**ELEC 291 Section 20C**

Project 2 Progress Report

L2C

Team 2A and 2B (Group 2)

March 31, 2016

*Student name Student number*

Andy Ruan 36863141

Kevin Wong 32105132

Clarence Su 36387132

Timothy Leung 14189147

Kevin Qiu 14188149

Derek Tam 24630148

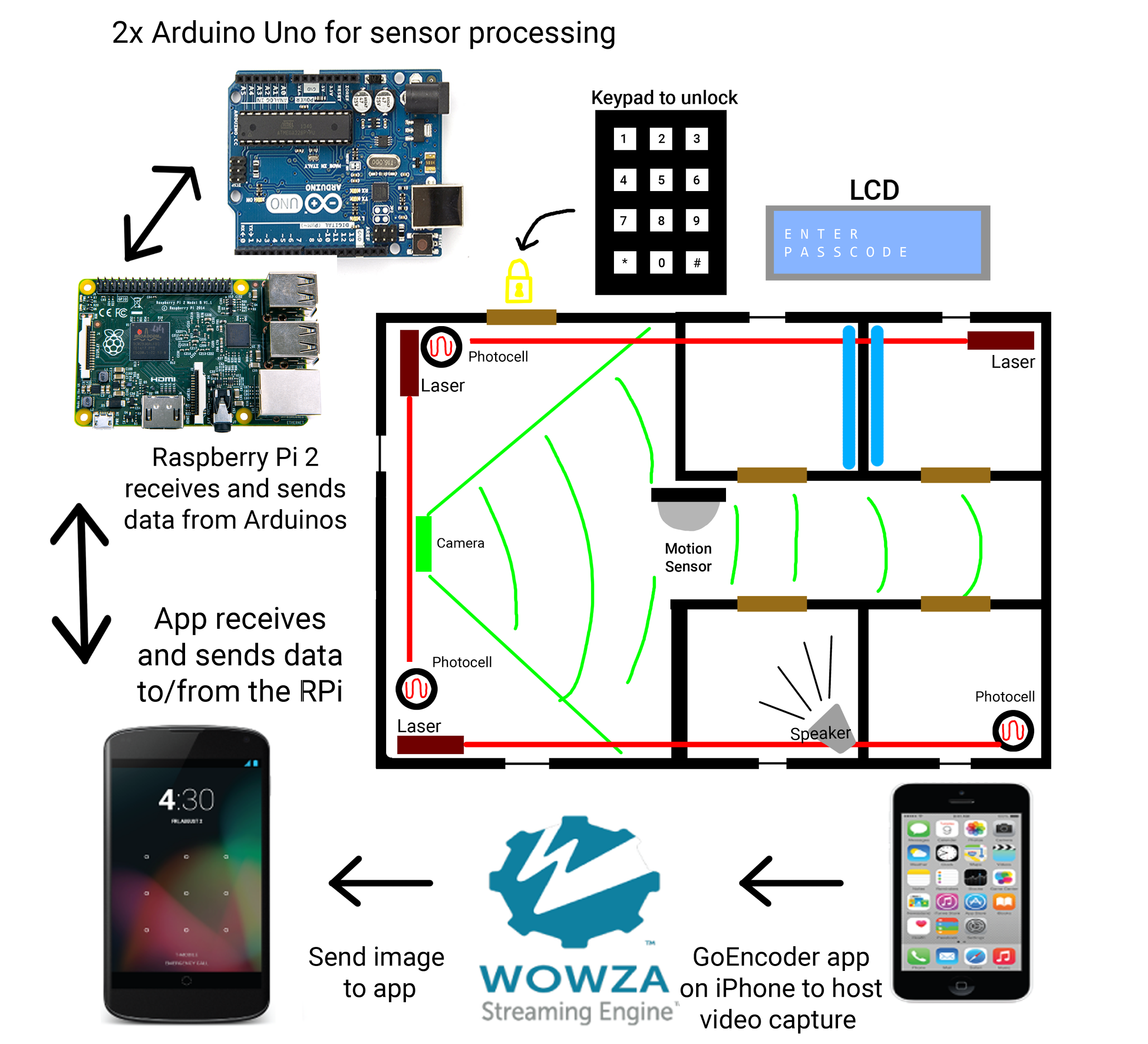
**Introduction**

This report outlines the progress that Group 2 has made in Project 2 of ELEC 291-20C from the 18th of March to the 31st and will be organized as follows: First, a concise description of the project will be given in the “Project Title” section. System diagrams will follow, both for software and hardware based on the current implementation. The report will go on to explain the project description and objectives more thoroughly. Proceeding that, we will show at which stage of our implementation we are in the process of completing. Individual contributions will be listed in reference to the assigned tasks in the project proposal. Finally, documentation such as images and code will be presented at the end of this report. Note that all work, including what we consider to be completed work, is subject to change before the due date as implementations may change depending on design decisions in the future.

**Project Title**

Smart Home System

**System Diagram**



System diagram updated to current implementations

**Project Description and Objectives**

This project is a smart home system that allows people to use their mobile device to control elements of their own household through an Android application while they are away from home. The purpose of choosing this project is because our team has a desire to work with mobile technologies and smart home features such as lighting control, alarms and cameras.

The project will consist of the following parts done in parallel:

The first part will involve constructing the model house and circuitry to fit all the electrical components such as cameras, LCD displays, photocells, and motion detectors. We will be planning out how everything is wired through our model house.

The second part will be installing cameras to the household that allow users to monitor their house in case of intruders.

The third part will be creating a mobile application to display data for each security feature. For instance, a video stream will be implemented to display the data coming from the cameras.

The fourth part will be a password requirement to enter the house. To implement this functionality, there will be a LCD Display and a Keypad to enter user’s passcode.

We have decided to scrap the voice recognition feature which we mentioned in our proposal in favour of focusing on making our implementations closer to the standards used in real world technologies today such as utilizing encryption and proper client-server architecture.

**Implementation**

As of the 31st of March, this project can be separated into four main parts: hardware, security, server communications, and Android application interfacing. The combination of these parts forms a complete smart home system that includes security features as well as lighting features in which the home owner has the ability to control multiple aspects of from distance through their phone.

**Hardware**

For our hardware, we have decided to implement several security features to demonstrate a large range of methods that can be used for a real security system.

First, we have implemented a keypad entry system that will allow entry into the household with a correct passcode. An LCD module accompanies the keypad at the front of the house and displays the status of the passcode entry.

Next, we have installed a laser system with the use of three red laser modules and three photocell resistors. By breaking the connection between laser and photocell, our system will be able to detect someone entering through any of the windows in the house. The change in the photocell resistance from a difference in light exposure will signal that the laser has been broken and will set the system status to be triggered.

Third, we have added a PIR motion sensor to detect movement within the house. The motion sensor will simply send a high signal when motion is detected. A flaw in this system is that one motion sensor will not provide enough coverage for the entire house as the walls prohibit the sensor from tracking each room.

Finally, interior lighting control was added to each room in the house. With lighting control in our Android application, we can turn our lights on and off as well as set a timer to turn them off. For lighting the house, we used 12V warm white LED strips at the top of the walls of each room. These strips are connected into a relay module and currently powered using the lab’s DC power supply.

To control every sensor and the light relay in our household system while also maintaining the ability to read keypad inputs and write to the LCD, we decided to implement I2C communications between two Arduinos. This prevents the algorithms that compute the correct pass code from introducing latency to the sensor monitoring. Our slave Arduino is connected to all sensors and the light relay while the master is connected the keypad and LCD. The reasoning behind this is due to the nature of the slave always listening to the master; it is necessary for the passcode Arduino to communicate to the monitoring Arduino whether a passcode has been entered properly or not without the monitoring Arduino polling for a response too frequently. This setup also increases our flexibility of pin layouts.

**Server**

An important feature for us was to allow for multiple devices to control the lights simultaneously; this meant we needed to be able to support multiple client connections to the server. We decided to use a Raspberry Pi as it has more computing power than an Arduino to handle these communications. We decided against a WiFi module for the Arduino as it actually cost more than the Raspberry Pi.

We used Java to write our server code because Android application also uses Java so it would be easy to communicate between them. The server utilizes threads to handle the many operations that are occurring. There are 2 main threads that are constantly running as well as additional client threads as needed when they connect.

The first main thread is a handler thread that listens for socket connections from the Android application. Once a client connects, the handler thread creates a new client thread that is used to communicate with the client. This thread has two objectives. The first one is to listen for commands from the application. Once a command is received, the thread adds it to a global command queue. Secondly, it will listen for an updated status string in the server. If there is a newly updated status string, the client thread will send in to the Android application. Once the client closes the connection, this thread is killed.

The second thread is the Arduino thread that handles the Serial communication between the Raspberry Pi and the Arduino. It has two objectives. When a status string is received by the server from the Arduino, it will update the global status string on the server. The client threads running will sense the update and send the string to the Android application. The second objective is to send any command currently in command queue to the Arduino. Utilizing a queue allows us to stack up commands from all the different client threads so that they can be sent to the Arduino.

All of the wireless connection between the Application and the Server is encrypted. The description of this is below in our security section.

**Android Application**

The Android application serves as the main way for the end-user to interface with the home security system. Through the app, the user can control the security features and lighting in individual rooms, view the home status and indoor camera, and receive remote notifications from the system. These operations are done by communicating over Wi-fi with the home’s RPi server, which will relay the corresponding commands to the Arduino. The RPi server also relays status information from the Arduino to the app, as mentioned previously in the report. Every time the app is re-opened, the user is required to enter a login PIN, ensuring that others cannot access and misuse the app features. There are four main UI windows available after entering the app: Overview, Security, Lights, and Camera.

The Overview windows provides a quick display of the current status of the security modules and lighting system, as well as a floor plan of the house itself. Status information is constantly received from the RPi server. The information is used to display text for the master security system, front door, motion sensor, lasers, and alarm, indicating if they are armed and/or if they have been triggered. There is also text for each individual room to indicate if the lighting is turned on or off.

On the Security window, the user can toggle each security module individually, or set the master control for all modules. Similarly, the lighting system can be controlled through the Lights window. The lights in each room can be toggled on/off or set on a timer, which will turn off the lights after a chosen amount of time.

The Camera window allows the Android application to view a stream of the indoor camera. To have this feature, our team needed to host the live stream from a phone camera on Wowza Streaming Engine. Using Wowza Streaming Engine’s specifications, a phone needed to install an application called GoCoder, which allows users to encode their live streaming data so that Wowza Streaming Engine can read it. Once that is complete, our team needed to implement the ability to view the stream on the Android application. Due to VideoView’s incompatibility with the RTMP protocol (a protocol to stream audio and video files over the Internet), we needed to implement the VideoView using an external API. For this task, we chose the Vitamio API which allows us to stream using RTMP protocol. Wowza Streaming Engine will provide a RTMP URL that will allow users to access the video stream. Our team created a Vitamio VideoView object with the RTMP URL given from Wowza Streaming Engine and started the video using the playback function for videos.

Another feature of the Android application is the ability to receive push notifications even when disconnected from the home’s RPi server. This allows the user to get important messages from the home security system, such as when the alarm is triggered while away from home. This feature was implemented using the Google Cloud Messaging (GCM) server, which is responsible for receiving messages from the RPi server and sending out the push notification to the Android device.

To use push notifications, the app must first connect to and register with a remote app server. In our app, this process is done with the RPi server upon enabling ‘Push notifications’ on the Settings page. Using the GCM API keys, the app receives a token from the GCM server unique to that device, allowing the server to select specific users to notify. This token is sent and stored in the RPi server using a ‘register’ command. After registration is complete, the device does not need to be connected to the RPi server anymore.

To notify users of an event, the RPi server calls sendPushNotification(), which sends a HTTP POST request to the GCM server containing the message and device tokens in JSON format. Upon receiving the POST request, the GCM server will send out the push notification to the selected devices. Our RPi server currently sends the notification to all registered devices since we assume that there is only one model house, however this can be easily changed.

**Security - Monitoring**

As explained above, most of the monitoring of sensors will be done on a single Arduino. Since both the hall sensor and PIR motion sensor return digital signals, we can attach these sensors to interrupt pins ensuring that a change in signal (and thus an intruder) can be detected. The state of the hall sensor is directly related to the state of the door so no further logic is needed past the ISR. In the case of the PIR motion sensor however, we want to avoid detecting false movements due to things like wind moving curtains around. This is accomplished through an implementation similar to debouncing a button; while the ISR will be constantly updating status from the PIR pin, it must return HIGH for a certain period of time before we trigger any flags.

For the laser trip-wire, we could not put the photoresistors on the interrupt pins because both pins are allocated already and due to the fact that the photoresistors return analog values instead of digital. Instead, these values will be read in the main loop, as we assume that it is physically impossible to cut the while loop and get into the house faster than the Arduino can loop back. To determine if a laser has been cut, we have set a threshold value for the photocells where if the value of the cell lowers past this point, then we assume the laser is cut. These statuses and whether that individual system has been armed at the time determine whether the alarm will be triggered or not.

When a change of status occurs, whether in systems armed, triggered, or lights, the Arduino will build and send a status string containing this information out through the serial port for further processing by the Raspberry Pi(RPi) server. Similarly, the RPi will be sending commands, generated by the Android application, through its serial port back to the Arduino in order to control the system.

**Security - Encryption**

Passwords are required when entering the house, connecting to the RPi server, or logging into the Android application. We have to make sure that the user’s passwords are secure in our implementation, so hash functions are implemented for each password. With the hash function, the hash value of the password will be stored in the program or the database instead of the plaintext password itself. We implemented at least two layers of protection to make sure the hashing process is difficult to hack or reverse. Therefore, even though we have no access to the actual password, the end-user’s privacy is secured.

For the password used to enter the house, it is an eight-character password consisting of only numbers. For the first layer of protection, we swap certain characters in the password. Since swapping is a injective process, it ensures that every output comes from an unique input. After swapping, we will map the swapping output into a power function to magnify the value, so that a small change of the password will result in a large change in the hash value. After this, we added two uniquely chosen prime number to the output, which forms the final output for the hash value.

For the password used to login in the Android app, it is a string with a minimum length of 6 characters consisting of the lower-case alphabet and numbers. Similar to the house password, we swap some characters in the string for the first layer of protection. Then, we use the ASCII value of each characters in the string to map to an array that contains 36 unique random strings. The final result of the mapping will be the hash value for the password.

