ECE 558
EMBEDDED SYSTEM PROGRAMMING

FINAL PROJECT REPORT

SMART HOME

BY

ATHARVA MAHINDRAKAR

&

DEEPAK MOHAN

Objective –

This project centres around the utilization of a simple communication to turn off/on our Home Appliances through our Android Devices. The user is given the platform by means of which he/she can control/screen the status of the devices in the homes and also detect any intrusion. The Project draws on the motivation that gradually yet in the end, the world is moving towards Automation and Smart Systems.

Project Description –

- An attempt to develop an IOT application for controlling various household peripherals and adding some security features.
- Developing the android app where all the house member could register to the services.
- Here Raspberry Pi will work as a server and PIC microcontroller would be connected to it via I2C to control peripherals like light LEDs, fan or air-conditioning system, temperature sensor, a PIR sensor and LDR.
- The process of lighting LEDs and spinning fan is automated on the basis of user's presence in particular room.
- The speed of the fan would be determined by the ambient temperature of the room and the intensity of light would be decided by ambient lighting present in the room. User may have override to control these peripherals even though it is automated.
- Using Geo location services user location is retrieved and if any of user is not present in house then automatically all the peripherals would get switched off or to desired standby state defined by the administrator.
- If none of house member are home and some intrusion activity detected by sensors at home then an alert will be given to all the users.

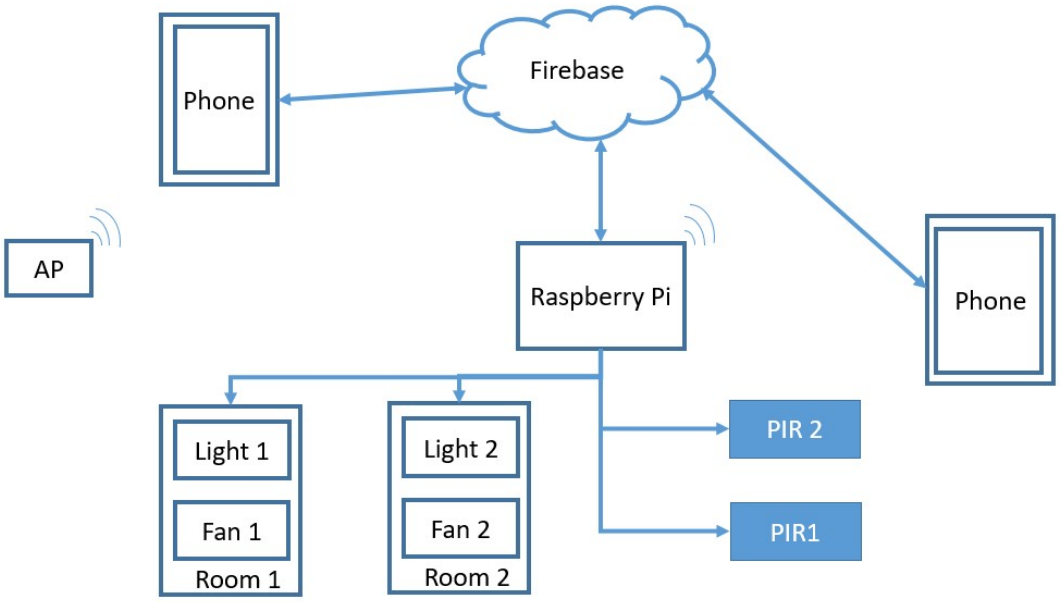
Hardware to be used –

- Raspberry pi 3B
- PIC microcontroller.
- Temperature sensor LM35.
- LEDs.
- DC motor to simulate fan (but we simulated this part using LEDs due to hardware contains).
- PIR sensor I052116.
- LDR.

Software tools –

- Android studio
- Arduino desktop application
- Android things.
- Firebase.
- Dialog Flow.
- Google assistant console.

