SMART PLANT CARE SYSTEM FINAL REPORT

by

BERIL AYGÜL, 119202033
DENIZ GÜNENÇ, 120200078
FATMA HILAL BÖRKLÜ, 119200087
MELIH FARUK ÇETIN, 119202070
RESUL ERDEM ARDUÇ, 119200056
SELIN YILMAZ, 119203057

Supervised by

ELIF PINAR HACIBEYOĞLU

Submitted to the
FACULTY OF ENGINEERING AND NATURAL SCIENCE
in partial fulfillment of the requirements for the

Bachelor of Science

in the
COMPUTER ENGINEERING & ELECTRICAL AND ELECTRONICS ENGINEERING

01.01.2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Aim of The Project

This system is planned for the people who have plants in their homes but do not have enough time to take care of them every day so the plants can be taken care of without human supervision. The intended features of the system are as follows:

- Location information of the plants will be captured with the help of GPS and with the captured location, sunrise and sunset times and the weather forecast will be obtained over the internet via an API. Curtains or blinds will be opened according to the light source and the weather condition.
- A heat sensor is placed on the pot and it measures the temperature of the environment where the plant is located and gives the user feedback accordingly.
- A moisture sensor device will be used to measure the soil's moisture rate on a regular basis, and this method will be used to carry out the necessary watering operation.
- The GPS location, environment temperature, soil moisture level, weather conditions for the day, and times for sunrise and sunset will be seen by the user through the mobile application.

The main purpose of this project is to care for the plant in the medium term, not the long term. Arduino will be used.

2 Purpose and Importance of The Project

The purpose of the smart plant caring project is to continue to care for and safeguard the plants that are absolutely essential to human life. The user may quickly install and maintain intelligent plant feeding, including all plant, sensor, and irrigation data. All information about plants, their growth, and health are always available for viewing.

The escalating speed of life often prevents people from being able to give their plants the care they need and require, leading to the need for a software system that can maintain plant care, particularly when the person is away on business or vacation. The balance of heat, temperature, and humidity will be ensured, the lifespan of the plants will be extended, and the protection and care of the plants will be made easier, thanks to the artificial intelligence that has been developed. The project will also benefit mobile devices because it facilitates automation. In this approach, protecting plants will be simpler, and people will favor automation in general. As a result plant feeding will become more common.

The correct amount of light must be provided to the plants, so blinds or curtains must be controlled. The blinds will only open when you use the app and stay closed at night because it will be hazardous for them to be open all day. In this approach, undesirable events like theft will be avoided. Additionally, when soil moisture levels are too high or too low, the rate of soil moisture is crucial because the plant may perish. It was decided that using a moisture sensor to control this was appropriate. When required, sufficient water support will be given to the plants.

3 Roles in Group

The tasks in the project have been distributed to team members according to their interests, with 4 CMPE students and 2 EEEN students comprising the team.

- Project Manager: Fatma Hilal Börklü
- Devs: Resul Erdem Arduç, Deniz Günenç, Fatma Hilal Börklü, Selin Yılmaz
- Designers: Melih Faruk Çetin, Beril Aygül
- Writers: Deniz Günenç, Selin Yılmaz
- Tester: Deniz Günenç

Table 1: Roles In Project

Activity	**Definition**	**Who is responsible?**
Project manager	Responsible for the overall administration and management of the project.	Fatma Hilal Börklü
Defining the project and preparing the proposal	Responsible for researching scientific articles about the project topic.	Beril Aygül, Deniz Günenç, Fatma Hilal Börklü, Melih Faruk Çetin, Resul Erdem Arduç, Selin Yılmaz
Design of electronic parts	Designing of hardware and mechatronic applications.	Beril Aygül, Melih Faruk Çetin
Coding and testing	Writing Arduino code and testing the modules.	Deniz Günenç, Fatma Hilal Börklü, Resul Erdem Arduç, Selin Yılmaz

4 Project Timeline

Table 2: Project Timeline

Activities/Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13
Topic Selection													
Literature Review													
Design													
Coding													
Model Making													
Testing													
Reporting													

- Topic Selection: The team is formed and after some discussions, the topic of the project is decided as a smart plant caring system.
- Literature Review: Articles about smart plant caring systems are read to get more ideas to make the project better.
- Design: Activity and use case diagrams of the project are designed.
- Coding: The codes needed to run the project are written.
- Model Making: The necessary materials are provided and the model is created.
- Testing: The codes are tested with the model to see if they are working properly.
- Reporting: The final report of the project is written.