For the RPi server communication security, we append an unique string (an authentication key) to the information we are sending, so that the server would know if the commands being sent are actually coming from the intended users. The command we are sending is already encrypted within the implementation of the server.

**Contributions**

Andy Ruan wrote the Arduino code used to control and monitor the security system. He implemented the ability to communicate and translate messages between the two Arduinos and RPi server using both serial and I2C. Additionally, he lead the documentation of this project.

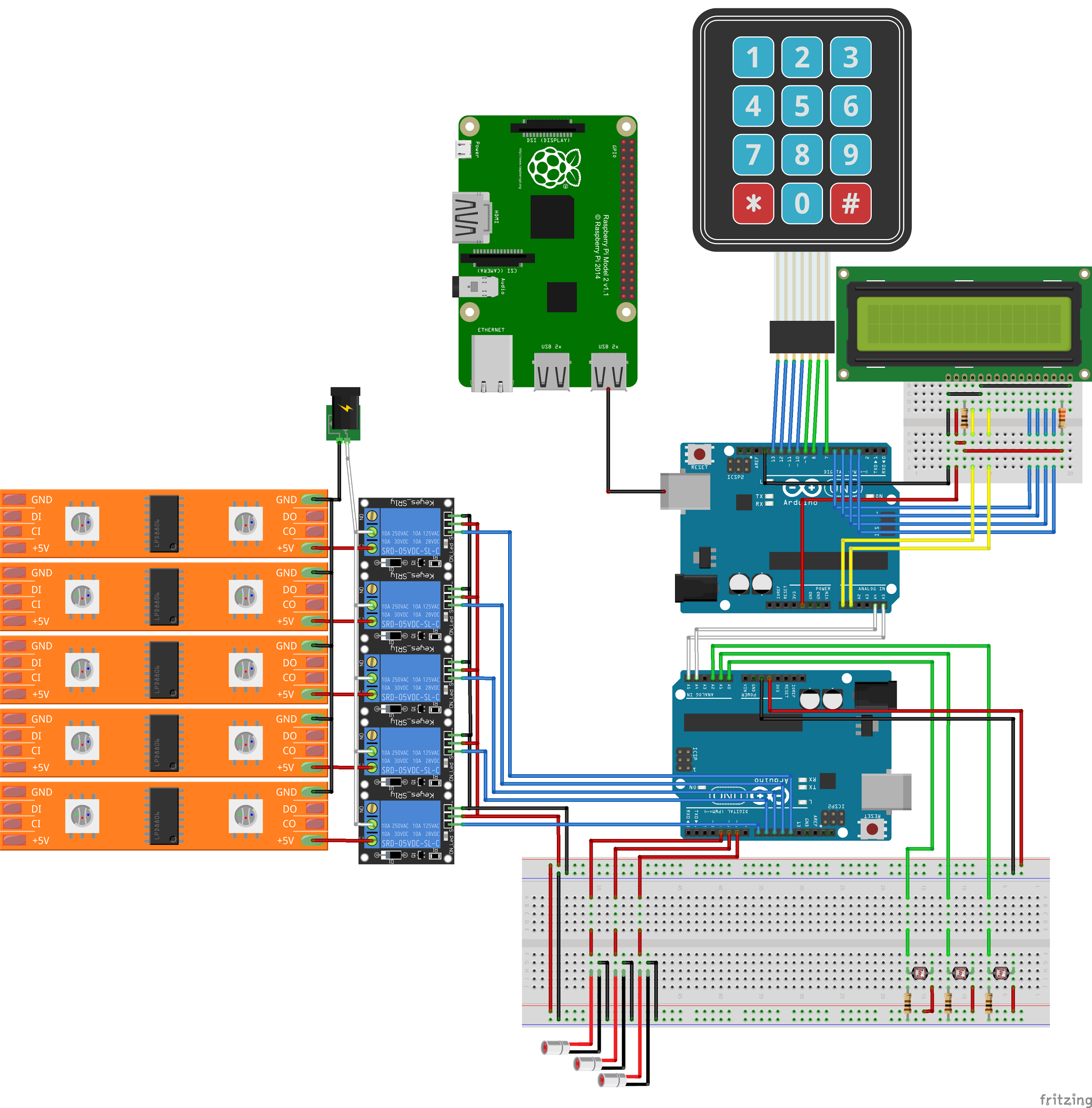
Kevin Wong wrote Android application to display camera to the application. He also designed and implemented the application icon.

Clarence Su worked on the encryption algorithm that uses the hash table data structure in order to encrypt passwords used to login the Android application.

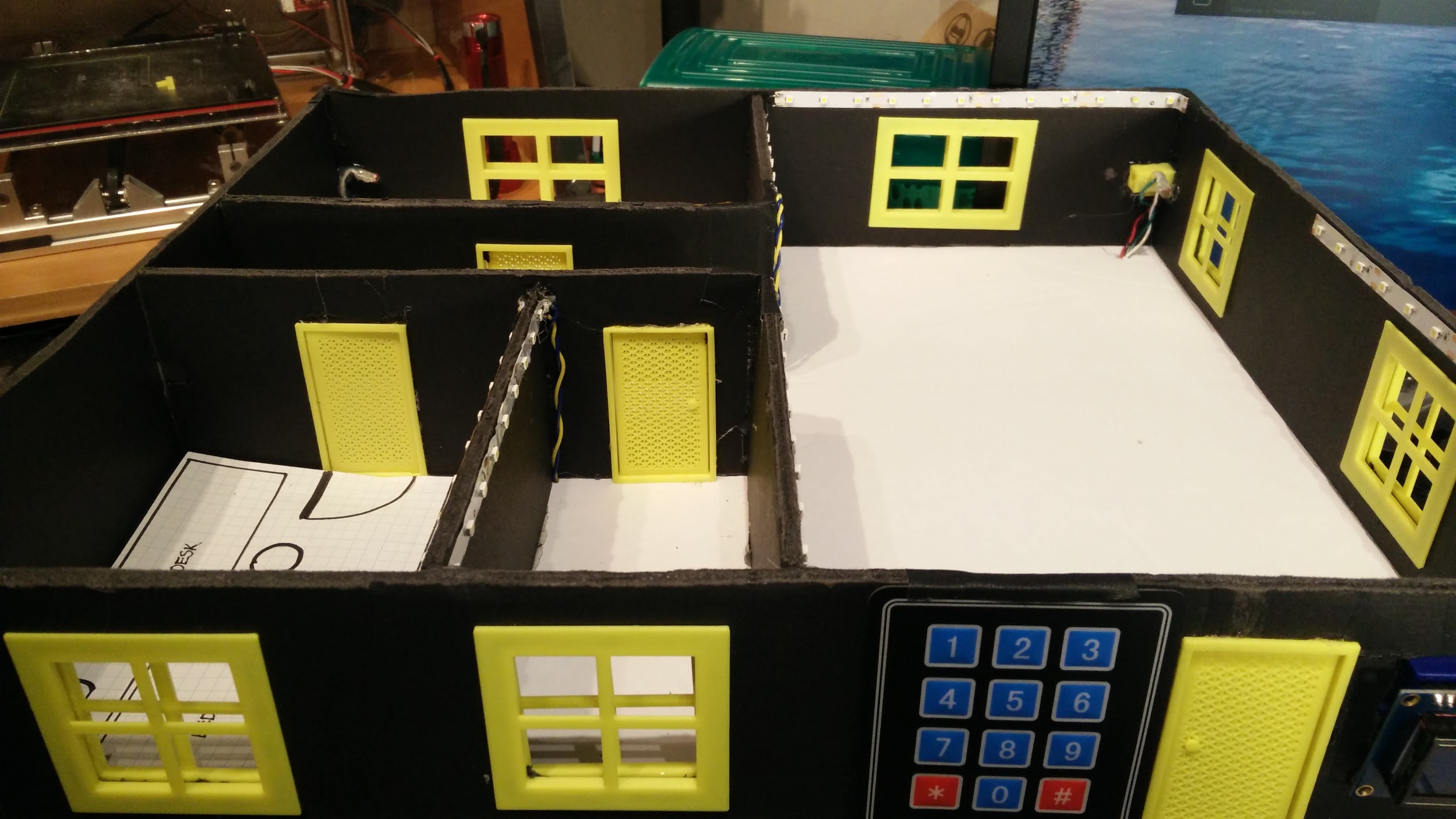
Timothy Leung worked on the server socket implementation in the Android application. He also managed the server communications between the app and the RPi. He also assisted in putting elements of the house together.

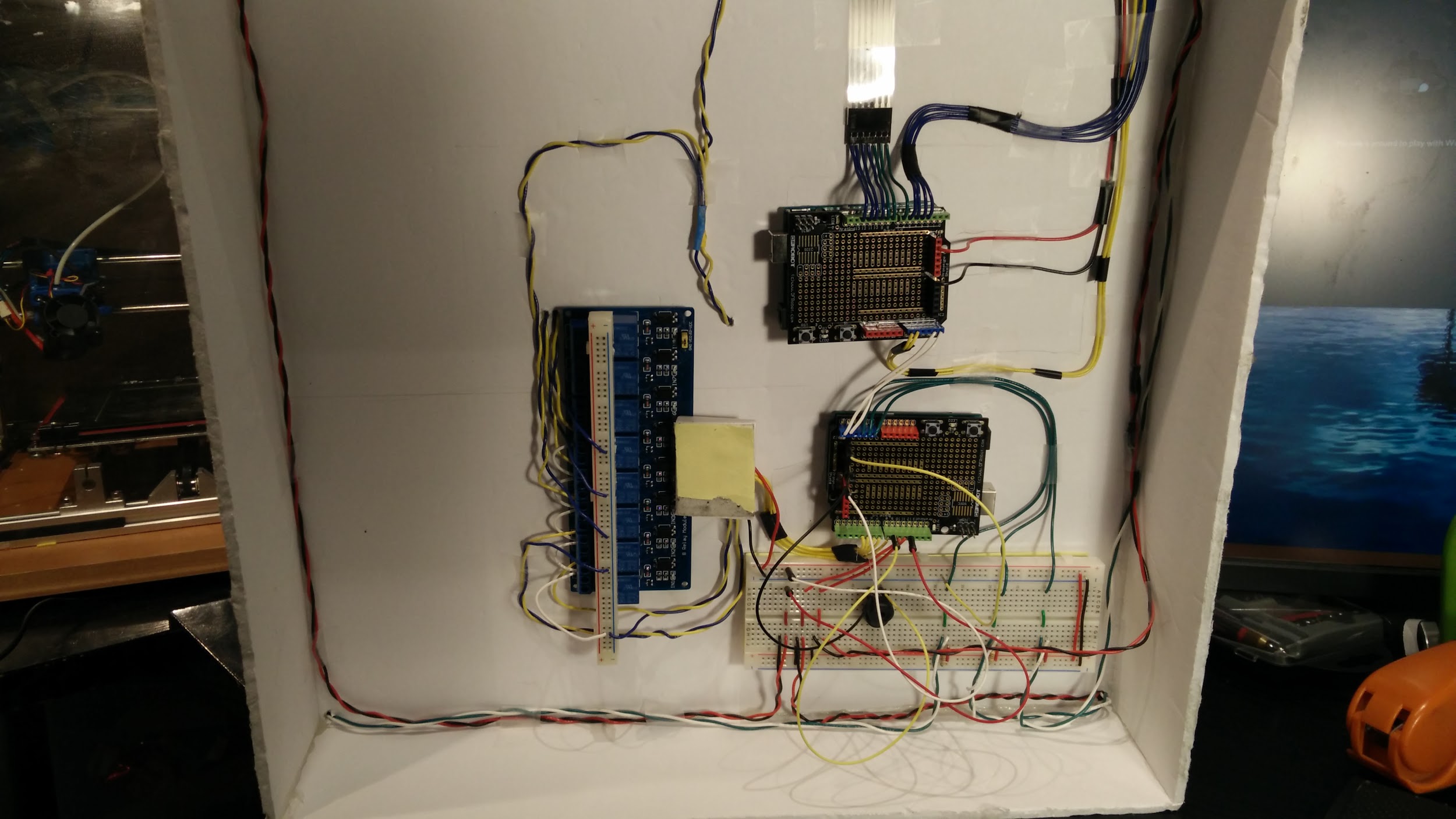
Kevin Qiu was in charge of the Android application’s UI elements, flow, and structure. He also created the design and layout for the model house as well as managed the circuitry for every component.

Derek Tam implemented the push server communication for the Android Application and RPi server. He also worked on the application UI and the model house hardware.

