A Project Report on

IoT BASED SMART IRRIGATION SYSTEM

Submitted in fulfillment for the award of the degree of

BCA

by

BARATH.G - (18BCA0083)

RUDRAGANESH.R – (18BCA0084)

SUDHARSUN.M – (18BCA0086)

Under the guidance of

Prof. Dr. BRINDHA K

SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VIT, VELLORE.

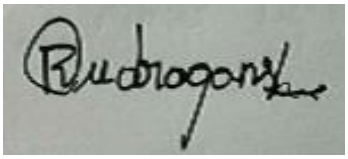
Winter Semester 2020-2021

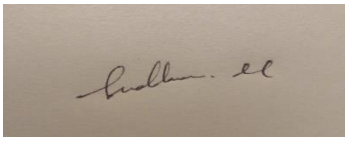
April, 2021

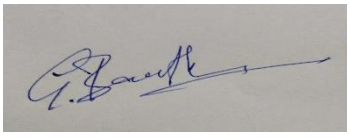
DECLARATION

We here by declare that the thesis entitled "IoT BASED SMART IRRIGATION SYSTEM" submitted by us, for the award of the degree of BCA VIT is a record of bonafide work carried out by us under the supervision of Prof. Dr. Brindha K.

We further declare that the work reported in thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any institute or university.







Place: Vellore

Date: 02/06/2021

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "IoT BASED SMART IRRIGATION SYSTEM" submitted by BARATH G (18BCA0083), RUDRAGANESH R (18BCA0084) & SUDHARSUN M (18BCA0086) School Of Information Technology and Engineering VIT, for the award of the degree of BCA is a record of bonafide work carried out by him/her under my supervision during the period of 12.07.2018 to 02.06.2021, as per the VIT code of academics and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any degree or diploma in this institute or any other institute or university. The Project report fulfils the requirements and regulations of VIT and in my opinion meets the necessary standards for submission.

Place: Vellore

Date: 02/06/2021 **Signature of the Guide**

Internal Examiner **External Examiner**

Head of the Department

Programme

ACKNOWLEDGEMENT

It is our pleasure to express with deep sense of gratitude to Prof. Dr. Brindha K, Associate Professor Grade 1, School Of Information Technology and Engineering, Vellore Institute of Technology, for his/her constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavour. Our associate with him/her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Internet of Things (IoT).

We would like to express my gratitude to Dr. G. Viswanathan, Mr. G.V. Selvam, Dr. Rambabu Kodali, Dr. S. Narayanan, Dr. Balakrushna Tripathy, School Of Information Technology and Engineering, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood We express ingeniously my whole-hearted thanks to Prof. Srinivas Koppu, Associate Professor Grade 1, all teaching staff and members working as limbs of our university for their not-self-centred enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize our course study successfully. We would like to thank our parents for their support.

It is indeed a pleasure to thank our friends who persuaded us to take up and complete this task. At last but not least, we express our gratitude and appreciation to all those who have helped us directly or indirectly towards the successful completion of this project.

BARATH G

RUDRAGANESH R

Place: Vellore **SUDHARSUN M**

Date: 02/06/2021 **Name of the Students**

ABSTRACT

The scarcity of clean water resources around the globe has generated a need for their optimum utilization. Internet of Things (IoT) solutions, based on the application specific sensors' data acquisition and intelligent processing, are bridging the gaps between the cyber and physical worlds. IoT based smart irrigation systems can help in achieving optimum water-resource utilization in the precision farming landscape. This paper presents an open-source technology based smart system to predict the irrigation requirements of a field using the sensing of ground parameter like soil moisture, soil temperature, and environmental conditions along with the weather forecast data from the Internet. The intelligence of the proposed system is based on a smart algorithm, which considers sensed data along with the weather forecast parameters like precipitation, air temperature, humidity, and UV for the near future. The complete system has been developed and deployed on a pilot scale, where the sensor node data is wirelessly collected over the cloud using web-services and a web-based information visualization and decision support system provides the real-time information insights based on the analysis of sensors data and weather forecast data. The paper describes the system and discusses in detail the information processing results of three weeks data based on the proposed algorithm. The system is fully functional and the prediction results are very encouraging.

Most of the farmers use large portions of farming land and it becomes very difficult to reach and track each corner of large lands. Sometime there is a possibility of uneven water sprinkles. This result in the bad quality crops which further leads to financial losses. In this scenario the Smart Irrigation System using Latest IoT technology is helpful and leads to ease of farming. The Smart irrigation System has wide scope to automate the complete irrigation system. Here we are building a IoT based Irrigation System using ESP8266 NodeMCU Module and DHT11 Sensor.

EXECUTIVE SUMMARY

A smart irrigation system was the desired outcome of this paper. The results and the visual outcomes indicate the fulfilment of the initial project goal. We had created our own Android Application to make the user-friendly Smart Irrigation System. We mainly focusing on the Smart irrigation based on the Internet of Things (IoT). The User can handle the Smart Irrigation from anywhere, anytime. Also, we had done the different types of irrigation using solenoid valve. Also, we worked on the alarm system for the plant-based soil moisture, temperature and humidity of the soil using the buzzer. Also, we have to monitor the level of water in the water tank using ultrasonic sensor. We are going to monitor the rain is coming or not.

The proposed loT based smart irrigation system has the capability of regulating soil moisture level as per requirement and user can remotely monitor, control, and collect data through the Android Application. The automated irrigation system presented in this work was found more viable, and can manage irrigation water supply more effectively. It helps to optimize the use of water for irrigation purpose. It shows that water consumption is reduced with the implementation of soil-moisture based automated irrigation system. The Software which we used is Arduino IDE platform to run the code. We are created our own Android Application using Android Studio to maintain the Smart Irrigation System based IoT (Internet of Things). We reduced the wastage of water level in our project also the mobile application is handled only by manually.

CONTENTS Page No.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS / ACRONYMS

1. IOT – INTERNET OF THINGS
2. IDE – INTEGRATED DEVELOPMENT ENVIRONMENT
3. RDBMS – REALTIME DATABASE MANAGEMANT SYSTEMS
4. API – APPLICATION PROGRAMMING INTERFACE

INTRODUCTION

Smart irrigation systems are a combination of an advanced technology of sprinklers with nozzles that improve coverage and irrigation controllers that are watering and water conservation systems that monitor moisture-related conditions on your property and automatically adjust watering to optimal levels.

It is estimated that India's population will surpass 1.69 billion by 2050 Coale and Hoover, and the current food availability is itself not sufficient to feed the people. Till date, traditional farming that is used in fields could be one of the reasons behind reduced crop production even though ample arable land is available. It thus becomes essential to merge the developing technologies with agriculture and perform intelligent farming. Digital India concept has led to tremendous growth in digital information storage, retrieval and communication and, one such concept Internet of Things (IoT).

The idea of IoT has been blooming since decades, but it got recognized only after it was mentioned by Kevin Ashton co-founder and executive director of the Auto-ID Center at MIT. His words in Ashton (2009) describe IoT as: "Today computers and, therefore, the Internet are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (a petabyte is 1024 terabytes) of data available on the Internet were first captured and created by human beings by typing, pressing a record button, taking a digital picture or scanning a bar code". This concept is the crux of smart technology and can be applied to gardens/farms for proper irrigation management. Smart farming is a well-established domain and there are numerous advantages of combining technology with farmers' experience like: improved crop health, better poultry farm animal health and tracking, water management, food from farm to plate, etc.

Numerous works has been done and is being carried out by various research groups in the smart agriculture domain, and details of a few works have been presented in this section. Water is one of the most important resources when it comes to farming. This has led to an extensive work related to efficient utilization of this limited resource. A large amount of water, approximately 100x more than personal use is consumed by food and agriculture and almost 70% of river and groundwater is utilized in irrigation making them the largest consumers of water resources. Currently, out of 3600 km^3 of fresh water, nearly half is lost due to evaporation, transpiration from the crops, etc. while the remaining half adds the groundwater level. Most of the water involved in agriculture is provided directly by rainfall, but the places with drought have only irrigation as an option for farming purpose.

A typical irrigation system comprises of a layout: irrigation method, source: nearby water body, carriers: pipelines to the farm, actuators: on/off mechanism to control water, valves: water flow control. There are methods as mentioned by Food and Agriculture Organization of the United Nations to compute irrigation requirements of crops. To achieve these needs of crops, there are various ways to irrigate them. These include drip, sprinkler, furrow, manual, etc. Out of these ways to irrigate, drip has the highest efficiency and a well-designed one can even reach to 100% when it comes to water usage.

1.1 Objective

The main objective of this project is to develop an intelligent system that solves most problems related to irrigation and agriculture such as controlling and saving both the water and electricity, increasing agricultural production using small quantities of water, minimize manual intervention in watering operations with increasing watering speed, easy to use. All these features make these methods sustainable option to be considered to improve the agriculture and irrigation efficiency.

1.2 Scope of the Project

This system can be the more intelligent system which predicts user actions, nutrient level of the plants, time to harvest, etc. With using Machine Learning algorithms more advancements can be done in the future which will help farmer a lot and water consumption can also be reduced in agriculture.

- Focused on the prevention of crops from insect attack which damages the crop leaves and root so it automatically affects the crop yield.
- Climatic conditions also affect the growth of crops, like temperature increases the water requirement also increases so it can also be monitored.

1.3 Background

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

PROJECT DESCRIPTION AND GOALS

Smart irrigation is an innovative scenario where many researchers are taking interest and for decades it is developing and emerging. Pressure on the water distribution system is increasing and the significance of water management has increased due to the sustainability irrigated farming. Generally, the main purpose of smart irrigation is to reduce manpower, water resources and power consumption. Routine maintenance is required for the legitimate performance of all sensors. Very smart irrigation system works automatic and use the moisture sensor to systematically water the plants without human observation. Therefore, the main purpose of the work is to design the irrigation system, which provides all the above quality with the traditional feature available in irrigation system such as measuring moisture analysis of the area to prevent crop damage issues. Temperature is observed so that the surrounding temperature can be examined as the crop temperature is also sensitive.

Today's technology requires a user-friendly device which is economical in cost and most effective as well and which is provided by Arduino board. In today's generation, it is very convenient to send the notification to IoT platform because of widespread use of smartphones and PC. The whole system is distributed into two parts. The first part comprises of setting up an Arduino Board and interfacing it with the several sensors. The second part consists of developing the IoT platform and connecting it to the server. The ESP 8266 Wi-Fi module is used because of which the transmission becomes simpler and faster. It consists of two motors one for water pump, and second for fertilizers pump. It also helps the owner to monitor system globally throughout the world. After getting the proper information about the field, farmer can turn ON/OFF the motors parenthetically. This system is simple to implement. This System is designed to improve the security, flexibility and to remove the flaws of the existing system.

TECHNICAL SPECIFICATION

HARWARE SPECIFICATIONS:

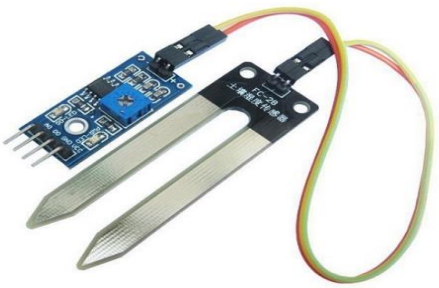
NODEMCU ESP8266:

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

NodeMCU is an open-source firmware for which open-source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. NodeMCU started on 13 Oct 2015, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the Gerber file of an ESP8266 board, named devkit v0.9.

SOIL MOISTURE YL-69 SENSOR:

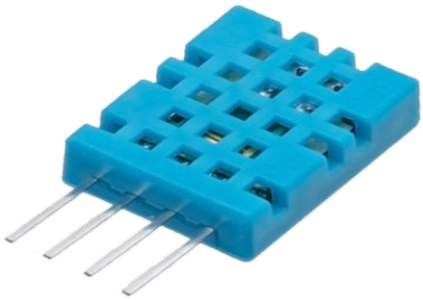


The soil moisture sensor or the hygrometer is usually used to detect the humidity of the soil. So, it is perfect to build an automatic watering system or to monitor the soil moisture of your plants. The sensor is set up by two pieces: the electronic board (at the right), and the probe with two pads, that detects the water content (at the left).