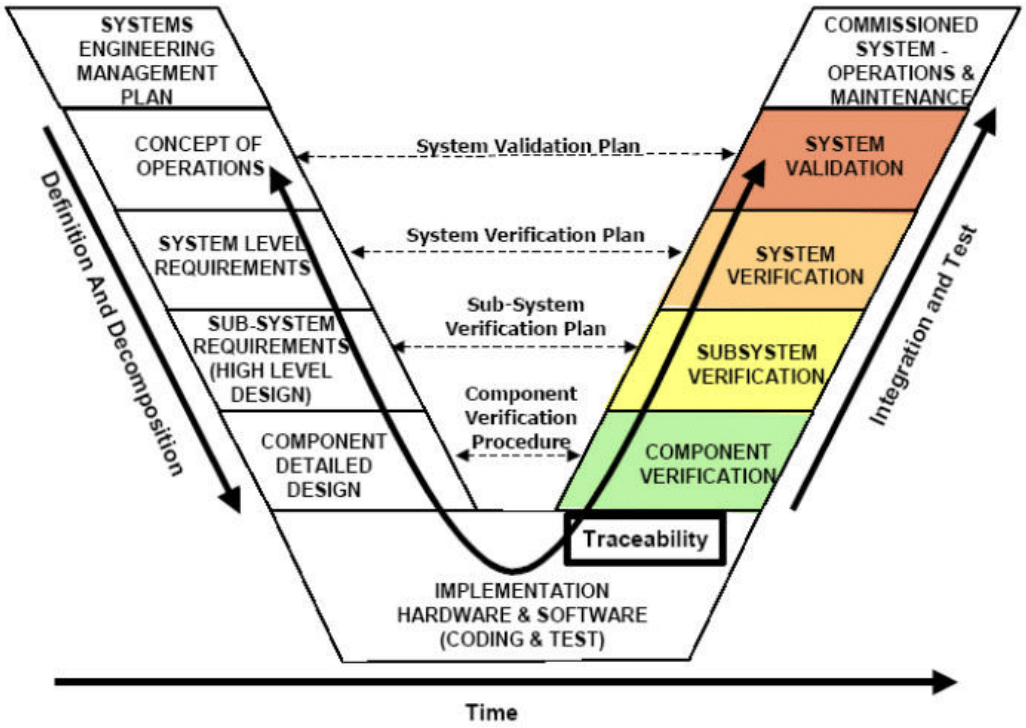

Figure 1: Project Life Cycle Model

The V-shaped model in systems engineering is a development process that follows a specific sequence of steps in order to deliver a system or product. It consists of the following phases:

1. Systems Engineering Management Plan: The team develops a management plan that outlines the overall approach and processes to be used for the development of the system or product.

2. Concept of Operations (ConOps): The team defines the concept of operations for the system or product, outlining how it will be used and the goals it is intended to achieve.

3. System Level Requirements: The team defines the high-level requirements for the system or product, including functional, performance, and interface requirements.

4. Sub-System Requirements: The team defines the requirements for each subsystem or component of the system or product.

5. Component Design: The team designs the individual components or sub-systems of the system or product.

6. Implementation (Hardware/Software): The team implements the hardware and software components of the system or product.

7. Component Verification: The team verifies that each component or sub-system of the system or product functions as intended.

8. Subsystem Verification: The team verifies that the sub-systems or components of the system or product function correctly when integrated together.

9. System Verification: The team verifies that the entire system or product functions correctly and meets the requirements.

10. System Validation: The team validates that the system or product functions correctly in the intended operational environment.

11. Commissioned System Operations & Maintenance: The system or product is deployed to production and the team provides ongoing maintenance and support.

This process helps ensure that the system or product is developed and tested in a systematic and thorough manner and that it meets the requirements and functions as intended when it is deployed.

5 Requirement List

Table 3: Requirement Table

Identifier	**Priority**	**Requirement**
REQ1	3	Opening or closing the shutters in front of the plant according to the sunrise and sunset times
REQ2	1	Watering the potted plant
REQ3	1	The system's decision to irrigate according to the humidity of the potted plant to be irrigated
REQ4	1	The system measures the humidity and temperature of the potted plant to be watered
REQ5	2	Seeing the humidity and temperature data of the plant via a mobile app
REQ6	3	Obtaining sunrise and sunset times over the internet according to the location of the plant
REQ7	2	Transferring the humidity and temperature data of the plant to the internet

