi 3) and send it to AWS for text-speech translation. In order to do this, we need to use a few libraries, and the most important one is Flask-ask, which is a Python-framework that simplifies the Alexa skill development process. Also, we used a software called ngrok to use Alexa Voice Service as an API, by creating an endpoint to the AVS API and thereby able to send/receive data from there. The code on local machine was written in Python in our project. The Python code contains 20 functions and each corresponds to one intent. Whenever the user triggers an intent by saying the corresponding sample utterance, the corresponding Python function will be executed. It sends the expected bytes to Arduino to trigger the corresponding C function, and also send the expected response to AWS, to let it be translated to speech and sent back as an audio.

The voice control functionality diagram is shown below:


Spoken Responses
Voice commands
Echo Device
Voice audio for processing / audio responses
Amazon Alexa Service Platform
Custom Skill code (AWS Lambda, or other HTTPS endpoint)
Tokenized JSON requests / response
Response Cards
Alexa Companion Phone App
Handles speech recognition
Handles text to speech
Maps voice commands to JSON intents

Android App

The Android App can be separated into 4 stages, which are GUI design, Bluetooth implementation, SocketIO implementation and Chatroom design.

1. GUI Design: Since we should display several data transferred from Arduino, we design to display temperature, humidity, light level and ultrasonic sensor reading in one page. And the 3 buttons in the first column are used to control automatic security mode, which are turn on, reset and turn off. The 3 buttons in the other column are to control LED, Buzzer and Servo. The two image buttons at the bottom are disconnect and chat, which can disconnect bluetooth and jump to chat room page.
2. Bluetooth:Since we created an app for the first project and we successfully implemented Bluetooth data transfer from Android to Arduino in project1, we modified the bluetooth connection and create new event for this project this time.

3. SocketIO: SocketIO is used to communicate with Node.js Server. Both server and Android App uses SocketIO. The Android app constantly sends 4 events for the 4 input data to the server which then the server sends to other html clients. The server send 6 different commands to the Android (not constantly) when the corresponding buttons are clicked for LED control command, Buzzer control command, Servo control command, Auto/Security control command, Reset control command and Clear control command. These 6 commands will trigger 6 different methods in the Android app to send data to Arduino via Bluetooth to control different components. SocketIO is also used to implement chatroom function with other html clients. In particular, we define a new Socket in chatroom class and create an event to listen on.
4. Chatroom: the chatroom is implemented basing on SocketIO. Since chatroom is a different class (in Java) to other Bluetooth Control page, we have to redefine SocketIO one more time. We create an event call ”chat” and the method will listen to this event if web page will send data through “chat” event. When android sending data, we create an imageButton and each time we press the button will trigger the method called “sendChat” method, which will add the text to arrayadapter to display on screen and also will send to WebServer from Socket IO default method. We receive the EditText and convert it to String type and directly send it to Web Server. All these message will be convert to String type and add into an ArrayList called “messages”, and this “messages” is linked to an ArrayAdapter.After receiving or sending message, we will update the ArrayAdapter to display it on screen.

Arduino

Since our project uses the sensors which we are familiar with from previous labs, consolidating the individual codes from various labs was not too difficult. It was only a bit time consuming testing all the sensors and the combined code. The sensors that we used are LM35, DHT11, Ultrasonic Sensor, Photo Sensor, Servo Motor (door), LCD Display, Piezo Buzzer and LED.

The more challenging works regarding Arduino was coordinating Bluetooth (HC-06) with Android app and Serial communication (wired) with the Raspberry Pi.

The most challenging work regarding Arduino however was writing the logic for the security mode. It is coded to handle three different commands from either Bluetooth or Serial communication.

1. Security mode command: Upon receiving this command, the Arduino starts monitoring the ultrasonic sensor if the input value ever reaches below the threshold value we set which is the distance from the sensor and the wall that it is facing; this acts as sort of a motion sensor. If it does detect a lower input value then the threshold, we know that there is an intruder in the house and the door of the house closes, Piezo buzzer alarm turns on and red LED flashes. We call this situation that the house is triggered.
2. Reset command: This command only works when the house is in security mode. Upon receiving this command, if the house is triggered, then all the alarm and LED turns off, but the house is still in security mode. If the house was not already in security mode, it doesn't do anything.
3. Turn off command: This command does same thing as the reset command, except if it is in security mode and the house is triggered, the alarm and led turns off and the house is no longer in security mode. Therefore, the house is non-triggerable.

Html client website

The html client website is used for users to remotely control their home facility and work in the same way as the Android App. The website has an user-friendly interface which allows clients to use easily. It has three basic functionalities: monitoring, chat room and video streaming.

1. The monitoring page is designed for users to read the real-time values of sensors remotely on the html web page. The data is updated instantaneously, which means that when the sensor values change then data on the html will be changed as well. Also, the monitoring page has the feature of remotely control the LEDs, piezo buzzers and servo motor. It allows users to manually monitor their home and make their home safe. In addition, it also has the feature of turning on the Auto Mode.
2. The chat room page is created for clients to communicate on both web pages and Android App. People can set their names during the setting, and send messages. It is designed for users to deal with emergency and easily contact with their family members. The chat room also allows users to send messages that can be displayed on LCD.
3. The video streaming page is used for people to see their home via a camera that is

attached on a Raspberry Pi, and it acts like a surveillance camera.

We use two HTML5 templates to design our webpages, and we also use CSS to design styles and customize our website. There are also Javascripts which are used to import libraries and connect to websockets for communications.

Node.js server

The basic idea of our network communications across Raspberry Pi, Android App and html pages is to use websockets. We set up the Node.js server on the virtual machine and make it constantly listen to the 9000 port. It uses express framework to deal with http request and socket.io to build websockets connections. Other devices are client sides and build a websocket connection with the Node.js server. We use Javascript to write the client side code for web pages, Java for Android App and Python for Raspberry Pi. All the events are passing through the server.

For example, if a user clicks a button on the web page to turn on the LED, the web page client side will emit this event and send it to the server, and the server has the code to handle this event and emit another event to the Android. Unlike http requests, the socket connection is stable and fast because it does not need to wait for server to respond back. Different devices are designed to handle different events so that the communication does not affect each other. In addition, the server can handle many client requests with nearly no latency.

Virtual Machine main website

We used the cloud based virtual machine provided (38.88.74.87) as our server. We started setting up the virtual machine with installing Apache, MySQL, and PHP, following a tutorial found online. With the server set up well, we designed our homepage website using a HTML template. Our homepage was designed to provide a brief introduction to our project and team, and a control panel which could navigate the user to our client website and video streaming page. The biggest challenge in this part was to understand the HTML and CSS code, which we have never used before. We found it helpful to track the tags in these two programming language, so that we could easily and quickly find the part of the code we need.

Raspberry Pi Camera

1. Introduction to the topic: The basic idea of video streaming application of a Raspberry Pi Camera is as such: it captures image, send it through internet and then the front end constantly refresh the image it received.
2. Challenges: The implementation mentioned above is simple and naive. Due to limited bandwidth and poor compression of each image, as well as lack of synchronization between both ends, the streaming turns out to be very laggy and slow.
3. Testing and improvements: To solve this problem, we simply followed a tutorial named "Build a Raspberry Pi Webcam Server in Minutes", as mentioned in the tutorial, two existing project is used: "Motion" will allow programs to monitor the video signal from one or more cameras and detect if a significant part of the picture has changed, and "motioneye OS" for Raspberry Pi, which is a Linux distribution that turns your single board computer into a video surveillance system. Just as their name, it works roughly the same way as motion pictures: the significant changes are called "keyframes" are captured and sent through socket, and the "middle frames" are not actually captured, but generated, which allows fluent video streaming to be displayed on the website.
4. Evaluation of the solution: It requires CPU useage of Raspberry Pi for about 10%, while the naive implementation requires maximum 5%. We think such an increasement in the CPU usage is acceptable consider the boost of video streaming quality it gives us.
5. Comments: For video streaming, we used another Raspberry Pi, but it is not because this feature is not compatible with Alexa. Video streaming and voice recognition can work perfectly with each other. We did it that way because we originally worked in parallel on two raspberry Pi board, one for Alexa and the other for video streaming to be more effective.

E. Test, Evaluations and Challenges

Explain the evaluation/testing procedure, test cases, and results both for hardware and software.

Include the problems or challenges you encounter and how you resolve them, as well as best practices you have learned.

Hardware

- **Arduino, Bluetooth and Sensors**
    - Testing Procedure
        - We individually checked each input/output sensors that they were properly functioning before adding more code for sensors
    - Challenges
        - Major challenge was designing the logic for security mode, specifically being able to change modes and coordinating each feature dedicated to a certain mode. This was hard because the mode system had to be implemented in a loop() which runs forever.
            - Solution was to use many flags and use if statements (modularly) in the loop().
- **Raspberry Pi Camera**
    - The major challenge of video streaming is that if we send all uncompressed images we captured, the video we turn out getting is laggy and late.
    - We tried to limit the frame rate and lower the quality of the video, but it still has a great dependency on the bandwidth and does not completely solve the lag problem.
    - By implementing “Motion” and “Motion Eye OS”, as mentioned above, we find that the video is fluent, but it consumes a maximum 10% (about 5% minimum) CPU usage of raspberry Pi, and also requires some, but minor resource on the client end. We decided the increment in the video streaming fluency and quality is worth the cost of those reasonable computation power.

Software

- **Voice Control Functionality**

The biggest challenge for implementing voice control functionality is finding a good API, or similar related speech-text translation services. Initially, we tried to use 3 open source voice control programs on the Raspberry Pi, however, it turned out that all of them are outdated. Before we decided to use Amazon Alexa Voice Service, we had a really hard time configuring and modifying those open source code to make them work. However, we started to make noticeable progress only after we found Alexa.

- **<u>Android App</u>**
    - Testing
        - First, we tested for successful Bluetooth connection with Arduino.
        - Then we tested SocketIO with Node.js server
        - Then we tested SocketIO communication with Node.js server and its html clients
    - Challenges
        - A lot of time was spent into designing the GUI.
            - Despite a lot of time spent, GUI techniques on Android Studios were learnt not very efficiently since Android Studio is a very powerful GUI design tool and it is hard to master all functions in a very short period of time. We use Sketch to design the GUI pages before we write code for GUI in Android Studio. The page in Sketch is much better than the page we actually make, and we find out it is not easy to make the same ones. And after we tried very hard to make a similar one, in fact, the real one is not as good as we see in Sketch,
            - Hard to solve bugs from Android Studio

          We met several times that the program did not work because of some strange reasons. And we google online to look for the solutions but they did not work all the time. Android Studio is new for us and we have less understanding about other file in an App file since we only know how to code in java file and design GUI in layout file.

- **<u>Node.js server</u>**
    - Testing
        - SocketIO tests were first conducted with other html clients and the server only. Only after that we tested SocketIO with Android app.
    - Challenges
        - The most challenging work regarding to this part is to set up the server that can work across different networks. We first tested our server code with the local host. And after making it worked on the local network, we moved it to the virtual machine. Then we got a problem

that the port on the virtual machine was not open for us to use, so then we asked professor to help us open the port. After the port was opened, the problem wa solved, and we can add more features onto our server.

F. References and bibliography

Amazon Alexa Voice

- https://developer.amazon.com/blogs/post/TxDJWS16KUPVKO/New-Alexa-Skills-Kit-Template-Build-a-Trivia-Skill-in-under-an-Hour
- https://developer.amazon.com/alexa-skills-kit/alexa-skill-quick-start-tutorial
- https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/getting-started-guide
- https://www.pluralsight.com/guides/node-js/amazon-alexa-skill-tutorial

Arduino Code & Fritzing

- Previous 5 labs for sensor code and fritzing drawing for reference

Fritzing Drawing for Raspberry Pi

- Couldn't find Raspberry PI in Fritzing app so I downloaded Raspberry Pi drawing from the internet and modified it.