The output can be a digital signal (D0) LOW or HIGH, depending on the water content. If the soil humidity exceeds a certain predefined threshold value, the modules outputs LOW, otherwise it outputs HIGH. The threshold value for the digital signal can be adjusted using the potentiometer.

The output can be an analog signal and so you'll get a value between 0 and 1023.

DHT11 SENSOR:

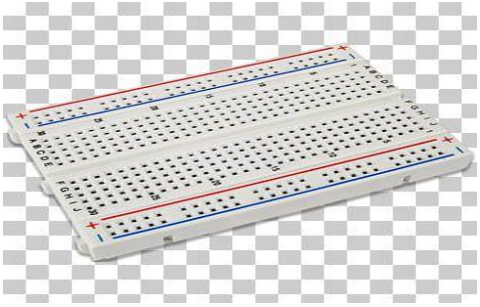


The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data. You can get new data from it once every 2 seconds, so when using the library from Adafruit, sensor readings can be up to 2 seconds old.

Comes with a 4.7K or 10K resistor, which you will want to use as a pullup from the data pin to VCC.

Compared to the DHT22, this sensor is less precise, less accurate and works in a smaller range of temperature/humidity, but its smaller and less expensive.

BREAD BOARD:



A breadboard, or protoboard, is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used for slicing bread. In the 1970s the solderless breadboard (a.k.a. plugboard, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.

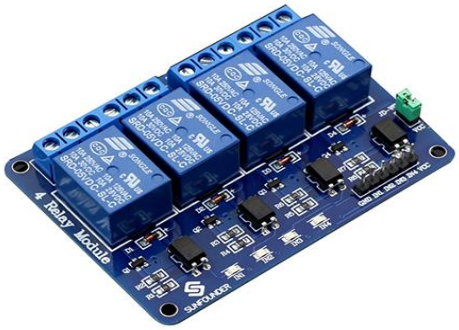
Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason, solderless breadboards are also popular with students and in technological education. Older breadboard types did not have this property. A stripboard (Veroboard) and similar prototyping printed circuit boards, which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

JUMPER WIRES:

A jump wire (also known as jumper, jumper wire, jumper cable, DuPont wire or cable) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

RELAY MODULE 4 CHANNEL:



This is a 5V 4-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller.

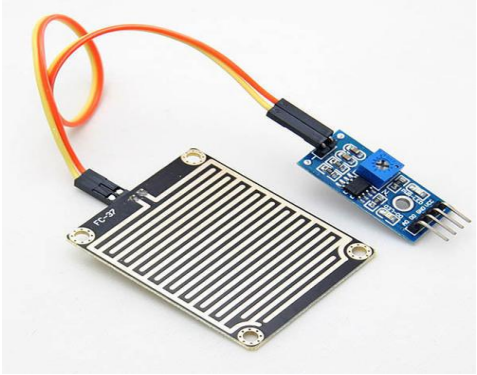
The four-channel relay module contains four 5V relays and the associated switching and isolating components, which makes interfacing with a microcontroller or sensor easy with minimum components and connections. The contacts on each relay are specified for 250VAC and 30VDC and 10A in each case, as marked on the body of the relays.

SOLENOID VALVE:

A solenoid valve is an electrically controlled valve. The valve features a solenoid, which is an electric coil with a movable ferromagnetic core (plunger) in its center. In the rest position, the plunger closes off a small orifice. An electric current through the coil creates a magnetic field. The magnetic field exerts an upwards force on the plunger opening the orifice. This is the basic principle that is used to open and close solenoid valves.

Solenoid valves are used in a wide range of applications, with high or low pressures and small or large flow rates. These solenoid valves use different operating principles that are optimal for the application. The three most important ones are explained in this article: direct acting, indirect acting, and semi-direct acting operation.

RAIN DROP SENSOR:



Raindrop Sensor is a tool used for sensing rain. It consists of two modules, a rain board that detects the rain and a control module, which compares the analog value, and converts it to a digital value. The raindrop sensors can be used in the automobile sector to control the windshield wipers automatically, in the agriculture sector to sense rain and it is also used in home automation systems.

Raindrop sensor is basically a board on which nickel is coated in the form of lines. It works on the principal of resistance.

Raindrop Sensor module allows to measure moisture via analog output pins and it provides a digital output when a threshold of moisture exceeds.

The module is based on the LM393 op amp. It includes the electronics module and a printed circuit board that “collects” the rain drops. As rain drops are collected on the circuit board, they create paths of parallel resistance that are measured via the op amp.

The sensor is a resistive dipole that shows less resistance when wet and more resistance when dry. When there is no rain drop on board it increases the Resistance so we get high voltage according to V=IR.

When rain drop present, it reduces the resistance because water is a conductor of electricity and presence of water connects nickel lines in parallel so reduces resistance and reduces voltage drop across it.

ULTRASONIC SENSOR:

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e., the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety systems. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles (though the physical components are still affected by variables such as heat).

BUZZER:

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke. A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

SOFTWARE SPECIFICATIONS:

ARDUINO IDE:

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

Arduino IDE is a derivative of the Processing IDE, however as of version 2.0, the Processing IDE will be replaced with the Visual Studio Code-based Eclipse Theia IDE framework.

With the rising popularity of Arduino as a software platform, other vendors started to implement custom open-source compilers and tools (cores) that can build and upload sketches to other microcontrollers that are not supported by Arduino's official line of microcontrollers.

ANDROID STUDIO:

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

FIREBASE CONSOLE:

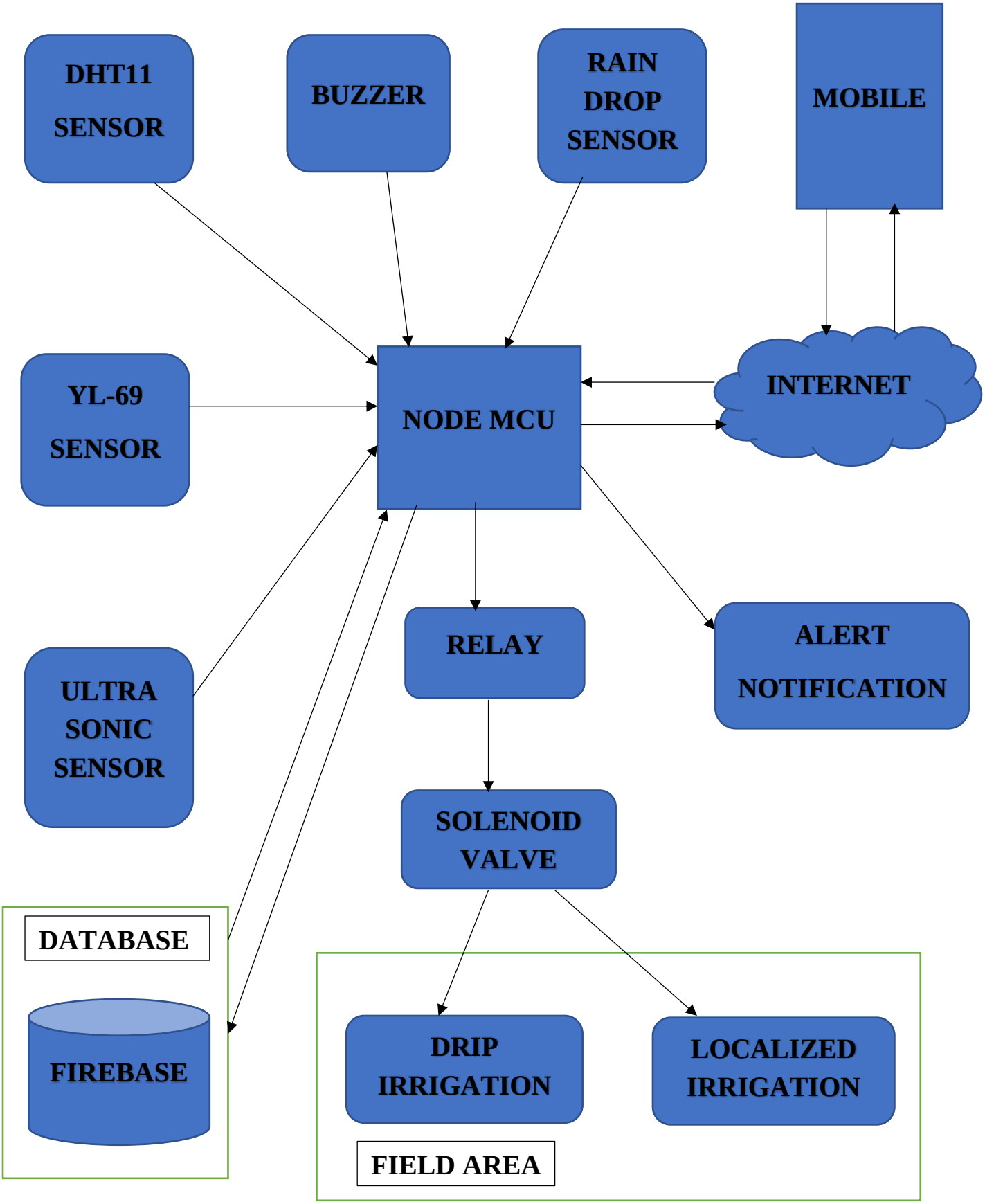
Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

Firebase's first product was the Firebase Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications.

In May 2012, a month after the beta launch, Firebase raised $1.1 million in seed funding from venture capitalists Flybridge Capital Partners, Greylock Partners, Founder Collective, and New Enterprise Associates. In June 2013, the company further raised $5.6 million in Series A funding from Union Square Ventures and Flybridge Capital Partners.

DESIGN APPROACH AND DETAILS

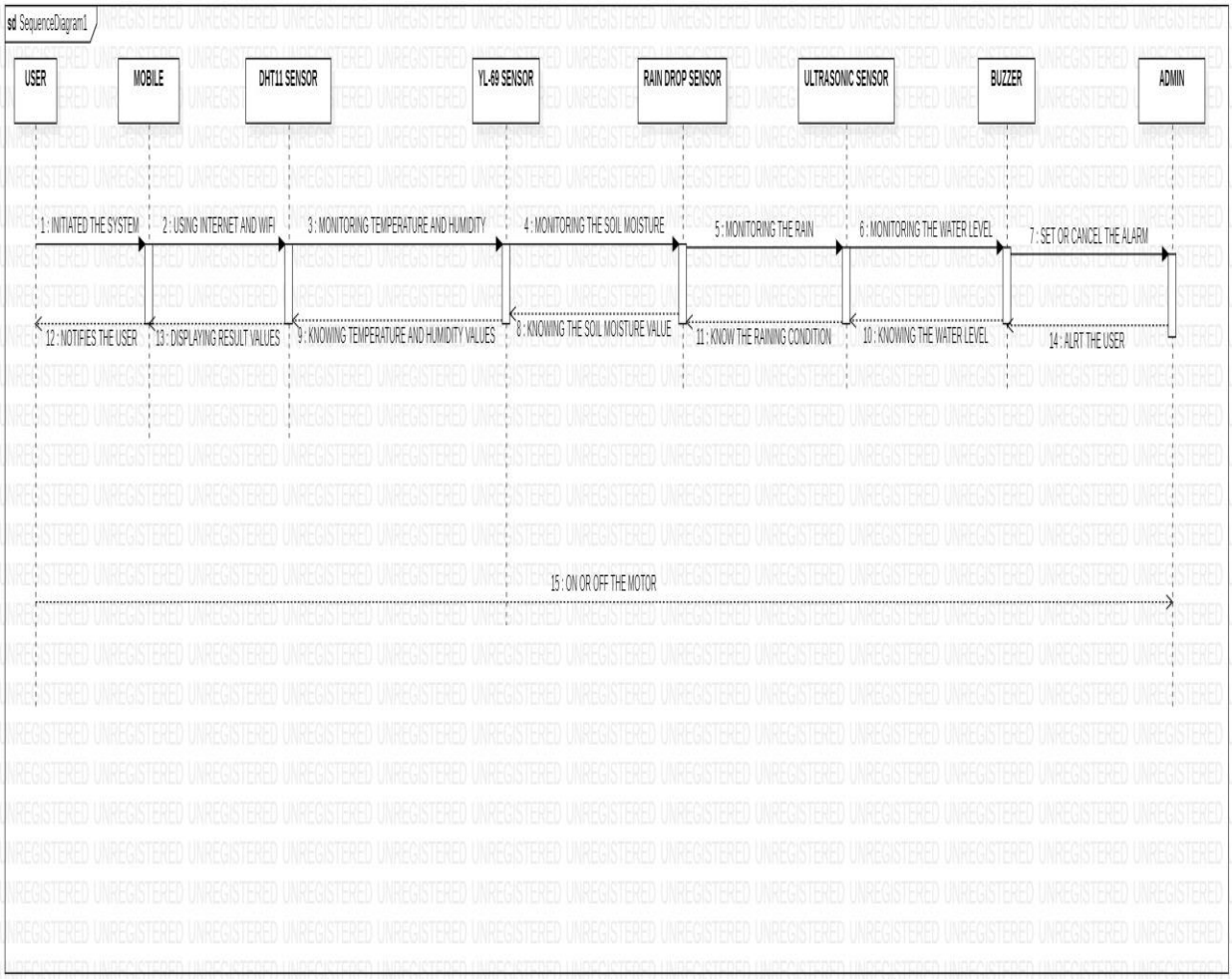
SYSTEM ARCHITECTURE:



UML DIAGRAMS:

1. USECASE DIAGRAM:

2. SEQUENCE DIAGRAM:



MODULES

5.1 WATER LEVEL MONITORING SYSTEM

- This Module represents the water level monitoring system which helps to monitoring the water level of the tank or well without the use of man power.
- Very Effortless, human energy is not required.
- Low Efficient Cost.
- Farmer is not required to go around to the tank and check the availability of water is enough or not at any time.
- It is very helpful for the Farmers.

