Automatic Watering System Smart Farm

Assumption University

CE4301

Fundamentals Internet of Things

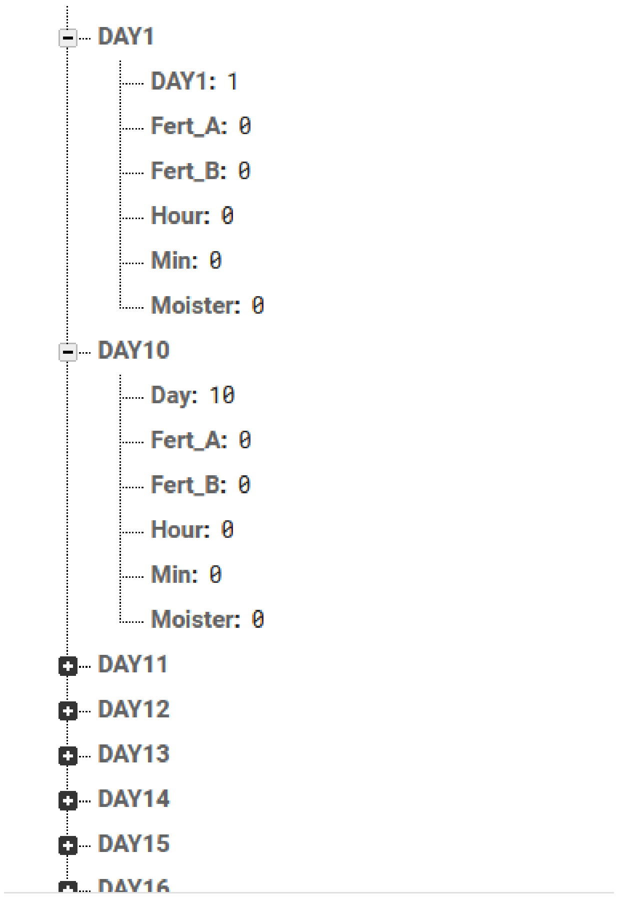
Submitted to:	A. Sunpong Thanok.
Submitted by:	Abdur R. 6128137
	Taksin D. 6121007
	Rajay 6018127

Abstract

This system can take care of a vegetable for the length of its growing life, the user has to set the parameters before using for each day and after setting the parameters for the whole 38 days the user can toggle the Auto mode to start the operation for the whole 38 days, the user can monitor the live data of moister sensor at all time from either the mobile app or the web application that we designed. Configurable parameters are desired moister level and 2 different types of fertilizers. There is an manual mode for the user to test the other hardware like the pump or solenoid valve.

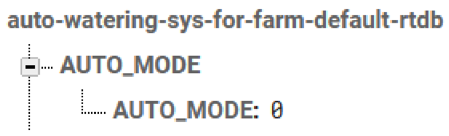
Data server (Firebase)

Since any application weather web or mobile, needs a place to store all the data, for our project we chose firebase to store al the data, the mobile app, and the web app, as well as the controller can read from fire base.



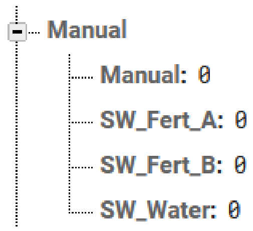
```
DAY1
    DAY1: 1
    Fert_A: 0
    Fert_B: 0
    Hour: 0
    Min: 0
    Moister: 0
DAY10
    Day: 10
    Fert_A: 0
    Fert_B: 0
    Hour: 0
    Min: 0
    Moister: 0
DAY11
DAY12
DAY13
DAY14
DAY15
DAY16
```

We made parent for every single day, and we made child for every configurable parameter, so that the user can write the data here through mobile app. And the controller can read it and put it into effect.



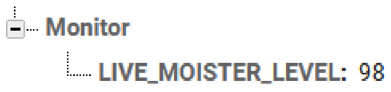
```
auto-watering-sys-for-farm-default-rtdb
  AUTO_MODE
    AUTO_MODE: 0
```

To trigger the whole 38 days operation, we the user can set this child to high.



```
Manual
  Manual: 0
  SW_Fert_A: 0
  SW_Fert_B: 0
  SW_Water: 0
```

How ever for the manual mode we have made a parent with the needed child, it is completely different then the automatic mode datas.

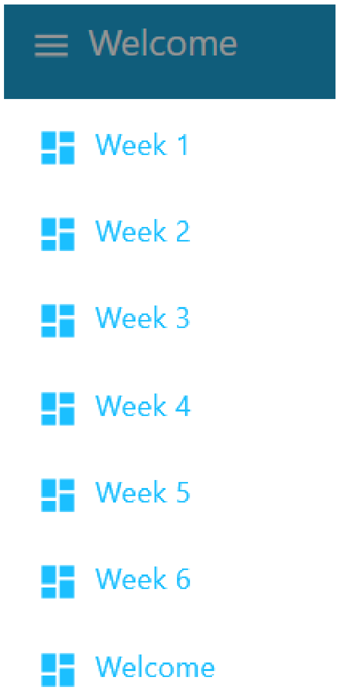


```
Monitor
  LIVE_MOISTER_LEVEL: 98
```

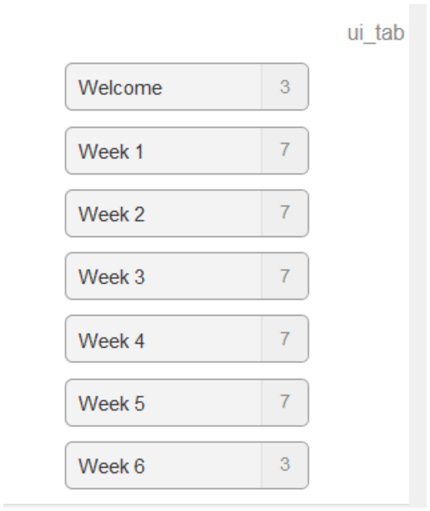
And for the data from the moister sensor that the controller reading, we have made a child so the controller will keep updating it every 1 second.