Android SocketIO Library Use

- https://www.youtube.com/watch?v=WFfd4QbmWpI&t=499s

Node.js & Web Socket.io

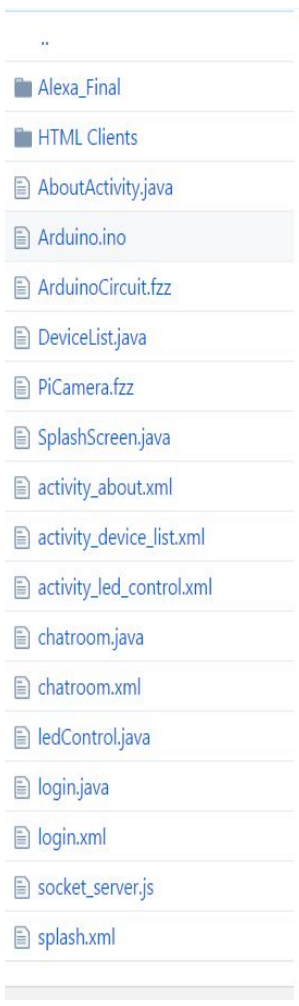
- https://socket.io/
- https://nodejs.org/en/
- https://www.youtube.com/watch?v=HZWmrt3Jy10&t=414s
- https://www.youtube.com/watch?v=TPLAd-j9tMs

Raspberry Pi camera video streaming

- https://pimylifeup.com/raspberry-pi-webcam-server/

File Submission Descriptions

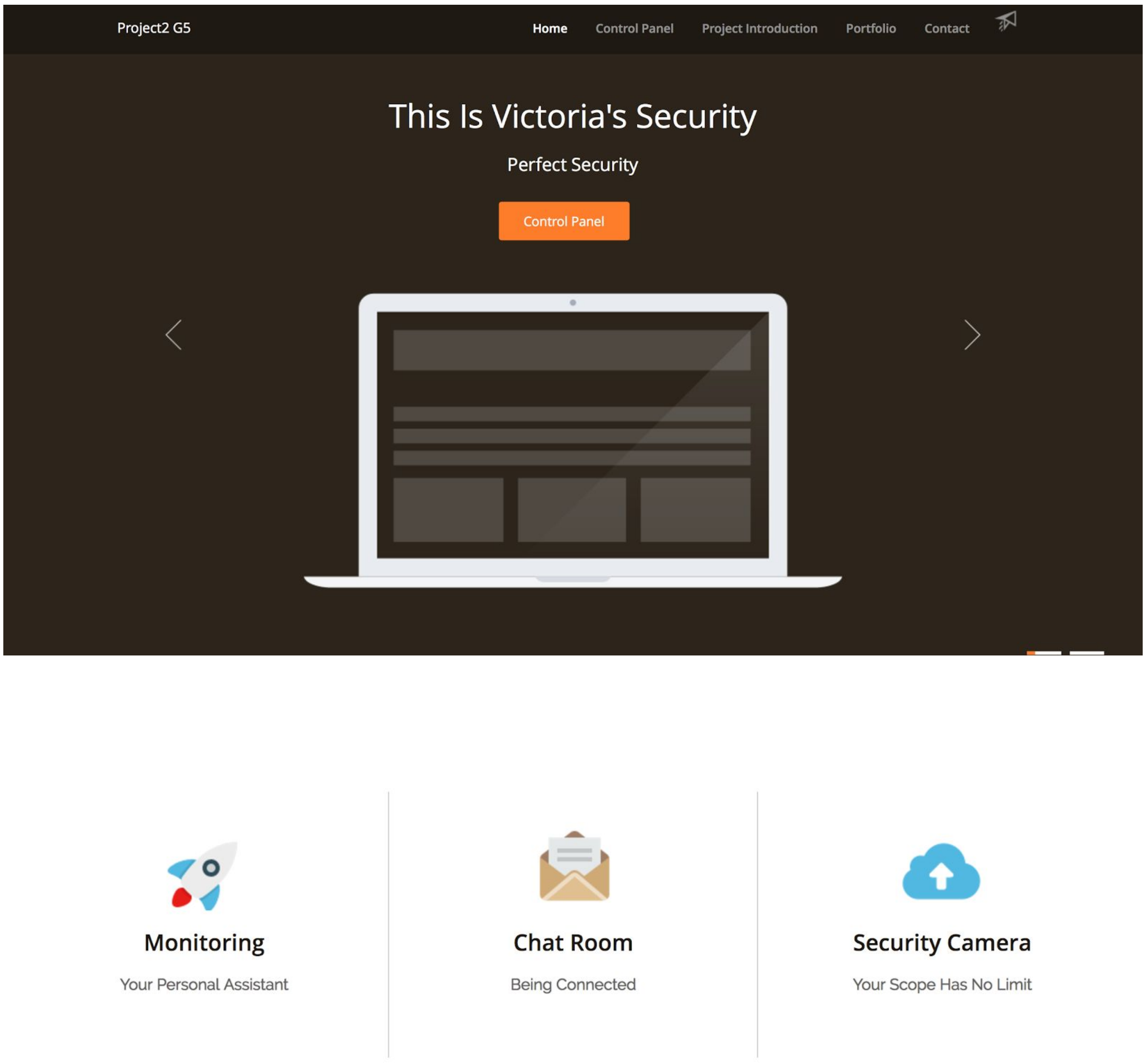
All the submitted files are listed below, the file descriptions are the file names

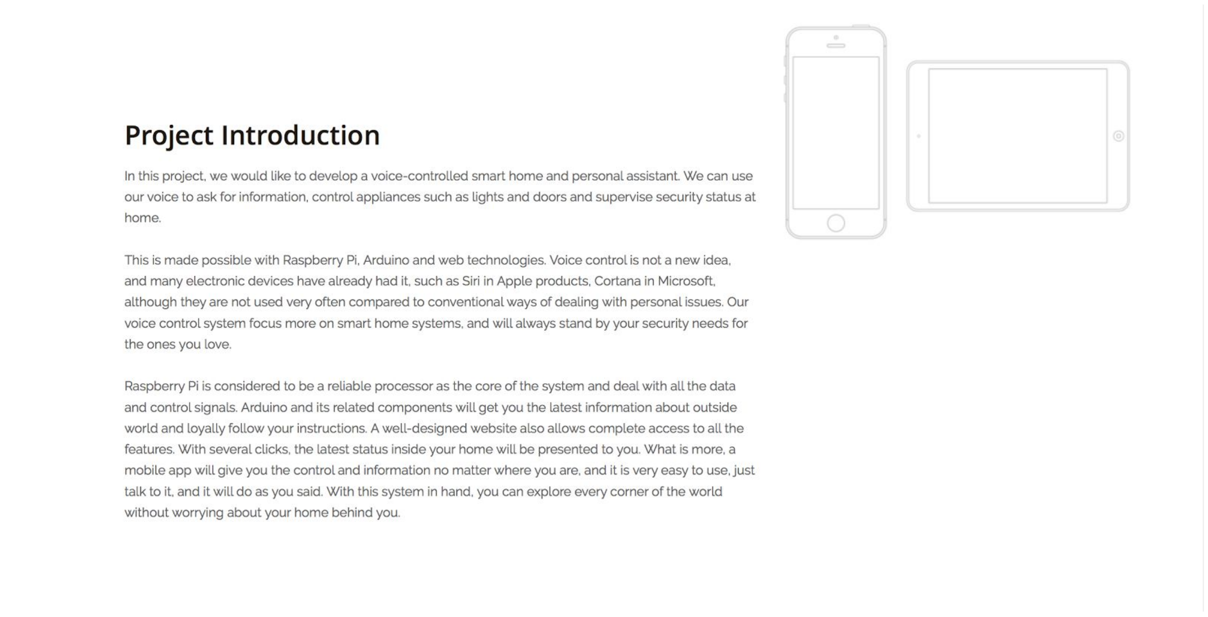


..
Alexa_Final
HTML Clients
AboutActivity.java
Arduino.ino
ArduinoCircuit.fzz
DeviceList.java
PiCamera.fzz
SplashScreen.java
activity_about.xml
activity_device_list.xml
activity_led_control.xml
chatroom.java
chatroom.xml
ledControl.java
login.java
login.xml
socket_server.js
splash.xml

Appendix A – Project pictures

- **Homepage**



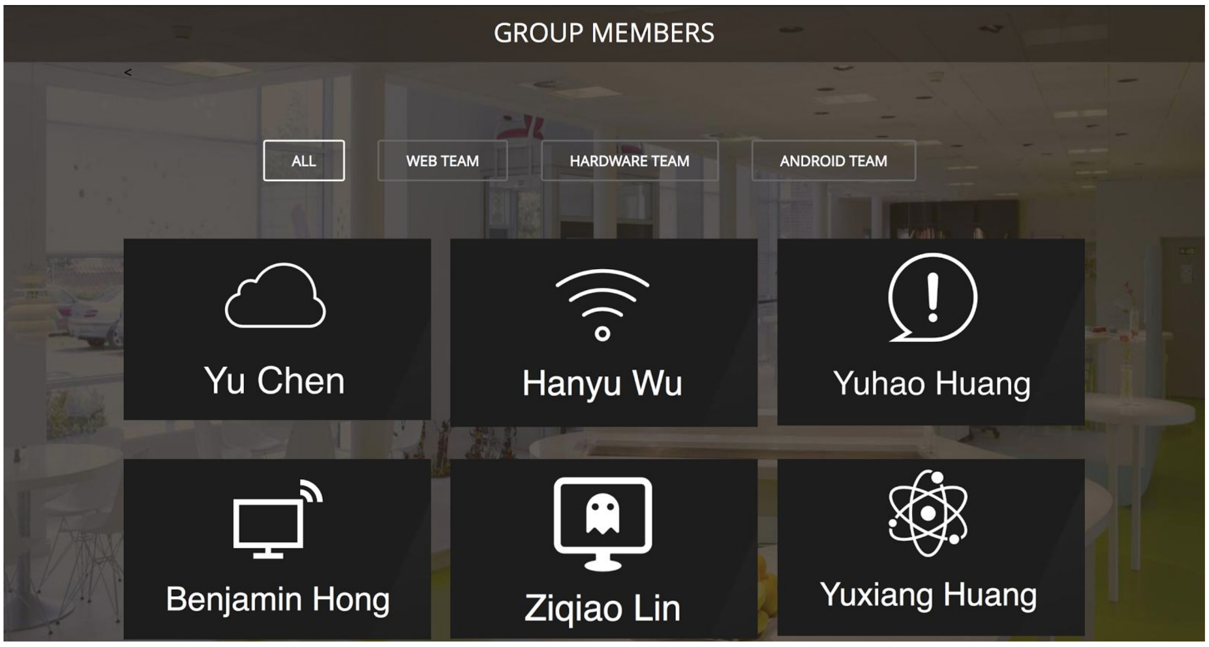


Project Introduction

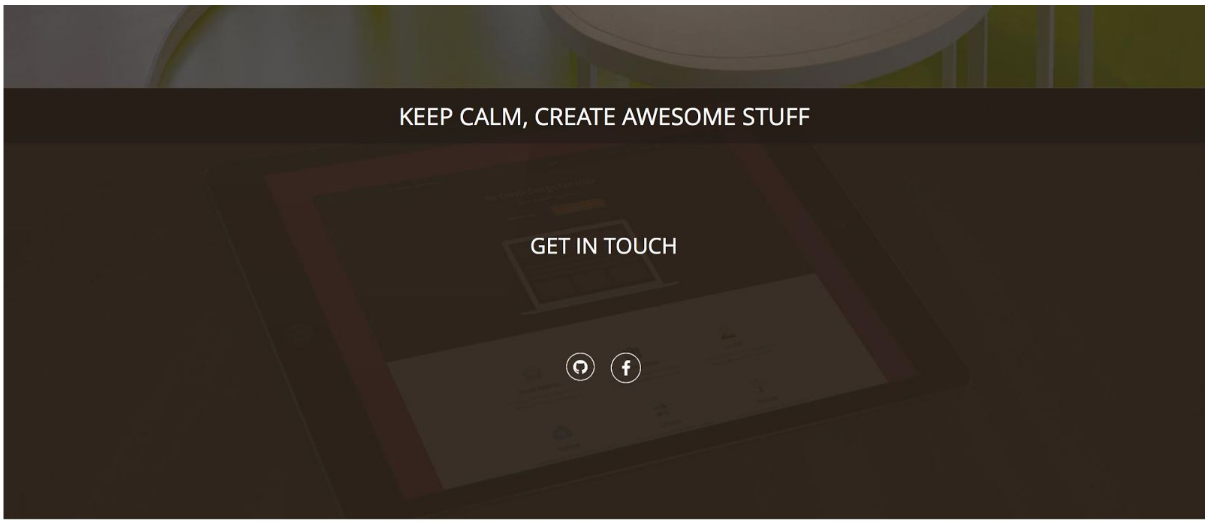
In this project, we would like to develop a voice-controlled smart home and personal assistant. We can use our voice to ask for information, control appliances such as lights and doors and supervise security status at home.

This is made possible with Raspberry Pi, Arduino and web technologies. Voice control is not a new idea, and many electronic devices have already had it, such as Siri in Apple products, Cortana in Microsoft, although they are not used very often compared to conventional ways of dealing with personal issues. Our voice control system focus more on smart home systems, and will always stand by your security needs for the ones you love.

Raspberry Pi is considered to be a reliable processor as the core of the system and deal with all the data and control signals. Arduino and its related components will get you the latest information about outside world and loyally follow your instructions. A well-designed website also allows complete access to all the features. With several clicks, the latest status inside your home will be presented to you. What is more, a mobile app will give you the control and information no matter where you are, and it is very easy to use, just talk to it, and it will do as you said. With this system in hand, you can explore every corner of the world without worrying about your home behind you.