Block Diagram –



Block Diagram

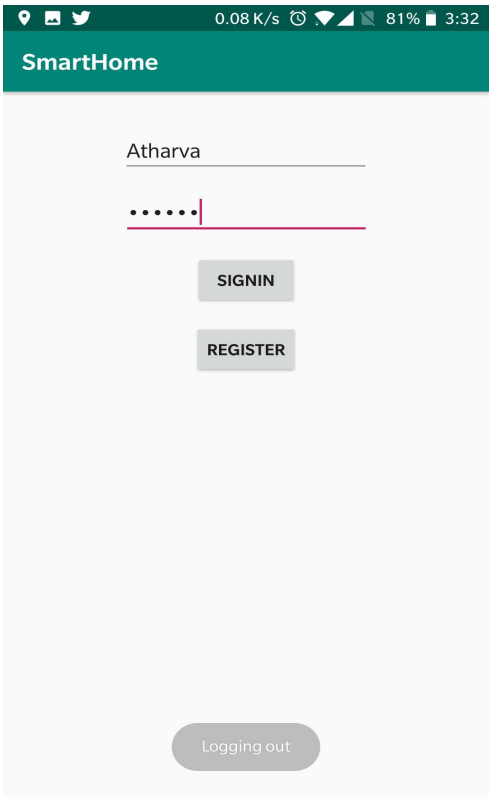
Design Flow –

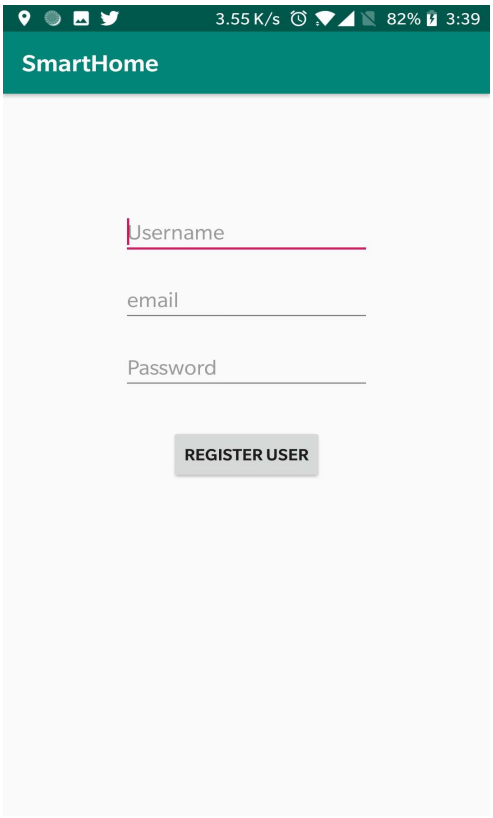
At first, we created the Database on firebase for all the peripherals. As we are using 2 LEDs and 2 Motors for each room, we named our database's nodes PWM1 to PWM4. We added an extra node called OVERRIDE, which is set by user in the application to start or stop the designed automation and give custom inputs. The rest of the nodes are discussed further in this report.

After setting up the firebase we enabled the Anonymous sign in methods for our database. We selected this method to customise the user registration and sign in methods.

User Registration and Sign in process –

After launching the app, it asks user to either login or register. We have designed our custom method for User Registration and Sign in.





- **User Registration –**

When user clicks Register button, an intent is called which takes us to **RegistrationActivity**. Here user enters his/hers credentials like email id, user name and Password. When user clicks **Register User** button, a node is created on our database with name of **User ID**, and at that node, various sub child are created to store password, email id, Home Location in the form of latitude and longitude, device MAC address, the access point which is being used and RSSI parameter.

We decided to use the parameters like MAC id, Access point name and RSSI for checking out user's location when he is indoors and actual GPS signals not available.

To help user's registration, we have created one java class named User. An instance of this class is declared in the **RegistrationActivity** which holds all the parameter discussed above. Finally data from this object is posted on firebase database.

Following snippet of code gives glimpse of User registration process –

```
NetworkInformation apInfo = new NetworkInformation();
apInfo =
MainActivity.getAPInfo(getApplicationContext().getApplicationContext());
final User user = new User(userId, password);
user.setEmail(email);
user.setSSID(apInfo.getSSID());
user.setRSSI(apInfo.getRSSI());
user.setMAC(apInfo.getMAC());

mUsers.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if(dataSnapshot.child(user.getUserId()).exists()) {
            Toast.makeText(RegistrationActivity.this, "User already exists",
Toast.LENGTH_SHORT).show();
        }else{
            mUsers.child(userId).setValue(user);
            Toast.makeText(RegistrationActivity.this, "User Registered
successfully", Toast.LENGTH_SHORT).show();
            Intent myIntent = new Intent();
            myIntent.putExtra("userId", userId);
            setResult(RESULT_OK, myIntent);
            finish();
        }
    }
```

We have included the test cases which would expel user from registering with existing User ID or with illegal User ID (a user id in which illegal characters are used like curly braces, brackets, etc).