**Images**

Fritzing schematic

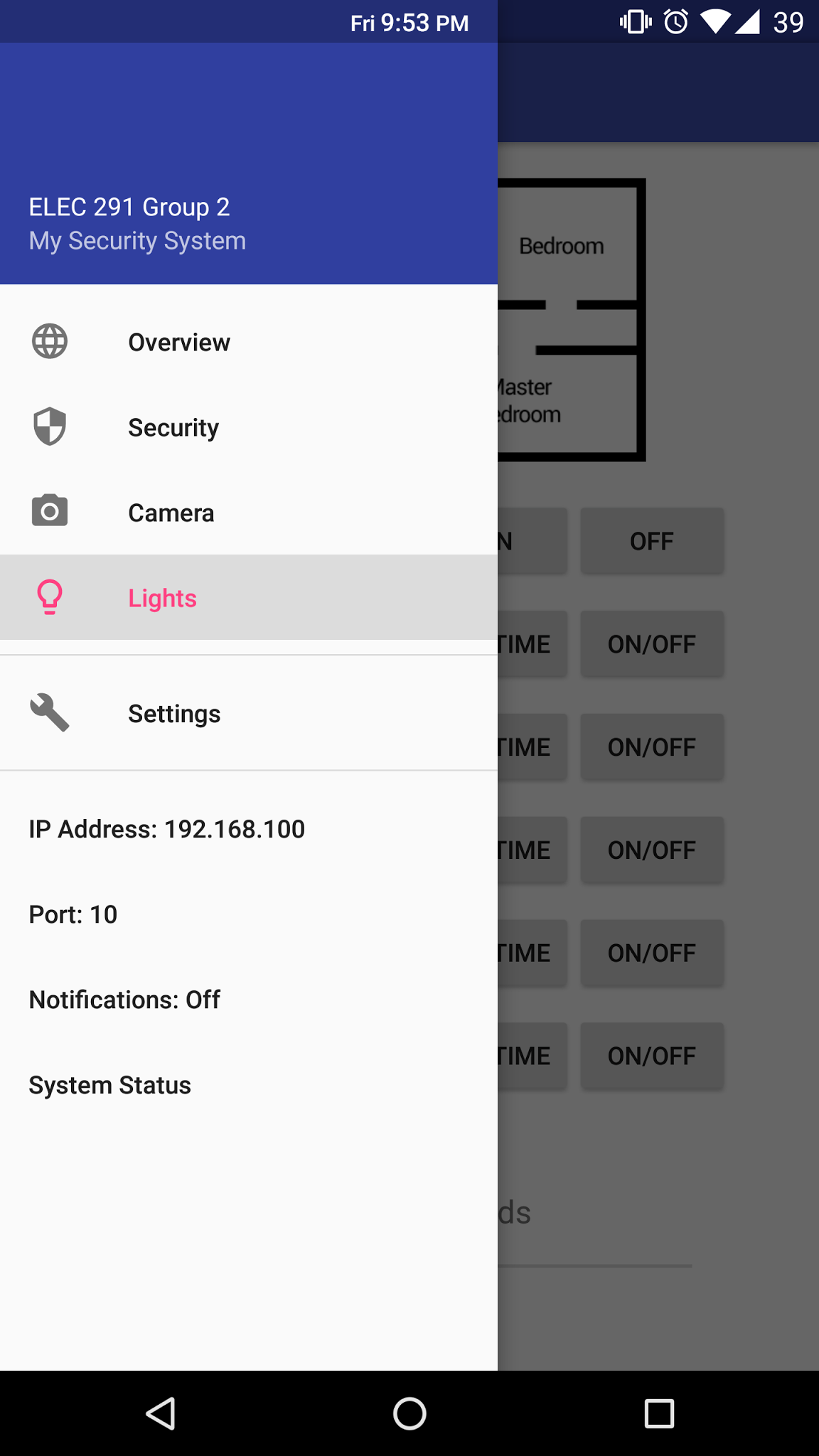
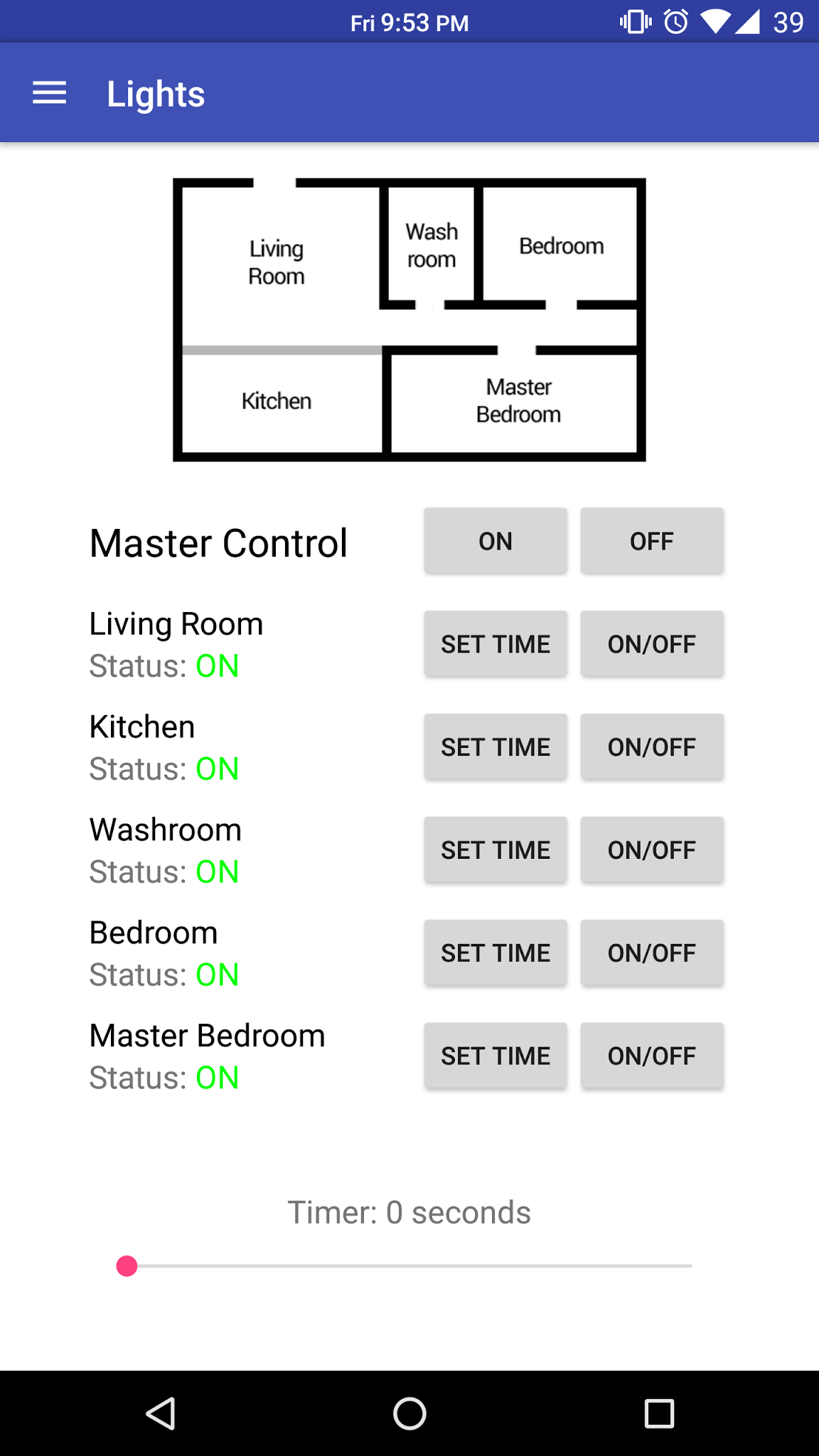
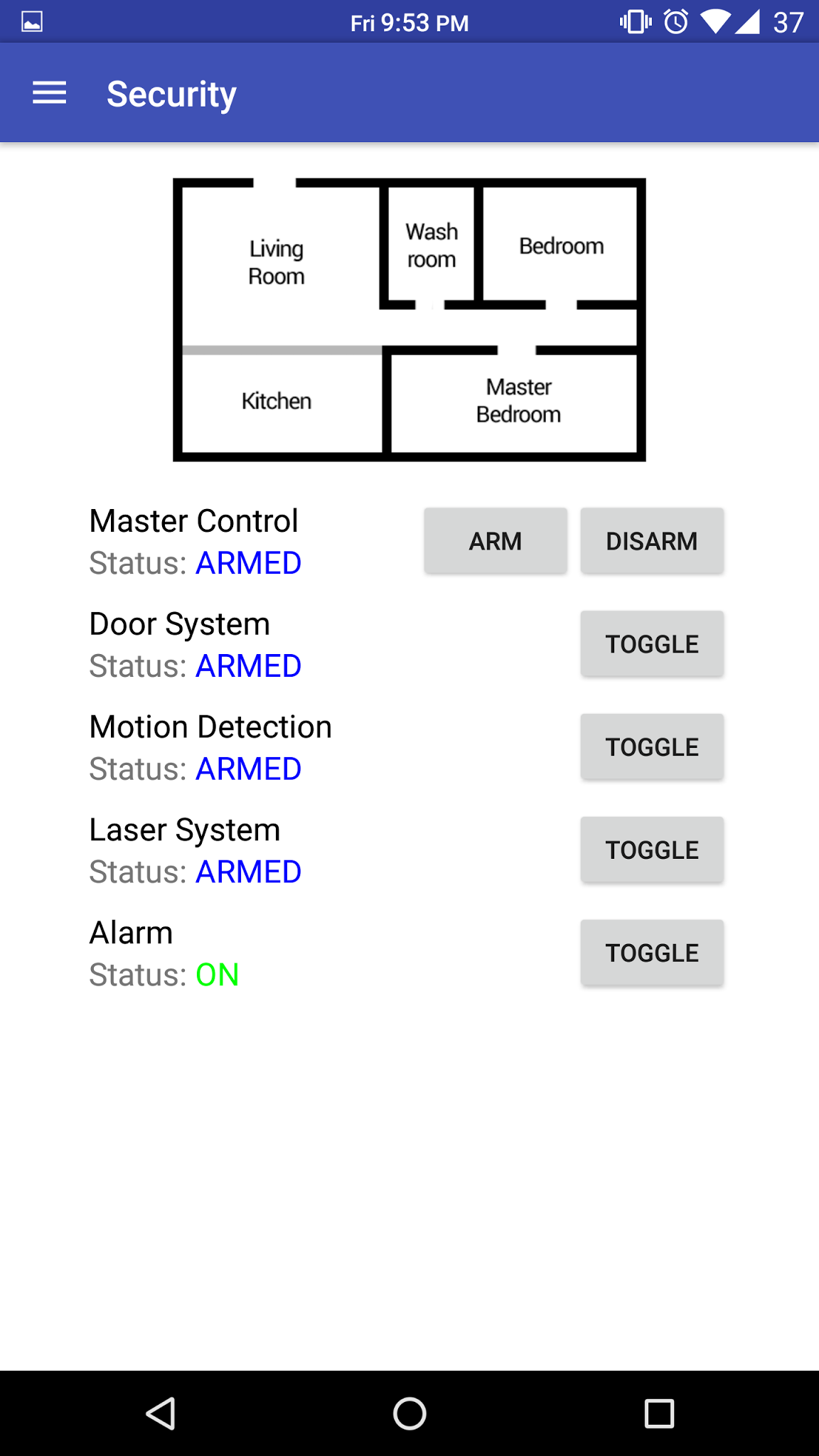


Overview of the house layout 

Bottom of the house showing the two arduinos and the relay (Some of the wiring is temporary)

5

Front of the house showing the keypad, LCD

Screenshots of the Android application

**Code**

Code/Snippets will be separated by platform.