5.2 SOIL MOISTURE MONITORING SYSTEM

- This Module represents the soil moisture monitoring system which is very helpful to monitoring the moisture of the soil.
- Low Efficient Cost and cheap, also very Effortless.
- YL-69 Sensor is used to check the moisture level of soil.
- Very Useful for the Farmers and Nursery Gardeners.

5.3 TEMPERATURE AND HUMIDITY MONITORING SYSTEM

- This Module represents the Temperature and Humidity monitoring system which is useful to monitor the Temperature and Humidity Level of Soil.
- DHT11 Sensor is Used to monitor the Temperature and Humidity level of soil.
- Low Cost, Very Useful and cheap in Market.
- Very helpful for Nowadays.
- User can check the Temperature and Humidity level at anywhere and anytime.

5.4 RAIN MONITORING SYSTEM

- This Module represents the Rain Monitoring with the use of Rain drop sensor.
- Raindrop sensor is basically a board on which nickel is coated in the form of lines.
- Rain Sensor module allows to measure moisture via analog output pins and it provides a digital output when a threshold of moisture exceeds.
- With the use of Rain drop sensor we send the digital data to ESP8266, then ESP8266 send the data to Realtime database of firebase.
- With the help of firebase our Android application shows the status of Rain.
- It works on the principle of resistance.

TYPES OF IRRIGATION

LOCALIZED IRRIGATION:

- Localized irrigation is a method of applying water that results in wetting only a small area of the soil surface and sometimes only part of the root zone.
- Water is applied near the base of the plant so that the application is concentrated in the root zone.
- Water is generally applied at a low flow rate, in small amounts, and frequently.
- The application devices may be small tubes, orifices, nozzles, or perforated pipes. The water may either be applied above or below the soil surface.
- The main components of a localized irrigation system are the water supply (including flow and pressure regulators), the filtration system, main lines, sub-main lines, laterals, and distributors.
- The primary advantages of localized irrigation systems are the high efficiency rates that can be achieved, sometimes as high as 90%.
- High efficiency may result in very significant water savings.
- Often a localized irrigation system will allow a farmer to irrigate twice the area possible with surface irrigation.
- Precise control of water and nutrient application often results in much higher yields and quality.

DRIP IRRIGATION

- Drip irrigation is sometimes called trickle irrigation and involves dripping water onto the soil at very low rates (2-20 litres/hour) from a system of small diameter plastic pipes fitted with outlets called emitters or drippers.

- Water is applied close to plants so that only part of the soil in which the roots grow is wetted, unlike surface and sprinkler irrigation, which involves wetting the whole soil profile.

- With drip irrigation water, applications are more frequent (usually every 1-3 days) than with other methods and this provides a very favourable high moisture level in the soil in which plants can flourish.

- Drip irrigation system delivers water to the crop using a network of mainlines, sub-mains and lateral lines with emission points spaced along their lengths.

- Each dripper/emitter, orifice supplies a measured, precisely controlled uniform application of water, nutrients, and other required growth substances directly into the root zone of the plant.

ALERT OR ALARM SYSTEM

- This Module represents the alarm system with the use of Buzzer.

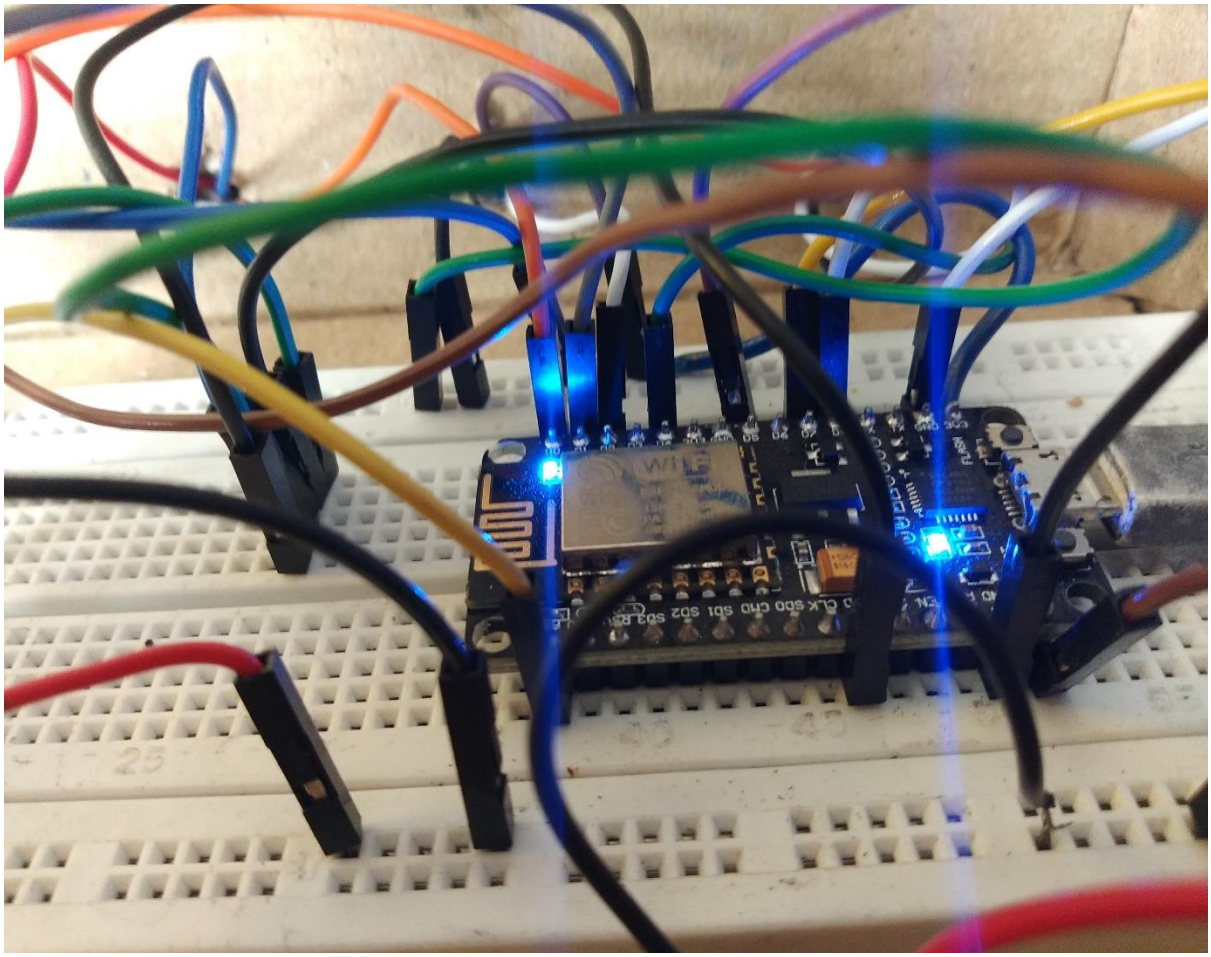
- Buzzers are used for making beeps, tones and alerts. To use, connect short pin to ground and the other pin to 5 voltage level.

- User can set the alarm for the soil moisture, temperature and humidity with the use of our Android Application.

- Once you set the alarm our application compares the sensor given data with user data, if the sensor given data is less than or equal to user data buzzer will turn on.

RESULT AND DISCUSSION

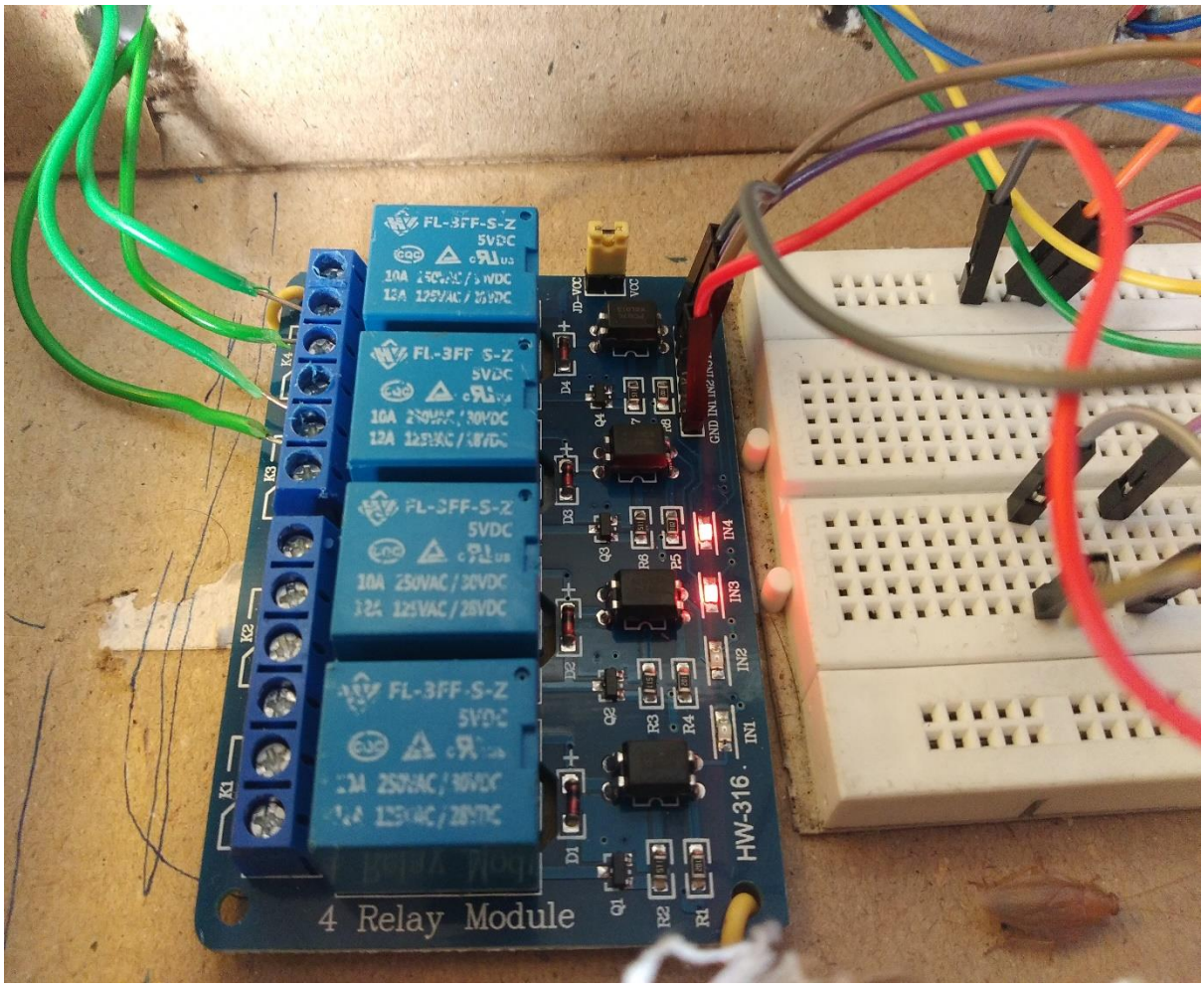
NodeMCU Working Principle:



NodeMCU is an open-source firmware for which open-source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

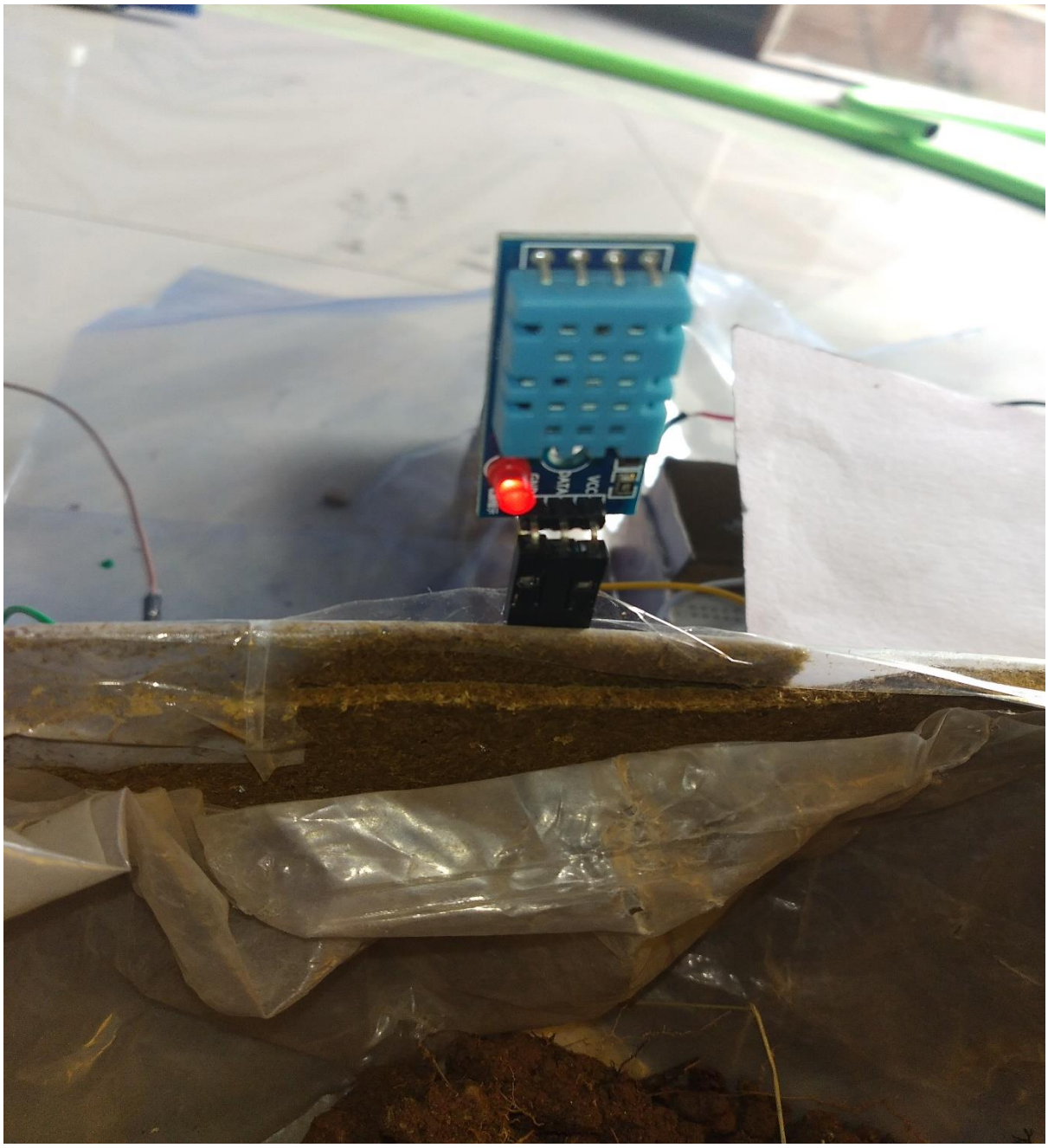
Relay Working Principle:



The 4 Channel Relay Module is a convenient board which can be used to control high voltage, high current load such as motor, solenoid valves, lamps and AC load. It is designed to interface with microcontroller such as Arduino, PIC and etc. The relays terminal (COM, NO and NC) is being brought out with screw terminal. It also comes with a LED to indicate the status of relay.

This is a 5V 4-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller.

DHT11 Sensor Working:



The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

Comes with a 4.7K or 10K resistor, which you will want to use as a pullup from the data pin to VCC.

Compared to the DHT22, this sensor is less precise, less accurate and works in a smaller range of temperature/humidity, but its smaller and less expensive.

YL-69 Sensor Working:

The soil moisture sensor or the hygrometer is usually used to detect the humidity of the soil. So, it is perfect to build an automatic watering system or to monitor the soil moisture of your plants. The sensor is set up by two pieces: the electronic board (at the right), and the probe with two pads, that detects the water content (at the left).

The output can be a digital signal (D0) LOW or HIGH, depending on the water content. If the soil humidity exceeds a certain predefined threshold value, the modules outputs LOW, otherwise it outputs HIGH. The threshold value for the digital signal can be adjusted using the potentiometer.

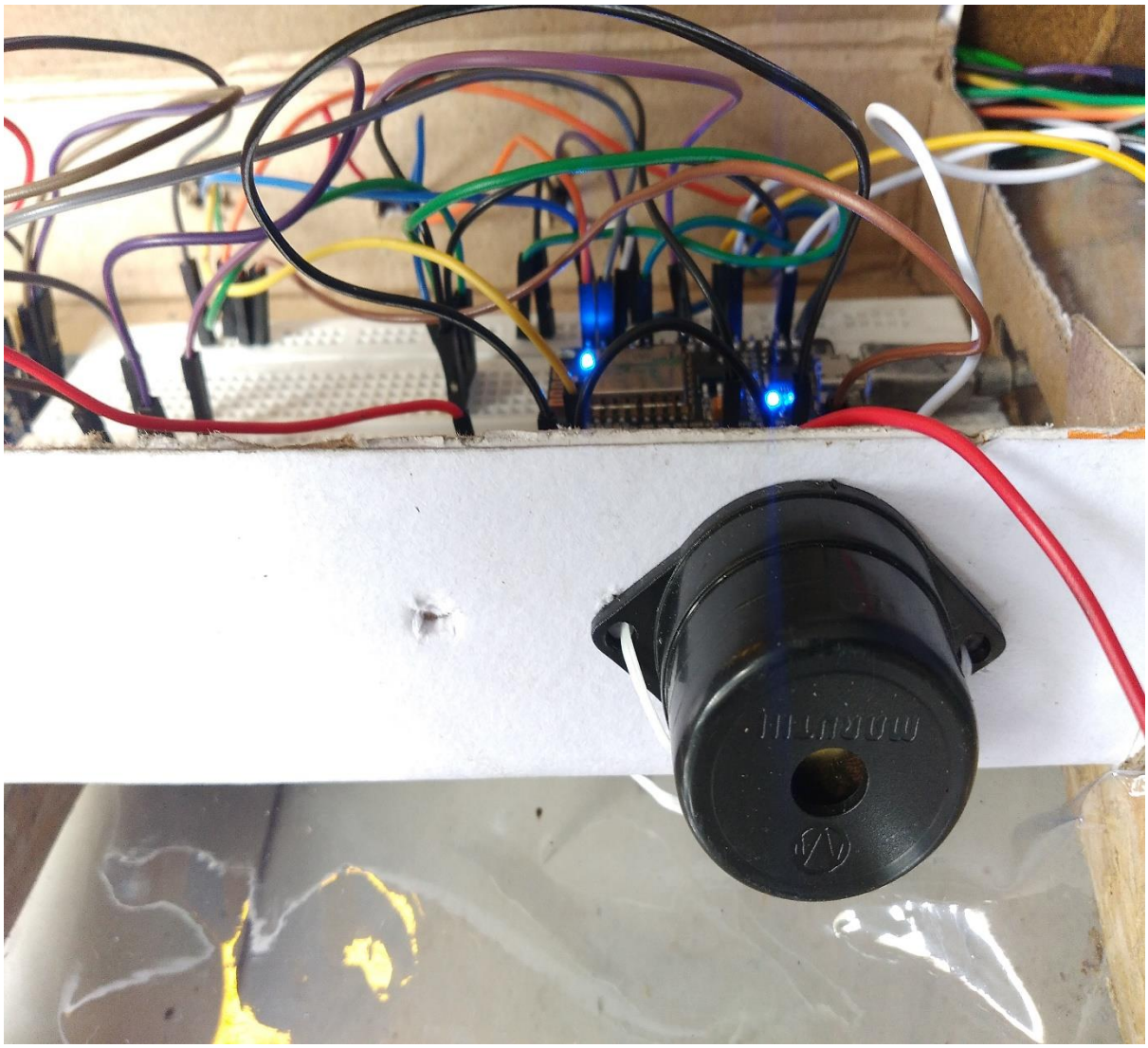
The output can be an analog signal and so you'll get a value between 0 and 1023.

Ultrasonic Sensor Working Principle:

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e., the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

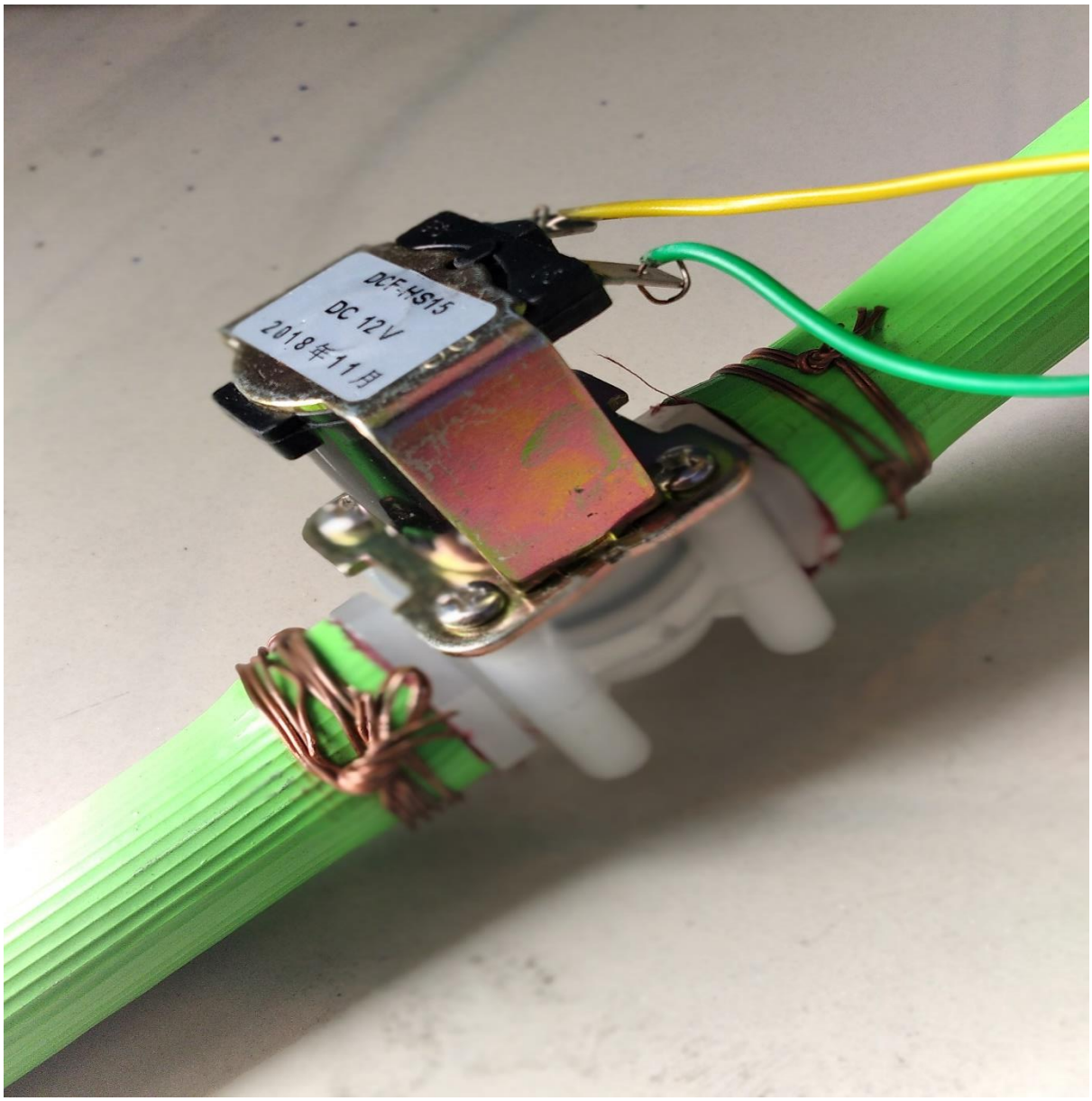
Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety systems. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles (though the physical components are still affected by variables such as heat).