6 Detailed design

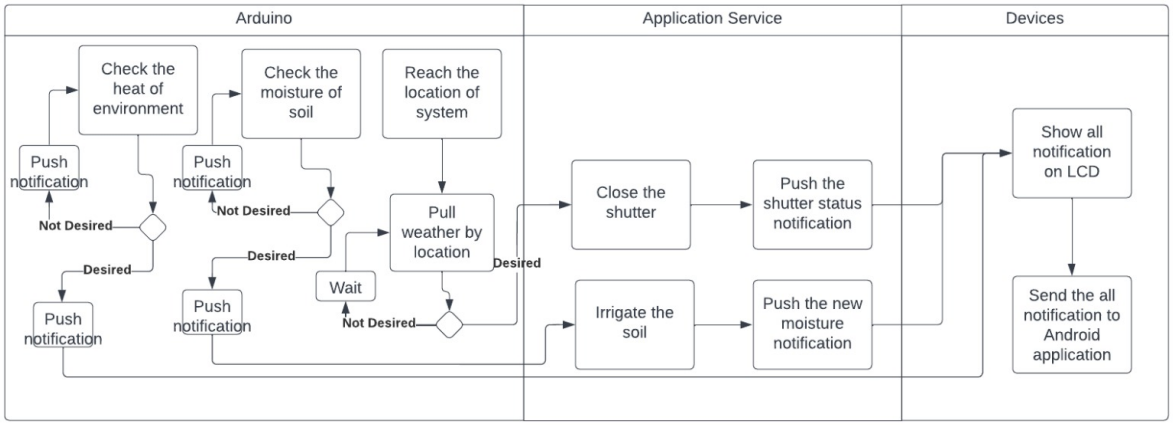

Figure 2: Activity Diagram

The project's activity diagram is shown in the image above. The temperature and humidity are first reached by the mechanism and the code of that here A. It subsequently transmits the accessed data to the LCD display that can be seen in its code B and it sends data to API and its code C then the mobile application will reach data D. The temperature and humidity are examined to see if they are appropriate for the project. The system checks again until the values are appropriate if the values are not appropriate for the project. But if appropriate, it will irrigate automatically by using this method E. Through the mobile application, the location is discovered on F. On the other side, the data is utilized to determine the weather and the times of sunset and sunrise and its code is available at G. Then Arduino will get these data from API. H is the code for reaching data. The mechanism moves the curtain open and shut in accordance with these times. The curtain mechanism runs as it seems in this method I.

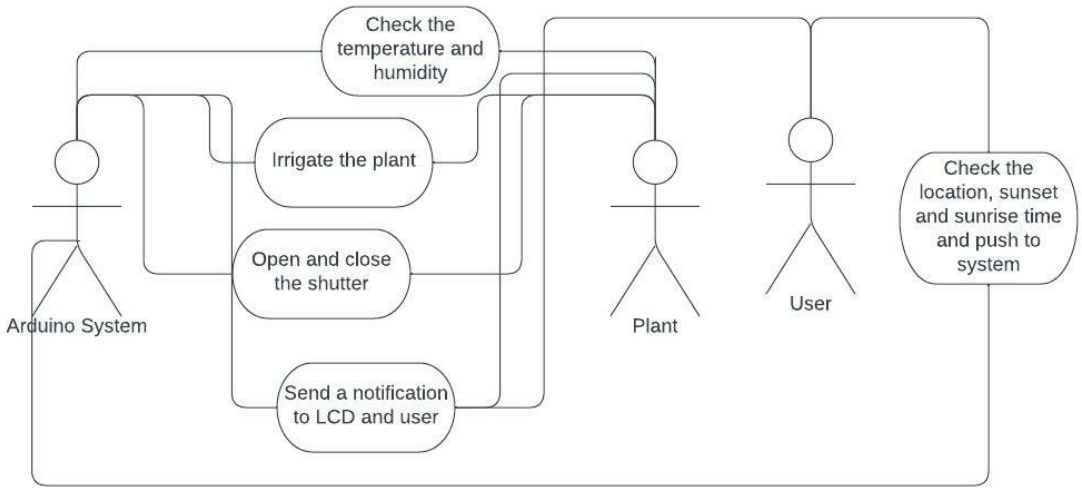

Figure 3: Use Case Diagram

A use case diagram represents a visual depiction of a system that monitors temperature and humidity, displays this information on an LCD screen and alerts the user, and automatically irrigates the plant based on humidity levels. The system also receives location information from a mobile application when the user is near the plant and adjusts the curtains accordingly.