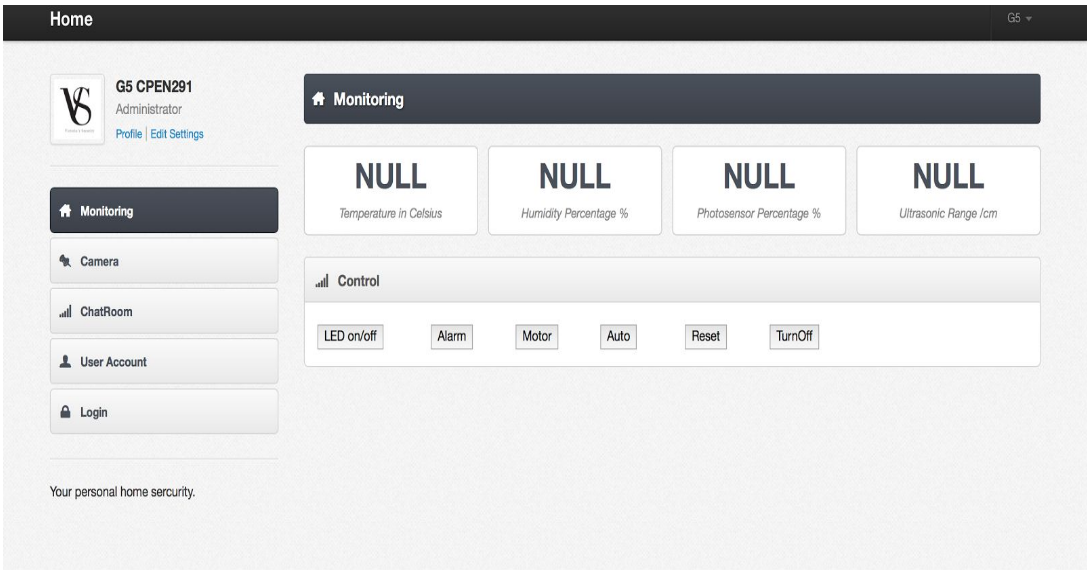


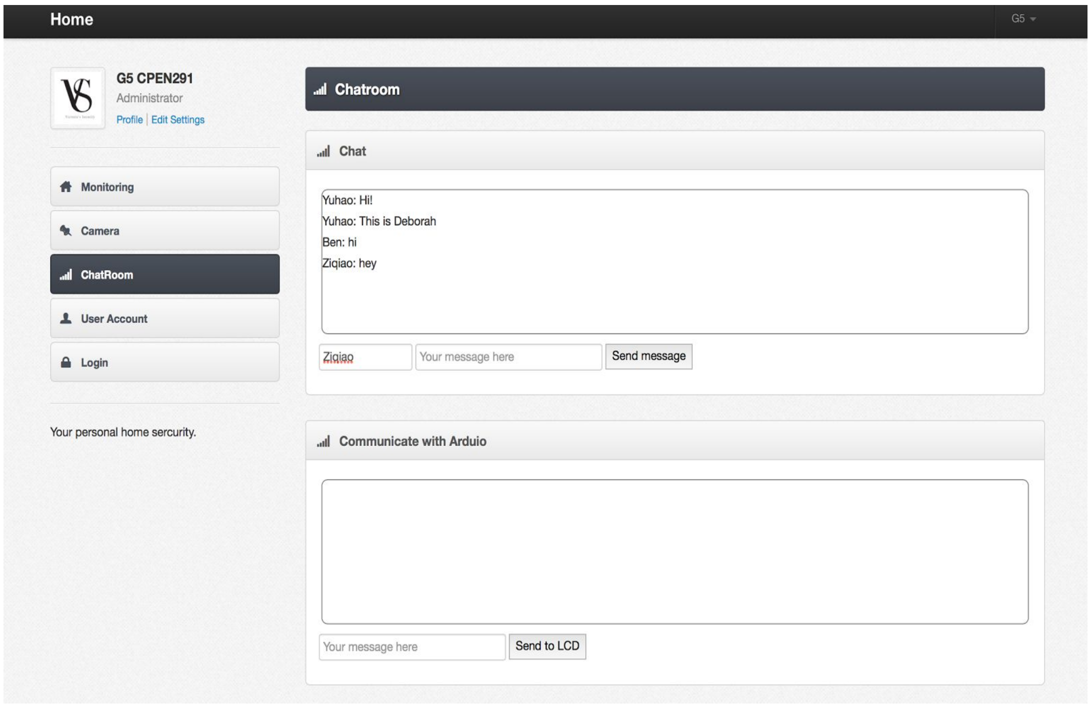
GROUP MEMBERS



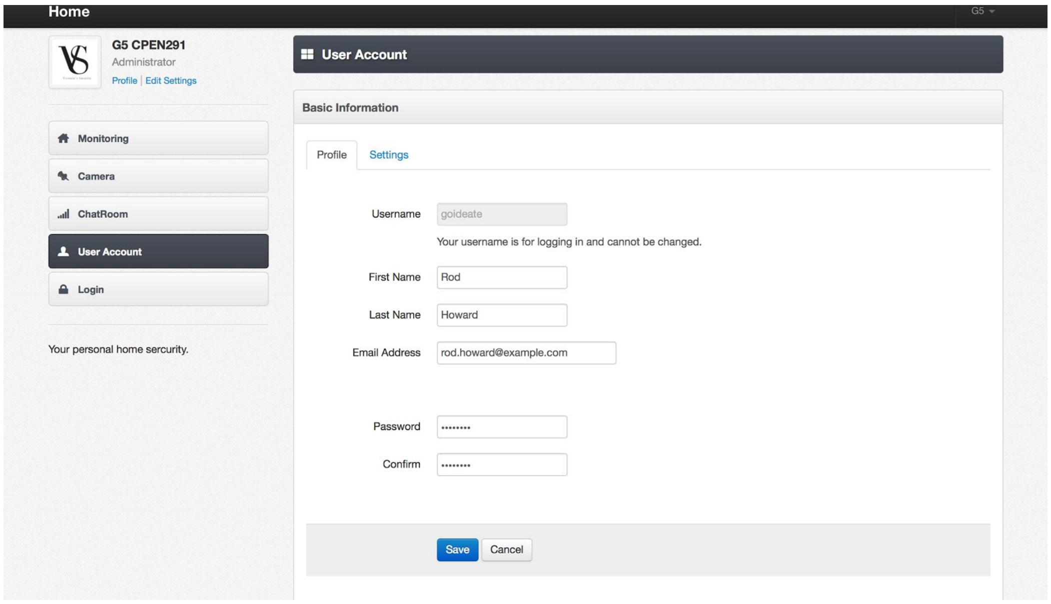
KEEP CALM, CREATE AWESOME STUFF

- **Html Client Page**





Yuhao: Hi!
Yuhao: This is Deborah
Ben: hi
Ziqiao: hey


Home

G5 CPEN291
Administrator
Profile | Edit Settings

Monitoring
Camera
ChatRoom
User Account
Login

Your personal home sercurity.

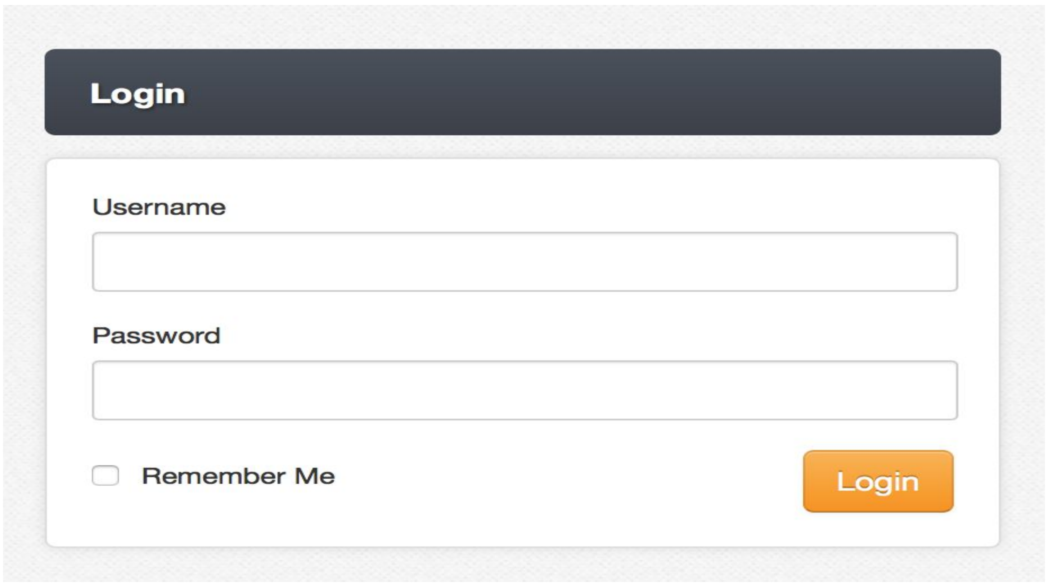
User Account

Basic Information

Profile Settings

Username: goideate
Your username is for logging in and cannot be changed.
First Name: Rod
Last Name: Howard
Email Address: rod.howard@example.com
Password: ••••••••
Confirm: ••••••••