Buzzer Working Principle:



A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke. A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

Solenoid Valve Working Principle:



A solenoid valve is an electrically controlled valve. The valve features a solenoid, which is an electric coil with a movable ferromagnetic core (plunger) in its centre. In the rest position, the plunger closes off a small orifice. An electric current through the coil creates a magnetic field. The magnetic field exerts an upwards force on the plunger opening the orifice. This is the basic principle that is used to open and close solenoid valves.

Solenoid valves are used in a wide range of applications, with high or low pressures and small or large flow rates. These solenoid valves use different operating principles that are optimal for the application. The three most important ones are explained in this article: direct acting, indirect acting, and semi-direct acting operation.

TYPES OF IRRIGATION WORKING:

LOCALIZED IRRIGATION:

- Localized irrigation is a method of applying water that results in wetting only a small area of the soil surface and sometimes only part of the root zone.

- Water is applied near the base of the plant so that the application is concentrated in the root zone.

- Water is generally applied at a low flow rate, in small amounts, and frequently.

- The application devices may be small tubes, orifices, nozzles, or perforated pipes. The water may either be applied above or below the soil surface.

- The main components of a localized irrigation system are the water supply (including flow and pressure regulators), the filtration system, main lines, sub-main lines, laterals, and distributors.

- The primary advantages of localized irrigation systems are the high efficiency rates that can be achieved, sometimes as high as 90%.

- High efficiency may result in very significant water savings.

- Often a localized irrigation system will allow a farmer to irrigate twice the area possible with surface irrigation.

- Precise control of water and nutrient application often results in much higher yields and quality.

DRIP IRRIGATION

- Drip irrigation is sometimes called trickle irrigation and involves dripping water onto the soil at very low rates (2-20 litres/hour) from a system of small diameter plastic pipes fitted with outlets called emitters or drippers.

- Water is applied close to plants so that only part of the soil in which the roots grow is wetted, unlike surface and sprinkler irrigation, which involves wetting the whole soil profile.

- With drip irrigation water, applications are more frequent (usually every 1-3 days) than with other methods and this provides a very favourable high moisture level in the soil in which plants can flourish.

- Drip irrigation system delivers water to the crop using a network of mainlines, sub-mains and lateral lines with emission points spaced along their lengths.

- Each dripper/emitter, orifice supplies a measured, precisely controlled uniform application of water, nutrients, and other required growth substances directly into the root zone of the plant.

Rain Drop Sensor Working Principle:

Raindrop Sensor is a tool used for sensing rain. It consists of two modules, a rain board that detects the rain and a control module, which compares the analog value, and converts it to a digital value. The raindrop sensors can be used in the automobile sector to control the windshield wipers automatically, in the agriculture sector to sense rain and it is also used in home automation systems.

Raindrop sensor is basically a board on which nickel is coated in the form of lines. It works on the principal of resistance.

Raindrop Sensor module allows to measure moisture via analog output pins and it provides a digital output when a threshold of moisture exceeds.

The module is based on the LM393 op amp. It includes the electronics module and a printed circuit board that "collects" the rain drops. As rain drops are collected on the circuit board, they create paths of parallel resistance that are measured via the op amp.

The sensor is a resistive dipole that shows less resistance when wet and more resistance when dry. When there is no rain drop on board it increases the Resistance so we get high voltage according to V=IR.

When rain drop present, it reduces the resistance because water is a conductor of electricity and presence of water connects nickel lines in parallel so reduces resistance and reduces voltage drop across it.

COST ANALYSIS

PRODUCTS	COST
NodeMCU	400
DHT11 Sensor	245
YL-69 Sensor	230
HC-SR04 Sensor	175
Rain Drop Sensor	170
Breadboard	200
Jumper Wire	230
Solenoid Valve	650
Buzzer	90
TOTAL COST	**2,390**

CONCLUSION

A smart irrigation system was the desired outcome of this paper. The results and the visual outcomes indicate the fulfilment of the initial project goal. A mobile application offering a user-friendly experience contributes to the lessened sophistication of the system. Moreover, the developed website providing rich content for the farmer plays a huge role in enlightening the user too. The prototype in question possesses adequate room for improvement nonetheless but the sustainability period can be predicted from three to five years.

The IoT-Based Smart Irrigation System is an innovation in the step towards smart irrigation to increase plant health and conserve water by implementing IoT concepts using sensors and microcontrollers over a network to facilitate communication and efficient action. With IoT as its backbone, the IoT Based Smart Irrigation System uses soil moisture sensors to accurately measure and compare the moisture of the soil to a threshold value to ensure that the soil is watered only when the plant needs it. Additionally, if the pH of the water being given to the soil is unsafe for the plant the IoT Based Smart Irrigation Smart Irrigation uses our own Android Application on IoT to communicate to sub-level microcontrollers which use plant-healthy acid and base pH solutions to modify the pH of the water being given to the plant. The system can be managed by the user via the Application created by us, which allows the user to select the type of plant that they are watering so that the system can accurately manipulate the pH of the water given to the plant. The app can also be used by the user to monitor the pH and moisture levels monitored by the system. From the results of the experimental evaluation, it can be concluded that such a system is possible for use in autonomously and efficiently watering specific crops. Overall, the system acts as an efficient method to conserve water by reducing human error and increase efficiency in terms of large-scale agriculture.

SUMMARY

IOT based smart agriculture system can prove to be very helpful for farmers since over as well as less irrigation is not good for agriculture. Threshold values for climatic conditions like humidity, temperature, moisture can be fixed based on the environmental conditions of that particular region. The system also senses the invasion of animals which is a primary reason for reduction in crops. This system generates irrigation schedule based on the sensed real time data from field and data from the weather repository. This system can recommend farmer whether or not, is there a need for irrigation. Continuous internet connectivity is required. This can be overcome by extending the system to send suggestion via SMS to the farmer directly on his mobile using GSM module instead of mobile app.

The proposed loT based smart irrigation system has the capability of regulating soil moisture level as per requirement and user can remotely monitor, control, and collect data through the Android Application. The automated irrigation system presented in this work was found more viable, and can manage irrigation water supply more effectively. It helps to optimize the use of water for irrigation purpose. It shows that water consumption is reduced with the implementation of soil-moisture based automated irrigation system. The Software which we used is Arduino IDE platform to run the code. We are created our own Android Application using Android Studio to maintain the Smart Irrigation System based IoT (Internet of Things). We reduced the wastage of water level in our project also the mobile application is handled only by manually.

APPENDICES

Appendix 1:

- Writing the connections between Other Devices and NodeMCU.

OTHER DEVICES	NODE MCU PIN CONFIGURATION
DHT11 Sensor (Data)	D2
Buzzer (+ve)	D7
YL-69 Sensor (A0)	A0
Rain Drop Sensor (D0)	D0
Ultrasonic Sensor (Trigger)	D4
Ultrasonic Sensor (Echo)	D8
Relay (Inp 3)	D1
Relay (Inp 4)	D5

- Writing the connections between Relay and Other Devices.

OTHER DEVICES	RELAY PIN CONFIGURATION
Solenoid Valve 1 (+ve)	K4 Normally Open
Solenoid Valve 2 (+ve)	K3 Normally Open
12V Power Supply 1 (+ve)	K4 Common
12V Power supply 2 (+ve)	K3 Common

Appendix 2:

- Materials and their features that have been used in this project.

MATERIALS	FEATURES
NodeMCU	❖ NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. ❖ Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. ❖ NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.
DHT11 Sensor	❖ Ultra-low cost. ❖ 3 to 5V power and I/O. ❖ 2.5mA max current use during conversion (while requesting data). ❖ Good for 20-80% humidity readings with 5% accuracy. ❖ Good for 0-50°C temperature readings ±2°C accuracy. ❖ No more than 1 Hz sampling rate (once every second). ❖ Body size 15.5mm x 12mm x 5.5mm. ❖ 4 pins with 0.1" spacing.
YL-69 Sensor	❖ Operating voltage: DC 3.3V - 5V. ❖ Output voltage signal: 0 ~ 4.2V. ❖ Current: 35mA. ❖ LED: Power indicator (Red) and Digital switching output indicator (Green).

	❖ Size: 60 x 20 x 5mm
Breadboard	❖ 2 Distribution Strips, 200 tie-points. ❖ 630 tie-points in IC/ circuit areas. ❖ ABS plastic with color legend. ❖ Dimension: 6.5*4.4*0.3 inch. ❖ Hole/Pitch Style: Square wire holes (2.54mm). ❖ ABS heat Distortion Temperature: 84° C (183° F). ❖ Rating: 300/3 to 5Amps. ❖ Insulation Resistance: 500MΩ / DC500V. ❖ Withstanding Voltage: 1,000V AC / 1 minute. ❖ Insertion Wire Size: 21 to 26 AWG Wire.
Jumper Wire	❖ It is a connector pins at each end, allowing them to be used to connect two points to each other without soldering. ❖ Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed.
Relay Module 4 Channel	❖ 5V 4-Channel Relay interface board. ❖ Requires 15-20mA signal drive Current. ❖ TTL logic compatible. ❖ High-current AC250V/10A, DC30V/10A relay. ❖ Status LED.

	❖ Equipped with 3.1mm screw holes for easy installation. ❖ 75 x 55 x 19.3mm (2.95 x 2.16 x 0.76″).
Ultrasonic Sensor	❖ Operating voltage: +5V. ❖ Theoretical Measuring Distance: 2cm to 450cm. ❖ Practical Measuring Distance: 2cm to 80cm. ❖ Accuracy: 3mm. ❖ Measuring angle covered: <15°. ❖ Operating Current: <15mA. ❖ Operating Frequency: 40Hz.
Rain Drop Sensor	❖ Working voltage 5V. ❖ Output format: Digital switching output (0 and 1), and analog voltage output A0. ❖ Potentiometer adjust the sensitivity. ❖ Uses a wide voltage LM393 comparator. ❖ Comparator output signal clean waveform is good, driving ability, over 15mA. ❖ Anti-oxidation, anti-conductivity, with long use time. ❖ With bolt holes for easy installation. ❖ Small board PCB size: 3.2cm x 1.4cm.
Solenoid Valve	❖ Double Viton O-ring for improved sealing at the tube inlet and outlet. ❖ The threaded armature top screws into the armature tube, and then the

	connection is welded together. The threaded connection combined with welding allows for a longer lifespan of the valve. ❖ The armature tube is machined from a solid stainless-steel bar - with no additional joints or connections, it can withstand higher pressure and more lifecycles.
Buzzer	❖ Rated Voltage: 6V DC. ❖ Operating Voltage: 4-8V DC. ❖ Rated current: <30mA. ❖ Sound Type: Continuous Beep. ❖ Resonant Frequency: ~2300 Hz. ❖ Small and neat sealed package. ❖ Breadboard and Perf board friendly.