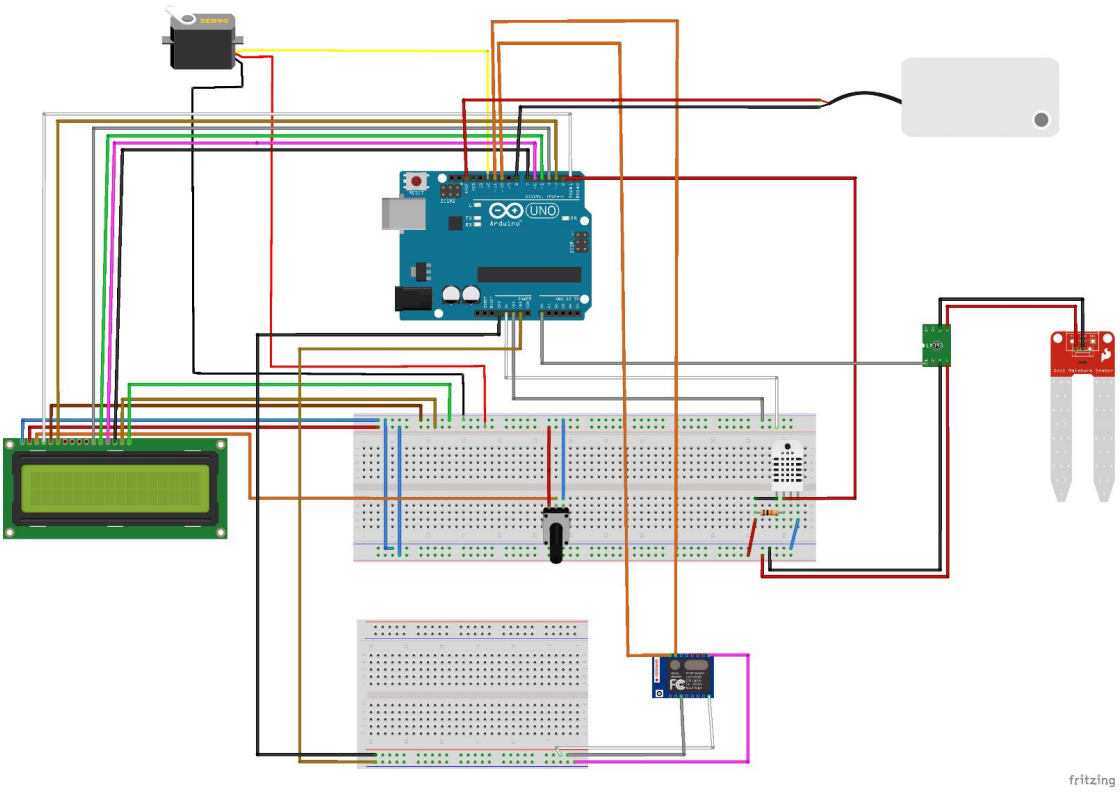

Figure 4: Arduino Design

Since Arduino will be used in the project, it is necessary to first design the Arduino. The Arduino, as seen above, has a servo motor that will turn the shutter. The servo motor will operate in accordance with the sunrise and sunset times provided by the mobile application. In addition, there is a water pump to re-moisten the water when the humidity drops. The temperature and humidity sensors, which are central to the project, are shown on the far right of the figure. After the data from the sensors are processed in the Arduino, it will power the water engine according to the situation. Finally, an LCD screen displays the received data.

7 Components of The Project

1. Arduino Uno

   Arduino processed the information to be received from the plant with the help of sensors throughout the project. It determined if the plant needs water, and if it does, it actuated the diver pump and irrigated the plant. It made curtains open and close in the house at sunrise and sunset times. In this way, it ensured that the plant receives the sunlight it needs. At night, when there was no daylight, it closed the blinds and kept the house safe. At the same time, information was sent to the mobile application with the help of the WiFi module, and the system was manually controlled remotely.

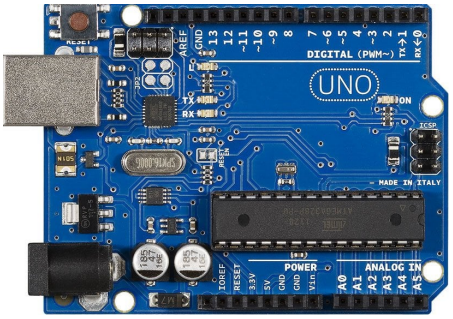

Figure 5: Arduino Uno

2. WiFi Module

   The WiFi module ensured connecting the system to the internet and enabling the user to access all information remotely.

Figure 6: ESP8266 WiFi Module

3. Moisture Sensor

   A moisture sensor is a sensor that the user can use to measure the amount of moisture in the soil or the level of a small liquid. The moisture meter was used by immersing it in the medium to be measured. It was decided whether to give additional water to the plant according to the humidity of the soil.

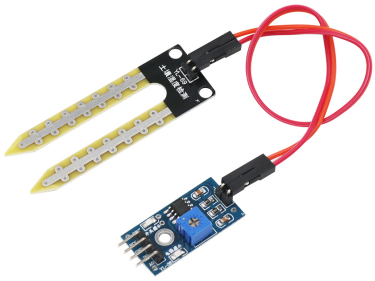

Figure 7: Soil Moisture Sensor

4. Temperature Sensor

   The temperature sensor was used by immersing it in the ground. The information received about the temperature of the soil is checked and a decision is made on whether to open or close the blinds.

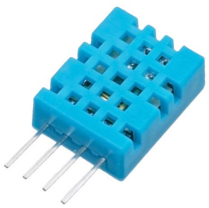

Figure 8: DHT11

5. LCD Display

   LCD screen ensures to inform the user about humidity and temperature.