Save Cancel

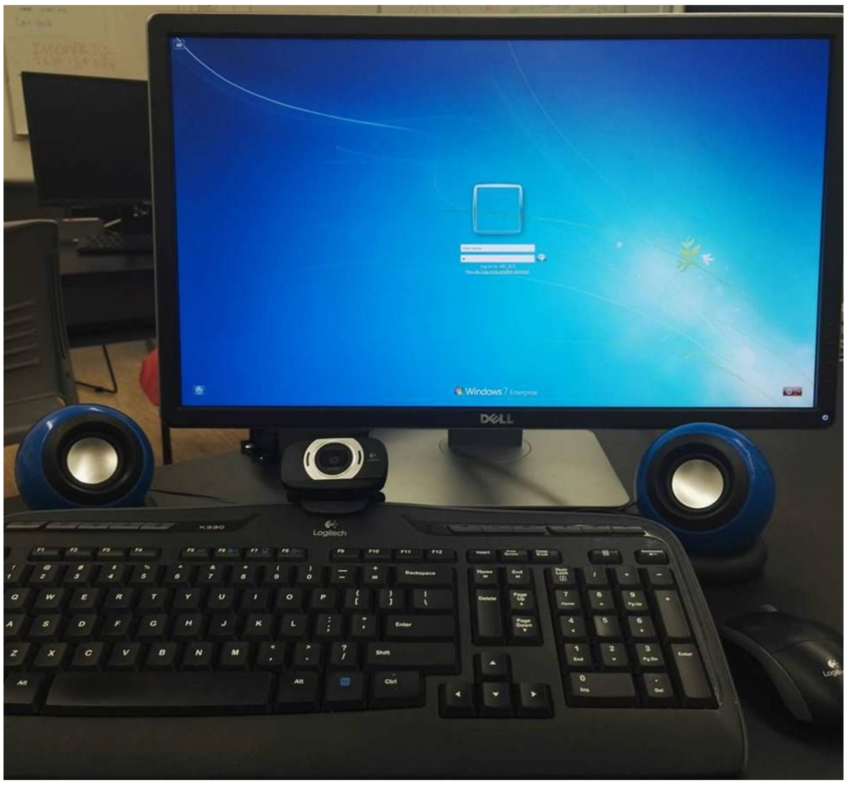

Login

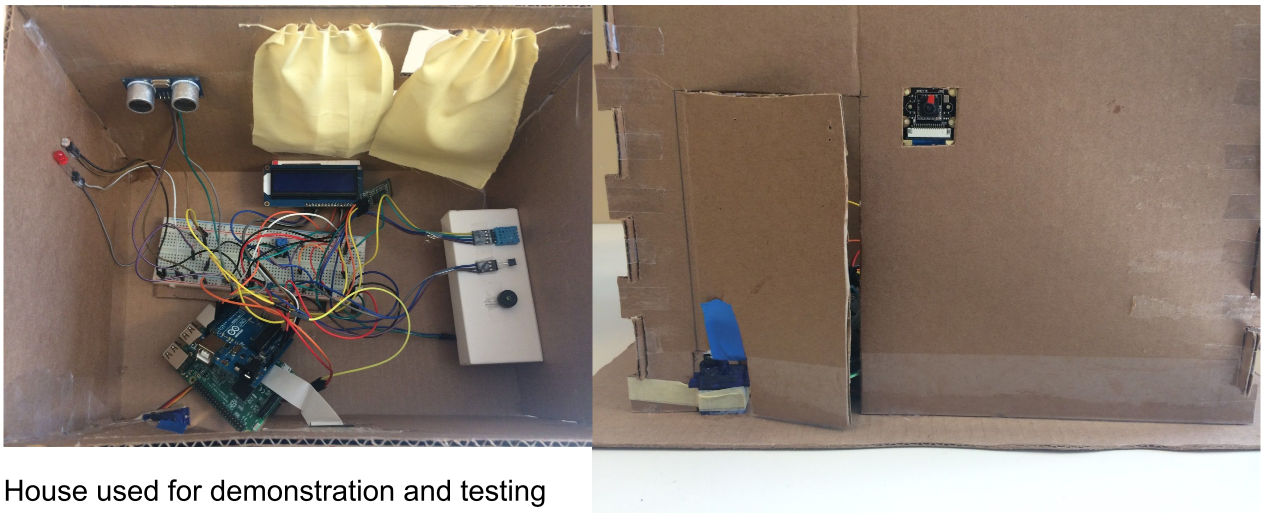
Username

Password

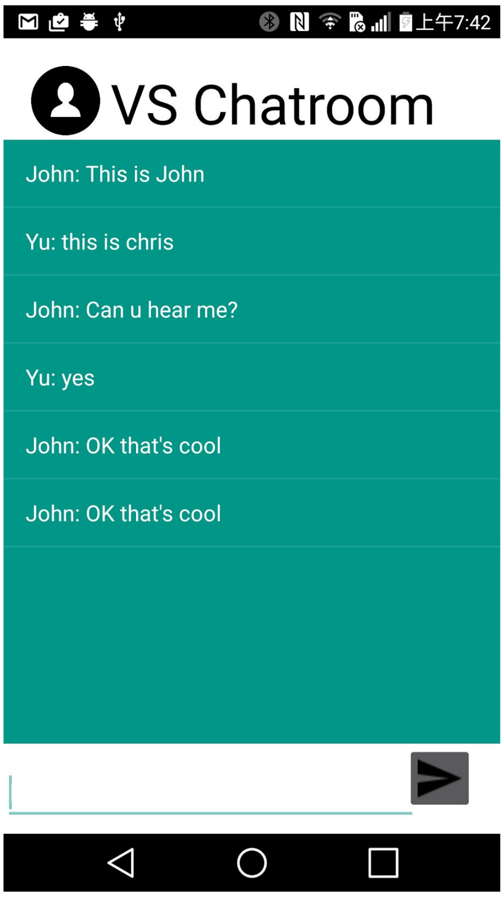
Remember Me

Login


Alexa Implementation with speaker and microphone


House used for demonstration and testing

- **Android App**



VS Chatroom

John: This is John

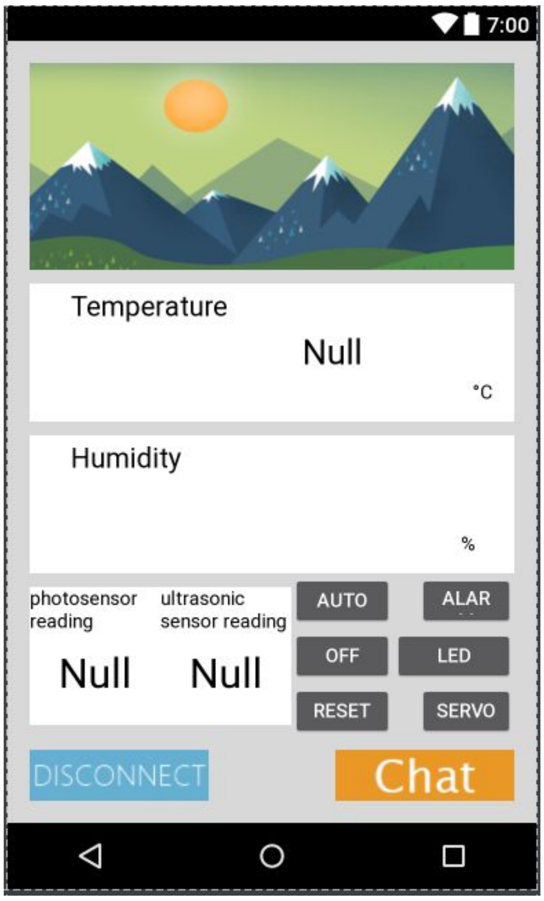
Yu: this is chris

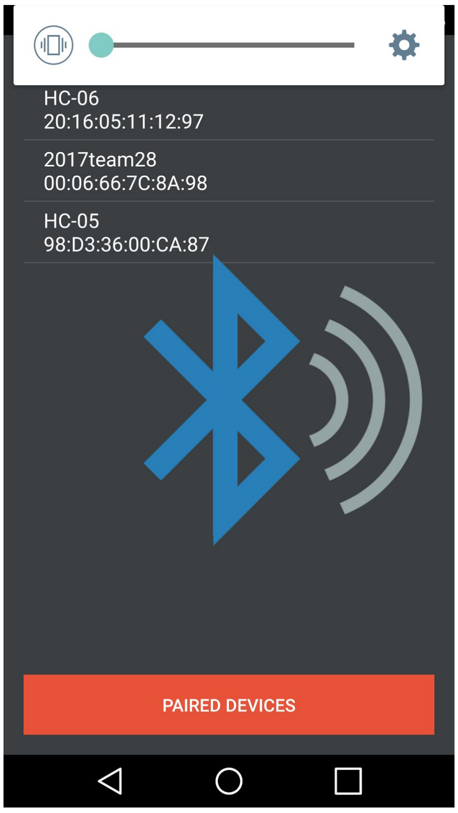
John: Can u hear me?

Yu: yes

John: OK that's cool

John: OK that's cool





HC-06
20:16:05:11:12:97

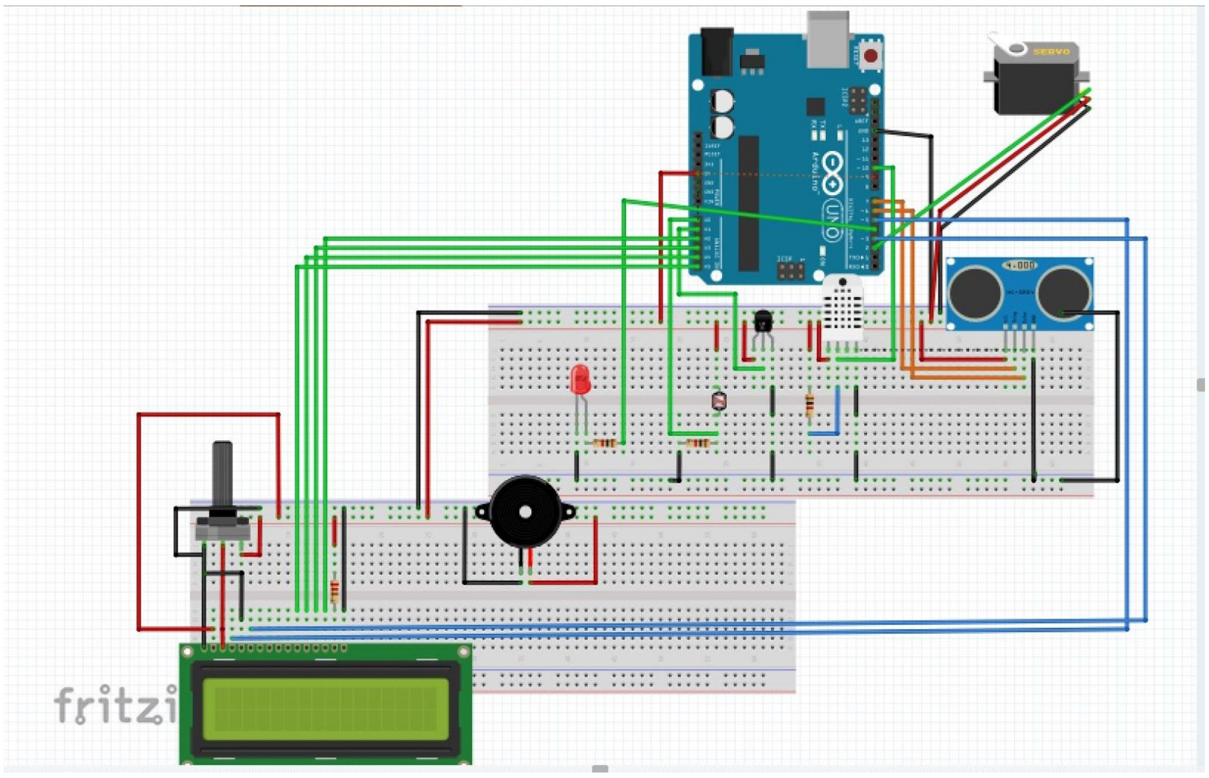
2017team28
00:06:66:7C:8A:98

HC-05
98:D3:36:00:CA:87

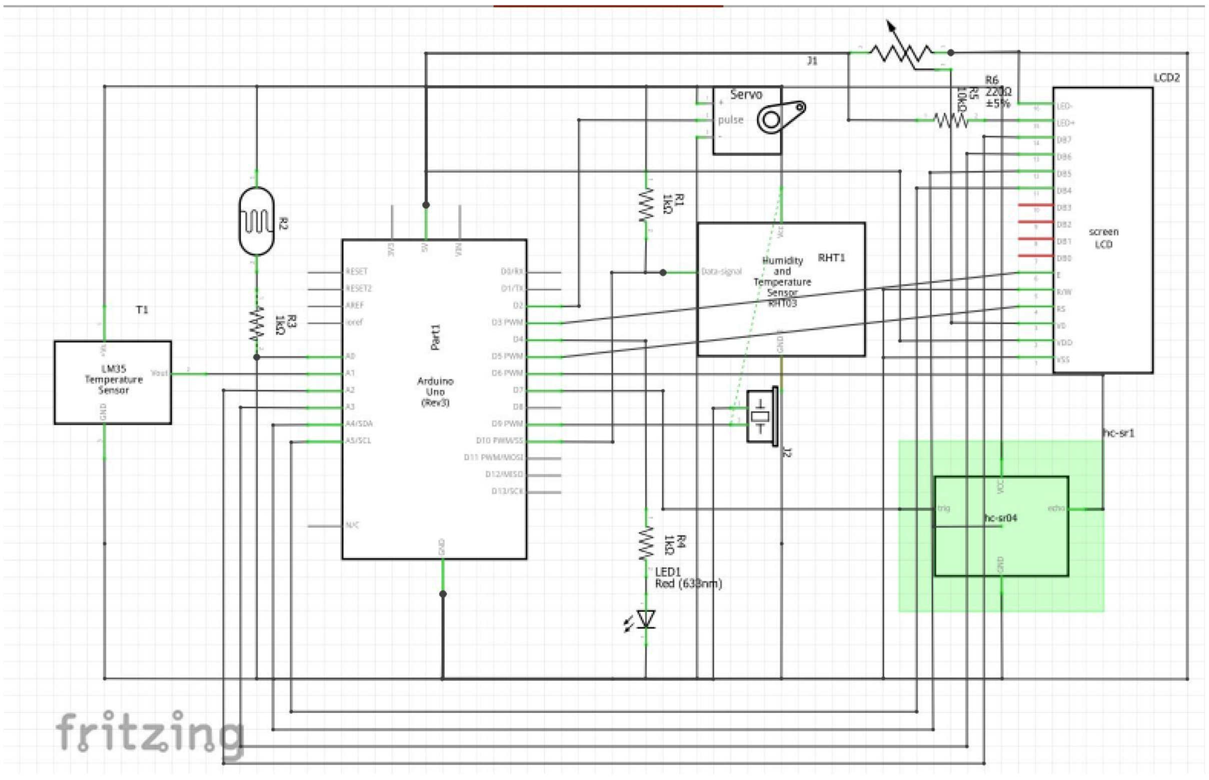
Appendix B – Component list

- Logitech C170 Webcam with microphone (Quantity: 1)(Provided by Yuxiang)
- Speakers(Quantity: 2)(Provided by Yuxiang)
- HC-SR04: Ultrasonic Range Finder Sensor Module (Quantity: 1)
- Arduino UNO Rev3 Board(Quantity:1)
- 1.5m USB Cable Type A to B(Quantity: 1)
- Standard LCD 16x2 + 10K POT + header(Quantity: 1)
- On-Off Power Button / Pushbutton Toggle Switch (Quantity: 1)
- Perma-Proto Half-sized Breadboard PCB [perf-board prototyping board](Quantity: 1)
- Breadboard from ELEC 201 lab kit(Quantity: 1)
- Micro servo motor(Quantity: 1)
- Raspberry Pi:
    - Raspberry Pi 3 with 16G MicroSD(Quantity: 2)(Purchased one from RP Electronics: $60/item)
    - Raspberry Pi 3 Official Power Supply(Quantity: 2)
    - HDMI cable(Quantity: 2)
    - Raspberry Pi camera(Quantity: 1)(Provided by Yuxiang)
- Alexa:
    - Dual stereo speaker set(Quantity 1)(Provided by Yuxiang)
    - Video Camera with microphone(Quantity 1)(Provided by Yuxiang)
- Resistors(Quantity: 4)
    - 150 ohms ¼ watt 5%(Quantity: 3)
    - 1k ohms ¼ watt 5%(Quantity: 1)
    - TRIMMER 10K OHM 0.5W TH [variable resistor/POT](Quantity: 1)
- LM-35 Temperature sensor(Quantity: 1)
- DHT11 basic temperature-humidity sensor(Quantity: 1)
- HC-05/HC-06 Bluetooth module(Quantity: 1)(Purchases from Amazon: $11/item)
- Photo cell (CdS photoresistor)(Quantity: 1)
- Piezo Buzzer - PS1240(Quantity: 1)
- Red LED(Quantity: 1)
- Cupboard, small boxes, tape and glue(Quantity: Several)(Provided by Yuhao and YuXiang)

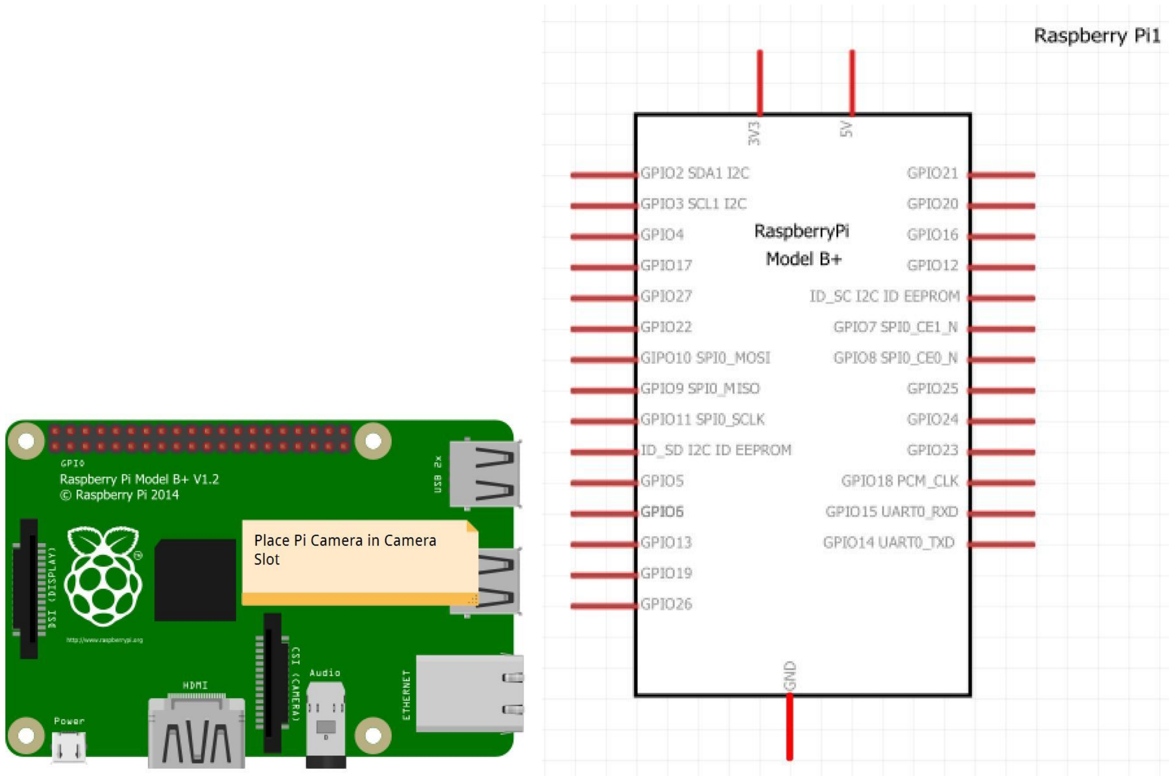
Appendix C - Fritzing



Arduino Breadboard Diagram



Arduino Schematic Diagram



Raspberry Pi with Camera (inside the house)