**Arduino**

This is code used to monitor sensors and communicate with the Raspberry Pi

|  |
| --- |
| #include <Wire.h> // I2C not yet fully implemented  // sensor pins  #define DOOR\_HALL\_PIN 2 // ON ISR  #define PIR\_PIN 3 // ON ISR  #define PCELL0\_PIN A0  #define PCELL1\_PIN A1  #define PCELL2\_PIN A2  #define LASER0\_PIN 10  #define LASER1\_PIN 11  #define LASER2\_PIN 12  // light pins  #define LIGHT0\_PIN 4 // living room  #define LIGHT1\_PIN 5 // kitchen  #define LIGHT2\_PIN 6 // washroom  #define LIGHT3\_PIN 7 // bedroom  #define LIGHT4\_PIN 8 // master bedroom  // alarm pins  #define ALARM\_PIN 9  const long motionDetectTime = 1500; // max time that motion is detected before alarm triggers  const int pCellAlarmThreshold = 300; // max pcell value before alarm triggers  const long maxLightOnTime = 10000; // max time that a light will remain on  // arm/disarm alarms  int doorArmed = 0;  int pirArmed = 0;  int pCellArmed = 0;  // light status  int light0Status = 0;  int light1Status = 0;  int light2Status = 0;  int light3Status = 0;  int light4Status = 0;  // timers  unsigned long light0Timer = 0;  unsigned long light1Timer = 0;  unsigned long light2Timer = 0;  unsigned long light3Timer = 0;  unsigned long light4Timer = 0;  // alarms  int doorAlarm = 0;  int pirAlarm = 0;  int pirStatus = 0;  int pCellAlarm = 0;  int triggered = 0;  int manualAlarm = 0;  String prevStatusStr = "";  void setup() {  Serial.begin(250000);  prevStatusStr.reserve(16); // reserve space for status string  attachInterrupt(digitalPinToInterrupt(DOOR\_HALL\_PIN), doorHallISR, CHANGE);  attachInterrupt(digitalPinToInterrupt(PIR\_PIN), motionSensISR, CHANGE);  // initialize lights and lasers  pinMode(LIGHT0\_PIN, OUTPUT);  pinMode(LIGHT1\_PIN, OUTPUT);  pinMode(LIGHT2\_PIN, OUTPUT);  pinMode(LIGHT3\_PIN, OUTPUT);  pinMode(LIGHT4\_PIN, OUTPUT);  pinMode(LASER0\_PIN, OUTPUT);  pinMode(LASER1\_PIN, OUTPUT);  pinMode(LASER2\_PIN, OUTPUT);  digitalWrite(LIGHT0\_PIN, HIGH);  digitalWrite(LIGHT1\_PIN, HIGH);  digitalWrite(LIGHT2\_PIN, HIGH);  digitalWrite(LIGHT3\_PIN, HIGH);  digitalWrite(LIGHT4\_PIN, HIGH);  digitalWrite(LASER0\_PIN, HIGH);  digitalWrite(LASER1\_PIN, HIGH);  digitalWrite(LASER2\_PIN, HIGH);  // initialize and send the first status string  prevStatusStr = createStatusStr();  Serial.print(prevStatusStr);  }  void loop() {  checkAndTriggerAlarms();  // send a new status string if something changed  String statusStr = createStatusStr();  if( prevStatusStr == statusStr ) {  Serial.print(statusStr);  prevStatusStr = statusStr;  }  powerSaveLights();  }  /\*  \* Times and turns off lights which have been on for longer than maxLightOnTime.  \*/  void powerSaveLights() {  unsigned long currTime = millis();  if( light0Status && currTime - light0Timer >= maxLightOnTime ) {  digitalWrite(LIGHT0\_PIN, LOW);  light0Status = 0;  }  if( light1Status && currTime - light1Timer >= maxLightOnTime ) {  digitalWrite(LIGHT1\_PIN, LOW);  light0Status = 0;  }  if( light2Status && currTime - light2Timer >= maxLightOnTime ) {  digitalWrite(LIGHT2\_PIN, LOW);  light0Status = 0;  }  if( light3Status && currTime - light3Timer >= maxLightOnTime ) {  digitalWrite(LIGHT3\_PIN, LOW);  light0Status = 0;  }  if( light4Status && currTime - light4Timer >= maxLightOnTime ) {  digitalWrite(LIGHT4\_PIN, LOW);  light0Status = 0;  }  }  /\*  \* Creates a status string based on the current system status.  \*/  String createStatusStr() {  String s;  s.concat(triggered); // 0  s.concat(doorAlarm); // 1  s.concat(pirAlarm); // 2  s.concat(pCellAlarm); // 3  s.concat(doorArmed); // 4  s.concat(pirArmed); // 5  s.concat(pCellArmed); // 6  s.concat(manualAlarm); // 7  s.concat(light0Status); // 8  s.concat(light1Status); // 9  s.concat(light2Status); // 10  s.concat(light3Status); // 11  s.concat(light4Status); // 12  s.concat('\n');  return s;  }  /\*  \* Triggered on recieving serial from RPI.  \* Posible actions include, arming/disarming alarms, turning on/off the buzzer, and turning on/off lights.  \*/  void serialEvent() {    if( Serial.available() == 1 ) {  int x = Serial.read();  switch(x) {  case 0:  turnOffAndResetAlarms();  break;  case 1:  manualAlarmOn();  break;  case 2:  doorArmed = 1;  pirArmed = 1;  pCellArmed = 1;  break;  case 3:  doorArmed = 0;  pirArmed = 0;  pCellArmed = 0;  break;  case 4:  doorArmed = 1;  break;  case 5:  pirArmed = 1;  break;  case 6:  pCellArmed = 1;  break;  case 7:  doorArmed = 0;  break;  case 8:  pirArmed = 0;;  break;  case 9:  pCellArmed = 0;  break;  default: break;  }  }  else if( Serial.available() == 2 ) {  int light = Serial.read();  int state = Serial.read();  int set = state ? HIGH : LOW;  switch(light) {  case 0:  digitalWrite(LIGHT0\_PIN, set);  light0Status = state;  light0Timer = millis();  break;  case 1:  digitalWrite(LIGHT1\_PIN, set);  light1Status = state;  light1Timer = millis();  break;  case 2:  digitalWrite(LIGHT2\_PIN, set);  light2Status = state;  light2Timer = millis();  break;  case 3:  digitalWrite(LIGHT3\_PIN, set);  light3Status = state;  light3Timer = millis();  break;  case 4:  digitalWrite(LIGHT4\_PIN, set);  light4Status = state;  light4Timer = millis();  break;  case 5:  digitalWrite(LIGHT0\_PIN, set);  digitalWrite(LIGHT1\_PIN, set);  digitalWrite(LIGHT2\_PIN, set);  digitalWrite(LIGHT3\_PIN, set);  digitalWrite(LIGHT4\_PIN, set);  light0Timer = light1Timer = light2Timer = light3Timer = light4Timer = millis();  light0Status = light1Status = light2Status = light3Status = light4Status = state;  break;  default: break;  }  }    }  //////////////////////////////////////////////////////  // MONITORING AND ALARM CODE //  //////////////////////////////////////////////////////  void checkAndTriggerAlarms() {  checkAlarms();  triggerAlarm();  }  /\*  \* Triggers the alarm if any part of the system is both armed and triggered.  \*/  void triggerAlarm() {  if( (doorArmed & doorAlarm) | (pirArmed & pirAlarm) | (pCellArmed & pCellAlarm) ) {  tone(ALARM\_PIN, 520);  if( !triggered ) {  triggered = 1;  manualAlarm = 1;  }  }  }  /\*  \* Turns off any current alarms and resets the status of each alarm.  \*/  void turnOffAndResetAlarms() {  noTone(ALARM\_PIN);  doorAlarm = pirAlarm = pCellAlarm = triggered = manualAlarm = 0;  }  /\*  \* Manually turn on the alarm, does not count as an intruder triggered alarm  \*/  void manualAlarmOn() {  tone(ALARM\_PIN, 620);  manualAlarm = 1;  }  void checkAlarms() {  checkPIR();  checkPCell();  }  /\*  \* Triggers the PIR alarm if motion has been detected for at least motionDetectTime  \*/  void checkPIR() {  unsigned long currTime = millis();  while( pirStatus ) {  if( millis() - currTime >= motionDetectTime )  pirAlarm = 1;  }  }  /\*  \* Triggers the laser system alarm if any laser is cut from its respective photoresistor.  \*/  void checkPCell() {  if( analogRead(PCELL0\_PIN) < pCellAlarmThreshold || analogRead(PCELL1\_PIN) < pCellAlarmThreshold || analogRead(PCELL2\_PIN) < pCellAlarmThreshold)  pCellAlarm = 1;  }  void doorHallISR() {  doorAlarm = digitalRead(DOOR\_HALL\_PIN);  }  void motionSensISR() {  pirStatus = digitalRead(PIR\_PIN);  } |

This is code used for the keypad and passcode feature. The logic of the feature is completed, however the implementation of the UI is to be decided.

|  |
| --- |
| /////////////////////////////////////////////////  // KeyPad //  /////////////////////////////////////////////////  //keypad layout  char keyPad[4][3] = {  {'1', '2', '3'},  {'4', '5', '6'},  {'7', '8', '9'},  {'\*', '0', '#'}  };  //pin number  const int KeyRowP[] = {3, 4, 5, 6};  const int KeyColP[] = {7, 8, 9};  //other variables for keypad  int debounceTime = 1000;  char lastInput = ' ';  /////////////////////////////////////////////////  // LCD //  /////////////////////////////////////////////////  // LCD variables  int numArray[] = { LOW, HIGH };  int LCDDisplay = HIGH;  int LCDCursor = LOW;  int LCDCursorBlink = LOW;  String lcdOffSet=" ";  // LCD analog pins  const int LCDrs = A0;  const int LCDenable = A1;  const int LCDdata[4] = {A5, A4, A3, A2};  //LCD character  char Binary[][9] =  {  "01000001", //A 0  "01000010", //B 1  "01000011", //C 2  "01000100", //D 3  "01000101", //E 4  "01000110", //F 5  "01000111", //G 6  "01001000", //H 7  "01001001", //I 8  "01001010", //J 9  "01001011", //K 10  "01001100", //L 11  "01001101", //M 12  "01001110", //N 13  "01001111", //O 14  "01010000", //P 15  "01010001", //Q 16  "01010010", //R 17  "01010011", //S 18  "01010100", //T 19  "01010101", //U 20  "01010110", //V 21  "01010111", //W 22  "01011000", //X 23  "01011001", //Y 24  "01011010", //Z 25  "00110000", //0 26  "00110001", //1 27  "00110010", //2 28  "00110011", //3 29  "00110100", //4 30  "00110101", //5 31  "00110110", //6 32  "00110111", //7 33  "00111000", //8 34  "00111001", //9 35  "00100110", //& 36  "00101010", //\* 37  "00100001", //! 38  "00101101", //Hyphen 39  "00100011", //# 40  "00101110", //Period (.) 41  "00100000", //Space 42  "01100001", //a 43  "01100010", //b 44  "01100011", //c 45  "01100100", //d 46  "01100101", //e 47  "01100110", //f 48  "01100111", //g 49  "01101000", //h 50  "01101001", //i 51  "01101010", //j 52  "01101011", //k 53  "01101100", //l 54  "01101101", //m 55  "01101110", //n 56  "01101111", //o 57  "01110000", //p 58  "01110001", //q 59  "01110010", //r 60  "01110011", //s 61  "01110100", //t 62  "01110101", //u 63  "01110110", //v 64  "01110111", //w 65  "01111000", //x 66  "01111001", //y 67  "01111010", //z 68  "00111111" //? 69  };  //password  unsigned long password\_hash\_value = 0;  boolean password\_exist = false;  const int PASSWORD\_MAX\_LENGTH = 8;  boolean verified\_log\_in = false;  const int time\_between\_input = 2000;  int verify\_attempts = 5;  //todo:  //1.need to implement a time lock to prevent brute force break  //2.forgot password  /////////////////////////////////////////////////  // Execution //  /////////////////////////////////////////////////  void setup() {  // put your setup code here, to run once:  }  void loop() {  // put your main code here, to run repeatedly:  if(!password\_exist) initialize\_menu();  while(!verified\_log\_in){  input\_password();  }  //execute other parts    }  /////////////////////////////////////////////////  // Interface //  /////////////////////////////////////////////////  void initialize\_menu(){  String welcomeStringp1="Initial set up ";  //welcome message  lcdClear();  lcdNoAutoScroll();  lcdPrint(welcomeStringp1);  lcdPrint(lcdOffSet);  delay(2000);  }  /////////////////////////////////////////////////  // KeyPad //  /////////////////////////////////////////////////  //parameter: none  //return the key pressed by the user  char readKey() {  int startTime = millis();  int col, row;  //read the pins of Keypad  //implemented a super loop to keep keypad listening to the input  while (true) {  boolean noKey = true;  //when there is no input keep the arduino scanning for the keys  while (noKey) {  for (int index = 0; index < 3; index++) {  digitalWrite(KeyColP[index], HIGH);  }  for (int index = 0; index < 3; index++) {  digitalWrite(KeyColP[index], LOW);  for (int rowIndex = 0; rowIndex < 4; rowIndex++) {  if (digitalRead(KeyRowP[rowIndex]) == LOW) {  row = rowIndex;  col = index;  noKey = false;  break;  }  }  if (!noKey) break;  }  delay(25);  }  if (keyPad[row][col] != lastInput || (millis() - startTime >= debounceTime))break;  //prevent the arduino from reading too many inputs. It only read new input when the new input does not equal to last input or exceeds the debounce time  }  //return the result  lastInput = keyPad[row][col];  return keyPad[row][col];  }  /////////////////////////////////////////////////  // LCD //  /////////////////////////////////////////////////  // Flashes the LCDenable pin to read instructions/data  void flashEnable() {  digitalWrite(LCDenable, LOW);  delay(1);  digitalWrite(LCDenable, HIGH);  delay(1);  digitalWrite(LCDenable, LOW);  delay(1);  }  // Sets the LCDdata pins  void setLCDPins(int rs, int d3, int d2, int d1, int d0) {  digitalWrite(LCDrs, rs);  digitalWrite(LCDdata[0], d3);  digitalWrite(LCDdata[1], d2);  digitalWrite(LCDdata[2], d1);  digitalWrite(LCDdata[3], d0);  }  // Sends the data/instruction to the lcd and flashes enable  void sendCode(int rs, int d7, int d6, int d5, int d4, int d3, int d2, int d1, int d0) {  setLCDPins(rs, d7, d6, d5, d4);  flashEnable();  setLCDPins(rs, d3, d2, d1, d0);  flashEnable();  }  // Increments the cursor position to the right (moves to next line on the left if end of line)  void incrementCursor() {  sendCode(LOW, LOW, LOW, LOW, HIGH, LOW, HIGH, HIGH, LOW);  }  // Decrements the cursor position to the left (moves to previous line on the right if end of line)  void decrementCursor() {  sendCode(LOW, LOW, LOW, LOW, HIGH, LOW, HIGH, LOW, LOW);  }  // Shifts entire display to the right, cursor follows  void displayRight() {  sendCode(LOW, LOW, LOW, LOW, HIGH, HIGH, HIGH, LOW, LOW);  }  // Shifts entire screen to the left, cursor follows  void displayLeft() {  sendCode(LOW, LOW, LOW, LOW, HIGH, HIGH, LOW, LOW, LOW);  }  // Prints a single char onto the lcd  void printChar(char c) {  //Convert character to ASCII number  int value = (int) c;  //ASCII for 0-9  if (value >= 48 && value <= 57) {  value = value - 22;  getCharacter(value);  }  //ASCII for A-Z  else if (value >= 65 && value <= 90) {  value = value - 65;  getCharacter(value);  }  else if (value >= 97 && value <= 122) {  value = value - 54;  getCharacter(value);  }  //ASCII for period  else if (value == 46) {  getCharacter(41);  }  // ASCII For exclamation  else if (value == 33) {  getCharacter(38);  }  // ASCII For &  else if (value == 38) {  getCharacter(36);  }  //ASCII For \*  else if (value == 42) {  getCharacter(37);  }  //ASCII for - (Hyphen)  else if (value == 45) {  getCharacter(39);  }  //ASCII for #  else if (value == 35) {  getCharacter(40);  }  //ASCII for Space  else if (value == 32) {  getCharacter(42);  }  //ASCII for ?  else if (value == 63) {  getCharacter(69);  }  }  int convertToInt(char character) {  int aNumber = character - '0';  return aNumber;  }  void getCharacter(int value) {  String inputNumber = Binary[value];  sendCode(HIGH, numArray[convertToInt(inputNumber.charAt(0))],  numArray[convertToInt(inputNumber.charAt(1))],  numArray[convertToInt(inputNumber.charAt(2))],  numArray[convertToInt(inputNumber.charAt(3))],  numArray[convertToInt(inputNumber.charAt(4))],  numArray[convertToInt(inputNumber.charAt(5))],  numArray[convertToInt(inputNumber.charAt(6))],  numArray[convertToInt(inputNumber.charAt(7))]  );  }  // LCD FUNCTIONS - HIGH LEVEL  // Initializes the lcd to 4-bit mode at power on  void lcdInit() {  delay(50); // wait for VDD  for (int i = 0; i < 3; i++) { // repeat x3  setLCDPins(LOW, LOW, LOW, HIGH, HIGH);  flashEnable();  delay(40);  }  setLCDPins(LOW, LOW, LOW, HIGH, LOW); // 4-bits mode  flashEnable();  setLCDPins(LOW, HIGH, LOW, LOW, LOW); // display lines and font  flashEnable();  setLCDPins(LOW, HIGH, LOW, LOW, LOW); // display off  flashEnable();  setLCDPins(LOW, LOW, LOW, LOW, HIGH); // display clear  flashEnable();  setLCDPins(LOW, LOW, HIGH, HIGH, LOW); // increment cursor, no display shift  flashEnable();  }  // Clears the LCD screen and returns the cursor home  void lcdClear() {  sendCode(LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH);  delay(5);  }  // Returns the cursor home  void lcdCursorHome() {  sendCode(LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH, LOW);  delay(5);  }  // Turns ON the display  void lcdDisplay() {  LCDDisplay = HIGH;  sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);  }  // Turns OFF the display  void lcdNoDisplay() {  LCDDisplay = LOW;  sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);  }  // Displays the cursor  void lcdCursor() {  LCDCursor = HIGH;  sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);  }  // Hides the cursor  void lcdNoCursor() {  LCDCursor = LOW;  sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);  }  // Blinks the cursor  void lcdBlink() {  LCDCursorBlink = HIGH;  sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);  }  // Stops blinking the cursor  void lcdNoBlink() {  LCDCursorBlink = LOW;  sendCode(LOW, LOW, LOW, LOW, LOW, HIGH, LCDDisplay, LCDCursor, LCDCursorBlink);  }  // Turn ON scrolling when writing  void lcdAutoScroll() {  sendCode(LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH, HIGH);  }  // Turn OFF scrolling when writing  void lcdNoAutoScroll() {  sendCode(LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH, LOW);  }  // Increments cursor position k times  void lcdMoveCursorRight(int k) {  for (int i = 0; i < k; i++) {  incrementCursor();  }  }  // Decrements cursor position k times  void lcdMoveCursorLeft(int k) {  for (int i = 0; i < k; i++) {  decrementCursor();  }  }  // Scrolls the display to the right k times  void lcdScrollRight(int k) {  for (int i = 0; i < k; i++) {  displayRight();  }  }  // Scrolls the display to the left k times  void lcdScrollLeft(int k) {  for (int i = 0; i < k; i++) {  displayLeft();  }  }  // Prints a string onto the lcd  void lcdPrint(String s) {  for (int i = 0; i < s.length(); i++) {  printChar(s.charAt(i));  }  }  /////////////////////////////////////////////////  // password //  /////////////////////////////////////////////////  void initial\_password() {  //message to prompt the user for password  int input[PASSWORD\_MAX\_LENGTH] = {0};  boolean finished = false;  //keep reading until the user finish input  char currentReading;  while (!finished) {  for (int index = 0; index < PASSWORD\_MAX\_LENGTH; index++) {  currentReading = readKey();  if (currentReading == '#') {  finished = true;  break;  }  else {  input[index] = convertToInt(currentReading);  }  }  }  password\_hash\_value = hash\_function(input);  }  void change\_password() {  int input[PASSWORD\_MAX\_LENGTH] = {0};  boolean finished = false;  //keep reading until the user finish input  char currentReading;  while (!finished) {  for (int index = 0; index < PASSWORD\_MAX\_LENGTH; index++) {  currentReading = readKey();  if (currentReading == '#') {  finished = true;  break;  }  else {  input[index] = convertToInt(currentReading);  }  }  }  password\_hash\_value = hash\_function(input);  }  boolean verify\_password(int\* input) {  if (hash\_function(input) == password\_hash\_value) return true;  return false;  }  //read password from user  boolean input\_password() {  int input[PASSWORD\_MAX\_LENGTH] = {0};  boolean finished = false;  //keep reading until the user finish input  char currentReading;  while (!finished) {  for (int index = 0; index < PASSWORD\_MAX\_LENGTH; index++) {  currentReading = readKey();  if (currentReading == '#') {  finished = true;  break;  }  else {  input[index] = convertToInt(currentReading);  }  }  }  return verify\_password(input);  }  long hash\_function(int\* input) {  first\_layer\_enscription(input);  int i;  unsigned long hash\_value;  unsigned long intermediate;  for (i = 0; i < PASSWORD\_MAX\_LENGTH; i++) {  intermediate += input[i] \* pow(10, i);  }  hash\_value = intermediate + 3 \* 7823 + 2 \* 5783;  return hash\_value;  }  //first layer of encription: encript the int array  void first\_layer\_enscription(int\* input) {  swap(input, 0, 8);  swap(input, 3, 6);  swap(input, 4, 5);  }  //swap two elements in the array  void swap(int\* input, int index1, int index2) {  int intermediate = 0;  intermediate = input[index1];  input[index1] = input[index2];  input[index2] = intermediate;  } |