Appendix 3:

- Arduino Code

```
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
#include <DHT.h>
#define FIREBASE_HOST "https://smrtirri-brs-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "j5ibUzwkXkhcixZWIpDjiV22iUoEjPHeJt3XRGLG"
#define WIFI_SSID "Free"
#define WIFI_PASSWORD "12345qbs"
#define BUZZPIN D7
#define DHTPIN D2
#define DRIPIRR D5
#define OTHERIRR D1
#define RAINPIN D0
#define TRIGPIN D4
#define ECHOPIN D8
#define DHTTYPE    DHT11
DHT dht(DHTPIN, DHTTYPE);
int rainState=0;
long duration, distance, average;
long a[3];
int dry=0,low=0,medium=0,full=0,veryfull=0;
FirebaseData firebaseData;
FirebaseData getData;
FirebaseData getDrip;
FirebaseData getOther;
FirebaseJson json;

void setup() {
```

```
  Serial.begin(9600);
  dht.begin();
    pinMode(DRIPIRR,OUTPUT);
  pinMode(OTHERIRR,OUTPUT);
    pinMode(RAINPIN, INPUT);
    pinMode(TRIGPIN, OUTPUT);
    pinMode(ECHOPIN, INPUT);
    pinMode(A0,OUTPUT);
    pinMode(BUZZPIN, OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }

  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

}
void measure()
{
digitalWrite(TRIGPIN, LOW);
delayMicroseconds(5);
```

```
digitalWrite(TRIGPIN, HIGH);
delayMicroseconds(15);
digitalWrite(TRIGPIN, LOW);
pinMode(ECHOPIN, INPUT);
duration = pulseIn(ECHOPIN, HIGH);
distance = (duration/2) / 29.1;
}
void sensorUpdate(){
  float m = analogRead(A0);
  rainState = digitalRead(RAINPIN);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  Serial.println(F("Rain State"));
  Serial.println(rainState);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("C  ,"));
  Serial.print(f);
  Serial.println(F("F  "));

   if (Firebase.setFloat(firebaseData, "/dataR/soil", m))
  {
    Serial.println("PASSED");
```

```
}
else
{
  Serial.println("FAILED");

}
if (Firebase.setFloat(firebaseData, "/dataR/temprature", t))
{
  Serial.println("PASSED");

}
else
{
  Serial.println("FAILED");

}

if (Firebase.setFloat(firebaseData, "/dataR/humidity", h))
{
  Serial.println("PASSED");
}
else
{
  Serial.println("FAILED");

}

if (Firebase.setFloat(firebaseData, "/dataR/rain", rainState))
{
```

```
  Serial.println("PASSED");
 }
 else
 {
  Serial.println("FAILED");

 }
  if (Firebase.getString(getDrip, "/dataM/other")){
  Serial.println(getDrip.stringData());
  if (getDrip.stringData() == "0") {
  digitalWrite(OTHERIRR, HIGH);
  }
 else if (getDrip.stringData() == "1"){
     digitalWrite(OTHERIRR, LOW);

  }
 }
  if (Firebase.getString(getDrip, "/dataM/dripOn")){
  Serial.println(getDrip.stringData());
  if (getDrip.stringData() == "0") {
  digitalWrite(DRIPIRR, HIGH);
  }
 else if (getDrip.stringData() == "1"){
     digitalWrite(DRIPIRR, LOW);

  }
 }
  /* */

}
```

```
void loop() {
 if (Firebase.getString(getData, "/dataS/alarm")){
   Serial.println(getData.stringData());
   if (getData.stringData() == "0") {
       digitalWrite(BUZZPIN,LOW);
   }
  else if (getData.stringData() == "1"){
       digitalWrite(BUZZPIN,HIGH);
   }
  }
   for (int i=0;i<=2;i++)
  {   //average distance
  measure();
  a[i]=distance;
  }
 distance=(a[0]+a[1]+a[2])/3;
   if (Firebase.setFloat(firebaseData, "/dataR/waterlvl", distance))
  {
   Serial.println("PASSED WWW");
   Serial.print(distance);
  }
  else
  {
   Serial.println("FAILED");
  }
  sensorUpdate();
  delay(100);
}
```

Appendix 4:

- Android Studio Code

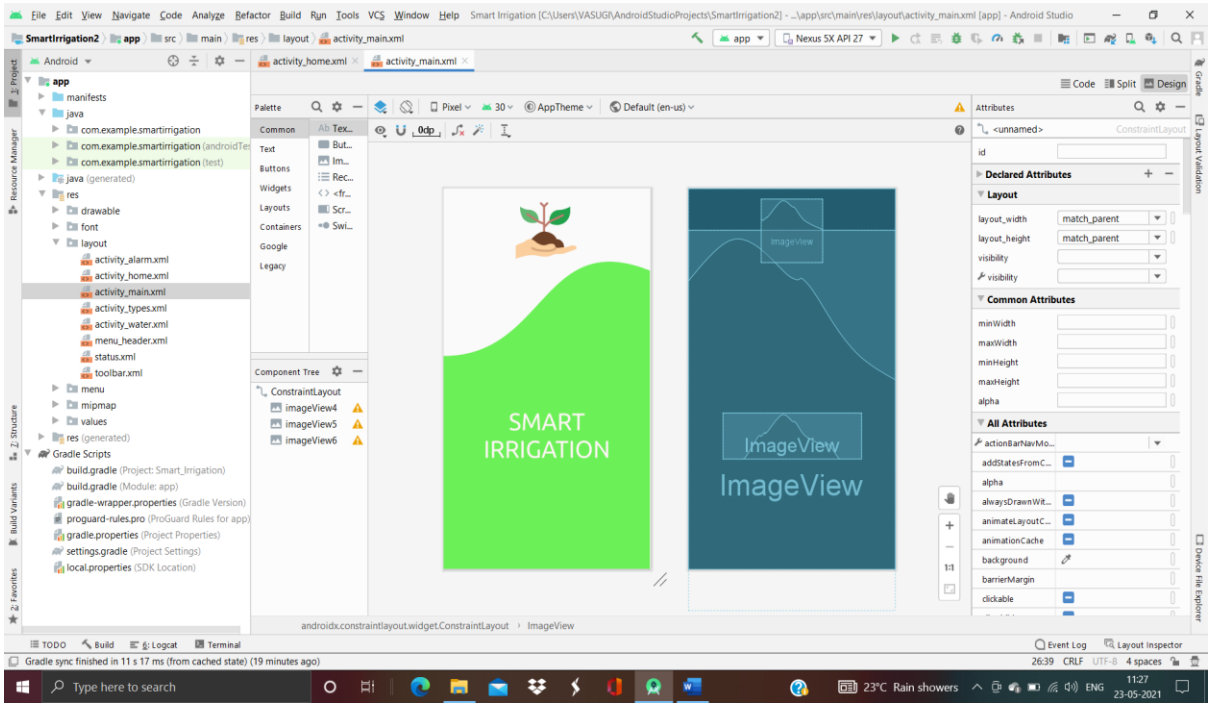
XML CODES:

MAIN.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <ImageView
        android:id="@+id/imageView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="96dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.166"
        app:srcCompat="@drawable/rec" />

    <ImageView
        android:id="@+id/imageView5"
        android:layout_width="275dp"
        android:layout_height="88dp"
        android:layout_marginBottom="212dp"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/name" />
    <ImageView
        android:id="@+id/imageView6"
        android:layout_width="121dp"
        android:layout_height="121dp"
        android:layout_marginStart="144dp"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="20dp"
        android:layout_marginBottom="485dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:srcCompat="@drawable/welimg" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



HOME.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/dra"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeActivity">

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="left"
        android:background="@color/theme"
        android:theme="@style/nav_theme"
        app:headerLayout="@layout/menu_header"
        app:menu="@menu/main_menu"

        />

    <RelativeLayout
        android:id="@+id/home_data"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
    android:background="#6CF057"
    android:padding="20dp">

    <ImageView
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentRight="true"
        android:src="@drawable/planting" />

    <ImageView
        android:id="@+id/menuopen"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:src="@drawable/menusym" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:fontFamily="@font/acme"
        android:text="SMART IRRIGATION"
        android:textColor="#000000"
        android:textSize="20dp" />
</RelativeLayout>

<TextView
    android:id="@+id/wat"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:visibility="invisible" />

<include
    layout="@layout/status"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="0dp"
    android:layout_marginLeft="0dp"
    android:layout_marginTop="80dp"
    android:layout_marginBottom="0dp" />

<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_marginTop="120dp"
    android:layout_marginEnd="0dp"
    android:layout_marginRight="0dp"
    android:alignmentMode="alignMargins"
    android:columnCount="2"
    android:columnOrderPreserved="false"
    android:rowCount="4">

    <androidx.cardview.widget.CardView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_margin="16dp"
        app:cardBackgroundColor="#BBF8B1"
        app:cardCornerRadius="12dp"
```

```
    app:cardElevation="5dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="4dp">

        <ImageView
            android:layout_width="110dp"
            android:layout_height="110dp"
            android:src="@drawable/sun" />

        <TextView
            android:id="@+id/t"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="115dp"
            android:fontFamily="@font/acme"
            android:text="0°C"
            android:textColor="#000000"
            android:textSize="20dp" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:layout_margin="16dp"
    app:cardBackgroundColor="#BBF8B1"
```

```
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="4dp">

        <ImageView
            android:layout_width="110dp"
            android:layout_height="110dp"
            android:src="@drawable/humidityy" />

        <TextView
            android:id="@+id/h"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="115dp"
            android:fontFamily="@font/acme"
            android:text="0%"
            android:textColor="#000000"
            android:textSize="20dp" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="170dp"
    android:layout_columnSpan="2"
```

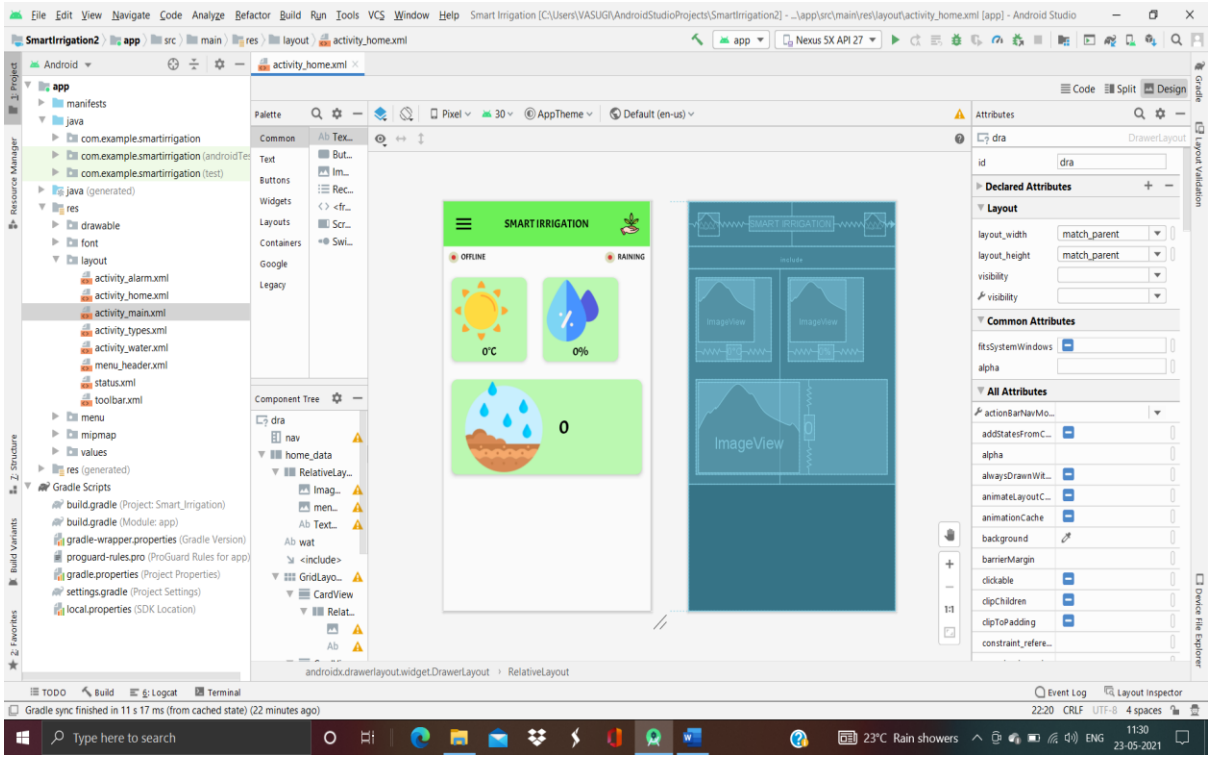
```
        android:layout_margin="16dp"
        app:cardBackgroundColor="#BBF8B1"
        app:cardCornerRadius="12dp"
        app:cardElevation="5dp">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:padding="4dp">