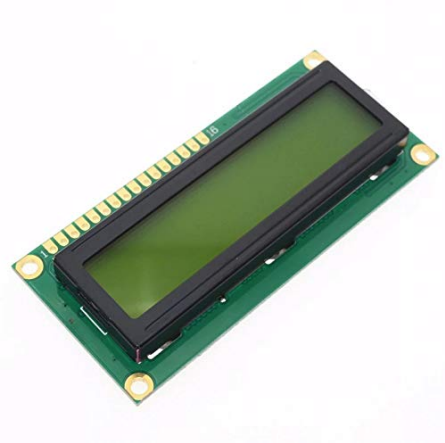

Figure 9: LCD Display

6. Breadboard

   The whole system was set up on the breadboard. Breadboard provides the communication between the Arduino and the rest of the system.

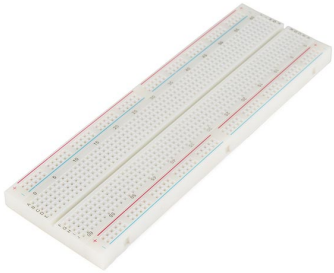

Figure 10: Breadboard

7. Jumper Cable

   A jumper is a small metal connector used to close or open part of an electrical circuit. A jumper has two or more ports regulating an electrical circuit board. Jumper cables are used to establish a data and power connection between the breadboard and the Arduino, or between the sensors and the Arduino.

Figure 11: Jumper Cable

8. Mini Diver Pump

   The mini diver pump provides the necessary flow for the plant to carry the water it needs from the water tank to the flower pot when necessary.

Figure 12: Mini Diver Pump

9. Water Pipe

   The water pipe is responsible for carrying the water needed by the plant to the pot.

Figure 13: Water Pipe

10. Servo-motor

    Servo-motor enables the curtains to be opened and closed.

Figure 14: SG90 Servo Motor

8 Tests

Table 4: Test Cases and Expected Results

Test Number	Test Name	Test Type	Expected Result	Result
1	GPS	Unit	Show the location on console	Pass
2	Heat Sensor	Unit	Show the temperature on console	Pass
3	Moisture Sensor	Unit	Show the moisture on console	Pass
4	Curtain Control	Integration	Opens and closes curtains	Pass
5	LCD	Integration	Shows the current environment's status	Pass
6	Water Pump	Integration	Pumps water whenever plant needs	Pass
7	Mobile App	End to end	Shows the weather information Shows the moisture level Shows the heat level Shows the GPS longitude and latitude Shows the sunrise and sunbath times	Pass
8	V&V	Acceptance	Verification and Validation	Pass

1. The GPS component was tested to ensure its correct functioning by printing data from the component to the console. If the data was close to the actual location, the test was considered passed. This was a unit test because it did not require connection to other components and only used the GPS component.

2. The Heat Sensor was tested to ensure its ability to accurately measure the temperature of the environment. The data from the sensor was compared to the actual temperature of the room or place, and the data was printed to the console. This was a unit test because it did not require any other components in the project, only the heat sensor.

3. The Moisture Sensor was tested to determine its accuracy in measuring the moisture level in the soil of the plant. Data from the sensor was printed to the console. This was also a unit test because it only depended on the moisture sensor and did not require any other components.

4. The Curtain Control was tested to ensure that it opened and closed the curtain according to data from other components. The curtain was connected to a servo motor, which was also tested. To pass this test, the curtain had to open when the sun rose and the weather was sunny, and it had to close during all other conditions. This was an example of an integration test because it required data from other components to determine whether to open or close the curtain.

5. The LCD was tested to ensure that it displayed data from other components on its screen. It did not matter if the screen displayed the correct temperature or moisture level, only that it was able to receive data from other components. This was an integration test because it required data from other components to function.

6. The Water Pump was tested to ensure that it watered the plant as needed. Because the watering of the plant depended on data from other components, this was an integration test.

7. The Mobile App was tested to ensure the correct functioning of its five features, and its ability to communicate with the Arduino and local host. It was also tested to ensure that it could communicate with the local host. This was an end-to-end test because it required the use of the Arduino and mobile app with a network connection, and required a server to facilitate communication.

8. The final step in testing was the acceptance test, which employed the V&V method (Verification & Validation). During this test, the entire project was checked to ensure its correct functioning and conformity to expectations. The project was used as a daily user would, and tested to ensure that it met its intended purpose. Verification involved asking whether the project was built as intended, and validation involved asking whether the project met the needs of the user.

9 Future Work

This chapter will be focused on potential changes that can be addressed in future work.

- The performance of the project can also be improved in terms of the speed of operation and memory, by using another high-end controller
- The water pump system irrigates the same amount of water for each plant. Therefore, the current system can be extended to control the irrigation process by choosing a suitable amount of water for plants.
- The testing process was performed in the same place. Testing in different locations can make the project more accurate.
- The current mobile application design can be more developed.