**Raspberry Pi**

**Server**

This is the server code that will be run on the Raspberry Pi. It handles the serial communication with the Arduino as well as the socket code that talks to the application.

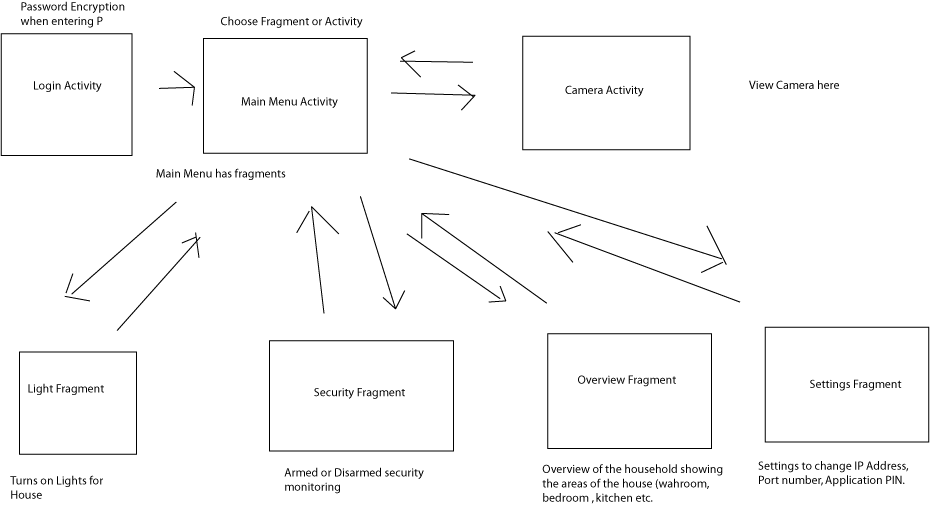
|  |
| --- |
| package elec291group2.com.SmartHomeServer;  import java.io.BufferedReader; import java.io.IOException; import java.io.InputStream; import java.io.InputStreamReader; import java.io.OutputStream; import java.io.OutputStreamWriter; import java.io.PrintWriter; import java.net.HttpURLConnection; import java.net.ServerSocket; import java.net.Socket; import java.net.URL; import java.util.HashSet; import java.util.LinkedList; import java.util.Set; import java.util.Timer; import java.util.Queue; import java.util.Scanner; import org.apache.commons.io.IOUtils; import org.json.JSONObject; import org.json.JSONException; import com.pi4j.io.serial.\*; import elec291group2.com.SmartHomeServer.Constants;   public class SmartHomeServer {  private static String AUTHENTICATION\_KEY = "1234567";  private static String hashed\_key;    private Set<String> deviceTokens;  private ServerSocket serverSocket;   // This is the status variable that will be true if a new status string  // needs to be sent   // This is the status string that the Arduino will send to the Android  // Device  String status = "";   // This is the Queue of commands that the Android device is sending  Queue<String> commandQueue = new LinkedList<String>();   /\*\*  \* @param port  \* Port Number to create server on  \*/  public SmartHomeServer(int port) throws IOException {  serverSocket = new ServerSocket(port);  deviceTokens = new HashSet<String>();  }   public void arduino() {  System.out.println("...Arduino communication thread started...");  // create an instance of the serial communications class  final Serial serial = SerialFactory.createInstance();   // create and register the serial data listener  serial.addListener(new SerialDataListener() {  @Override  public void dataReceived(SerialDataEvent event) {  // update the status with the one received on RX  System.out.println("=== NEW STATUS RECIEVED ===");  String rx = event.getData();  System.out.println("RX: " + rx);  StringBuilder rxSB = new StringBuilder();   // systemStatus  if (rx.charAt(0) == '1') {  rxSB.append('2');  sendPushNotification("ALERT: INTRUDER DETECTED");  } else if (rx.charAt(4) == '0' || rx.charAt(5) == '0' || rx.charAt(6) == '0')  rxSB.append('1');  else  rxSB.append('0');  System.out.println("systemStatus: " + rxSB.charAt(0));   // doorStatus  if (rx.charAt(1) == '1' && rx.charAt(4) == '1')  rxSB.append('3');  else if (rx.charAt(1) == '1' && rx.charAt(4) == '0')  rxSB.append('2');  else if (rx.charAt(1) == '0' && rx.charAt(4) == '1')  rxSB.append('1');  else  rxSB.append('0');  System.out.println("doorStatus: " + rxSB.charAt(1));   // motionStatus  if (rx.charAt(2) == '1' && rx.charAt(5) == '1')  rxSB.append('3');  else if (rx.charAt(2) == '1' && rx.charAt(5) == '0')  rxSB.append('2');  else if (rx.charAt(2) == '0' && rx.charAt(5) == '1')  rxSB.append('1');  else  rxSB.append('0');  System.out.println("motionStatus: " + rxSB.charAt(2));   // laser  if (rx.charAt(3) == '1' && rx.charAt(6) == '1')  rxSB.append('2');  else if (rx.charAt(6) == '1')  rxSB.append('1');  else  rxSB.append('0');  System.out.println("laserStatus: " + rxSB.charAt(3));   // manualAlarm  if (rx.charAt(7) == '1')  rxSB.append('1');  else  rxSB.append('0');  System.out.println("manualAlarm: " + rxSB.charAt(4));   // lights  rxSB.append(rx.substring(8));  System.out.println("Light 0: " + rxSB.charAt(5));  System.out.println("Light 1: " + rxSB.charAt(6));  System.out.println("Light 2: " + rxSB.charAt(7));  System.out.println("Light 3: " + rxSB.charAt(8));  System.out.println("Light 4: " + rxSB.charAt(9));   // update status  String rxPro = rxSB.toString();  System.out.println("=== statusString: " + rxPro + " ===");  status = rxPro;  }  });   try {  serial.open("/dev/ttyACM0", 38400); // open up default USB port for  // communication  while (true) {  if (!commandQueue.isEmpty()) { // New command  try {  // Retrieve and send the command through serial  String commandOut = commandQueue.poll();  System.out.println("Command sent: " + commandOut);   // process the command then send it to the Arduino  // PROCESS / DECIPHER COMMAND HERE  serial.write("commandOut");  } catch (Exception e) {  e.printStackTrace();  }  }  Thread.yield();  }  } catch (SerialPortException ex) {  System.out.println(" ==>> SERIAL SETUP FAILED : " + ex.getMessage());  return;  }  }   /\*\*  \* Run the server, listening for connections and handling them.  \*   \* @throws IOException  \* if the main server socket is broken  \*/  public void serve() throws IOException {  System.out.println("Connection handling server started.");  while (true) {  // block until a client connects  final Socket socket = serverSocket.accept();  // create a new thread to handle that client  Thread handler = new Thread(new Runnable() {  public void run() {  try {  try {  handle(socket);  } finally {  socket.close();  }  } catch (IOException ioe) {  // this exception wouldn't terminate serve(),  // since we're now on a different thread, but  // we still need to handle it  ioe.printStackTrace();  }  }  });  // start the thread  handler.start();  }  }   /\*\*  \* Handle one client connection. Returns when client disconnects.  \*   \* @param socket  \* socket where client is connected  \* @throws IOException  \* if connection encounters an error  \*/  private void handle(Socket socket) throws IOException {   System.err.println("\nA client has connected, new communication thread started.");   // get the socket's input stream, and wrap converters around it  // that convert it from a byte stream to a character stream,  // and that buffer it so that we can read a line at a time  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));   // similarly, wrap character=>bytestream converter around the  // socket output stream, and wrap a PrintWriter around that so  // that we have more convenient ways to write Java primitive  // types to it.  PrintWriter out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);  boolean authenticated = false;  int authentication\_timeout = 1000;   try {   long startTime = System.currentTimeMillis();  while ((System.currentTimeMillis() - startTime) < authentication\_timeout) // Set  // time  // out  {  if (in.ready()) // If command is retrieved  {  String s = in.readLine();  System.out.println("The key recieved is : " + s);   if(s.equals(hashed\_key))  {  authenticated = true;  System.err.println("The client has been verified.");  out.println("Verified");  out.flush();  break;  } else {  authenticated = false;  System.err.println("The client has sent an incorrect key.");  out.println("Wrong key");  out.flush();  break;  }  } else {  System.err.println("Waiting for authentication key.");  }  Thread.sleep(750);  }   String lastStatus = "";  while (authenticated == true) {  if (in.ready()) // Retrieve command from Android device, add to  // device queue  {  String s = in.readLine();  if (s.equals("exit")) {  System.err.println("A client has ended the connection.");  break;  }   System.out.println("The new command is: " + s);  commandQueue.add(s);  }   if (!lastStatus.equals(status)) // Send new status to Android  // device  {  System.out.println("The new status is : " + status);  out.println(status);  out.flush();  lastStatus = status;   }  Thread.sleep(250);  Thread.yield();  //out.println("bob");  }  } catch (Exception e) {   }   if (authenticated == false) {  System.err.println("Authentication failed, invalid key or timeout reached.");  }  System.err.println("Thread closed.");  }   /\*  \* Registers a mobile device token retrieved from the GCM server to the  \* local database.  \*/  private void registerDeviceToken(String token) {  if (token != null) {  deviceTokens.add(token);  System.out.println("Device token successfully registered!");  sendPushNotification("Push notification test");  }  }   /\*  \* Sends notification message to all tokens/devices registered to the  \* server. Uses HTTP POST protocol to send a downstream message to GCM  \* server. Maximum # of recipients per push: 1000  \*/  private void sendPushNotification(String message) {  try {  JSONObject jMessage = new JSONObject();  JSONObject jGcmData = new JSONObject();  String[] recipients = deviceTokens.toArray(new String[0]);   // Set main message 'data' field  jMessage.put("message", message);  jGcmData.put("data", jMessage);  // Set message recipients (which device tokens to push to)  jGcmData.put("registration\_ids", recipients);   // Create connection to send GCM Message Request  URL url = new URL("https://android.googleapis.com/gcm/send");  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestProperty("Content-Type", "application/json");  conn.setRequestProperty("Authorization", "key=" + Constants.API\_KEY);  conn.setRequestMethod("POST");  conn.setDoOutput(true);   // Send GCM message content.  OutputStream outputStream = conn.getOutputStream();  outputStream.write(jGcmData.toString().getBytes());   System.out.println("\nHTTP POST request sent: \n" + jGcmData.toString(4));   InputStream inputStream = conn.getInputStream();  String resp = IOUtils.toString(inputStream);  System.out.println("GCM server response:" + resp);  } catch (IOException | JSONException e) {  System.out.println("Unable to send GCM message. ");  e.printStackTrace();  }  }   public static void main(String[] args) throws IOException {  SmartHomeServer server = new SmartHomeServer(90);   hashed\_key = encryptionFunction.password\_hash(AUTHENTICATION\_KEY);  //System.out.println(hashed\_key);   // Socket communication between Server and Android device  Thread serverThread = new Thread(new Runnable() {  public void run() {  // Start the multithreaded server  try {  server.serve();  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }   // Run the server   }  });   // Serial communication between server and Arduino  Thread arduinoThread = new Thread(new Runnable() {  public void run() {  server.arduino();  }  });   // Start the threads  // arduinoThread.start();  serverThread.start();   } } |