- **Login in process –**
  The user has to enter valid login credentials to use the services. If the login credentials are not matched or found in our database, then Toast is displayed stating either "No such User exits" or "Incorrect Credentials".

  Following snippet of code gives glimpse of User Login process –

```
if(dataSnapshot.child(userId).exists()){

    Log.d(TAG, "Child exists for the user " + dataSnapshot.child(userId));

    User login = (User) dataSnapshot.child(userId).getValue(User.class);


    if(login.getPassword().equals(password)){
        Log.d(TAG, "Passwords match.. loging you in");

        Toast.makeText(MainActivity.this,"Login Success",
Toast.LENGTH_SHORT).show();
```

```
        apInfo = getAPInfo(getApplicationContext().getApplicationContext());

        login.setMAC(apInfo.getMAC());
        login.setRSSI(apInfo.getRSSI());
        login.setSSID(apInfo.getSSID());
        login.setLatitude(mLatitude);
        login.setLongitude(mLongitude);

        mUsers.child(login.getUserId()).setValue(login);
        //Database references used to write data (location/wifi) into User
class
        userDB =
mFirebaseHelper.getDatabaseChildRef(mUserId.getText().toString());
        latitudeRef =
ref.child(mUserId.getText().toString()).child("latitude");
        longitudeRef =
ref.child(mUserId.getText().toString()).child("longitude");

        loggedIn = true;

        Intent intent = new
Intent(getApplicationContext().getApplicationContext(), HomeActivity.class);
        intent.putExtra("userId", userId);
        startActivity(intent);

    }else{
        Toast.makeText(MainActivity.this, "Incorrect Credentials",
Toast.LENGTH_SHORT).show();
    }
}else{
    Toast.makeText(MainActivity.this,"No such user exisits",
Toast.LENGTH_SHORT).show();
}
```

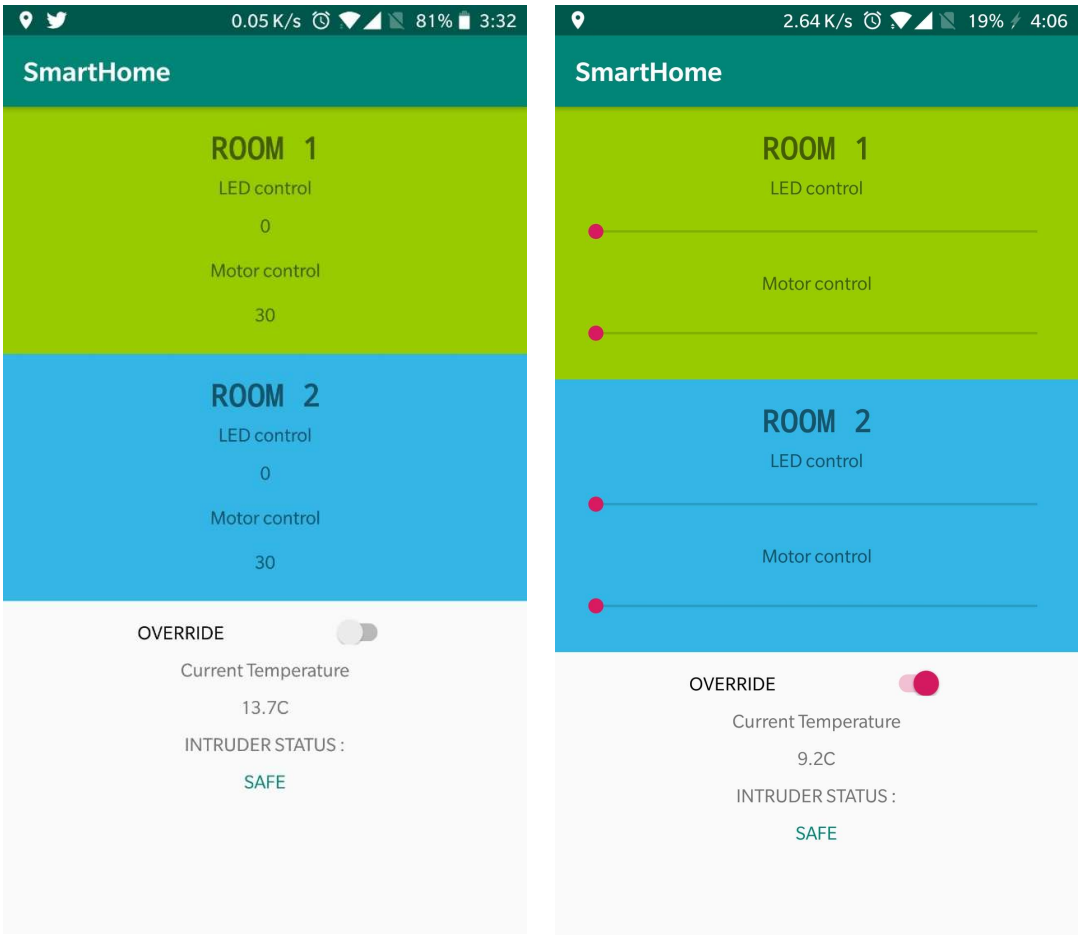
After successful login, this activity starts to send the data of location like Latitude and Longitude to the database with the help of Geolocation services and we get transferred to HomeActivity using intent.