10 Useful Links

- GitHub Project Page: This page provides access to the source code, documentation, and other materials related to the Smart Plant Caring System project. (https://github.com/SeaBenSea/Smart-Plant-Caring-System)
- Thingspeak Data Preview Page: This page allows users to view the data collected by the Smart Plant Caring System, including temperature, humidity, GPS coordinates, and more. (https://thingspeak.com/channels/1985536)
- Thingspeak Last 5 Data: This page provides access to the most recent data points collected by the Smart Plant Caring System, in JSON format. (https://api.thingspeak.com/channels/1985536/feeds.json?results=5)

References

[1] Starting Electronics. (2017, May 11). Which Arduino for Beginners. https://startingelectronics.org/articles/arduino/which-arduino-for-beginners/

[2] Waldo, W.G. (2013). Chapter 20 - Managing Embedded Software Development. In Software Engineering for Embedded Systems (pp. 671-730).

[3] Mucit Pilot. (2020, Sep 15). Arduino ile ESP8266 Kullanımı ve İnternet Erişimi Serisi. https://mucitpilot.blogspot.com/2020/09/arduino-ile-esp8266-kullanm-ve-internet.html

A Code of Sensors

```
void read_values() {                    //Read values from sensors
  sensorValue = analogRead(soilPin);    //Read and set sensor value of humidty
  temp = dht.getTemperature();          //Read and set sensor value of
    temperature
  humi = (sensorValue / 1023.0) * 100;  //Set humidity value as percentage
}
```

Code 1: Code of Sensor

B Code of Updating LCD

```
void update_lcd() {      //Updating the texts on the LCD screen
  lcd.setCursor(0, 0);   //Set cursor on starting of the first line
  lcd.print("Temp: ");   //Write Temp:
  lcd.print(temp);       //Write the temperature value that we stored
  lcd.print(" C");       //Write the unit of the temperature
  lcd.setCursor(0, 1);   //Set cursor on starting of the second line
  lcd.print("Humi: ");   //Write Humi:
  lcd.print(humi);       //Write the humidty percentage value that we stored
  lcd.print(" %");       //Write the percentage sign
}
```

Code 2: Code of Updating LCD

C Code of Posting Data From Arduino

```
void sendth(float temp, float humi) {
        //Send stored temperature and humidty percentage values to cloud
  esp.println("AT+CIPSTART=\"TCP\",\"" + ip + "\",80");
        //Connecting to Thingspeak
  if (esp.find("Error")) {
        //Check for the connection error
    Serial.println("AT+CIPSTART Error");
        //LOG ERROR
  }
        //
  String data = "GET https://api.thingspeak.com/update?api_key=";  //Our api
    call address
  data += "&field1=";
        //field 1 (temperature)
  data += String(temp);
        //field 1 (temperature) data
  data += "&field2=";
        //field 2 (humidty)
  data += String(humi);
        //field 2 (humidty) data
  data += "\r\n\r\n";
        //End of the call
  esp.print("AT+CIPSEND=");
        //Sending the length of our data that we will send to ESP
  esp.println(data.length() + 2);
        //Add the length here
  delay(2000);
        //Sleep 2 seconds
  if (esp.find(">")) {
        //Runs the commands inside of the ESP when it is ready
    Serial.println(esp.read());
        //Send the data
    Serial.println(esp.print(data));
        //Send the data
    Serial.println("Data sent");
        //LOG data sent
    delay(1000);
        //Sleep 1 second
  }
        //
  Serial.println("Connection Closed.");
        //LOG Connection closed
  esp.println("AT+CIPCLOSE");
        //Closing the connection with ESP
  delay(5000);
        //Sleep 5 seconds
}
```

Code 3: Code of Posting Data From Arduino

D Code of Getting Temperature and Humidities from Api to Android

```
1     public void getSensors(){
2         String url = "https://api.thingspeak.com/channels//feeds/last.json?
      api_key=";
3         JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(url, null,
      new com.android.volley.Response.Listener<JSONObject>() {
4             @Override
5             public void onResponse(JSONObject response) {
6                 try {
7                     temp.setText(response.getString("field1"));
8                     humi.setText( response.getString("field2"));
9
10                } catch (JSONException e) {
11                    e.printStackTrace();
12                }
13            }
14        }, new Response.ErrorListener() {
15            @Override
16            public void onErrorResponse(VolleyError error) {
17                temp.setText("error ");
18            }
19        });
20
21        RequestQueue requestQueue = Volley.newRequestQueue(this);
22        requestQueue.add(jsonObjectRequest);
23    }
24 }
```

Code 4: Code of Getting Temperature and Humidities from Api to Android

E Code of Irrigation

```
1 void read_values() {                    //Read values from sensors
2   sensorValue = analogRead(soilPin);    //Read and set sensor value of humidty
3   temp = dht.getTemperature();          //Read and set sensor value of
      temperature
4   humi = (sensorValue / 1023.0) * 100;  //Set humidity value as percentage
5 }
```