**Android Application**

The following class is tested on the Eclipse platform. However, it does not work as expected in the back end of the Android app as there is memory limit when running in Android. Therefore, further modification is needed.

|  |
| --- |
| public class encrytionFunction {  static private String[] keys =  {"ke", "vI", "ca", "da", "la", "HA", "HS", "8\*", "-o", "8^",  "Ke", "vP", "cA", "#a", "~a", "]A", ",S", ".\*", "-O", "8?",  "kE", "v%", "cy", "dz", "lc", "HU", "H|", "8)", "Po", "8-",  "TR", "1I", "3a", "4a", "09a", "HsA"};    static String message\_id = "elec291project2";    //for password login  //length of the password has to be greater or equal to 6  static public String passWord\_hash(String input){  int password\_length = input.length();  StringBuffer to\_return = new StringBuffer();  assert(password\_length<100);  //first layer  String swap1 = swap(input, 0, password\_length-1);  String swap2 = swap(swap1, 1, password\_length-2);  String swap3 = swap(swap2, 2, password\_length-3);    //second layer  char[] to\_map = swap3.toCharArray();  for (int i = 0; i < password\_length; i++){  int ascii\_value = (int) to\_map[i];  if (ascii\_value >= 97 && ascii\_value <= 122){  to\_return.append(keys[ascii\_value-97]);  }  else if(ascii\_value >= 48 && ascii\_value <= 57){  to\_return.append(keys[ascii\_value-48+26]);  }  }    return to\_return.toString();  }    //messages have to share the same length  static public String encrypt\_message(String message){  int message\_length = message.length();  StringBuffer to\_return = new StringBuffer();  to\_return.append(message\_id);  to\_return.append(message);  return to\_return.toString();  }    static public String decrypt\_message(String message){  int message\_length = message.length();  String useful\_info = message.replace("elec291project2", "");  return useful\_info;  }    //swap the characters inside a string  static private String swap(String s, int p1, int p2){  char[] to\_swap = s.toCharArray();  char temp = to\_swap[p1];  to\_swap[p1] = to\_swap[p2];  to\_swap[p2] = temp;  return new String(to\_swap);  }  } |

Here is the flow of the Android Application. Our group discussed with Farshid about how to display the code for the Android Application. He suggested that we create a flowchart, mention any libraries used and have snippets of code since Android Applications have too many lines of code with the XML and Java code. We used two libraries in our Android Application. One is Vitamio Library for Videoview. The other is Pi4J for serial communication to Raspberry Pi. The full source code is available at : https://github.com/kevqiu/project2 if necessary.



This is the code for Login Activity:

|  |
| --- |
| package elec291group2.com.project2;  import android.app.AlertDialog;  import android.content.DialogInterface;  import android.content.Intent;  import android.content.SharedPreferences;  import android.os.Bundle;  import android.preference.PreferenceManager;  import android.support.v7.app.AppCompatActivity;  import android.support.v7.widget.Toolbar;  import android.text.InputType;  import android.text.method.PasswordTransformationMethod;  import android.view.View;  import android.view.inputmethod.InputMethodManager;  import android.widget.Button;  import android.widget.EditText;  import android.widget.LinearLayout;  import android.widget.Toast;  public class Login extends AppCompatActivity  {  SharedPreferences sharedPreferences;  EditText pinField;  String pin;  Button loginBtn;  @Override  protected void onCreate(Bundle savedInstanceState)  {  super.onCreate(savedInstanceState);  setContentView(R.layout.login);  Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  setSupportActionBar(toolbar);  PreferenceManager.setDefaultValues(this, R.xml.preferences, false);  sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);  loginBtn = (Button) findViewById(R.id.login\_button);  pinField = (EditText) findViewById(R.id.pin\_field);  pin = sharedPreferences.getString("PIN", "Not set");  if (pin.equals("Not set"))  {  AlertDialog.Builder prompt = new AlertDialog.Builder(this);  prompt.setMessage("Please set a security PIN.");  final EditText input = new EditText(this);  prompt.setView(input);  input.setLayoutParams(new LinearLayout.LayoutParams(50, 30));  input.setInputType(InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);  input.setTransformationMethod(PasswordTransformationMethod.getInstance());  prompt.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener()  {  public void onClick(DialogInterface dialog, int which)  {  SharedPreferences.Editor editor = sharedPreferences.edit();  editor.putString("PIN", input.getText().toString());  editor.commit();  Toast.makeText(getApplicationContext(), "Pin set.", Toast.LENGTH\_SHORT).show();  pin = sharedPreferences.getString("PIN", "Not set");  }  });  prompt.setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener()  {  public void onClick(DialogInterface dialog, int which)  {  Intent exitApp = new Intent(Intent.ACTION\_MAIN);  exitApp.addCategory(Intent.CATEGORY\_HOME);  exitApp.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);  startActivity(exitApp);  }  });  AlertDialog dialog = prompt.create();  dialog.setOnShowListener(new DialogInterface.OnShowListener()  {  @Override  public void onShow(DialogInterface dialog)  {  InputMethodManager imm = (InputMethodManager) getSystemService(getApplicationContext().INPUT\_METHOD\_SERVICE);  imm.showSoftInput(input, InputMethodManager.SHOW\_IMPLICIT);  }  });  dialog.show();  }  loginBtn.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(final View v)  {  if (pinField.getText().toString().equals(pin))  {  Intent main = new Intent(getApplicationContext(), MainMenu.class);  startActivity(main);  }  else  {  pinField.setText("");  Toast.makeText(getApplicationContext(), "Wrong PIN, try again.", Toast.LENGTH\_SHORT).show();  }  }  });  }  } |

This is the code for MainMenu Activity:

|  |
| --- |
| package elec291group2.com.project2;  import android.app.AlertDialog;  import android.app.Fragment;  import android.app.FragmentManager;  import android.content.DialogInterface;  import android.content.Intent;  import android.content.SharedPreferences;  import android.os.Bundle;  import android.preference.PreferenceManager;  import android.support.design.widget.NavigationView;  import android.support.v4.view.GravityCompat;  import android.support.v4.widget.DrawerLayout;  import android.support.v7.app.ActionBarDrawerToggle;  import android.support.v7.app.AppCompatActivity;  import android.support.v7.widget.Toolbar;  import android.view.Menu;  import android.view.MenuItem;  public class MainMenu extends AppCompatActivity  implements NavigationView.OnNavigationItemSelectedListener  {  SharedPreferences sharedPreferences;  @Override  protected void onCreate(Bundle savedInstanceState)  {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  setSupportActionBar(toolbar);  toolbar.setTitle("Overview");  sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);  ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(  this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close);  drawer.setDrawerListener(toggle);  toggle.syncState();  NavigationView nv = (NavigationView) findViewById(R.id.nav\_view);  Menu menu = nv.getMenu();  nv.setNavigationItemSelectedListener(this);  String ip = sharedPreferences.getString("IP", "Not set");  String port = sharedPreferences.getString("Port", "Not set");  boolean notifStatus = sharedPreferences.getBoolean("Notifications", false);  menu.findItem(R.id.ip\_address).setTitle("IP Address: " + ip);  menu.findItem(R.id.port).setTitle("Port: " + port);  menu.findItem(R.id.notifications).setTitle("Notifications: " + (notifStatus ? "On" : "Off"));  if (ip.equals("Not set") || port.equals("Not set"))  {  toolbar.setTitle("Settings");  AlertDialog.Builder prompt = new AlertDialog.Builder(this);  prompt.setMessage("Please enter your IP address and port.");  prompt.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener()  {  public void onClick(DialogInterface dialog, int which)  {  }  });  prompt.show();  getFragmentManager().beginTransaction().replace(R.id.relativeLayout, new Settings()).commit();  }  else  {  getFragmentManager().beginTransaction().replace(R.id.relativeLayout, new Overview()).commit();  }  }  @Override  public void onBackPressed()  {  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);  if (drawer.isDrawerOpen(GravityCompat.START))  {  drawer.closeDrawer(GravityCompat.START);  }  else  {  ((Toolbar) findViewById(R.id.toolbar)).setTitle("Overview");  getFragmentManager().beginTransaction().replace(R.id.relativeLayout, new Overview()).commit();  }  }  @SuppressWarnings("StatementWithEmptyBody")  @Override  public boolean onNavigationItemSelected(MenuItem item)  {  Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  Fragment fragment = null;  switch(item.getItemId())  {  case R.id.nav\_overview:  toolbar.setTitle("Overview");  fragment = new Overview();  break;  case R.id.nav\_security:  toolbar.setTitle("Security");  fragment = new Security();  break;  case R.id.nav\_camera:  Intent camera = new Intent(getApplicationContext(), Camera.class);  startActivity(camera);  overridePendingTransition(R.anim.slide\_in\_left, R.anim.slide\_out\_left);  break;  case R.id.nav\_lights:  toolbar.setTitle("Lights");  fragment = new Lights();  break;  case R.id.nav\_settings:  toolbar.setTitle("Settings");  fragment = new Settings();  break;  default:  break;  }  if(fragment != null)  {  FragmentManager fm = getFragmentManager();  fm.beginTransaction().replace(R.id.relativeLayout, fragment).commit();  DrawerLayout dl = (DrawerLayout) findViewById(R.id.drawer\_layout);  dl.closeDrawers();  }  return true;  }  } |

The following code is used for Lights Fragment

|  |
| --- |
| package elec291group2.com.project2;  import android.app.Fragment;  import android.content.SharedPreferences;  import android.graphics.Color;  import android.os.Bundle;  import android.os.Handler;  import android.os.Looper;  import android.preference.PreferenceManager;  import android.support.annotation.Nullable;  import android.util.Log;  import android.view.LayoutInflater;  import android.view.View;  import android.view.ViewGroup;  import android.widget.Button;  import android.widget.SeekBar;  import android.widget.TextView;  import android.widget.Toast;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.OutputStreamWriter;  import java.io.PrintWriter;  import java.net.Socket;  import java.net.UnknownHostException;  /\*\*  \* Created by Kevin on 2016-03-24.  \*/  public class Lights extends Fragment  {  final boolean ON = true, OFF = false;  //Server stuff  BufferedReader in;  PrintWriter out;  Handler handler;  SharedPreferences sharedPreferences;  View view;  Button masterOnButton,  masterOffButton,  livingRoomButton,  kitchenButton,  washroomButton,  bedroomButton,  masterBedroomButton,  livingRoomTimer,  kitchenTimer,  washroomTimer,  bedroomTimer,  masterBedroomTimer;  TextView livingText,  kitchenText,  washroomText,  bedroomText,  masterBedroomText,  timerValue;  boolean livingRoomStatus = false,  kitchenStatus = false,  washroomStatus = false,  bedroomStatus = false,  masterBedroomStatus = false;  int duration = 0;  private Socket socket;  private String ipField;  private String portField;  private String status = "1111111111"; //temp status placeholder  private String auth\_key;  private Runnable getStatus = new Runnable()  {  @Override  public void run()  {  /\* do what you need to do \*/  getStatus();  // Call itself every 500 ms  }  };  @Nullable  @Override  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)  {  sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.getContext());  ipField = sharedPreferences.getString("IP", "Not set");  portField = sharedPreferences.getString("Port", "Not set");  auth\_key = sharedPreferences.getString("auth\_key", "abc123");  view = inflater.inflate(R.layout.lights, container, false);  masterOnButton = (Button) view.findViewById(R.id.master\_on\_button);  masterOffButton = (Button) view.findViewById(R.id.master\_off\_button);  livingRoomButton = (Button) view.findViewById(R.id.livingroom\_button);  kitchenButton = (Button) view.findViewById(R.id.kitchen\_button);  washroomButton = (Button) view.findViewById(R.id.washroom\_button);  bedroomButton = (Button) view.findViewById(R.id.bedroom\_button);  masterBedroomButton = (Button) view.findViewById(R.id.mbedroom\_button);  livingRoomTimer = (Button) view.findViewById(R.id.livingroom\_timer);  kitchenTimer = (Button) view.findViewById(R.id.kitchen\_timer);  washroomTimer = (Button) view.findViewById(R.id.washroom\_timer);  bedroomTimer = (Button) view.findViewById(R.id.bedroom\_timer);  masterBedroomTimer = (Button) view.findViewById(R.id.mbedroom\_timer);  livingText = (TextView) view.findViewById(R.id.livingroom\_status);  kitchenText = (TextView) view.findViewById(R.id.kitchen\_status);  washroomText = (TextView) view.findViewById(R.id.washroom\_status);  bedroomText = (TextView) view.findViewById(R.id.bedroom\_status);  masterBedroomText = (TextView) view.findViewById(R.id.mbedroom\_status);  SeekBar timerSlider = (SeekBar) view.findViewById(R.id.timer\_slider);  timerValue = (TextView) view.findViewById(R.id.timer\_text);  masterOnButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("MasterLights ON");  }  });  masterOffButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("MasterLights OFF");  }  });  livingRoomButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Living " + (livingRoomStatus ? "OFF" : "ON"));  }  });  kitchenButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Kitchen " + (kitchenStatus ? "OFF" : "ON"));  }  });  washroomButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Washroom " + (washroomStatus ? "OFF" : "ON"));  }  });  bedroomButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Bedroom " + (bedroomStatus ? "OFF" : "ON"));  }  });  masterBedroomButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("MasterBedroom " + (masterBedroomStatus ? "OFF" : "ON"));  }  });  livingRoomTimer.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("LivingRoomTimed " + duration);  }  });  kitchenTimer.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("KitchenTimed " + duration);  }  });  washroomTimer.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("WashroomTimed " + duration);  }  });  bedroomTimer.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("BedroomTimed " + duration);  }  });  masterBedroomTimer.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("MasterBedroomTimed " + duration);  }  });  timerSlider.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()  {  @Override  public void onProgressChanged(SeekBar seekBar, int progress,  boolean fromUser)  {  timerValue.setText("Timer: " + String.valueOf(progress \* 60 / 100) + " seconds");  duration = progress \* 60 / 100;  }  @Override  public void onStartTrackingTouch(SeekBar seekBar)  {  }  @Override  public void onStopTrackingTouch(SeekBar seekBar)  {  }  });  new Thread(new ClientThread()).start();  updateText();  return view;  }  public void updateText()  {  int livingRoomLights = Character.getNumericValue(status.charAt(5)),  kitchenLights = Character.getNumericValue(status.charAt(6)),  washroomLights = Character.getNumericValue(status.charAt(7)),  bedroomLights = Character.getNumericValue(status.charAt(8)),  masterBedroomLights = Character.getNumericValue(status.charAt(9));  // 0 = on (R), 1 = off (G)  livingRoomStatus = livingRoomLights == 0 ? OFF : ON;  livingText.setText(livingRoomLights == 0 ? "OFF" : "ON");  livingText.setTextColor(livingRoomLights == 0 ? Color.RED : Color.GREEN);  kitchenStatus = kitchenLights == 0 ? OFF : ON;  kitchenText.setText(kitchenLights == 0 ? "OFF" : "ON");  kitchenText.setTextColor(kitchenLights == 0 ? Color.RED : Color.GREEN);  washroomStatus = washroomLights == 0 ? OFF : ON;  washroomText.setText(washroomLights == 0 ? "OFF" : "ON");  washroomText.setTextColor(washroomLights == 0 ? Color.RED : Color.GREEN);  bedroomStatus = bedroomLights == 0 ? OFF : ON;  bedroomText.setText(bedroomLights == 0 ? "OFF" : "ON");  bedroomText.setTextColor(bedroomLights == 0 ? Color.RED : Color.GREEN);  masterBedroomStatus = masterBedroomLights == 0 ? OFF : ON;  masterBedroomText.setText(masterBedroomLights == 0 ? "OFF" : "ON");  masterBedroomText.setTextColor(masterBedroomLights == 0 ? Color.RED : Color.GREEN);  }  @Override  public void onPause()  {  if (socket != null)  {  sendCommand("exit");  try  {  in.close();  out.close();  socket.close();  } catch (Exception e)  {  e.printStackTrace();  }  // Toast.makeText(this.getContext(), "Client has closed the connection.", Toast.LENGTH\_SHORT).show();  }  super.onPause();  }  private void sendCommand(String command)  {  if (out != null)  {  try  {  out.println(command);  } catch (Exception e)  {  e.printStackTrace();  }  }  }  public void updateStatusUI()  {  getActivity().runOnUiThread(new Runnable()  {  @Override  public void run()  {  updateText();  //updateAllButtons();  }  });  }  private void getStatus()  {  try  {  if (in.ready()) // Retrieve command from Android device, add to device queue  {  status = in.readLine();  Log.v("System.out", status);  if (status.length() == 10)  {  updateStatusUI();  }  handler.postDelayed(getStatus, 1000);  }  } catch (Exception e)  {  e.printStackTrace();  handler.removeCallbacksAndMessages(getStatus);  }  }  private void showToast(String message)  {  final String msg = message;  new Handler(Looper.getMainLooper()).post(new Runnable()  {  @Override  public void run()  {  Toast.makeText(getContext(), msg, Toast.LENGTH\_LONG).show();  }  });  }  class ClientThread implements Runnable  {  @Override  public void run()  {  try  {  socket = new Socket(ipField, Integer.parseInt(portField));  if (socket != null) // TODO: Find a valid condition to check  {  in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())), true);  sendCommand(auth\_key);  String verification\_status = in.readLine();  Log.v("System.out", verification\_status);  if (verification\_status.equals("Verified"))  {  showToast("Connected.");  Looper.prepare();  handler = new Handler();  handler.postDelayed(getStatus, 1000);  Looper.loop();  }  else  {  showToast("Authentication key is incorrect");  }  }  else  {  showToast("Server information is incorrect.");  }  } catch (UnknownHostException e1)  {  e1.printStackTrace();  } catch (IOException e1)  {  e1.printStackTrace();  } catch (NumberFormatException e1)  {  e1.printStackTrace();  }  }  }  } |