Following snippet of code gives glimpse of Geolocation services –

```
mLocationManager = (LocationManager)this.getSystemService(LOCATION_SERVICE);

mLocationListener = new android.location.LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        Log.d(TAG, location.toString());

        if(loggedIn) {
                mLatitude = location.getLatitude();
                mLongitude = location.getLongitude();

                latitudeRef.setValue(mLatitude);
                longitudeRef.setValue(mLongitude);

            }

        }else {
            Log.d(TAG, "Not logged in.. discarding the write to DB");
        }
```

After logging in, the parameters like home location are sent to nodes “HOMELAT” AND “HOMELNG”. Along with that the current location are continuously updated to database’s nodes “LATITUDE” and “LONGITUDE”.

Home Automation & control –



HomeActivity UI

After logging in, user gets this UI on the device. Here user can use this part in two modes. In first mode where Override switch is off, user gets to see current statuses of the peripherals in his house and in second mode, if user switches on Override method, he/she gets an option of setting up the peripherals to desired levels using Seekbar. In this activity, two fragment listeners are defined to send data like peripheral statuses to each room’s fragment when first mode and in second mode when override switch is on, it helps in receiving data from each room fragments to get statuses of seekbars. We have also added scroll view when device is in landscape mode so that app UI would be accessable.

The user also gets to see the current Intruder status. If user is not home and none of the Intruder activity is registered by PIR sensors then it displays “SAFE”, otherwise it shows “INTRUDER DETECTED”.

Following snippet of code gives glimpse of communication between fragments and activity –

This interface is defined in FragmentA.

```
public interface FragmentAListener {
    void onInputASent(String progress, String childname);
}

if(seekBarID == PWM1){
    listener.onInputASent(String.valueOf(seekbar_led.getProgress()),
"PWM1");
}
else if(seekBarID == PWM2){
    listener.onInputASent(String.valueOf(seekbar_motor.getProgress()),
"PWM2");
}
```

this is a implementation of interface's function in HomeActivity

```
public void onInputASent(String progress, String childname) {
    Log.d(TAG,"Room 1 Data - "+childname+"  "+"Status - "+progress );

    fbHelper.writeToDataBase(childname, progress);
}
```

Android Things application for Raspberry Pi –

We have built this part on top of Project 3's application. We have used 4 PWMs to control 2 sets of peripherals (LEDs and Motors) simulated as rooms. To detect the presence in room, we have used PIR sensors. The PIR sensors are connected to GPIOs of Raspberry Pi.

For getting PIR triggers, we have used "registerGpioCallback" method, which works as an interrupt method. It helps registering the status of the sensors as quickly as possible.

The following snippet of code helps understand setting up of GPIO callback method –

```
PIR1.setEdgeTriggerType(Gpio.EDGE_BOTH);
PIR1.registerGpioCallback(mPIR1callback);

PIR2.setEdgeTriggerType(Gpio.EDGE_BOTH);
PIR2.registerGpioCallback(mPIR2callback);
```

Gpio callback for PIR sensor 1 –

```
private GpioCallback mPIR1callback = new GpioCallback() {
    @Override
    public boolean onGpioEdge(Gpio gpio) {

        try{
            if(gpio.getValue()) {
                room1_status = true;
                if(!geolocation && room1_status) {
                    intrusion_detection = true;
                    fbHelper.writeToDataBase("INTRUSION", "1");
                }
                System.out.println("PIR1 status - " + room1_status+" Intrusion
status - "+intrusion_detection);
```

```
            }
            else {
                room1_status = false;
                intrusion_detection = false;
                fbHelper.writeToDataBase("INTRUSION", "0");
                System.out.println("PIR1 status - "+room1_status);
            }
        }catch (IOException e){
            System.out.println("Error in PIR1 GPIO callback");
        }

        return true;
    }
};
```

We are also using a FirebaseHelper class developed for Project 3 to connect Raspberri Pi to firebase.