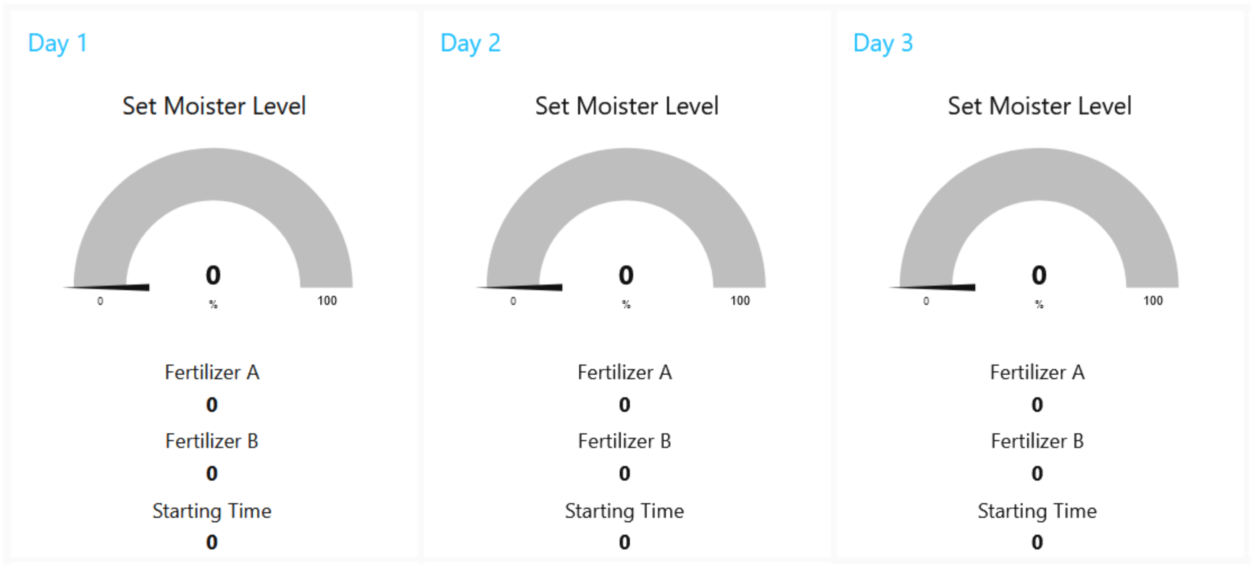
Node red

We also have designed a web application using Node-red, the user can see all the detail of the 38 days of operation in the web app, for example how much of each fertilizer will be applied or what is the desired moister level for a specific day etc. as well as they can monitor the current moister level and the status of auto mode weather it is on or not.