This is the code for Security Fragment:

|  |
| --- |
| package elec291group2.com.project2;  import android.app.Fragment;  import android.content.SharedPreferences;  import android.graphics.Color;  import android.os.Bundle;  import android.os.Handler;  import android.os.Looper;  import android.preference.PreferenceManager;  import android.support.annotation.Nullable;  import android.util.Log;  import android.view.LayoutInflater;  import android.view.View;  import android.view.ViewGroup;  import android.widget.Button;  import android.widget.TextView;  import android.widget.Toast;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.OutputStreamWriter;  import java.io.PrintWriter;  import java.net.Socket;  import java.net.UnknownHostException;  /\*\*  \* Created by Kevin on 2016-03-23.  \*/  public class Security extends Fragment  {  final boolean ON = true, OFF = false;  SharedPreferences sharedPreferences;  View view;  BufferedReader in;  PrintWriter out;  Handler handler;  Button masterArmButton,  masterDisarmButton,  doorButton,  motionButton,  laserButton,  alarmButton;  boolean systemStatus = false,  doorStatus = false,  motionStatus = false,  laserStatus = false,  alarmStatus = false;  TextView systemText, doorText, motionText, laserText, alarmText; // security system  private Socket socket;  private String ipField;  private String portField;  private String status = "1111111111";  private String auth\_key = "";  private Runnable getStatus = new Runnable()  {  @Override  public void run()  {  /\* do what you need to do \*/  getStatus();  // Call itself every 500 ms  handler.postDelayed(this, 500);  }  };  @Nullable  @Override  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)  {  view = inflater.inflate(R.layout.security, container, false);  // get the IP and port for socket  sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.getActivity());  ipField = sharedPreferences.getString("IP", "Not set");  portField = sharedPreferences.getString("Port", "Not set");  auth\_key = sharedPreferences.getString("auth\_key", "1234");  // system arming buttons  masterArmButton = (Button) view.findViewById(R.id.master\_arm\_button);  masterDisarmButton = (Button) view.findViewById(R.id.master\_disarm\_button);  doorButton = (Button) view.findViewById(R.id.door\_button);  motionButton = (Button) view.findViewById(R.id.motion\_button);  laserButton = (Button) view.findViewById(R.id.laser\_button);  alarmButton = (Button) view.findViewById(R.id.alarm\_button);  // security system status  systemText = (TextView) view.findViewById(R.id.system\_status);  doorText = (TextView) view.findViewById(R.id.door\_status);  motionText = (TextView) view.findViewById(R.id.motion\_status);  laserText = (TextView) view.findViewById(R.id.laser\_status);  alarmText = (TextView) view.findViewById(R.id.alarm\_status);  masterArmButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("System ON");  }  });  masterDisarmButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("System OFF");  }  });  doorButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Door " + (doorStatus ? "OFF" : "ON"));  }  });  motionButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Motion " + (motionStatus ? "OFF" : "ON"));  }  });  laserButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Laser " + (laserStatus ? "OFF" : "ON"));  }  });  alarmButton.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  sendCommand("Alarm " + (alarmStatus ? "OFF" : "ON"));  }  });  new Thread(new ClientThread()).start();  //updateAllButtons();  updateText();  return view;  }  public void updateText()  {  int systemValue = Character.getNumericValue(status.charAt(0)),  doorValue = Character.getNumericValue(status.charAt(1)),  motionValue = Character.getNumericValue(status.charAt(2)),  laserValue = Character.getNumericValue(status.charAt(3)),  alarmValue = Character.getNumericValue(status.charAt(4));  // systemValue: 0 = unarmed (G), 1 = armed (B), 2 = triggered (R)  systemStatus = systemValue == 0 ? OFF : ON;  systemText.setText(systemValue == 0 ? "UNARMED" :  systemValue == 1 ? "ARMED" : "TRIGGERED");  systemText.setTextColor(systemValue == 0 ? Color.GREEN :  systemValue == 1 ? Color.BLUE : Color.RED);  // doorValue: 0 = closed (G), 1 = armed (B), 2 = open (M), 3 = triggered (R)  doorStatus = doorValue == 0 || doorValue == 2 ? OFF : ON;  doorText.setText(doorValue == 0 ? "CLOSED" :  doorValue == 1 ? "ARMED" :  doorValue == 2 ? "OPEN" : "TRIGGERED");  doorText.setTextColor(doorValue == 0 ? Color.GREEN :  doorValue == 1 ? Color.BLUE :  doorValue == 2 ? Color.MAGENTA : Color.RED);  // motionValue: 0 = idle (G), 1 = armed (B), 2 = detected (M), 3 = triggered (R)  motionStatus = motionValue == 0 || motionValue == 2 ? OFF : ON;  motionText.setText(motionValue == 0 ? "IDLE" :  motionValue == 1 ? "ARMED" :  motionValue == 2 ? "DETECTED" : "TRIGGERED");  motionText.setTextColor(motionValue == 0 ? Color.GREEN :  motionValue == 1 ? Color.BLUE :  motionValue == 2 ? Color.MAGENTA : Color.RED);  // laserValue: 0 = unarmed (G), 1 = armed (B), 2 = triggered (R)  laserStatus = laserValue == 0 ? OFF : ON;  laserText.setText(laserValue == 0 ? "UNARMED" :  laserValue == 1 ? "ARMED" : "TRIGGERED");  laserText.setTextColor(laserValue == 0 ? Color.GREEN :  laserValue == 1 ? Color.BLUE : Color.RED);  // alarmValue: 0 = off (R), 1 = on (G)  alarmStatus = laserValue == 0 ? OFF : ON;  alarmText.setText(alarmValue == 0 ? "OFF" : "ON");  alarmText.setTextColor(alarmValue == 0 ? Color.RED : Color.GREEN);  }  /\*  public void updateAllButtons()  {  // update Value of the lights  systemStatus = Character.getNumericValue(status.charAt(0)) == 1 ? ON : OFF;  doorStatus = Character.getNumericValue(status.charAt(1)) == 1 ? ON : OFF;  motionStatus = Character.getNumericValue(status.charAt(2)) == 1 ? ON : OFF;  laserStatus = Character.getNumericValue(status.charAt(3)) == 1 ? ON : OFF;  alarmStatus = Character.getNumericValue(status.charAt(4)) == 1 ? ON : OFF;  // update buttons with new statuses  updateButton(doorButton, doorStatus);  updateButton(motionButton, motionStatus);  updateButton(laserButton, laserStatus);  updateButton(alarmButton, alarmStatus);  }  public void updateButton(Button btn, boolean status)  {  btn.setText(status ? "ARMED" : "DISARMED");  btn.getBackground().setColorFilter(status ? Color.GREEN : Color.RED, PorterDuff.Mode.MULTIPLY);  }  \*/  @Override  public void onPause()  {  if(socket != null)  {  sendCommand("exit");  try  {  in.close();  out.close();  socket.close();  } catch (Exception e)  {  e.printStackTrace();  }  // Toast.makeText(this.getContext(), "Client has closed the connection.", Toast.LENGTH\_SHORT).show();  }  super.onPause();  }  private void sendCommand(String command)  {  if(out != null)  {  try  {  out.println(command);  } catch (Exception e)  {  e.printStackTrace();  }  }  }  public void updateStatusUI()  {  getActivity().runOnUiThread(new Runnable()  {  @Override  public void run()  {  //updateAllButtons();  updateText();  }  });  }  private void getStatus()  {  try  {  if (in.ready()) // Retrieve command from Android device, add to device queue  {  status = in.readLine();  Log.v("System.out", status);  if(status.length() == 10)  {  updateStatusUI();  }  handler.postDelayed(getStatus, 1000);  }  } catch (Exception e)  {  e.printStackTrace();  handler.removeCallbacksAndMessages(getStatus);  }  }  class ClientThread implements Runnable  {  @Override  public void run()  {  try  {  socket = new Socket(ipField, Integer.parseInt(portField));  if(socket != null) // TODO: Find a valid condition to check  {  in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())), true);  sendCommand(auth\_key);  String verification\_status = in.readLine();  Log.v("System.out", verification\_status);  if(verification\_status.equals("Verified"))  {  showToast("Connected.");  Looper.prepare();  handler = new Handler();  handler.postDelayed(getStatus, 1000);  Looper.loop();  }  else  {  showToast("Authentication key is incorrect");  }  }  else  {  showToast("Server information is incorrect.");  }  }  catch (UnknownHostException e1)  {  e1.printStackTrace();  }  catch (IOException e1)  {  e1.printStackTrace();  }  catch (NumberFormatException e1)  {  e1.printStackTrace();  }  }  }  private void showToast(String message) {  final String msg = message;  new Handler(Looper.getMainLooper()).post(new Runnable() {  @Override  public void run() {  Toast.makeText(getActivity(), msg, Toast.LENGTH\_LONG).show();  }  });  }  } |