            <ImageView
                android:layout_width="200dp"
                android:layout_height="200dp"
                android:src="@drawable/soilmoisture" />

            <TextView
                android:id="@+id/m"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:layout_marginLeft="210dp"
                android:fontFamily="@font/acme"
                android:text="0"
                android:textColor="#000000"
                android:textSize="35dp" />
        </RelativeLayout>
    </androidx.cardview.widget.CardView>
</GridLayout>
<TextView
    android:id="@+id/stemp"
```

```
        android:visibility="invisible"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
        <TextView
            android:id="@+id/shumi"
            android:visibility="invisible"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"   />
        <TextView
            android:id="@+id/ssoil"
            android:visibility="invisible"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"  />
    </RelativeLayout>
</androidx.drawerlayout.widget.DrawerLayout>
```



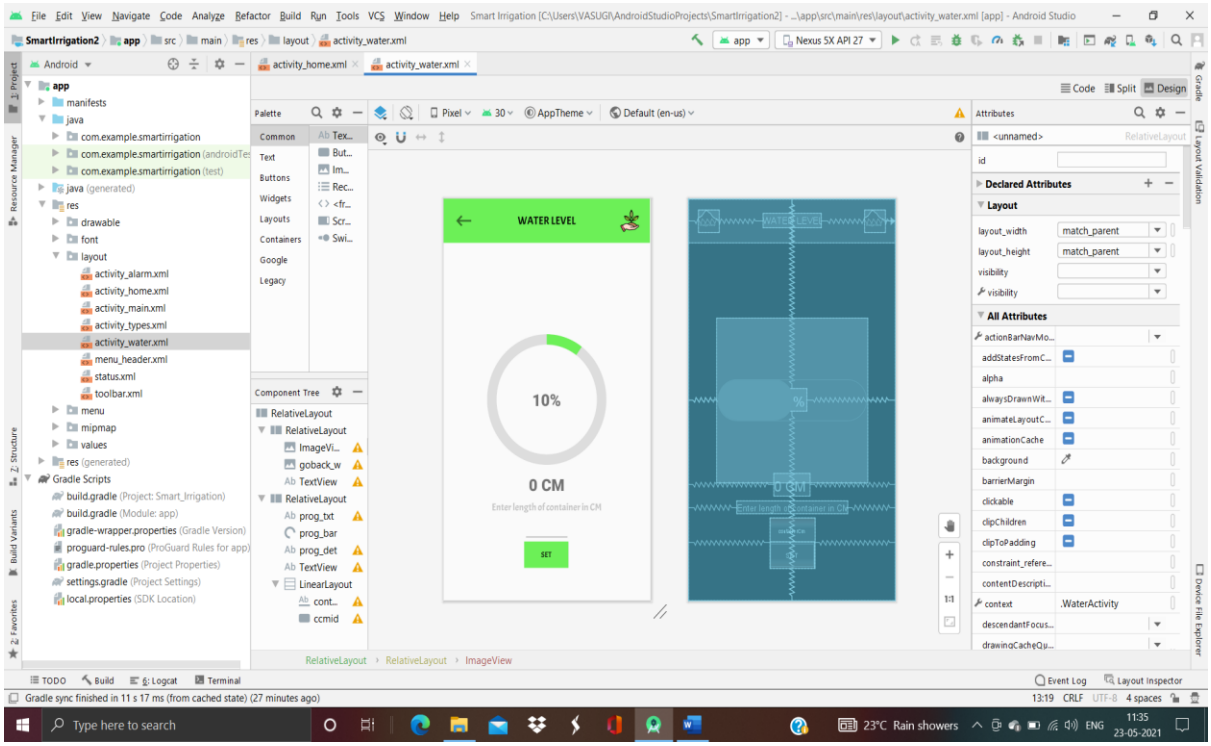
WATER LEVEL.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".WaterActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:background="#6CF057">
        <ImageView
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/planting"
            android:layout_alignParentRight="true"
            />
        <ImageView
            android:id="@+id/goback_w"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/goback"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
```

```
        android:fontFamily="@font/acme"
        android:text="WATER LEVEL"
        android:textColor="#000000"
        android:textSize="20dp" />
</RelativeLayout>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:id="@+id/prog_txt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
           android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:fontFamily="sans-serif-black"
            android:text="10%"
            android:textSize="30dp" />
        <ProgressBar
            android:id="@+id/prog_bar"
            android:layout_width="300dp"
            android:layout_height="300dp"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:indeterminateOnly="false"
            android:progress="10"
            android:progressDrawable="@drawable/circle"
            android:rotation="-90" />
        <TextView
            android:id="@+id/prog_det"
            android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="500dp"
    android:layout_marginEnd="172dp"
    android:layout_marginRight="168dp"
    android:layout_marginBottom="164dp"
    android:fontFamily="sans-serif-black"
    android:text="0 CM"
    android:textSize="30dp" />
<TextView
    android:textSize="18dp"
    android:fontFamily="@font/acme"
    android:layout_width="wrap_content"
    android:textColor="#C3C3C3"
    android:layout_marginTop="550dp"
    android:layout_height="wrap_content"
    android:text="Enter length of container in CM"
    android:layout_centerHorizontal="true" />
<LinearLayout
    android:layout_centerHorizontal="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="580dp"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/containerCm"
        android:fontFamily="@font/acme"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:inputType="number"  />
```

```
            <Button
                android:id="@+id/ccmid"
                android:fontFamily="@font/acme"
                android:background="@color/theme"
                android:text="set"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
        </LinearLayout>
    </RelativeLayout>
</RelativeLayout>
```



TYPES.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TypesActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:background="#6CF057">
        <ImageView
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/planting"
            android:layout_alignParentRight="true"
            />
        <ImageView
            android:id="@+id/goback2"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/goback"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
```

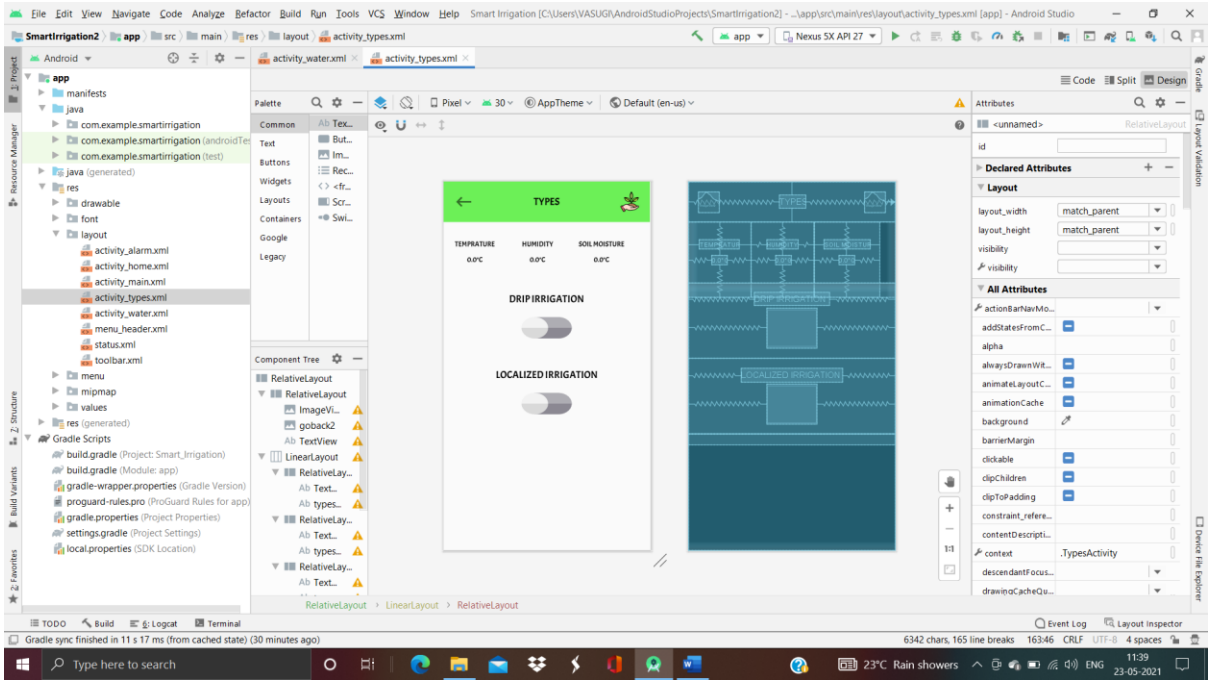
```
        android:fontFamily="@font/acme"
        android:text="TYPES"
        android:textColor="#000000"
        android:textSize="20dp" />
    </RelativeLayout>
    <LinearLayout
       android:layout_width="match_parent"
       android:layout_height="150dp"
       android:orientation="horizontal"
       android:layout_marginTop="80dp">>
       <RelativeLayout
          android:layout_height="match_parent"
          android:layout_width="125dp"
          android:padding="10dp">
          <TextView
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_centerHorizontal="true"
             android:layout_marginTop="25dp"
             android:fontFamily="@font/acme"
             android:text="TEMPRATURE"
             android:textColor="#000" />
          <TextView
             android:id="@+id/typestemp"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_centerHorizontal="true"
             android:layout_centerVertical="true"
             android:fontFamily="@font/acme"
             android:text="0.0*C"
```

```
        android:textColor="#000" />
    </RelativeLayout>
    <RelativeLayout
        android:layout_height="match_parent"
        android:layout_width="125dp"
        android:padding="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="25dp"
            android:fontFamily="@font/acme"
            android:text="HUMIDITY"
            android:textColor="#000" />
        <TextView
            android:id="@+id/typeshumi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:fontFamily="@font/acme"
            android:text="0.0*C"
            android:textColor="#000" />
    </RelativeLayout>
    <RelativeLayout
        android:layout_height="match_parent"
        android:layout_width="130dp"
        android:padding="10dp">
        <TextView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="25dp"
            android:fontFamily="@font/acme"
            android:text="SOIL MOISTURE"
            android:textColor="#000" />
        <TextView
            android:id="@+id/typesoil"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:fontFamily="@font/acme"
            android:text="0.0*C"
            android:textColor="#000" />
    </RelativeLayout>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="200dp"
    android:orientation="vertical">
<RelativeLayout
    android:padding="20dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DRIP IRRIGATION"
        android:textSize="20dp"
```

```
        android:textColor="#000"
        android:fontFamily="@font/acme"
        android:layout_centerHorizontal="true"
        />
    <ToggleButton
        android:background="@drawable/tbbtn"
        android:text=""
        android:id="@+id/dripirr"
        android:layout_width="100dp"
        android:layout_height="80dp"
        android:textOn=" "
        android:textOff=" "
        android:layout_marginTop="30dp"
        android:layout_centerHorizontal="true"
        />
</RelativeLayout>
    <RelativeLayout
        android:padding="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="LOCALIZED IRRIGATION"
            android:textSize="20dp"
            android:textColor="#000"
            android:fontFamily="@font/acme"
            android:layout_centerHorizontal="true"
            />
        <ToggleButton
```