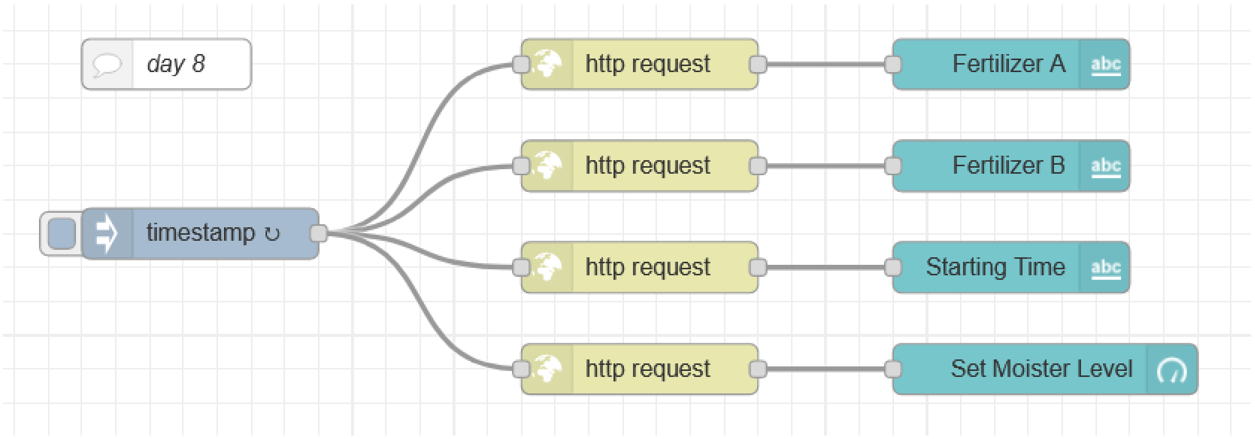


To design different layers, I have created using the ui_tab in settings

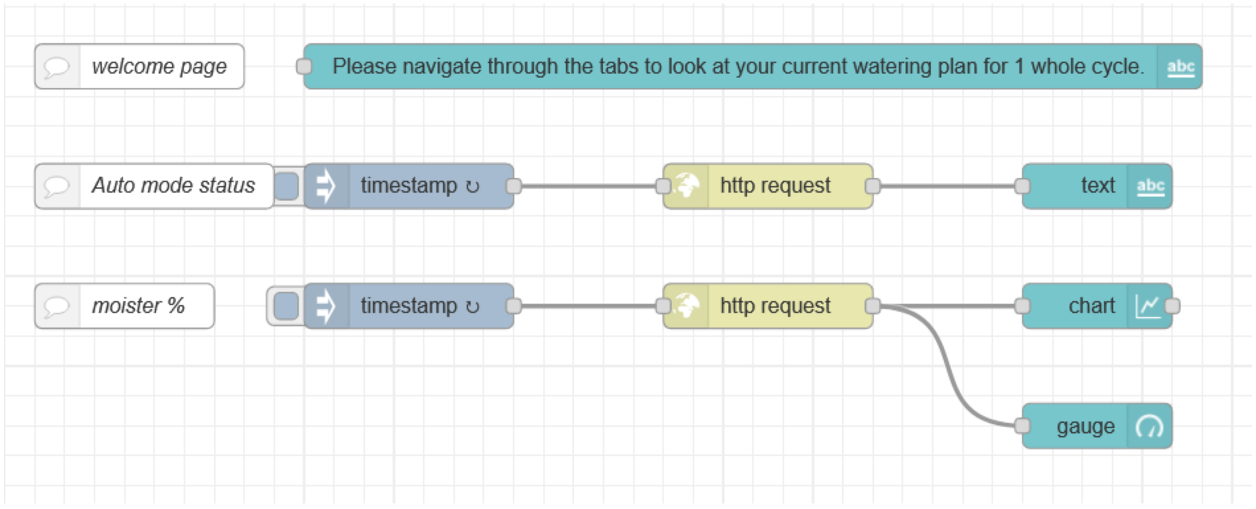




As you can see in the figure above, you are able to check all the 4 parameters.



To read those 4 parameters from firebase I have used the function Http request, used inject to trigger http request function, and to display them in the User interface I have used Text and gauge from dashboard. And repeat this exact process for all the other days.



To design those data reading firebase, I have used the function Http request, I used inject to trigger http request function, and to display them in the User interface I have used Text and gauge and chart from dashboard. As you might notice that the moister data is being transmitted parallelly to both the gouge and the chart.

Android Studio

We have designed for our android studio code for Menu Screen, Monitor Screen, Congfiguration Screen, and Manual Screen. For four screens, we create four classes for each. In the Menu Screen, we will see config button, manual button, monitor button, setting button, and last ON/OFF button for automation mode. Menu Screen is the following given in figure1.

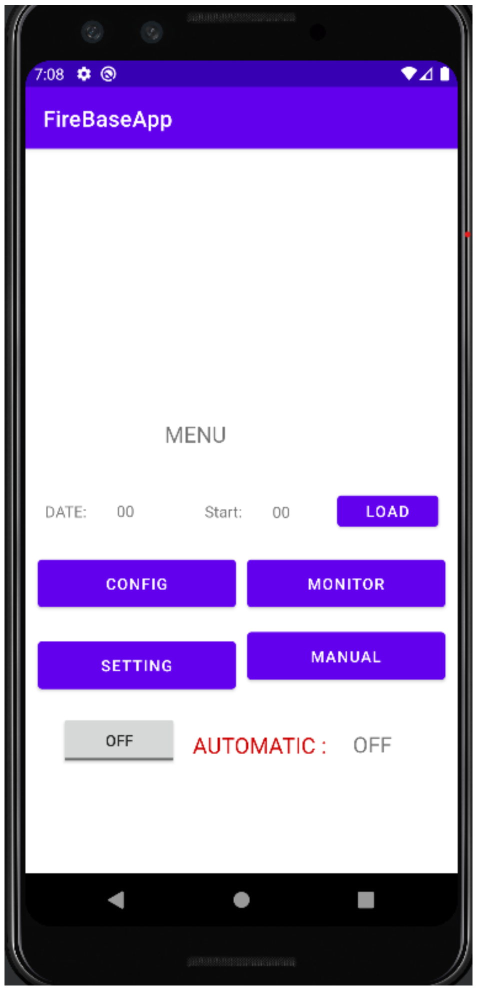
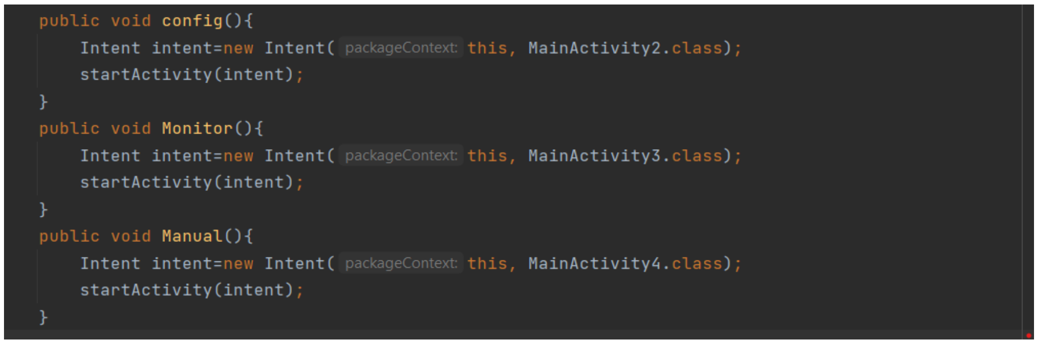


Figure: 1

For Menu screen, MainActivity.java class within class we create some other method which are config method to call MainActivity2.java class for config Screen, monitor method for MainActivity3.java class for Monitor screen and last method is Manual is for MainActivity4.java class which is Manual screen.