Appendix D – Arduino Code

```
#include "DHT.h"
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
#include <Servo.h>

SoftwareSerial BT(9, 10);            //bluetooth object
LiquidCrystal lcd(4,12,13,A0,A3,A4);  //lcd object

const int DHT_READ_PIN = 11;       //pin assignment of DHT sensor
const int DHT_TYPE=11;            //DHT model 11
const int LM_READ_PIN = A5;       //pin assignment of LM35 sensor
const int PTC_READ_PIN = A1;      //pin assignment of photocell
const int PRECISION=4;            //precision of data display
const int LED_PIN = 3;
const int PIEZO_PIN = 8;
const int THRESHOLD = 10;         //threshold for motion sensor/ultrasonic sensor
                                   //change the threshold to the farthest distance from the
direction the sensor points at

//For sonar
const int ECHO_PIN = 6;           //Pin for ECHO pin on the sonar
const int TRIG_PIN = 7;           //Pin for ECHO pin on the sonar
const int MOTOR_PIN = 5;          //Pin to control the motor    NOT USING FOR NOW

char data;
char dataSerial;                //
int led_status = 0;             //used to indicate if led is turned on or off
int alarm_status = 0;           //used to indicate if alarm is turned on or off
int alarm_alternation = 0;       //used to alternate between different pitches of the alarm
(piezo buzzer)
int servo_status = 0;           //used to indicate if the door is open or closed
int auto_mode_status = 0;        //Auto mode = security mode. This is used to indicate if
this mode is on or off
int triggered = 0;                //In Auto(Security) mode, this indicates if the intruder
protocol is triggered or not
int led_trigger_status = 0;     //used to indicate if led blinking pattern is on or off
int led_alternation = 0;        //used to alternate between on and off so that it blinks
float TmpLM;                    //holds temperature read from LM35
float lightLevel;               //holds photosensor input value
float humidity;                 //holds humidty input value from DHT 11
float distance;                 //holds distance value read from the ultrasonic sensor

Servo myServo;                  //global servo object

DHT dht(DHT_READ_PIN, DHT_TYPE);  //creator of dht library
```

```
void setup() {
  Serial.begin(9600);
  BT.begin(9600);

  //sensors
  pinMode(DHT_READ_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);

  //door initialization
  myServo.attach(MOTOR_PIN);

  //lcd initialization
  lcd.begin(16, 2);

  //for sonar
  pinMode(ECHO_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(PTC_READ_PIN, INPUT);

  doorClose();                  //have door shut
  digitalWrite(LED_PIN, LOW);   //have led start off for demo
}

/**
 * Read the switch status, and decide which style of display it shall follow
 */
void loop() {

  checkForCommand();

  TmpLM=readTmpLM();
  lightLevel=readLight();
  humidity=readHumDHT();
  distance=readSonar();

  checkAutoMode();

  checkAlarmStatus();     //checks alarm flag and turns on/off alarm accordingly
  checkLEDStatus();       //checks led trigger flag and turns on/off alarm accordingly

  String inputData = String(TmpLM) + " " + String(humidity) + " " + String(distance) + " "
+ String(lightLevel);
  Serial.println(inputData);     //Print it to Serial for the Raspberry Pi to be able to
read the values
                                //if the user asks Alexa what the input sensor values are

  updateViaBluetooth();     //Sends the input sensor values over Bluetooth
}
```

```
/*
 * Checks for auto_mode_status flag and evaluates if the distance
 *
 * Must come after reading input values
 * Must come before checkAlarmStatus() and checkLEDStatus()
 */
void checkAutoMode() {
  if(auto_mode_status == 1) {
     if(distance < THRESHOLD) {            //threshold is set to 10cm. Can be adjusted from
constants
      triggered = 1;                       //this flag will later turn on the alarm and turn
the blinking pattern in next if statement
      doorClose();                         //close door
    }
    if(triggered == 1) {
       alarm_status = 1;                  //alarm_status flag raised which will be evaluated
later in the loop() to turn the alarm on
        led_trigger_status = 1;             //led_trigger_status flag raised which will be
evalued later in the loop() to turn the led pattern on
    } else {                             //if not triggered, don't do the trigger activity
      alarm_status = 0;                  //don't raise the alarm flag
      led_trigger_status = 0;            //don't raise the led trigger pattern flag
    }
  }

  //if not auto)mode, don't even worry about the flags
}

/*
 * Checks if alarm_status flag is on, and if it is on, then turn the alarm on and vice
versa.
 */
void checkAlarmStatus() {
  if(alarm_status == 1) {            //if flag on
    if(alarm_alternation == 0){      //these two if statements allow the alternation between
the tones
      tone(PIEZO_PIN, 1865);         //hardcoded lower tone
      alarm_alternation = 1;         //switch to higher tone next time
    }
    else {
      tone(PIEZO_PIN, 1976);         //hardcoded higher tone
      alarm_alternation = 0;         //switch to lower tone next time
    }
  }
  else
    noTone(PIEZO_PIN);               //if flag is not raised, then don't turn on the alarm
}
```

```
/*
 * Checks for the LED trigger flag and turns it on and off accordingly
 */
void checkLEDStatus() {
  if(led_trigger_status == 1) {          //if trigger flag is on, then go into alternation
of light on and off for blinking effect
    if(led_alternation == 0) {
      digitalWrite(LED_PIN, HIGH);
      led_alternation = 1;
    } else {
      digitalWrite(LED_PIN, LOW);
      led_alternation = 0;
    }
   } else {                                    //if trigger flag is off, then there is no
alternation of blinking
         led_alternation = 0;                  //Either led is turned on or off from the
checkForCommands(), not here
  }
}

/*
 * Passes the input sensor readings to the bluetooth socket so that the Android phone can
liten to the values and update
 * the readings on the phone. Then the Android will send these data to the server which
then the server will send the data to other
 * html clients as well
 *
 * Refer to function below this one for the importance of addOneToLeft function
 */
void updateViaBluetooth() {
  BT.print(addOneToLeft(TmpLM, "1"));
  delay(200);
  BT.print(addOneToLeft(lightLevel, "2"));
  delay(200);
  BT.print(addOneToLeft(distance, "3"));
  delay(200);
  BT.print(addOneToLeft(humidity, "4"));
  delay(200);
}

/*
 * This function is VERY importent in Bluetooth communication.
 * This function addresses two problems.
 * Problems: 1) When passing values from Arduino to Bluetooth, the first character in a
string is always lost on the other side.
 *             Soln: We just attach a random "9" in the beginning so no data is lost
 *            2) There is a limit on how many bytes we can send over Bluetooth.
 *              Soln: We number code the values we want to pass and send multiple smaller
bytes.
 *                This number coding is followed after the first character that is lost and
is specified by the input parameter
 */
float addOneToLeft(float value, String beginNum) {
  String string1 = String(value);
  String string2 = "9" + beginNum + string1;
  return string2.toFloat();
}



/*
  * Checks for command given by the Bluetooth (Android) and the Raspberry Pi of voice
control (Alexa)
 * The commands are letter coded
 */
void checkForCommand() {
  //checks Bluetooth for commands
   if(BT.available()) {         //some function names are self-explanatory given the letter
coding
    data = BT.read();
    if(data == 'L') {
      LedOnOff();

    } else if(data == 'P') {
      AlarmOnOff();

    } else if(data == 'S') {
      ServoOnOff();

    } else if (data == 'A') { //turn on auto/security mode
      auto_mode_status = 1;

    } else if (data == 'R') { //reset the auto/security mode
      triggered = 0;          //lower trigger flag
      digitalWrite(LED_PIN, LOW);   //turn off the led

    } else if (data == 'O') { //turn off auto/security mode
      triggered = 0;            //lowerering trigger activity flags
      alarm_status = 0;         //""
      led_trigger_status = 0;   //""
      auto_mode_status = 0;     //lowers auto/security mode flag

     } else if (data == 'T') {    //Command to update the LCD clock time. This was not used
during demo
      String time = BT.readString();
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(time);

    } else if (data == 'C') {   //command to write messages (from html client) to the lcd
      String text = BT.readString();
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(text);
    }
  }

  //checks Serial (Raspberry pi of voice control) for commands
  //these commands are activated if you talk to Alexa
  if(Serial.available()) {
     dataSerial = Serial.read();         //function names are self-explanatory given the
letter coding
    if(dataSerial == 'N') {
      LedOn();

    } else if(dataSerial == 'F') {
      LedOff();

    } else if(dataSerial == 'O') {
      doorOpen();

    } else if(dataSerial == 'C') {
      doorClose();

    } else if(dataSerial == 'A') {
      alarmOn();

    } else if(dataSerial == 'B') {
      alarmOff();
    }
  }
}
/*
 * Raises alarm flag
 */
void alarmOn() {
   alarm_status = 1;
}
/*
 * Lowers alarm flag
 */
void alarmOff() {
   alarm_status = 0;
}
```

```
/*
 * Opens the house door and raises the servo_status flag to indicate that the door is open
right now
 */
void doorOpen() {
  myServo.write(0);
  servo_status = 1;
}
/*
 * Closes the door and lowers the servo_status flag
 */
void doorClose() {
  myServo.write(90);
  servo_status = 0;
}
/*
 * Same as the two functions above, but this function makes logic easier for button (on
Android and html client) implementation
 */
void ServoOnOff() {
  if(servo_status == 0){
    myServo.write(0);
    servo_status = 1;
  }
  else {
    myServo.write(90);
    servo_status = 0;
  }
}
/*
 * Used for button logic (in Android and html client)
 * If alarm flag on, then turn it off. If alarm flag is off, then turn it on
 */
void AlarmOnOff() {
  if(alarm_status == 0)
    alarm_status = 1;
  else
    alarm_status = 0;
}

/*
 * Used for button logic (in Android and html client)
 * If led flag is on, turn led off and turn flag off
 * If led flag is off, turn led on and turn the flag on
 */
void LedOnOff() {
  if(led_status == 0) {
   digitalWrite(LED_PIN, HIGH);
   led_status = 1;
  }
  else {
   digitalWrite(LED_PIN, LOW);
   led_status = 0;
  }
}

/*
 * Turns led on and sets flag
 */
void LedOn() {
  digitalWrite(LED_PIN, HIGH);
  led_status = 1;
}

/*
 * Turns led off and sets flag
 */
void LedOff() {
  digitalWrite(LED_PIN, LOW);
  led_status = 0;
}

//__________________Functions    below    were    brought    from    the    previous    labs
1-5______________________

/**
 * Function: Read from analog pin the LM35 temperature sensor is connected to, and map the
voltage to temperature in degree celsuis
 * Param: None
 * Return: Float value indicates the temperature reading from LM35, nan if the sensor is
not ready
*/
float readTmpLM(){
  float readVoltage = analogRead(LM_READ_PIN);
  delay(10);
  readVoltage = analogRead(LM_READ_PIN);        //Read twice and discard the first value so
that the reading is not interfered by noise
  float Tmp = mapping(readVoltage, 0, 225, 0, 100);
  return Tmp;
  }

/**
 * Function: Read from digital pin the DHT11 sensor is connected to, and map the voltage to
temperature in degree celsuis
 * Param: None
 * Return: Float value indicates the temperature reading from DHT11, 'nan' if the sensor is
not ready
*/
float readTmpDHT(){
  return dht.readTemperature();
  }

/**
 * Function: Read from digital pin the DHT11 sensor is connected to, and map the voltage to
humidity in relative(%)
 * Param: None
 * Return: Float value indicates the humidity reading from DHT11, 'nan' if the sensor is
not ready
*/
float readHumDHT(){
  return dht.readHumidity();
  }

/**
 * Function: Read from analog pin the photocell sensor is connected to, and map the voltage
to ambient light level
 * Param: None
 * Return: the light level reading from the photocell
*/
float readLight(){
  float readVoltage=analogRead(PTC_READ_PIN);
  delay(10);
  readVoltage=1023-analogRead(PTC_READ_PIN);
   float reversedValue=1023-readVoltage;             //The voltage and light level have a
reversed propotional relationship
  float lightLevel=mapping(reversedValue, 0, 1023, 0, 100);
  return lightLevel;
  }

/**
 * Funtion: Basically same as the map() function in the standard ardurino math library, but
able to convert it into float type.
 * The mapping is linear.
 * Param: float value: the value to be mapped
 *        inputLowerBond: The possible minium of the input value.
 *        inputUpperBond: The possible maxium of the output value.
 *        outputLowerBond: The wished value of which the minium input is mapped to
 *        outputUpperBond: The wished value of which the maxium input is mapped to
 * Return: float type being the value after mapping.
 * Preconditions: inputLowerBond<inputUpperBond && outputLowerBond<outputUpperBond
*/
```

```
float    mapping(float    value,    float    inputLowerBond,    float    inputUpperBond,    float
outputLowerBond, float outputUpperBond){
      float scale =(outputUpperBond - outputLowerBond)/(inputUpperBond - inputLowerBond);
      float offset = outputLowerBond - scale * inputLowerBond;
      return scale*value+offset;      //The process above can be mathmatically proved to be
effective
  }

/**
 * Function: Control the sonar to send a pulse, and measure the duration from the echo,
calculate the distance as per temperature
 * returns: a float that indicates the distance it gets, in cm, from 0 to 200, -1 if the
range is not reasonable.
*/
float readSonar(){
    float duration, distance;
    digitalWrite(TRIG_PIN, HIGH);           //set trigger pin to HIGH
    delayMicroseconds(1000);
    digitalWrite(TRIG_PIN, LOW);            //set trigger pin to LOW
    duration = pulseIn(ECHO_PIN, HIGH);     //read echo pin
    float temp=readTmpLM();                 //read temperature
     float sound_speed=331.5 + (0.6 * temp);                 //calculate the sound speed at
the point
    distance = (duration * sound_speed * 0.0001)/2;        //compute distance from duration
of echo Pin
    delay(200);
      if  (distance  >=  400  ||  distance  <=  0){    //deciding  whether  or  not  distance  is
reasonable
        return(-1);                         //if not, return -1
    }
    else{
        return(distance);
    }
}
```