This is the code for Overview Fragment:

|  |
| --- |
| package elec291group2.com.project2;  import android.app.Fragment;  import android.content.SharedPreferences;  import android.graphics.Color;  import android.os.Bundle;  import android.os.Handler;  import android.os.Looper;  import android.preference.PreferenceManager;  import android.support.annotation.Nullable;  import android.util.Log;  import android.view.LayoutInflater;  import android.view.View;  import android.view.ViewGroup;  import android.widget.TextView;  import android.widget.Toast;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.OutputStreamWriter;  import java.io.PrintWriter;  import java.net.Socket;  import java.net.UnknownHostException;  /\*\*  \* Created by Kevin on 2016-03-23.  \*/  public class Overview extends Fragment  {  View view;  SharedPreferences sharedPreferences;  //Server stuff  BufferedReader in;  PrintWriter out;  Handler handler;  TextView systemText, doorText, motionText, laserText, alarmText, // security system  livingText, kitchenText, washroomText, bedroomText, masterBedroomText; // lights  private Socket socket;  private String ipField;  private String portField;  private String auth\_key;  // status: { systemStatus, doorStatus, motionStatus, laserStatus, alarmStatus  // livingRoomLights, kitchenLights, washroomLights, bedroomLights, masterBedroomLights }  private String status = "1111111111";  private Runnable getStatus = new Runnable()  {  @Override  public void run()  {  /\* do what you need to do \*/  getStatus();  }  };  @Nullable  @Override  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)  {  view = inflater.inflate(R.layout.overview, container, false);  // get the IP and port for socket  sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.getActivity());  ipField = sharedPreferences.getString("IP", "Not set");  portField = sharedPreferences.getString("Port", "Not set");  auth\_key = sharedPreferences.getString(("auth\_key"),"1234567");  // security system status  systemText = (TextView) view.findViewById(R.id.system\_status);  doorText = (TextView) view.findViewById(R.id.door\_status);  motionText = (TextView) view.findViewById(R.id.motion\_status);  laserText = (TextView) view.findViewById(R.id.laser\_status);  alarmText = (TextView) view.findViewById(R.id.alarm\_status);  // lights status  livingText = (TextView) view.findViewById(R.id.livingroom\_status);  kitchenText = (TextView) view.findViewById(R.id.kitchen\_status);  washroomText = (TextView) view.findViewById(R.id.washroom\_status);  bedroomText = (TextView) view.findViewById(R.id.bedroom\_status);  masterBedroomText = (TextView) view.findViewById(R.id.mbedroom\_status);  new Thread(new ClientThread()).start();  return view;  }  /\*\*  \* Update security system status text on Overview with status string.  \*/  public void updateSecurity()  {  int systemStatus = Character.getNumericValue(status.charAt(0)),  doorStatus = Character.getNumericValue(status.charAt(1)),  motionStatus = Character.getNumericValue(status.charAt(2)),  laserStatus = Character.getNumericValue(status.charAt(3)),  alarmStatus = Character.getNumericValue(status.charAt(4));  // systemStatus: 0 = unarmed (G), 1 = armed (B), 2 = triggered (R)  systemText.setText(systemStatus == 0 ? "UNARMED" :  systemStatus == 1 ? "ARMED" : "TRIGGERED");  systemText.setTextColor(systemStatus == 0 ? Color.GREEN :  systemStatus == 1 ? Color.BLUE : Color.RED);  // doorStatus: 0 = closed (G), 1 = armed (B), 2 = open (M), 3 = triggered (R)  doorText.setText(doorStatus == 0 ? "CLOSED" :  doorStatus == 1 ? "ARMED" :  doorStatus == 2 ? "OPEN" : "TRIGGERED");  doorText.setTextColor(doorStatus == 0 ? Color.GREEN :  doorStatus == 1 ? Color.BLUE :  doorStatus == 2 ? Color.MAGENTA : Color.RED);  // motionStatus: 0 = idle (G), 1 = armed (B), 2 = detected (M), 3 = triggered (R)  motionText.setText(motionStatus == 0 ? "IDLE" :  motionStatus == 1 ? "ARMED" :  motionStatus == 2 ? "DETECTED" : "TRIGGERED");  motionText.setTextColor(motionStatus == 0 ? Color.GREEN :  motionStatus == 1 ? Color.BLUE :  motionStatus == 2 ? Color.MAGENTA : Color.RED);  // motionStatus: 0 = unarmed (G), 1 = armed (B), 2 = triggered (R)  laserText.setText(laserStatus == 0 ? "UNARMED" :  laserStatus == 1 ? "ARMED" : "TRIGGERED");  laserText.setTextColor(laserStatus == 0 ? Color.GREEN :  laserStatus == 1 ? Color.BLUE : Color.RED);  // alarmStatus: 0 = off (R), 1 = on (G)  alarmText.setText(alarmStatus == 0 ? "OFF" : "ON");  alarmText.setTextColor(alarmStatus == 0 ? Color.RED : Color.GREEN);  }  /\*\*  \* Update light status text on Overview with status string.  \*/  public void updateLights()  {  int livingRoomLights = Character.getNumericValue(status.charAt(5)),  kitchenLights = Character.getNumericValue(status.charAt(6)),  washroomLights = Character.getNumericValue(status.charAt(7)),  bedroomLights = Character.getNumericValue(status.charAt(8)),  masterBedroomLights = Character.getNumericValue(status.charAt(9));  // 0 = on (R), 1 = off (G)  livingText.setTextColor(livingRoomLights == 0 ? Color.RED : Color.GREEN);  kitchenText.setTextColor(kitchenLights == 0 ? Color.RED : Color.GREEN);  washroomText.setTextColor(washroomLights == 0 ? Color.RED : Color.GREEN);  bedroomText.setTextColor(bedroomLights == 0 ? Color.RED : Color.GREEN);  masterBedroomText.setTextColor(masterBedroomLights == 0 ? Color.RED : Color.GREEN);  }  @Override  public void onPause()  {  if(socket != null)  {  sendCommand("exit");  try  {  in.close();  out.close();  socket.close();  } catch (Exception e)  {  e.printStackTrace();  }  // Toast.makeText(this.getContext(), "Client has closed the connection.", Toast.LENGTH\_SHORT).show();  }  super.onPause();  }  private void sendCommand(String command)  {  if(out != null)  {  try  {  out.println(command);  } catch (Exception e)  {  e.printStackTrace();  }  }  }  public void updateStatusUI()  {  getActivity().runOnUiThread(new Runnable()  {  @Override  public void run()  {  updateSecurity();  updateLights();  }  });  }  private void getStatus()  {  try  {  if (in.ready()) // Retrieve command from Android device, add to device queue  {  status = in.readLine();  Log.v("System.out",status);  if(status.length() == 10)  {  updateStatusUI();  }  handler.postDelayed(getStatus, 1000);  }  } catch (Exception e)  {  e.printStackTrace();  handler.removeCallbacksAndMessages(getStatus);  }  }  class ClientThread implements Runnable  {  @Override  public void run()  {  try  {  socket = new Socket(ipField, Integer.parseInt(portField));  if(socket != null) // TODO: Find a valid condition to check  {  in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())), true);  Log.v("System.out", auth\_key);  sendCommand(auth\_key);  String verification\_status = in.readLine();  Log.v("System.out", verification\_status);  if(verification\_status.equals("Verified"))  {  showToast("Connected.");  Looper.prepare();  handler = new Handler();  handler.postDelayed(getStatus, 1000);  Looper.loop();  }  else  {  showToast("Authentication key is incorrect");  }  }  else  {  showToast("Server information is incorrect.");  }  }  catch (UnknownHostException e1)  {  e1.printStackTrace();  }  catch (IOException e1)  {  e1.printStackTrace();  }  catch (NumberFormatException e1)  {  e1.printStackTrace();  }  }  }  private void showToast(String message) {  final String msg = message;  new Handler(Looper.getMainLooper()).post(new Runnable() {  @Override  public void run() {  Toast.makeText(getActivity(), msg, Toast.LENGTH\_LONG).show();  }  });  }  } |

The following code is for Settings Fragment:

|  |
| --- |
| package elec291group2.com.project2;  import android.content.BroadcastReceiver;  import android.content.Context;  import android.content.Intent;  import android.content.IntentFilter;  import android.content.SharedPreferences;  import android.os.AsyncTask;  import android.os.Bundle;  import android.preference.CheckBoxPreference;  import android.preference.EditTextPreference;  import android.preference.ListPreference;  import android.preference.Preference;  import android.preference.PreferenceFragment;  import android.preference.PreferenceGroup;  import android.preference.PreferenceManager;  import android.support.design.widget.NavigationView;  import android.support.v4.content.LocalBroadcastManager;  import android.support.v4.content.res.ResourcesCompat;  import android.view.Menu;  import android.widget.Toast;  import com.google.android.gms.common.ConnectionResult;  import com.google.android.gms.common.GoogleApiAvailability;  import elec291group2.com.project2.gcm.RegistrationIntentService;  import elec291group2.com.project2.gcm.constants;  /\*\*  \* Created by Kevin Qiu on 2016-03-29.  \*/  public class Settings extends PreferenceFragment implements SharedPreferences.OnSharedPreferenceChangeListener  {  @Override  public void onCreate(Bundle savedInstanceState)  {  super.onCreate(savedInstanceState);  addPreferencesFromResource(R.xml.preferences);  getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeListener(this);  }  @Override  public void onResume()  {  super.onResume();  for (int i = 0; i < getPreferenceScreen().getPreferenceCount(); ++i)  {  Preference preference = getPreferenceScreen().getPreference(i);  if (preference instanceof PreferenceGroup)  {  PreferenceGroup preferenceGroup = (PreferenceGroup) preference;  for (int j = 0; j < preferenceGroup.getPreferenceCount(); ++j)  {  Preference singlePref = preferenceGroup.getPreference(j);  updatePreference(singlePref, singlePref.getKey());  }  }  else  {  updatePreference(preference, preference.getKey());  }  }  }  @Override  public void onSharedPreferenceChanged(final SharedPreferences sharedPreferences, final String key)  {  updatePreference(findPreference(key), key);  Menu menu = ((NavigationView) getActivity().findViewById(R.id.nav\_view)).getMenu();  menu.findItem(R.id.ip\_address).setTitle("IP Address: " + sharedPreferences.getString("IP", ""));  menu.findItem(R.id.port).setTitle("Port: " + sharedPreferences.getString("Port", ""));  //menu.findItem(R.id.auth\_key).setTitle("Authentication Key: " + sharedPreferences.getString("Authentication Key", ""));  // Register with GCM and app server if Notifications switches to 'enabled'  if ( key.equals("Notifications")  && sharedPreferences.getBoolean("Notifications", false)  && checkPlayServices() )  {  Context context = getActivity();  // Create broadcast receiver to update button when registration is done.  BroadcastReceiver registrationBroadcastReceiver = new BroadcastReceiver() {  @Override  public void onReceive(Context context, Intent intent) {  // Revert to unchecked button if registration failed.  if (!intent.getBooleanExtra("registrationResult", false)){  CheckBoxPreference notification\_checkbox =  (CheckBoxPreference) findPreference("Notifications");  notification\_checkbox.setChecked(false);  }  }  };  IntentFilter intentFilter = new IntentFilter(constants.BROADCAST\_REGISTRATION\_COMPLETE);  LocalBroadcastManager.getInstance(context).  registerReceiver(registrationBroadcastReceiver, intentFilter);  // Start IntentService to register this application with GCM.  Intent intent = new Intent(context, RegistrationIntentService.class);  context.startService(intent);  }  }  private void updatePreference(Preference preference, String key)  {  if(key.equals("auth\_key"))  {  SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.getActivity());  String un\_hashed = sharedPreferences.getString("auth\_key", "1234");  //sharedPreferences.edit().putString("auth\_key","1234").apply();  new hash().execute(un\_hashed);  return;  }  if (preference == null || key.equals("PIN")) return;  if (preference instanceof ListPreference)  {  ListPreference listPreference = (ListPreference) preference;  listPreference.setSummary(listPreference.getEntry());  SharedPreferences sharedPrefs = getPreferenceManager().getSharedPreferences();  preference.setSummary(sharedPrefs.getString(key, "Default"));  }  else if (preference instanceof EditTextPreference)  {  EditTextPreference editTextPreference = (EditTextPreference) preference;  editTextPreference.setSummary(editTextPreference.getText());  SharedPreferences sharedPrefs = getPreferenceManager().getSharedPreferences();  preference.setSummary(sharedPrefs.getString(key, ""));  }  else if (preference instanceof CheckBoxPreference)  {  CheckBoxPreference checkBoxPreference = (CheckBoxPreference) preference;  //checkBoxPreference.setSummary(checkBoxPreference.isChecked() ? "Enabled" : "Disabled");  SharedPreferences sharedPrefs = getPreferenceManager().getSharedPreferences();  //preference.setSummary(sharedPrefs.getBoolean(key, false) ? "Enabled" : "Disabled");  if (isAdded()){  if(!sharedPrefs.getBoolean(key,false))  preference.setIcon(ResourcesCompat.getDrawable(getResources(), R.drawable.ic\_dnd\_forwardslash\_24dp, null));  else  preference.setIcon(ResourcesCompat.getDrawable(getResources(), R.drawable.ic\_error\_24dp, null));  }  }  }  /\*\*  \* Check the device to make sure it has the Google Play Services APK. If  \* it doesn't, display a toast explaining the error code.  \*/  private boolean checkPlayServices() {  //Context context = this.getContext().getApplicationContext();  GoogleApiAvailability apiAvailability = GoogleApiAvailability.getInstance();  int resultCode = apiAvailability.isGooglePlayServicesAvailable(getActivity());  if (resultCode != ConnectionResult.SUCCESS)  {  // If failed, display error dialogs  String errorString = constants.MESSAGE\_PLAY\_SERVICES\_ERROR +  apiAvailability.getErrorString(resultCode);  Toast.makeText(getActivity(), errorString, Toast.LENGTH\_SHORT).show();  if (apiAvailability.isUserResolvableError(resultCode))  {  apiAvailability.getErrorDialog(getActivity(), resultCode,  constants.PLAY\_SERVICES\_RESOLUTION\_REQUEST).show();  }  return false;  }  else  {  return true;  }  }  private class hash extends AsyncTask<String, Void, String>  {  protected String doInBackground(String... params){  return "";//EncryptionFunction.password\_hash(params[0].toString());  }  protected void onPostExecute(String result) {  SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(getContext());  sharedPreferences.edit().putString("auth\_key", result).apply();  }  }  } |

The following code is used for Camera Activity:

|  |
| --- |
| import android.os.Bundle;  import android.support.design.widget.FloatingActionButton;  import android.support.design.widget.Snackbar;  import android.support.v7.app.AppCompatActivity;  import android.support.v7.widget.Toolbar;  import io.vov.vitamio.LibsChecker;  import io.vov.vitamio.MediaPlayer;  import io.vov.vitamio.widget.MediaController;  import io.vov.vitamio.widget.VideoView;  import android.app.Activity;  import android.widget.Toast;  import android.text.TextUtils;  import android.view.View;  import android.view.Menu;  import android.view.MenuItem;  public class MainActivity extends Activity {  // Create variables  private static final String TAG = "MainActivity";  private String path;  private VideoView mVideoView;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  if (!LibsChecker.checkVitamioLibs(this))  return;  setContentView(R.layout.content\_main);  // Select view to display camera stream on  mVideoView = (VideoView) findViewById(R.id.surface\_view);  path = /\*"rtmp://192.168.1.71:1935/live/myStream"\*/"rtmp://128.189.254.163:1935/live/myStream";  // set video path to play  mVideoView.setVideoPath(path);  mVideoView.setMediaController(new MediaController(this));  mVideoView.requestFocus();  mVideoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {  @Override  public void onPrepared(MediaPlayer mediaPlayer) {  // optional play video  mediaPlayer.setPlaybackSpeed(1.0f);  }  });  }  } |