Here code is some methods is giving the following figure2:



```
public void config(){
    Intent intent=new Intent( packageContext: this, MainActivity2.class);
    startActivity(intent);
}
public void Monitor(){
    Intent intent=new Intent( packageContext: this, MainActivity3.class);
    startActivity(intent);
}
public void Manual(){
    Intent intent=new Intent( packageContext: this, MainActivity4.class);
    startActivity(intent);
}
```

Figure:2

In CONFIG Screen, we will see data for Day, Time, moisture, NPK46(cc): and NPK15(cc) and three other buttons which are Ok, LOAD and BACK. In DAY, we can choose day 1 to 38 days, BACK button is going back to the Main screen which is Menu screen. When we click the LOAD Button, we will get data from firebase. The design for config screen is as follows given in figure3:

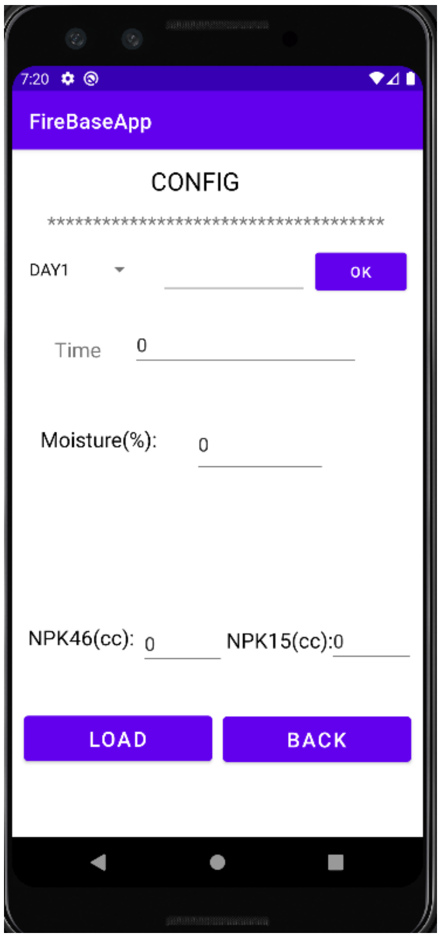
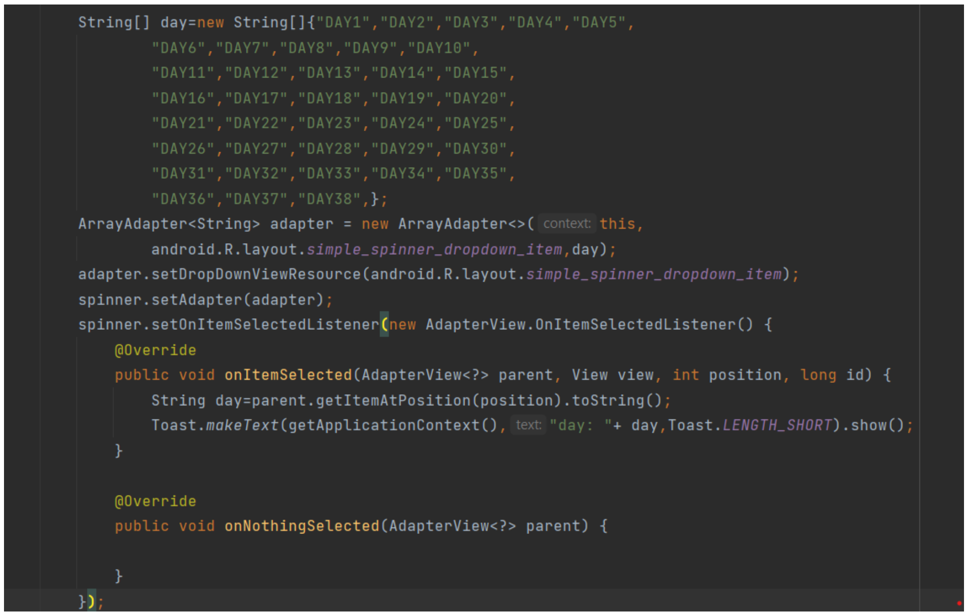