Google Assistant Integration –

As this was one of our stretched goal, we attempted to integrate Google assistant with firebase. For that purpose, we created one dummy database on Atharva's Google account. First we created two nodes called LED1 and LED2 to our database. Then using Dialog flow, we created two entities for this LEDs and created an application called "IOT controller". To setup communication between firebase and google assistant, we developed one Node JS script.

Steps to use google assistant to use IOT controller App –

1. Trigger google assistant
2. Say "Talk to IOT controller".
3. Wait for reply from firebase.
4. To turn on LED, say "turn on LED1"
5. To turn off LED, say "turn off LED1".

Firebase database output to above commands

Limitations we faced – Whenever we would start operating on LED 2, the state current state of LED 1 used to get erased.

For example I follow following commands, the output from database is as shown in the images –

1. Turn on LED 1

2. Turn on LED 2

As seen from these outputs, the status of LED 1 got erased as we gave command for LED 2. The reverse is also true for LED 1.

Goals Achieved –

- Automation of switching on/off peripherals using Temperature, LDR and PIR sensors.
- Override from user to turn off automation and set peripherals settings to user's desired state.
- User authentication for android application.
- Checking out user's presence in home using Geolocation services from android device.
- Detecting intrusion detection if user is not present in home.
- Integration of Google assistant (Partially, only integrated with firebase).

Challenges & Trade-offs –

- We first developed automation part on Arduino and were able to send sensors statuses from Arduino to Raspberry via UART, but we unable to send Override commands from Raspberry Pi to Arduino, latter came to know that this system would have worked better if we use RST& CST signals, but these are not present on Arduino board. So we decided to move to PIC microcontroller used in Project 3.
- The proposed system would have worked if we had connected Arduino board to an USB port of Raspberry Pi.
- We manually set a delta parameter which is difference between user's latitude & longitude with home's latitude and longitude. Our findings when experimented with Google maps showed that, a difference of 0.00004 between user's latitude & longitude with home's latitude and longitude gives us roughly 30 feet radius.

```
double latdiff = homelat - lat;
        double londiff = homelon - lon;

        Log.d(TAG,"Differeces - "+latdiff+ " "+londiff);

        if(latdiff > -0.00004 && latdiff < 0.00004){
            if (londiff > -0.00004 && londiff < 0.00004){

                geolocation = true;
                Log.d(TAG, "USER IS AT HOME. DETERMINED BY GEO LOCATION
SERVICES " + geolocation);
                //geolocation = true;
            }
        }
        else {
            geolocation = false;
            fbHelper.writeToDataBase("geolocation", "FALSE");
            Log.d(TAG, "USER IS NOT AT HOME. DETERMINED BY GEO LOCATION
SERVICES " + geolocation);
            //false
        }
```

- Instead of this method, we could have used Google Map's APIs.
- If user is present in his home, then we were going to switch from GPS based location services to network based. In this method, if user's device is connected to same Wi-Fi access point as our Raspberry Pi, then we will say user is home. This was back up method as GPS won't work indoors. We were not able to completely integrate this system with our application.
- While integrating Google assistant with firebase, we faced the problem as discussed above. Also while integrating this with our developed Firebase database, we faced some account access issues while installing Node JS's NPM package to our database.

Future Scope –

- Tracking individual User's locations to create multi user system.
- Giving option of Network based location detection.
- Intrusion detection, even user is present inside home.
- Complete integration with Google Assistant.
- Performing Analytics on Power consumption of all peripherals.

Contribution by Atharva –

- Worked on Arduino, PIC and Developed Android Things Application for Raspberry Pi.
- Developed UI for HomeActivity with Fragments in Android App.
- Developed Google assistant's integration with firebase.

Contribution by Deepak –

- Developed custom login authentication method for our database.
- Worked on Geolocation services.
- Worked on Network based user location tracking.