Appendix E – Raspberry Pi 3 Code

```
#import flask and flask_ask as Python framework
#to simplify the skill development procedure
from flask import Flask
from flask_ask import Ask, statement
import requests
import json

#import serial library to enable serial communication
import serial

#import os library in order to enable Voice Control system to
#execute Linux commands and shell scripts. This significantly
#expands the capability of voice control system. Opening web browser
#function is achieved using this method
import os

#Setting things up
ser = serial.Serial("/dev/ttyACM0", 9600)

app = Flask(__name__)
ask = Ask(app, '/')

@ask.launch

#link the function to the intent "LightOn"
@ask.intent("LightOn")
#Once the Alexa Web Service receives the words "LightOn", the
#intent "LightOn" will be triggered and the function on() below
#will be executed
def on():
    #use a for loop to send the byte 'N' to the
       #Arduino for 10 times, reducing the possibility
       #that Arduino fails to receive the signal
    for a in xrange(1, 10):
              #Send the byte 'N' to Arduino using serial communication
        ser.write(b'N')

       #This is the return statement which is supposed to be first sent
       #to Amazon Web Server, and later returned in audio format to be
       #played on local machine
    return statement("Hall light turned on.")

#The code below has the same structure and foramt as the previous code,
#so most of the comments are omitted
@ask.intent("LightOff")
def off():
    for b in xrange(1, 10):
        ser.write(b'F')

    return statement("Hall light turned off.")
@ask.intent("DoorOpen")
def dooron():
    for c in xrange(1, 10):
       ser.write(b'O')

    return statement("Door opened")

@ask.intent("DoorClose")
def dooroff():
    for d in xrange(1, 10):
        ser.write(b'C')

    return statement("Door closed")

@ask.intent("Temperature")
def temperature():
    data = ser.readline().split()
    temp = data[0]
    #return statement("Current Temperature is %." % (temp))
    return statement("Current temperature is" + temp + "Degree Celsius")

@ask.intent("Humidity")
def humidity():
    data = ser.readline().split()
    hum = data[1]
    #return statement("Current Temperature is %." % (temp))
    return statement("Current humidity is" + hum + "percent")

@ask.intent("Distance")
def distance():
    data = ser.readline().split()
    dis = data[2]
    #return statement("Current Temperature is %." % (temp))
    return statement("There is no object within" + dis + "centimeters")

@ask.intent("Light")
def light():
    data = ser.readline().split()
    light = data[3]
    #return statement("Current Temperature is %." % (temp))
    return statement("The light level is" + light)

@ask.intent("BuzzerOn")
def buzzerOn():
    for e in xrange(1, 10):
      ser.write(b'A')

    return statement("Alarm turned On")

@ask.intent("BuzzerOff")
def buzzerOff():
    for f in xrange(1, 10):
      ser.write(b'B')

    return statement("Alarm turned Off")

@ask.intent("Joke")
def joke():

     return statement("Where are you going he asked. I'm leaving you because I found out
you're a pedophile." + "Whoa whoa, he says. Pedophile." + "That's a pretty big word for an
eight-year-old")

@ask.intent("OpenWebsite")
def web():

       #Execute the Linux command below to open the url using
       #the default web browser
    os.system("xdg-open http://38.88.74.87")

    return statement("Victoria Security main page is opening")

#The code below has the same structure and foramt as the previous code,
#so the comments are omitted
@ask.intent("OpenGmail")
def gmail():

    os.system("xdg-open https://mail.google.com/mail/u/0/")

    return statement("Your gmail is opening")

@ask.intent("OpenChatroom")
def chatroom():

    os.system("xdg-open http://38.88.74.87/Subwebsite/chatRoom.html")

    return statement("Victoria Security chatroom is opening")

@ask.intent("Test")
def test():

    return statement("The computer is fine")
```

```
@ask.intent("OpenSurveillance")
def videoStream():

    os.system("xdg-open http://128.189.84.138:3000")

    return statement("Victoria Security video servallence is opening")

@ask.intent("OpenDatapage")
def datapage():

    os.system("xdg-open http://38.88.74.87/Subwebsite/Monitoring.html")

    return statement("Victoria Security monitoring page is opening")

@ask.intent("OpenFacebookPage")
def facebookPage():

    os.system("xdg-open https://www.facebook.com/Victoria-Security-1821395828176172/")

    return statement("Victoria Security facebook page is opening")

if __name__ == "__main__":
    app.run(debug=True)
```

Appendix F – Web app code

Index.html (homepage):

(Since the code is too long, we only include a portion of our codes that we used to create the basic feature of this html)

```
       <div class="top-grids" id="services" >

                             <div class="wrap">
                                    <div class="top-grid">
                                           <a               class="icon                man"
href="./Subwebsite/Monitoring.html"> </a>
                                           <a
href="./Subwebsite/Monitoring.html">Monitoring</a>
                                           <p>Your Personal Assistant</p>
                                    </div>
                                    <div class="top-grid">
                                           <a             class="icon             monitor"
href="./Subwebsite/chatRoom.html"> </a>
                                           <a       href="./Subwebsite/chatRoom.html">Chat
Room</a>
                                           <p>Being Connected</p>
                                    </div>
                                    <div class="top-grid">
                                           <a             class="icon              colors"
href="http://128.189.84.138:3000" target="_blank"> </a>
                                           <a            href="http://128.189.84.138:3000"
target="_blank">Security Camera</a>
                                           <p>Your Scope Has No Limit</p>
                                    </div>

                                    <div class="clear"> </div>
                             </div>
                      </div>
                      <!--- //End-top-services-grids---->
                      <!---start-process---->
                      <div class="process-grids" id="Introduction";>
                             <div class="wrap">
                                    <div class="process-top-grid">
                                           <div class="process-top-grid-left">
                                                  <h3>Project Introduction</h3>
                                                  <p>  Our Introduction here
</p>
                                           </div>
                                           <div             class="process-top-grid-right"
style="position: absolute;
```

```
    right: 50px;
    top: 1325px;">
                                                    <a                          herf="#"><img
src="images/divices.png" title="divices" /></a>
                                              </div>
                                              <div class="clear"> </div>
                                       </div>
                      </div>
                      <!---//End-process---->
                      <!---start-portfolio--->

                      <div class="portfolio" id="portfolio">
                             <div class="recent-works" id="portfolio">
                             <div class="recent-works-head">
                                    <h3>Group Members</h3>
                             </div>
                             <div class="wrap">
                             <!---start-mfp ---->


                      <div id="small-dialog1" class="mfp-hide">
                             <div class="pop_up">
                                    <h2>Yu Chen</h2>
                                    <p class="para">Web Development</p>
                             </div>
                      </div>

                      <div id="small-dialog2" class="mfp-hide">
                             <div class="pop_up">
                                    <h2>Hanyu WU</h2>
                                    <p class="para">Web Development</p>
                             </div>
                      </div>

                      <<div id="small" class="mfp-hide">
                             <div class="pop_up">
                                    <h2>Yuhao Huang</h2>
                                    <p class="para">Web Development</p>
                             </div>
                      </div>

                      <div id="small-dialog4" class="mfp-hide">
                             <div class="pop_up">
                                    <h2>Benjamin Hong</h2>
                                    <p class="para">Raspberry Pi</p>
                             </div>
                      </div>
                     <div id="small-dialog5" class="mfp-hide">
                             <div class="pop_up">
                                     <h2>Ziqiao Lin</h2>
                                     <p class="para">Android App</p>
                             </div>
                     </div>

                     <div id="small-dialog6" class="mfp-hide">
                             <div class="pop_up">
                                     <h2>Yuxiang Huang</h2>
                                     <p class="para">Arduino</p>
                             </div>
                     </div>
                     <!---end-mfp ---->
                             <div class="gallery">
                                     <div class="container">
                                     <ul id="filters" class="clearfix">
                                             <li><span         class="filter         active"
data-filter="yuchen hanyu yuhao ben ziqiao yuxiang">All</span></li>
                                             <li><span   class="filter"   data-filter="yuchen
hanyu yuhao">Web Team</span></li>
                                             <li><span  class="filter"  data-filter="yuxiang
ziqiao">Hardware Team</span></li>
                                             <li><span    class="filter"    data-filter="ben
ziqiao">Android Team</span></li>
                                     </ul>
                                             <div id="portfoliolist" style="     ">


                                             <div     class="portfolio     yuchen     mix_all"
data-cat="yuchen" style="display: inline-block; opacity: 1;">
                                                   <div class="portfolio-wrapper">
                                                          <a   class="popup-with-zoom-anim"
href="#small-dialog1">
                                                                      <img
src="images/yuchen-chris.png" alt="Image 2" style="top: 0px;">
                                                          </a>
                                                   </div>
                                             </div>
                                             <div     class="portfolio     hanyu     mix_all"
data-cat="hanyu" style="display: inline-block; opacity: 1;">
                                                   <div class="portfolio-wrapper">
                                                          <a   class="popup-with-zoom-anim"
href="#small-dialog2">
                                                                      <img
src="images/hanyu.png" alt="Image 2" style="top: 0px;">
                                                          </a>
                                                   </div>
                                             </div>
```

```
                        <!---start-contact---->
                        <div class="contact" id="contact">
                               <div class="contact-head">
                                      <h3>Keep calm, Create awesome Stuff</h3>
                               </div>
                               <div class="wrap">
                                      <div class="contact-info">
                                             <div class="getin-touch">
                                                    <h4>GET IN TOUCH</h4>
                                                    <ul>
                                                           <li><a          class="facebook"
href="#"><span> </span></a></li>
                                                           <li><a            class="github"
href="#"><span> </span></a></li>
                                                           <div class="clear"> </div>
                                                    </ul>
                                                    <a
href="https://github.com/CPEN-291/G5_A_P2">
  <img src="images/ghwhite.png" alt="Github Repo" style="width:44px;height:42px;border:0">
</a>
<a href="https://www.facebook.com/Victoria-Security-1821395828176172/">
<img src="images/fbwhite.png" alt="Facebook Page" style="width:48px;height:42px;border:0">
</a>
                                             </div>
                                             <div class="subscribe-form">
                                                    <form>
                                                    </form>
                                             </div>
                                      </div>
                               </div>
                        </div>
                        <!---//End-contact---->
                 </div>
                 <!----//End-content--->
                 <!----//End-wrap---->
<div style="display:none"><script src='http://v7.cnzz.com/stat.php?id=155540&web_id=155540'
language='JavaScript' charset='gb2312'></script></div>
</body>
</html>
```

Monitoring.html:

(Since the code is too long, we only include a portion of our codes that we used to create the basic feature of this html)

<!--Here are the codes we used to display real-time sensor values and control button -->

```
<div class="span9">
       <h1 class="page-title">
              <i class="icon-home"></i>
              Monitoring
       </h1>
       <div class="stat-container">
              <div class="stat-holder">
                     <div class="stat">
                            <span id="temp">NULL</span>
                            Temperature in Celsius
                     </div> <!-- /stat -->
              </div> <!-- /stat-holder -->
              <div class="stat-holder">
                     <div class="stat">
                            <span id="humidity">NULL</span>
                            Humidity Percentage %
                     </div> <!-- /stat -->
              </div> <!-- /stat-holder -->
              <div class="stat-holder">
                     <div class="stat">
                            <span id="lightlevel">NULL</span>
                            Photosensor Percentage %
                     </div> <!-- /stat -->
              </div> <!-- /stat-holder -->
              <div class="stat-holder">
                     <div class="stat">
                            <span id="distance">NULL</span>
                            Ultrasonic Range /cm
                     </div> <!-- /stat -->
              </div> <!-- /stat-holder -->
       </div> <!-- /stat-container -->
       <div class="widget">
              <div class="widget-header">
                     <i class="icon-signal"></i>
                     <h3>Control</h3>
              </div> <!-- /widget-header -->
              <div class="widget-content">
```

```
                                            <div id="bar-chart" >
                                                   <button id="led">LED on/off</button>
                                            </div>
                                            <div id="bar-chart" >
                                                   <button id="piezo" style="
    left: 150px;
    /* background-color: red; */
    top: 62px;
    position: absolute;
">Alarm</button>
                                            </div>
            <div id="bar-chart" >
              <button id="motor" style="
    left: 250px;
    /* background-color: red; */
    top: 62px;
    position: absolute;
  ">Motor</button>
            </div>
            <div id="bar-chart" >
              <button id="Auto-button" style="
    left: 350px;
    /* background-color: red; */
    top: 62px;
    position: absolute;
  ">Auto</button>
            </div>
            <div id="bar-chart" >
              <button id="reset-button" style="
    left: 450px;
    /* background-color: red; */
    top: 62px;
    position: absolute;
  ">Reset</button>
            </div>
            <div id="bar-chart" >
              <button id="turnOff-button" style="
    left: 550px;
    /* background-color: red; */
    top: 62px;
    position: absolute;
  ">TurnOff</button>
            </div>
               </div> <!-- /widget-content -->
                              </div> <!-- /widget -->
                                     </div> <!-- /widget-content -->
                              </div> <!-- /widget -->
                       </div> <!-- /span9 -->
               </div> <!-- /row -->
        </div> <!-- /container -->
</div> <!-- /content -->
```