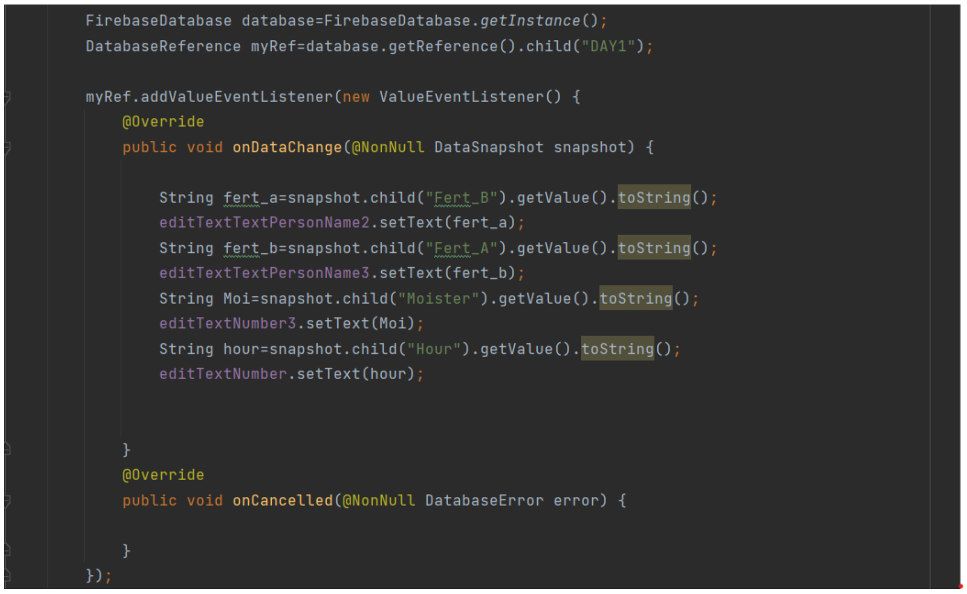
Figure:3

In Config Screen, for day data storage we use spinner for day1 to 38, we can select day by clicking spinner. The following given figure is how we store data in code for spinner to get day data.



```
String[] day=new String[]{"DAY1","DAY2","DAY3","DAY4","DAY5",
        "DAY6","DAY7","DAY8","DAY9","DAY10",
        "DAY11","DAY12","DAY13","DAY14","DAY15",
        "DAY16","DAY17","DAY18","DAY19","DAY20",
        "DAY21","DAY22","DAY23","DAY24","DAY25",
        "DAY26","DAY27","DAY28","DAY29","DAY30",
        "DAY31","DAY32","DAY33","DAY34","DAY35",
        "DAY36","DAY37","DAY38",};
ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this,
        android.R.layout.simple_spinner_dropdown_item,day);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String day=parent.getItemAtPosition(position).toString();
        Toast.makeText(getApplicationContext(), text: "day: "+ day,Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

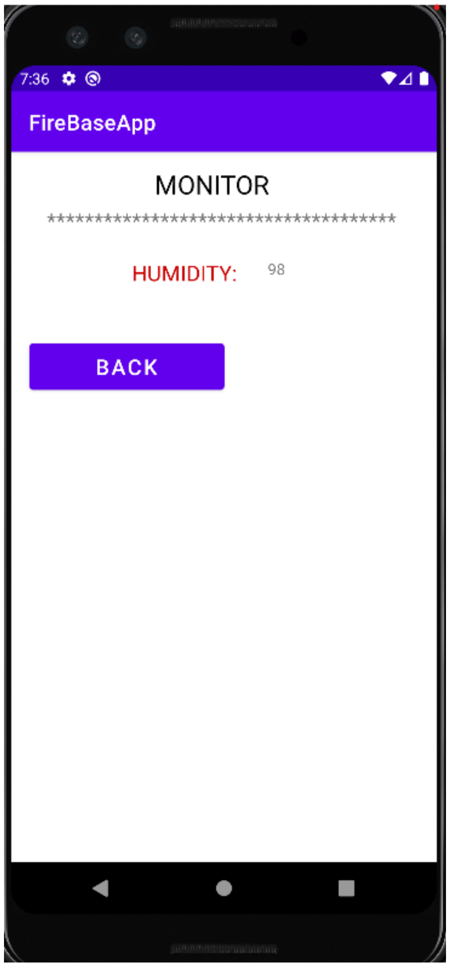
Day data for spinner



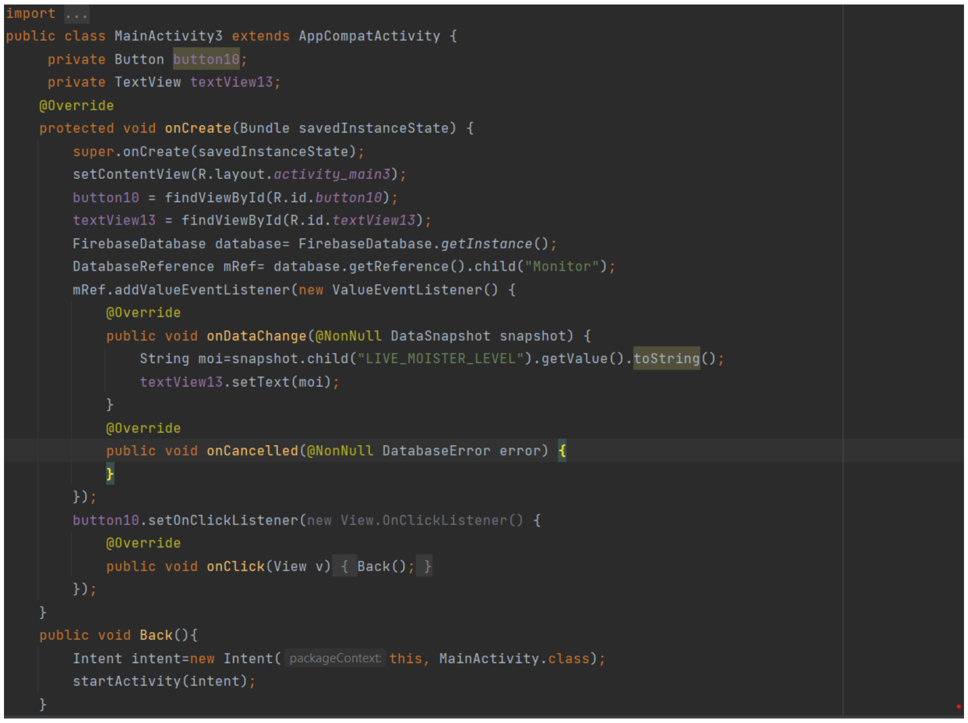
```
FirebaseDatabase database=FirebaseDatabase.getInstance();
DatabaseReference myRef=database.getReference().child("DAY1");

myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {

        String fert_a=snapshot.child("Fert_B").getValue().toString();
        editTextTextPersonName2.setText(fert_a);
        String fert_b=snapshot.child("Fert_A").getValue().toString();
        editTextTextPersonName3.setText(fert_b);
        String Moi=snapshot.child("Moister").getValue().toString();
        editTextNumber3.setText(Moi);
        String hour=snapshot.child("Hour").getValue().toString();
        editTextNumber.setText(hour);


    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Firebase Code

Next, for Monitor Screen we design as a simple GUI with one button and Text and Text View. The following given Figure is GUI for Monitor Screen.



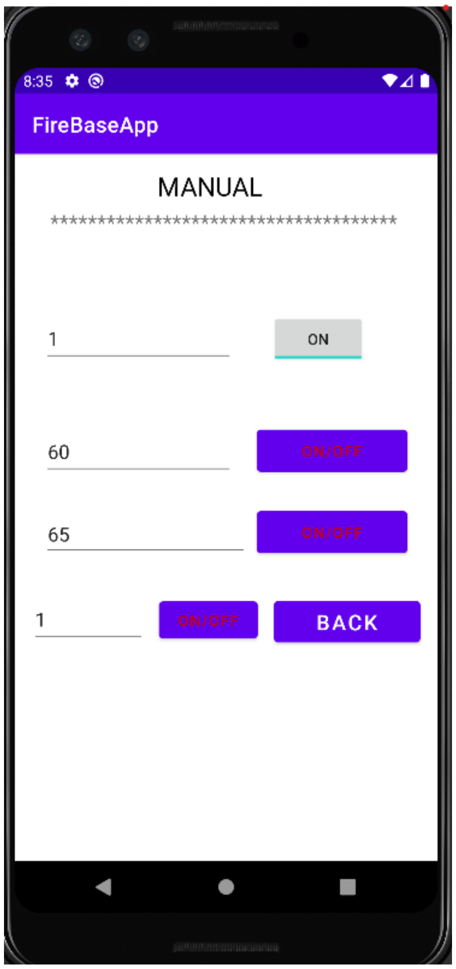
In this screen, we display data of HUMIDITY data from firebase in text View and Back button is to go back to Main Screen that is Menu Screen. Android Code for Monitor Screen is given as follows in the figure.



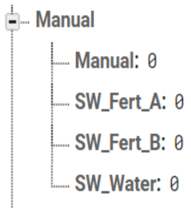
```
import ...
public class MainActivity3 extends AppCompatActivity {
     private Button button10;
     private TextView textView13;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
        button10 = findViewById(R.id.button10);
        textView13 = findViewById(R.id.textView13);
        FirebaseDatabase database= FirebaseDatabase.getInstance();
        DatabaseReference mRef= database.getReference().child("Monitor");
        mRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                String moi=snapshot.child("LIVE_MOISTER_LEVEL").getValue().toString();
                textView13.setText(moi);
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
            }
        });
        button10.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { Back(); }
        });
    }
    public void Back(){
        Intent intent=new Intent( packageContext: this, MainActivity.class);
        startActivity(intent);
    }
```

Within MainActivity3.java class we create another method which is called Back () that function is go for back button. In ("Button10. SetOnClickLister ")function, we call directly back () method to get action on button.

Last Screen, which is Manual screen, we display data that are SW_Fert_A, SW_Fert_B, SW_water frorm Realtime database. When we click toggle button ON, we get data from firebase and toggle button is OFF, we will not see any data in manual screen expect first edit text view is 0.