Code 5: Code of Irrigation

F Code of Getting Location

```
1  private void getLastLocation(){
2
3      if (ContextCompat.checkSelfPermission(this, Manifest.permission.
       ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED){
4
5
6          fusedLocationProviderClient.getLastLocation()
7                  .addOnSuccessListener(new OnSuccessListener<Location>() {
8                      @Override
9                      public void onSuccess(Location location) {
10
11                         if (location != null){
12
13
14
15                             try {
16                                 Geocoder geocoder = new Geocoder(MainActivity.
       this, Locale.getDefault());
17                                 List<Address> addresses = geocoder.
       getFromLocation(location.getLatitude(), location.getLongitude(), 1);
18
19
20                                 double latit = addresses.get(0).getLatitude();
21                                 double longit = addresses.get(0).getLongitude()
       ;
22                                 DecimalFormat twoDForm = new DecimalFormat("
       #.####");
23                                 latit = Double.valueOf(twoDForm.format(latit));
24                                 longit = Double.valueOf(twoDForm.format(longit)
       );
25                                 lati.setText("Lattitude: "+latit);
26                                 longi.setText("Longitude: "+longit);
27                                 getWeatherDetails(longit,latit);
28                             } catch (IOException e) {
29                                 msg4.setText(e.getMessage());
30                             }
31
32
33                         }
34
35                     }
36                 });
37
38
39     }else {
40
41         askPermission();
42
43
```

```
44      }
45
46
47  }
```

Code 6: Code of Getting Location

G Code of Getting Sunrise, Sunset and Weather

```
1
2   public void getWeatherDetails(double longit, double latit) {
3       String url = "https://api.openweathermap.org/data/2.5/weather";
4       String url2 = "https://api.openweathermap.org/data/2.5/weather?lat="+latit+
        "&lon="+longit+"&appid="+appid;
5
6           StringRequest stringRequest = new StringRequest(Request.Method.POST,
        url2, new Response.Listener<String>() {
7               @Override
8               public void onResponse(String response) {
9                   String output = "";
10                  try {
11                      JSONObject jsonResponse = new JSONObject(response);
12                      JSONArray jsonArray = jsonResponse.getJSONArray("weather");
13                      JSONObject jsonObjectWeather = jsonArray.getJSONObject(0);
14                      String description = jsonObjectWeather.getString("
        description");
15                      String mainWeather = jsonObjectWeather.getString("main");
16
17
18                      JSONObject jsonObjectSys = jsonResponse.getJSONObject("sys"
        );
19                      long sunset = Long.parseLong(jsonObjectSys.getString("
        sunset"));
20                      long sunrise = Long.parseLong(jsonObjectSys.getString("
        sunrise"));
21                      String countryName = jsonObjectSys.getString("country");
22                      String cityName = jsonResponse.getString("name");
23                      //msg.setTextColor(Color.rgb(68,134,199));
24
25                      //long unixSeconds = 1372339860;
26  // convert seconds to milliseconds
27                      Date date = new java.util.Date(sunset*1000L);
28                      Date date2 = new java.util.Date(sunrise*1000L);
29  // the format of your date
30                      SimpleDateFormat sdf = new java.text.SimpleDateFormat("HH:
        mm");
31  // give a timezone reference for formatting (see comment at the bottom)
```

```
32                      sdf.setTimeZone(java.util.TimeZone.getTimeZone("GMT+3"));
33                      String formattedDate = sdf.format(date);
34                      String formattedDate2 = sdf.format(date2);
35
36                      msg.setText("Current weather of " + cityName + " (" +
    countryName + ")");
37                      msg1.setText("DescriptionMain: " + mainWeather);
38                      //msg2.setText("Description: " + description);
39                      msg3.setText("Sunrise: " + formattedDate2);
40                      msg4.setText("Sunset: " + formattedDate);
41                      sendJsonPostRequest(longit,latit,formattedDate2,
    formattedDate,mainWeather);
42                  } catch (JSONException e) {
43                      msg.setText(e.getMessage());
44                  }
45              }
46          }, new Response.ErrorListener(){
47
48              @Override
49              public void onErrorResponse(VolleyError error) {
50                  msg.setText(error.getMessage());
51                  Toast.makeText(getApplicationContext(),error.toString().trim(),
    Toast.LENGTH_LONG).show();
52              }
53          });
54          RequestQueue requestQueue = Volley.newRequestQueue(
    getApplicationContext());
55          requestQueue.add(stringRequest);
56
57
58
59  }
```