<!--Here is the javascript code to emit events to server -->

```
<script>
              socket.on('temp', function(data) {
                     document.getElementById("temp").innerHTML=data;
              });
              socket.on('light', function(data) {
                     document.getElementById("lightlevel").innerHTML=data;
              });
              socket.on('humidity', function(data) {
                     document.getElementById("humidity").innerHTML=data;
              });
              socket.on('distance', function(data) {
                     document.getElementById("distance").innerHTML=data;
              });

              document.getElementById("led").onclick = function() {
                     socket.emit('led', 'led on/off');
              }
              document.getElementById("piezo").onclick = function() {
                     socket.emit('piezo', 'piezo on/off');
              }
    document.getElementById("motor").onclick = function() {
                     socket.emit('motor', 'motor on/off');
              }
    document.getElementById("Auto-button").onclick = function() {
                     socket.emit('Automode', 'Automode on/off');
              }
    document.getElementById("reset-button").onclick = function() {
```

```
                      socket.emit('Reset', 'reset on/off');
               }
    document.getElementById("turnOff-button").onclick = function() {
                      socket.emit('turnOff', 'turn on/off');
               }
</script>
```

chatroom.html:

(Since the code is too long, we only include a portion of our codes that we used to create the basic feature of this html)

<!--Here are the codes we used to design our chatroom -->

```
                      <div class="span9">
                             <h1 class="page-title">
                                    <i class="icon-signal"></i>
                                    Chatroom
                             </h1>

<div class="widget">


                                    <div class="widget-header">
                                           <i class="icon-signal"></i>
                                           <h3>Chat</h3>
                                    </div> <!-- /widget-header -->

                                    <div class="widget-content">
                                           <div       id="log_chat"       style="overflow:
scroll"></div>

                                           <div id="send-contrl-chat" >
                         <input  type="text"  placeholder="Your  name",  id="text-chat-name"
style="height: 18px;margin: 10px auto;width:100px">
                                                  <input   type="text"   placeholder="Your
message here", id="text-chat" style="height: 18px;margin: 10px auto;">
                                                  <button   id="send-chat"   style="height:
27px">Send message</button>
                                           </div>
                                    </div> <!-- /widget-content -->
                             </div> <!-- /widget -->
```

```
                              <div class="widget">

                                     <div class="widget-header">
                                            <i class="icon-signal"></i>
                                            <h3>Communicate with Arduio</h3>
                                     </div> <!-- /widget-header -->

                                     <div class="widget-content">
                                            <div   id="log_chat_arduino"   style="overflow:
scroll"></div>
                                            <div id="send-contrl-chat" >
                                                   <input   type="text"   placeholder="Your
message  here",  id="text-chat-arduino"  style="height:  18px;margin:  10px  auto;  overflow:
scroll">
                                                   <button           id="send-chat-arduino"
style="height: 27px">Send to LCD</button>
                                            </div>
                                     </div> <!-- /widget-content -->
                              </div> <!-- /widget -->
                                     </div> <!-- /widget-content -->
                              </div> <!-- /widget -->

                       </div> <!-- /span9 -->
               </div> <!-- /row -->
       </div> <!-- /container -->
</div> <!-- /content -->
<!--Here is the javascript code to emit events to server  -->
<script>
               socket.on('chat',function(msg) {
      console.log(msg);
                       log_chat.innerHTML+=msg+"</br>";
               });
               document.getElementById('send-chat').onclick=function () {
      var name=document.getElementById('text-chat-name').value;
      var text=document.getElementById('text-chat').value;
      if(name.length==0){
       name="Anon.";
       }
      var words=name+": "+text;
                       log_chat.innerHTML+=words+"</br>";
```

```
                    console.log('I send '+words);
                    socket.emit('message',words);
        document.getElementById('text-chat').value="";
            }
            socket.on('chat message to arduino',function(msg) {
                    console.log(msg);
                    log_chat_arduino.innerHTML+=msg+"</br>";
            });
            document.getElementById('send-chat-arduino').onclick=function () {
                    var text=document.getElementById('text-chat-arduino').value;
        log_chat_arduino.innerHTML+=text+"</br>";
                    console.log('I send '+text);
                    socket.emit('LCDtoServer',text);
            }
</script>
```

videostreaming.html:

(Since the code is too long, we only include a portion of our codes that we used to create the basic feature of this html)

```
<!--This is where we put our video streaming  -->
                                    <div    class="widget-content"    id="video    streaming"
style="text-align:center">
                <img  style="-webkit-user-select:  none;  "  src="http://128.189.84.138:3000/"
width="640" height="480">
                                    </div> <!-- /widget-content -->
```

Appendix G – Mobile app code

LED-Control Class

```
package com.led_on_off.led;

import android.net.Uri;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;

import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.AsyncTask;
import android.util.Log;

import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.UUID;
import android.os.Handler;

import org.w3c.dom.Text;

//import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.github.nkzawa.emitter.Emitter;
import com.github.nkzawa.socketio.client.IO;
import com.github.nkzawa.socketio.client.Socket;
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.appindexing.Thing;
import com.google.android.gms.common.api.GoogleApiClient;
import org.json.JSONException;
import org.json.JSONObject;

import java.net.URISyntaxException;


public class ledControl extends ActionBarActivity {

    // GUI Attributes
    ImageButton Discnt, chatRoomButton;   //BImage buttons
      TextView  Temperature, Photosensor,Humidity, ultrasonicSensor;   //define 4 different
reading textviews
                  Button     LED_Control,     AlarmButton,ServoButton,     AutoModelButton,
ResetButton,TureOffButton; // define 6 different buttons

    //Used for bluetooth
    String address = null;
    private ProgressDialog progress;
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;

    //Used by the thread that checks for incoming data from inputStream of btSocket
    boolean stopThread; //indicator to stop the thread or turn it on
    byte buffer[];      //if there is available info, store it here

    //SPP UUID. Look for it. Used to identify the bluetooth object to connect to
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    private Socket socket;
    /**
     * ATTENTION: This was auto-generated to implement the App Indexing API.
     * See https://g.co/AppIndexing/AndroidStudio for more information.
     */
    private GoogleApiClient client;

    {
        try {
                  socket  =  IO.socket("http://38.88.74.87:9000");   //  socket  connects  to
38.88.74.87:9000
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent newint = getIntent();
        address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS); //receive the address of
the bluetooth device

        //view of the ledControl
        setContentView(R.layout.activity_led_control);

        //call the widgets

        // connect each textView and Button by findViewById
        Discnt = (ImageButton) findViewById(R.id.discnt);
        chatRoomButton = (ImageButton) findViewById(R.id.chatRoomButton);
        Temperature = (TextView) findViewById(R.id.temperature);
        Humidity = (TextView) findViewById(R.id.humidity);
        ultrasonicSensor = (TextView) findViewById(R.id.ultrasonicsensor);
        Photosensor = (TextView)findViewById(R.id.photosensor);
        LED_Control = (Button) findViewById(R.id.led);
        AlarmButton = (Button) findViewById(R.id.alarmButton);
        ServoButton = (Button) findViewById(R.id.servobutton);
        AutoModelButton = (Button) findViewById(R.id.AutoModel);
        ResetButton = (Button) findViewById(R.id.Reset);
        TureOffButton = (Button) findViewById(R.id.TurnOfModel);

        new ConnectBT().execute(); //Call the class to connect

        //From here on until the method ends, all these codes attaches methods to buttons
        //That is, when a button is pressed, a method associated with the button activates


        Discnt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Disconnect(); //close connection
            }
        });

        LED_Control.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                LEDOnOff(); //close connection
            }
        });

        AlarmButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                BuzzerAlarmOnOff(); //close connection
            }
        });
        ServoButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnServo2(); //close connection
            }
        });
        AutoModelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Automodel(); //close connection
            }
        });

        ResetButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Reset(); //close connection
            }
        });

        TureOffButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TurnOff(); //close connection
            }
        });



        socket.connect();
        socket.on("ledToAndroid", led_control); // socket listen on ledToAndroid event
           socket.on("piezoToAndroid", piezo_control); // socket listen on piezoToAndroid
event
        socket.on("motorToAndroid",turnServo1);  // socket listen on motorToAndroid event
         socket.on("AutoModeAndroid",autoModel_control); //socket listen on AutoModeAndroid
event
        socket.on("ResetAndroid",reset_control);  //socket listen on ResetAndroid event
          socket.on("TurnOffAndroid",turnOff_control);   //socket listen on TurnOffAndroid
event
        //socket.on("TimeAndroid",time_control);
        socket.on("LCDAndroid",LCD_control);  //socket listen on LCDAndroid event


        // ATTENTION: This was auto-generated to implement the App Indexing API.
        // See https://g.co/AppIndexing/AndroidStudio for more information.
```

```
    client = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
}






private Emitter.Listener led_control = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        ledControl.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                LEDOnOff();
            }
        });
    }
};  // led_control wiil call LEDOnOff

private Emitter.Listener piezo_control = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        ledControl.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                BuzzerAlarmOnOff();
            }
        });
    }
};  // piezo_control will call BuzzerAlarmOnOff
private Emitter.Listener turnServo1 = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        ledControl.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                turnServo2();
            }
        });
    }
};  // turnServo1 will call turnServo2 method
private Emitter.Listener autoModel_control = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        ledControl.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Automodel();
                }
            });
        }
    };  // autoModel_control will call Autmodel method
    private Emitter.Listener reset_control = new Emitter.Listener() {
        @Override
        public void call(final Object... args) {
            ledControl.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Reset();
                }
            });
        }
    };  // reset_control will call Reset method
    private Emitter.Listener turnOff_control = new Emitter.Listener() {
        @Override
        public void call(final Object... args) {
            ledControl.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    TurnOff();
                }
            });
        }
    };  //turnOff_control will call TurnOff method


    /*
     *LCD_control will call this method directly, and this method will send data received
     * from Web Server, convert it to String and send it to LCD via Bluetooth
     *
     */
    private Emitter.Listener LCD_control = new Emitter.Listener() {
        @Override

        public void call(final Object... args) {
            ledControl.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                        String LCD_message = (String) args[0]; // convert args[0] to string
type
                    if (btSocket != null) {  // check btSocket is connected or not
                        try {
                              String data = "C" + LCD_message;  // send LCD_message to LCD
via Bluetooth
                            btSocket.getOutputStream().write(data.getBytes());
                        } catch (IOException e) {
                            msg("Error from LCD_control ");
                        }
                    }

                }
            });
        }
    };

    /*
     * this method will direct send "S" to Arduino via Bluetooth
     */
    private void turnServo2() {
        if (btSocket != null) {  // check whether Bluetooth is connected or not
            try {
                btSocket.getOutputStream().write("S".toString().getBytes()); //send data to
Bluetooth
            } catch (IOException e) {
                msg("Error from LEDOnOff ");
            }
        }
    }

    /*
    * this method will direct send "L" to Arduino via Bluetooth
    */
    private void LEDOnOff() {
        if (btSocket != null) {
            try {
                 btSocket.getOutputStream().write("L".toString().getBytes());  // send data
to Bluetooth
            } catch (IOException e) {
                msg("Error from LEDOnOff ");
            }
        }
    }
    /*
     *this method will get currentTime
     * @return string type time that format as HH:MM:SS
     */
    public static String getCurrentTime() {
        //date output format
        DateFormat dateFormat = new SimpleDateFormat("HH:mm:ss"); // get time format
        Calendar cal = Calendar.getInstance();  // receive current time
        return dateFormat.format(cal.getTime()); //return the time as format "HH:mm:ss"
    }

    /*
```

```
     * this method will direct send "P" to Arduino via Bluetooth
     */
    public void BuzzerAlarmOnOff(){
        if (btSocket != null) {
            try {
                btSocket.getOutputStream().write("P".toString().getBytes()); //send data to
Bluetooth
            } catch (IOException e) {
                msg("Error from BuzzerAlarmOnOff" );
            }
        }
    }


    /*
     * this method will direct send "A" to Arduino via Bluetooth
     */
    public  void Automodel(){
        //TOBE:
        if (btSocket != null) {
            try {
                btSocket.getOutputStream().write("A".toString().getBytes()); //send data to
Bluetooth
            } catch (IOException e) {
                msg("Error from Automodel method" );
            }
        }
    }

    /*
     * this method will direct send "R" to Arduino via Bluetooth
     */
    public void Reset(){
        if (btSocket != null) {
            try {
                  btSocket.getOutputStream().write("R".toString().getBytes()); // send data
to Bluetooth
            } catch (IOException e) {
                msg("Error from Reset" );
            }
        }
    }


    /*
     * this method will direct send "O" to Arduino via Bluetooth
     */
    public void TurnOff(){
        if (btSocket != null) {
            try {
                 btSocket.getOutputStream().write("O".toString().getBytes()); // send datat
to Bluetooth
            } catch (IOException e) {
                msg("Error from TurnOff method" );
            }
        }
    }

    public void Time(){
        if (btSocket != null) {
            try {
                String data = "T" + getCurrentTime();
                btSocket.getOutputStream().write(data.toString().getBytes());
            } catch (IOException e) {
                msg("Error from Time method" );
            }
        }
    }



    /**
        *  If Disconnect button is pressed, close the socket and stop the thread that
constantly checks for incoming Bluetooth data
     */
    private void Disconnect() {
        if (btSocket != null) //If the btSocket is busy
        {
            try {
                btSocket.close(); //close connection
                stopThread = true;  //stops the input Bluetooth thread
                socket.disconnect();
            } catch (IOException e) {
                msg("Error");
            }
        }
        finish(); //return to the first layout, that is, the DeviceList

    }

    // fast way to call Toast
    private void msg(String s) {
        Toast.makeText(getApplicationContext(), s, Toast.LENGTH_LONG).show();
    }

    /**
     * If About button is pressed, open a new intent, that is, a new page
     */
    public void about(View v) {
        if (v.getId() == R.id.chatRoomButton) { // check whether chatRoomButton is pressed
              Intent i = new Intent(this, AboutActivity.class); // switch to AboutActivity
class
            startActivity(i);
        }
    }

    public void switch_to_chatroom(View v){
        Intent i = new Intent(this, chatroom.class);
        startActivity(i);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_led_control, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //
    //

    /**
     * This function has been copied and modified from
     * https://www.allaboutcircuits.com/projects/control-an-arduino-using-your-phone/
     * <p>
      * This function runs in the background to listen for incoming Bluetooth data and will
change
     * the text
     */
    void beginListenForData() {
        final Handler handler = new Handler();
        stopThread = false;
        buffer = new byte[1024];
        Thread thread = new Thread(new Runnable() {
            public void run() {
                while (!Thread.currentThread().isInterrupted() && !stopThread) {
                    try {
                        int byteCount = btSocket.getInputStream().available();
                        if (byteCount > 0) {
                            byte[] rawBytes = new byte[byteCount];
                            int numBytes = btSocket.getInputStream().read(rawBytes);
                            final String string = new String(rawBytes);
                            handler.post(new Runnable() {
                                public void run() {
                                    int length = string.length();
                                    if (string.charAt(0) == '1') {
                                        Temperature.setText(string.substring(1, length));
                                            socket.emit("tempToServer", string.substring(1,
length));
                                          // temperature will start with '1' in string, and
this data will
                                        //send to Web Server as tempToServer event
                                    } else if(string.charAt(0) == '2') {
                                        Photosensor.setText(string.substring(1, length));
                                           socket.emit("lightToServer", string.substring(1,
length));
                                           //photosensor will start with '2' in string, and
this data will
                                        //send to WebServer as lightToServer event
                                    } else if (string.charAt(0) == '3') {

ultrasonicSensor.setText(string.substring(1,length));
                                        //ultrasonicSensor will start with '3' in string
                                        socket.emit("distanceToServer", string.substring(1,
length));
                                                    //socket IO will send to Web Server as
distanceToServer event
                                    } else if(string.charAt(0) == '4'){
                                        Humidity.setText(string.substring(1,length));
                                        //humidity is the last one
                                        socket.emit("humidityToServer", string.substring(1,
length));
                                                //socket IO will send data to WebServer as
humidityToServer event
                                    } else if(string.charAt(0) == '9') {
                                        //do nothing
                                    }
                                }
                            });

                        }
                    } catch (IOException ex) {
                        stopThread = true;
                        socket.disconnect();
                    }
                }
            }
        });

        thread.start();
    }

    /**
     * ATTENTION: This was auto-generated to implement the App Indexing API.
     * See https://g.co/AppIndexing/AndroidStudio for more information.
     */
    public Action getIndexApiAction() {
        Thing object = new Thing.Builder()
                .setName("ledControl Page") // TODO: Define a title for the content shown.
                // TODO: Make sure this auto-generated URL is correct.
                .setUrl(Uri.parse("http://[ENTER-YOUR-URL-HERE]"))
                .build();
        return new Action.Builder(Action.TYPE_VIEW)
                .setObject(object)
                .setActionStatus(Action.STATUS_TYPE_COMPLETED)
                .build();
    }

    @Override
    public void onStart() {
        super.onStart();

        // ATTENTION: This was auto-generated to implement the App Indexing API.
        // See https://g.co/AppIndexing/AndroidStudio for more information.
        client.connect();
        AppIndex.AppIndexApi.start(client, getIndexApiAction());
    }

    @Override
    public void onStop() {
        super.onStop();

        // ATTENTION: This was auto-generated to implement the App Indexing API.
        // See https://g.co/AppIndexing/AndroidStudio for more information.
        AppIndex.AppIndexApi.end(client, getIndexApiAction());
```

```
        client.disconnect();
    }


    public class ConnectBT extends AsyncTask<Void, Void, Void>  // UI thread
    {
        private boolean ConnectSuccess = true; //if it's here, it's almost connected

        @Override
        protected void onPreExecute() {
                 progress = ProgressDialog.show(ledControl.this, "Connecting...", "Please
wait!!!");  //show a progress dialog
        }

        @Override
            protected Void doInBackground(Void... devices) //while the progress dialog is
shown, the connection is done in background
        {
            try {
                if (btSocket == null || !isBtConnected) {
                        myBluetooth = BluetoothAdapter.getDefaultAdapter();//get the mobile
bluetooth device
                                                           BluetoothDevice dispositivo =
myBluetooth.getRemoteDevice(address);//connects to the device's address and checks if it's
available
                                                                               btSocket =
dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);//create a RFCOMM (SPP)
connection
                    BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                    btSocket.connect();//start connection
                }
            } catch (IOException e) {
                  ConnectSuccess = false;//if the try failed, you can check the exception
here
            }
            return null;
        }

        @Override
         protected void onPostExecute(Void result) //after the doInBackground, it checks if
everything went fine
        {
            super.onPostExecute(result);

            if (!ConnectSuccess) {
                msg("Connection Failed. Is it a SPP Bluetooth? Try again.");
                finish();
            } else {
                msg("Connected.");
                isBtConnected = true;
                beginListenForData();
            }
            progress.dismiss();
        }
    }
}
```