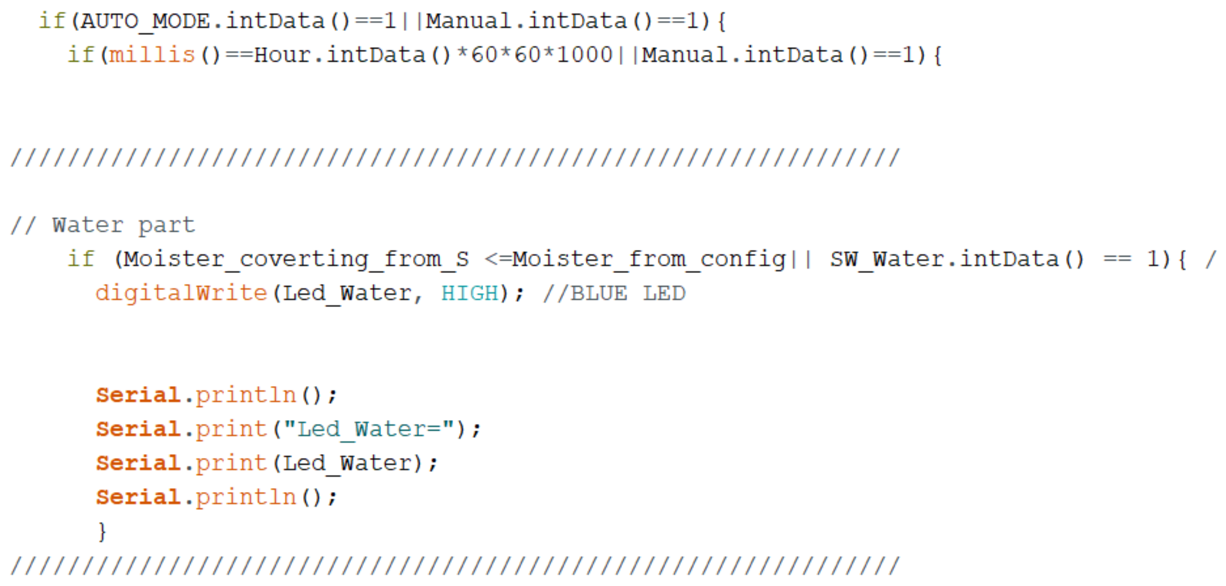
Manual Screen



```
Manual
    Manual: 0
    SW_Fert_A: 0
    SW_Fert_B: 0
    SW_Water: 0
```

Data for Manual Screen

Arduino IDE Coding



```
  if(AUTO_MODE.intData()==1||Manual.intData()==1){
    if(millis()==Hour.intData()*60*60*1000||Manual.intData()==1){


//////////////////////////////////////////////////////////////

// Water part
    if (Moister_coverting_from_S <=Moister_from_config|| SW_Water.intData() == 1){ /
      digitalWrite(Led_Water, HIGH); //BLUE LED


      Serial.println();
      Serial.print("Led_Water=");
      Serial.print(Led_Water);
      Serial.println();
      }
//////////////////////////////////////////////////////////////
```

The operation will be activated when Auto mode is equal to 1 or Manual is equal to 1. Then operation we see if the time has reached what we have configured or not. If it is, then it will continue the next step and if manual equal to 1 it will not check the time that we set. It is the big loop where it covers water part, day part, fertilizer A & B part.

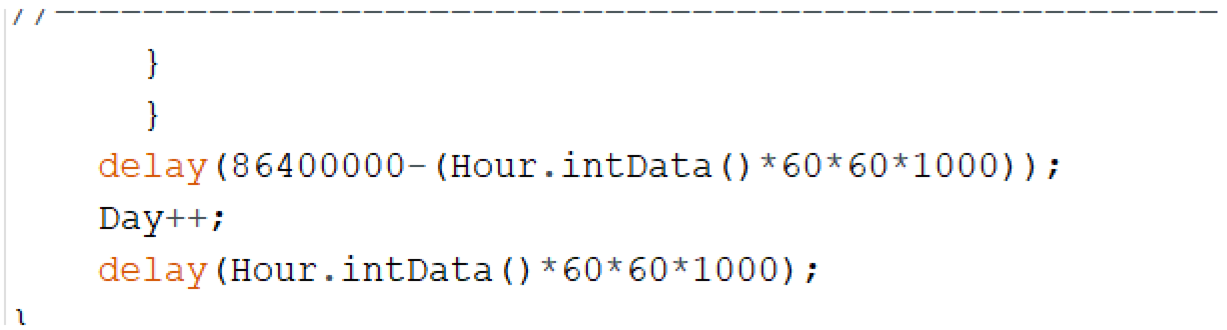
The water part will begin when the moister sensor reached the level of what we have configured. It will stop once it reached or beyond the configured moister level. We can also use water switch to activate the valve by making voltage equal to high.

```
Fert_A_time_converting = Fert_A_time*60000; // make it time 60seconds
    if(Fert_A_time_converting!=0||SW_Fert_A.intData()==1){
      digitalWrite(Led_Fert_A, HIGH); // GREEN LED

      Serial.println();
      Serial.print("Led_Fert_A=");
      Serial.print(Led_Fert_A);

      Serial.println();
      Serial.print("Fertilizer A time =");
      Serial.print(Fert_A_time_converting);
      delay(Fert_A_time_converting);
      Fert_A_time_converting=0;
      }
```

For fertilizer A to begin it will need to convert the configured time in fire base in millisecond time. Then it will check if it is not equal to. The valve will be activated once it reached a designated value of time it will stop, and the value of time will be equal to 0 so that it won’t be operate again.