Code 7: Code of Getting Sunrise Sunset and Weather

H Code of Getting Location Data from Api to Arduino

```
1  void get_data() {
               //For seperating the only important parts of our response data
2    esp.println("AT+CIPSTART=\"TCP\",\"" + ip + "\",80");
               //Connect Thingspeak with TCP
3    if (esp.find("Error")) {
               //Check for the connection errors
4      Serial.println("AT+CIPSTART Error");
               //LOG Error
5    }
               //
```

```
6  String rest = "AT+CIPSEND=90";
           //Create a string for rest API call
7  rest += "\r\n";
           //Add next lines end of the string for the ESP to understand the
     call is ended
8  sendData(rest, 2000, 0);
           //Send data
9  String host = "GET /channels//feeds.json?api_key=&results=1";  //Create
     another string for getting the result
10 host += "\r\n";
           //Add next lines end of the string for the ESP to understand the
     call is ended
11 host += "Host:api.thingspeak.com";
           //Set the host
12 host += "\r\n\r\n\r\n\r\n\r\n";
           //Add next lines end of the string for the ESP to understand the
     call is ended
13 String response = sendData(host, 2000, 0);
           //GET call ( GET /apps/thinghttp/send_request?api_key=
     XXXXXXXXXXXXXXXX Host: Host_server_name )
14
           //
15 int start_index = response.lastIndexOf(':');
           //Find and store last index of the : from response
16 hour_txt = response.substring(start_index - 139, start_index - 137);
           //Get the hour from the response
17 minute_txt = response.substring(start_index - 135, start_index - 134);
           //Get the minute from the response
18 temp_txt = response.substring(start_index - 104, start_index - 99);
           //Get the temperature from the response
19 humi_txt = response.substring(start_index - 87, start_index - 82);
           //Get the humidty percentage from the response
20 longi_txt = response.substring(start_index - 70, start_index - 63);
           //Get the longitude from the response
21 lat_txt = response.substring(start_index - 51, start_index - 44);
           //Get the latitude from the response
22 sunrise_txt = response.substring(start_index - 32, start_index - 27);
           //Get the sunrise time from the response
23 sunset_txt = response.substring(start_index - 15, start_index - 10);
           //Get the sunset time from the response
24 weather_txt = response.substring(start_index + 1, start_index + 2);
           //Get the weather condition from the response
25
           //
26 Serial.println("temp:" + temp_txt);
           //LOG temperature
27 Serial.println("humi:" + humi_txt);
           //LOG humidty percentage
28 Serial.println("longi:" + longi_txt);
```

```
            //LOG longitude
29  Serial.println("lat:" + lat_txt);
            //LOG latitude
30  Serial.println("sunrise:" + sunrise_txt);
            //LOG sunrise time
31  Serial.println("sunset:" + sunset_txt);
            //LOG sunset time
32  Serial.println("weather:" + weather_txt);
            //LOG weather condition
33  Serial.println("Connection Closed.");
            //LOG connection closed
34  esp.println("AT+CIPCLOSE");
            //Close the connection
35 }
```

Code 8: Code of Getting Location Data from Api to Arduino

I Code of Shutter Mechanism

```
1  void open_curtain() {        //For opening the curtain
2    curtain_servo.write(45);   //Rotate the motor counterclockwise
3    delay(2000);               //Keep rotating for 2 seconds
4    curtain_servo.write(90);   //Stop the motor
5    delay(1);                  //Stay stopped
6  }
7
8  void close_curtain() {        //For closing the curtain
9    curtain_servo.write(135);   //Rotate the motor clockwise
10   delay(2000);                //Keep rotating for 2 seconds
11   curtain_servo.write(90);    //Stop the motor
12   delay(1);                   //Stay stopped
13 }
14
15 bool sun() {                                //For checking if the sun is up or
       down
16   float sunrise = sunrise_txt.toFloat();   //Convert the sunrise time to float
17   float sunset = sunset_txt.toFloat();     //Convert the sunset time to float
18   int hour = hour_txt.toInt() + 3;         //Convert the hour to int and add 3
       hours for the timezone
19   int minute = minute_txt.toInt();         //Convert the minute to int
20   float time = hour + (minute / 100.0);    //Calculate the time
21   if (time > sunrise && time < sunset) {   //Check if the time is between
       sunrise and sunset
22     return true;                           //If it is return true
23   } else {                                 //If it is not
24     return false;                          //Return false
25   }                                        //
```

```
26 }
27
28 bool weather() {              //For checking the weather condition
29   if (weather_txt == "1") {  //Check if the weather Clear
30     return true;             //If it is return true
31   } else {                   //If it is not
32     return false;            //Return false
33   }                          //
34 }
35
36 //Functions for the checks
37 void check_curtain() {                    //For checking every needed data and
       deciding the curtain mode
38                                           //Open the curtain if the sun is up
       and the weather is clear
39   if (sun() && weather() && !curtains) {  //Check if the sun is up, the weather
        is clear and the curtain is closed
40     open_curtain();                       //Open the curtain
41     curtains = true;                      //Set the curtain mode to opened
42   } else if (!sun() && curtains) {        //Check if the sun is down and the
       curtain is opened
43     close_curtain();                      //Close the curtain
44     curtains = false;                     //Set the curtain mode to closed
45   }                                       //If the sun is down and the weather
       is not clear, do nothing
46 }
```

Code 9: Code of Shutter Mechanism