Chatroom class

```
package com.led_on_off.led;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.util.Log;



import com.github.nkzawa.emitter.Emitter;
import com.github.nkzawa.socketio.client.IO;
import com.github.nkzawa.socketio.client.Socket;
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.appindexing.Thing;
import com.google.android.gms.common.api.GoogleApiClient;

import java.lang.reflect.Array;
import java.net.URISyntaxException;
import java.util.ArrayList;


import org.json.JSONException;
import org.json.JSONObject;
/**
 * Created by linzi on 2017-03-27.
 */

public class chatroom extends ActionBarActivity {

    /*
    Used for chat room
```

```
     */
    String UserName = "Bob"; //default username
    String value;
    private Socket socket;  //define a IOsocket to communicate with Web Server
    EditText chatEditText;
    ArrayList<String> messages = new ArrayList<String>();
    ArrayAdapter<String> messageAdapter;

    /**
     * ATTENTION: This was auto-generated to implement the App Indexing API.
     * See https://g.co/AppIndexing/AndroidStudio for more information.
     */

    private GoogleApiClient client;

    {
        try {
            socket = IO.socket("http://38.88.74.87:9000");
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
    }




    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.chatroom);
        socket.connect(); // connect socket
        socket.on("chat", chatwithserver); // IO socket will listen to chat event

        client = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();

        /*
         *define message list and display them on screen
         */
           ListView  chatListView  =  (ListView)findViewById(R.id.charListView);  //get  text
message by ID
          messageAdapter  =  new  ArrayAdapter<String>(this,                          //this
arrayAdapter linked to the ArrayList called "messages"
                android.R.layout.simple_list_item_1,
                messages);

        chatListView.setAdapter(messageAdapter);

         value = getIntent().getStringExtra("string_name");
    }

    private Emitter.Listener chatwithserver = new Emitter.Listener() {
        @Override

        public void call(final Object... args) {
            chatroom.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    String message = (String) args[0];
                    messages.add(message);
                    messageAdapter.notifyDataSetChanged();

                }
            });
        }
    };

    public void login(View v){
        Intent i = new Intent(this, login.class);
        startActivity(i);
    }
    /*
     *This method will send data to Web Server and update the message on the screen
     *@param v
     *
     */
    public void sendChat(View v){

          chatEditText  =  (EditText)findViewById(R.id.message);         //get  EditText  type
message by finding ID
         String messageContent = chatEditText.getText().toString(); // Convert it to String
type
        if(value != null){
            UserName = value;
        }
         socket.emit("message",UserName+": " + messageContent);      // send message to Web
Server
         messages.add(UserName+": " + messageContent);               // add this message to
arrayList "messages"
           messageAdapter.notifyDataSetChanged();                             //  update  the
ArrayAdapter

    }

}
```

Login class

```
package com.led_on_off.led;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.util.Log;

/**
 * Created by linzi on 2017-03-27.
 */



public class login extends ActionBarActivity {

    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);
    }

    String getusername = "Bob"; //default username

    /*
     *This method will receive the new EditTect as username when Signup button is pressed.
      * After receive the new user name we will transfer this string back to chatroom class
and
     * jump back to chatroom class
     */
    public void setUserName(View v){
         EditText username = (EditText)findViewById(R.id.username); // link username to by
find the id

         getusername = username.getText().toString();
        Log.i("testing               :",getusername);
        Toast.makeText(this, "username: " + getusername, Toast.LENGTH_SHORT).show();
        //
        Intent intent = new Intent(this, chatroom.class);
        intent.putExtra("string_name", getusername);
        startActivity(intent);

    }
}
```

DeviceList class

```
package com.led_on_off.led;
```

```
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;


public class DeviceList extends ActionBarActivity
{
    //widgets
    Button btnPaired;
    ListView devicelist;
    //Bluetooth
    private BluetoothAdapter myBluetooth = null;
    private Set<BluetoothDevice> pairedDevices;
    public static String EXTRA_ADDRESS = "device_address";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device_list);

        //Calling widgets
        btnPaired = (Button)findViewById(R.id.button);
        devicelist = (ListView)findViewById(R.id.listView);

        //if the device has bluetooth
        myBluetooth = BluetoothAdapter.getDefaultAdapter();

        if(myBluetooth == null)
        {
            //Show a mensag. that the device has no bluetooth adapter
                 Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available",
Toast.LENGTH_LONG).show();
```

```
            //finish apk
            finish();
        }
        else if(!myBluetooth.isEnabled())
        {
                //Ask to the user turn the bluetooth on
                Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(turnBTon,1);
        }

        btnPaired.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v)
            {
                pairedDevicesList();
            }
        });

    }

    private void pairedDevicesList()
    {
        pairedDevices = myBluetooth.getBondedDevices();
        ArrayList list = new ArrayList();

        if (pairedDevices.size()>0)
        {
            for(BluetoothDevice bt : pairedDevices)
            {
                  list.add(bt.getName() + "\n" + bt.getAddress()); //Get the device's name
and the address
            }
        }
        else
        {
             Toast.makeText(getApplicationContext(), "No Paired Bluetooth Devices Found.",
Toast.LENGTH_LONG).show();
        }

                                        final      ArrayAdapter      adapter      =      new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
        devicelist.setAdapter(adapter);
          devicelist.setOnItemClickListener(myListClickListener); //Method called when the
device from the list is clicked

    }

               private    AdapterView.OnItemClickListener    myListClickListener    =    new
```

```
AdapterView.OnItemClickListener()
    {
        public void onItemClick (AdapterView<?> av, View v, int arg2, long arg3)
        {
            // Get the device MAC address, the last 17 chars in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Make an intent to start next activity.
            Intent i = new Intent(DeviceList.this, ledControl.class);

            //Change the activity.
                i.putExtra(EXTRA_ADDRESS, address); //this will be received at ledControl
(class) Activity
            startActivity(i);
        }
    };


    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_device_list, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Appendix H - GitHub

Hanyu Wu: I worked with Yu Chen for most of the time and did most adjustments together in combined effort. Most commits are made by him thus leaving me a minor amount of pushes.

Appendix I – Other code

Node.js server code

```
//import libraries, including express and socket.io framework
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);
var temp, humidity, light, distance;

//dealing with the http request and sent the index.html to clients
app.get('/',function(req,res){
    res.sendFile(__dirname+'/index.html');
})

//make the server listen to port 9000
http.listen(9000,function(){
    console.log('server listening on port 9000');
})

//once the web socket connection is on
io.on('connection',function(socket){
    //log in the console that there is a client connected
    console.log('one user connected '+socket.id);

    //log in the console that there is a client disconnectted
    socket.on('disconnect',function(){
        console.log('one user disconnected '+socket.id);
    })

    //deal with 'startTime' event
              socket.on('startTime', function(data) {
                     setInterval(function(){
                     io.sockets.emit("TimeAndroid", 'hi');
              }, 10000);
              });

    //deal with 'LCDtoServer' event
              socket.on('LCDtoServer', function(data) {
                     io.sockets.emit('LCDAndroid', data);
              });

        //request from html, send info to android
        socket.on('led', function(data) {
               console.log("led on/off");
               io.sockets.emit('ledToAndroid', 'led on/off');
        });

        //request from html, send info to android
        socket.on('piezo', function(data) {
```

```
        console.log("piezo on/off");
        io.sockets.emit('piezoToAndroid', 'led on/off');
});

//request from html, send info to android
socket.on('motor', function(data) {
        console.log("motor on/off");
        io.sockets.emit('motorToAndroid', 'motor on/off');
});

//request from html, send info to android
socket.on('Automode', function(data) {
  console.log("piezo on/off");
  io.sockets.emit('AutoModeAndroid', 'automode on/off');
});

//request from html, send info to android
socket.on('Reset', function(data) {
  console.log("piezo on/off");
  io.sockets.emit('ResetAndroid', 'automode on/off');
});

//request from html, send info to android
socket.on('turnOff', function(data) {
  console.log("piezo on/off");
  io.sockets.emit('TurnOffAndroid', 'turn on/off');
});

//request from android, send info to html
      socket.on('tempToServer', function(data) {
             temp = data;
             //console.log("Received temperature" + data);
             io.sockets.emit('temp', data);
      });

//request from android, send info to html
      socket.on('lightToServer', function(data) {
             light = data;
             //console.log("Received photosensor" + data);
             io.sockets.emit('light', data);
      });

//request from android, send info to html
      socket.on('distanceToServer', function(data) {
             distance = data;
             //console.log("Received temperature" + data);
             io.sockets.emit('distance', data);
      });
```

```
        //request from android, send info to html
               socket.on('humidityToServer', function(data) {
                       humidity = data;
                       //console.log("Received temperature" + data);
                       io.sockets.emit('humidity', data);
               });

               //request from PI
               socket.on('tempFromPi', function(data) {
                       console.log(data);
                       io.sockets.emit('tempFromServer', temp);    //send to android
               });

               socket.on('message', function(msg){
                       //io.emit('chat', msg);
                       socket.broadcast.emit('chat', msg);
               });

        socket.on('message-arduino', function(msg){
                       //io.emit('chat', msg);
                       socket.broadcast.emit('chat-arduino-android', msg);
               });
})
```