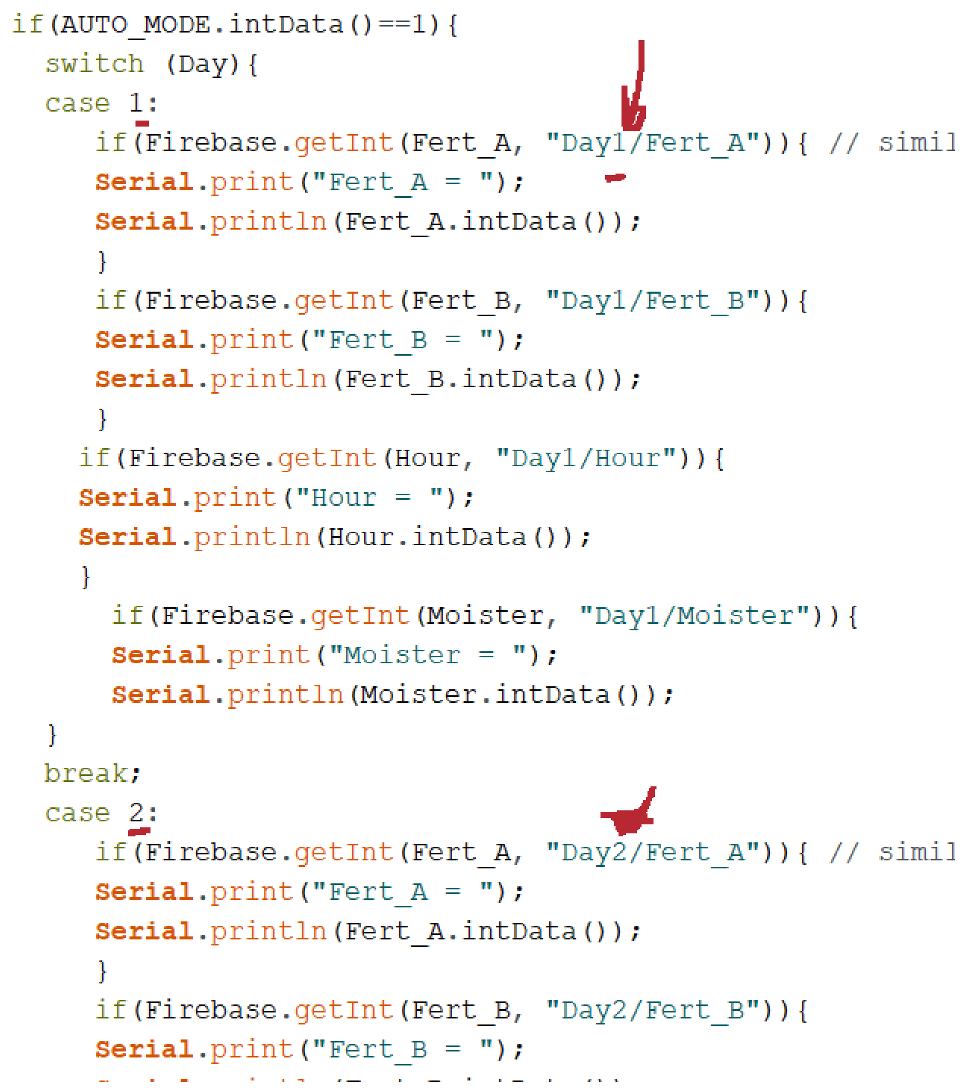
```
// Fertilizer B

Fert_B_time_converting = Fert_B_time*60000; // make it time 60seconds
      if(Fert_B_time_converting!=0||SW_Fert_B.intData()==1){
      digitalWrite(Led_Fert_B, HIGH); // GREEN LED

      Serial.println();
      Serial.print("Led_Fert_B=");
      Serial.print(Led_Fert_B);
      Serial.print("Fertilizer B time =");
      Serial.print(Fert_B_time_converting);
      delay(Fert_B_time_converting);
      Fert_B_time_converting=0;
      }
```

The concept is the same as the fertilizer A.



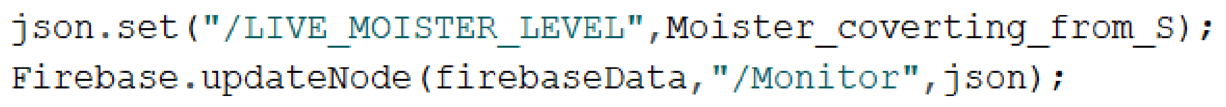
```
//------------------------------------------------------------
          }
          }
      delay(86400000-(Hour.intData()*60*60*1000));
      Day++;
      delay(Hour.intData()*60*60*1000);
}
```

After all the operation is done, we set the delay time shown above. We set Day = 1 as first so that we can receive from Data1. The first delay is to make the day to start a new day like 00.00AM. Then the value day will increase by 1 once it changes the day the second delay will be change into the set alarm.



```
if(AUTO_MODE.intData()==1){
  switch (Day){
  case 1:
     if(Firebase.getInt(Fert_A, "Day1/Fert_A")){ // simil
     Serial.print("Fert_A = ");
     Serial.println(Fert_A.intData());
     }
     if(Firebase.getInt(Fert_B, "Day1/Fert_B")){
     Serial.print("Fert_B = ");
     Serial.println(Fert_B.intData());
     }
    if(Firebase.getInt(Hour, "Day1/Hour")){
    Serial.print("Hour = ");
    Serial.println(Hour.intData());
    }
      if(Firebase.getInt(Moister, "Day1/Moister")){
      Serial.print("Moister = ");
      Serial.println(Moister.intData());
  }
  break;
  case 2:
     if(Firebase.getInt(Fert_A, "Day2/Fert_A")){ // simil
     Serial.print("Fert_A = ");
     Serial.println(Fert_A.intData());
     }
     if(Firebase.getInt(Fert_B, "Day2/Fert_B")){
     Serial.print("Fert_B = ");
```

Once the value of the data 1 increased, it will change into other cases which we can access to other configured day value.



```
json.set("/LIVE_MOISTER_LEVEL",Moister_coverting_from_S);
Firebase.updateNode(firebaseData,"/Monitor",json);
```

The moister level we detected from the sensor will be sent to firebase Data through this code.