```
            android:id="@+id/otherirr"
            android:layout_width="100dp"
            android:layout_height="80dp"
            android:textOn=" "
            android:textOff=" "
            android:background="@drawable/tbbtn"
            android:layout_marginTop="30dp"
            android:layout_centerHorizontal="true"
            />
    </RelativeLayout>
    <RelativeLayout
        android:padding="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </RelativeLayout>
</LinearLayout>
</RelativeLayout>
```



CUSTOM ALARM.xml CODE:

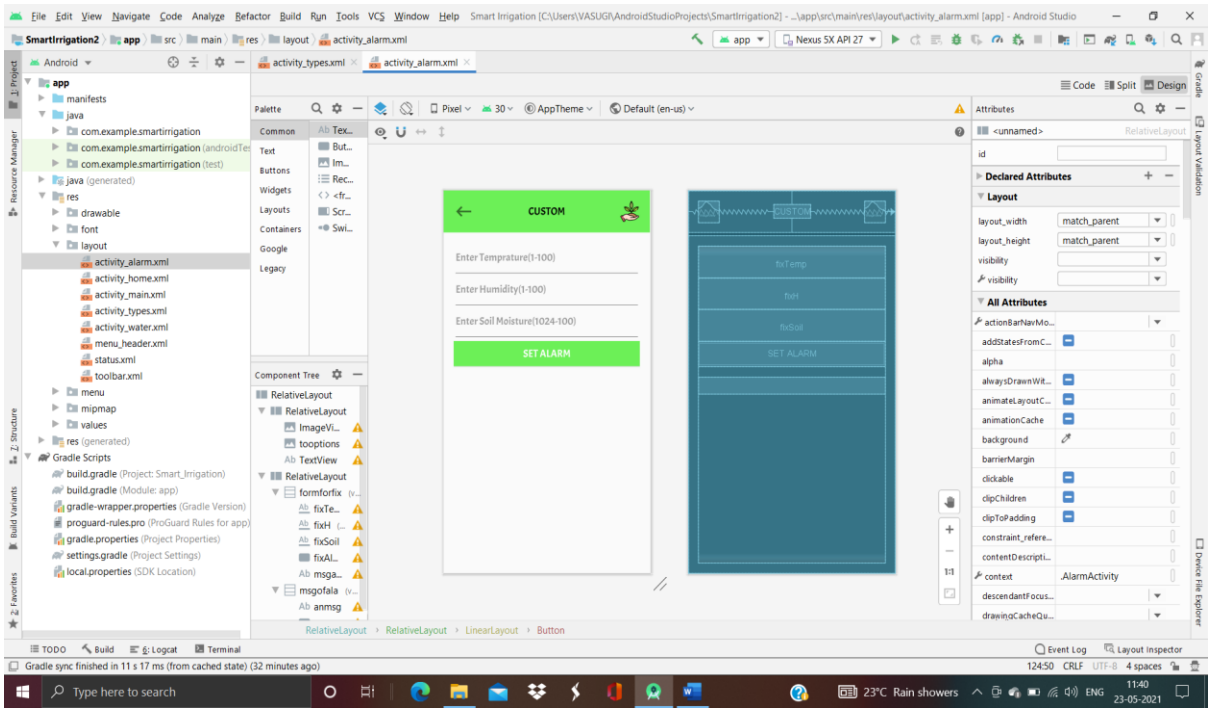
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AlarmActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:background="#6CF057">
        <ImageView
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/planting"
            android:layout_alignParentRight="true"
            />
        <ImageView
            android:id="@+id/tooptions"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/goback"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
```

```
            android:fontFamily="@font/acme"
            android:text="CUSTOM"
            android:textColor="#000000"
            android:textSize="20dp" />
    </RelativeLayout>
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="20dp"
    android:layout_marginTop="85dp">
    <LinearLayout
        android:id="@+id/formforfix"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        >
        <EditText
            android:id="@+id/fixTemp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Enter Temprature(1-100)"
            android:fontFamily="@font/acme"
            android:textColor="#000"
            android:paddingBottom="25dp"
            android:textSize="20dp"
            android:inputType="number"
            />
        <EditText
            android:id="@+id/fixH"
            android:paddingBottom="25dp"
```

```
        android:textSize="20dp"
        android:hint="Enter Humidity(1-100)"
        android:fontFamily="@font/acme"
        android:textColor="#000"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        />
    <EditText
        android:id="@+id/fixSoil"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Soil Moisture(1024-100)"
        android:fontFamily="@font/acme"
        android:textColor="#000"
        android:paddingBottom="25dp"
        android:textSize="20dp"
        android:inputType="number"
        />
    <Button
        android:id="@+id/fixAlarm"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/theme"
        android:text="SET ALARM"
        android:fontFamily="@font/acme"
        android:textColor="#fff"
        android:textSize="20dp"
        />
<TextView
```

```
android:text=""
android:textSize="25dp"
android:visibility="invisible"
android:layout_marginTop="20dp"
android:id="@+id/msgalar"
android:fontFamily="@font/acme"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textColor="#000"
/>
</LinearLayout>
<LinearLayout
   android:id="@+id/msgofala"
   android:visibility="gone"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <TextView
      android:text=""
      android:textSize="25dp"
      android:layout_marginTop="20dp"
      android:id="@+id/anmsg"
      android:fontFamily="@font/acme"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textColor="#000"
      />
       <Button
          android:id="@+id/cancelalarm"
          android:layout_width="wrap_content"
```

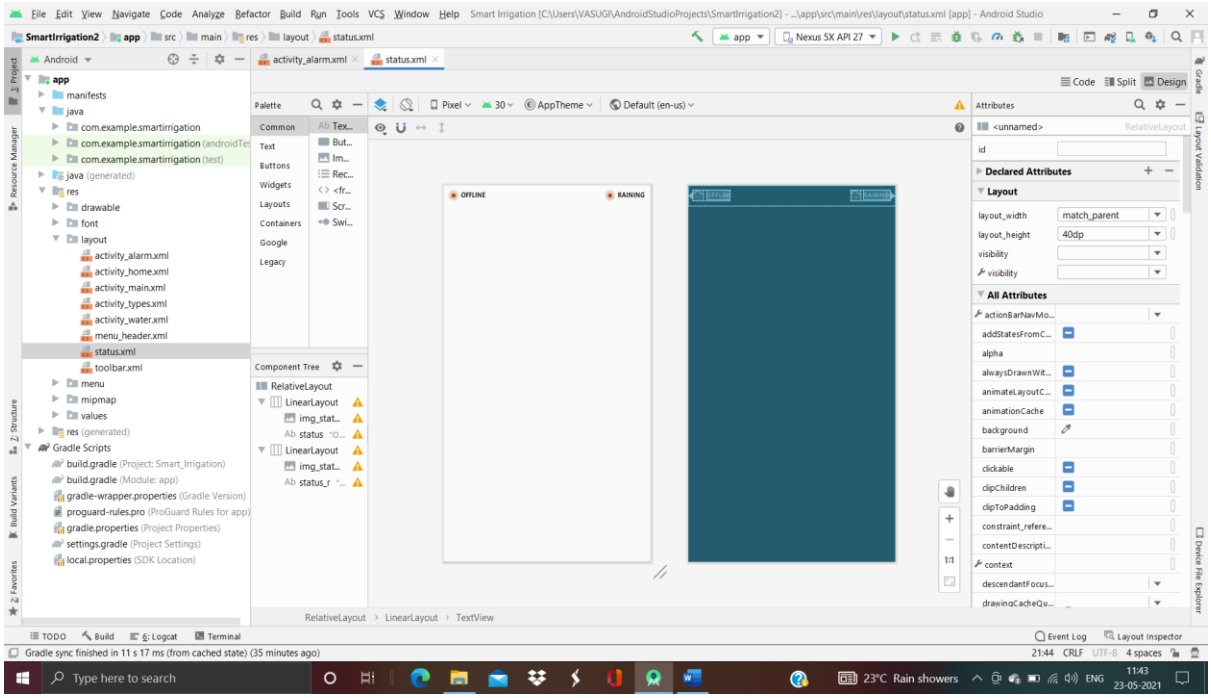
```
            android:layout_height="wrap_content"
            android:background="#F44336"
            android:text="cancel alarm"
            android:textColor="#fff"
            android:fontFamily="@font/acme"
            android:textSize="20dp"
            android:padding="5dp"
            />
    </LinearLayout>
</RelativeLayout>
</RelativeLayout>
```



STATUS.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
android:layout_height="40dp" android:padding="10dp">
   <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:layout_alignParentLeft="true"       >
   <ImageView
      android:id="@+id/img_status"
      android:layout_width="20dp"
      android:layout_height="20dp"
      android:src="@drawable/off"       />
      <TextView
         android:id="@+id/status"
         android:layout_marginLeft="5dp"
         android:layout_width="wrap_content"
         android:text="OFFLINE"
         android:fontFamily="@font/acme"
         android:textColor="#000000"
         android:layout_height="wrap_content"/>
   </LinearLayout>
   <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:layout_alignParentLeft="false"
      android:layout_alignParentRight="true"       >
```

```
        <ImageView
            android:id="@+id/img_status_r"
            android:layout_width="20dp"
            android:layout_height="20dp"
            android:src="@drawable/off"
            />
        <TextView
            android:id="@+id/status_r"
            android:layout_marginLeft="5dp"
            android:layout_width="wrap_content"
            android:text="RAINING"
            android:fontFamily="@font/acme"
            android:textColor="#000000"
            android:layout_height="wrap_content"/>
    </LinearLayout>
</RelativeLayout>
```



CIRCLE.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape
         android:shape="ring"
         android:thicknessRatio="16"
         android:useLevel="false">
         <solid android:color="#DDD" />
      </shape>
   </item>

   <item>
         <shape
            android:shape="ring"
            android:thicknessRatio="16"
            android:useLevel="true">
            <solid android:color="@color/theme" />
         </shape>

   </item>

</layer-list>
```

TOGGLE BUTTON OFF STATE.xml CODE:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="512dp"
    android:height="512dp"
    android:viewportWidth="512"
    android:viewportHeight="512">

  <path
      android:pathData="M376.5,120.5h-241c-8.285,0 -15,6.717 -15,15c0,8.284 6.715,15 15,15v211c-8.285,0 -15,6.717 -15,15c0,8.284 6.715,15 15,15h241c74.715,0 135.5,-60.785 135.5,-135.5S451.215,120.5 376.5,120.5z"
      android:fillColor="#8B8892"/>

  <path
      android:pathData="M376.5,120.5h-241c-8.285,0 -15,6.717 -15,15c0,8.284 6.715,15 15,15v105.499H512C511.999,181.285 451.214,120.5 376.5,120.5z"
      android:fillColor="#AEADB3"/>

  <path
      android:pathData="M271,256c0,-74.715 -60.785,-135.5 -135.5,-135.5S0,181.285 0,256s60.785,135.5 135.5,135.5S271,330.715 271,256z"
      android:fillColor="#F2F2F2"/>

  <path
      android:pathData="M135.5,391.5c74.715,0 135.5,-60.785 135.5,-135.5H0C0,330.715 60.785,391.5 135.5,391.5z"
      android:fillColor="#D9D9D9"/>
</vector>
```

TOGGLE BUTTON ON STATE:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="512dp"
    android:height="512dp"
    android:viewportWidth="512"
    android:viewportHeight="512">

  <path
      android:pathData="M376.5,361.5v-211c8.283,0 15,-6.716 15,-15c0,-8.283 -6.717,-15 -15,-15h-241C60.785,120.5 0,181.285 0,256s60.785,135.5 135.5,135.5h241c8.283,0 15,-6.716 15,-15C391.5,368.217 384.783,361.5 376.5,361.5z"
      android:fillColor="#13C37B"/>

  <path
      android:pathData="M135.5,120.5h241c8.284,0 15,6.717 15,15c0,8.284 -6.716,15 -15,15v105.499H0C0,181.285 60.785,120.5 135.5,120.5z"
      android:fillColor="#61DE56"/>

  <path
      android:pathData="M512,256c0,-74.715 -60.785,-135.5 -135.5,-135.5S241,181.285 241,256s60.785,135.5 135.5,135.5S512,330.715 512,256z"
      android:fillColor="#F2F2F2"/>

  <path
      android:pathData="M376.5,391.5c74.715,0 135.5,-60.785 135.5,-135.5H241C241,330.715 301.785,391.5 376.5,391.5z"
      android:fillColor="#D9D9D9"/>
</vector>
```

TOGGLE BUTTON ON/OFF FUNCTION.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android"
    android:enterFadeDuration="@android:integer/config_shortAnimTime"
    android:exitFadeDuration="@android:integer/config_shortAnimTime"
    >
    <item android:drawable="@drawable/ic_tog_on" android:state_checked="true"/>
    <item android:drawable="@drawable/ic_tog_off" android:state_checked="false"/>
</selector>
```

COLORS.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
    <color name="theme">#6CF057</color>
    <color name="cardv">#E1E1E1</color>
</resources>
```

FONT STYLE.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:app="http://schemas.android.com/apk/res-auto"
        app:fontProviderAuthority="com.google.android.gms.fonts"
        app:fontProviderPackage="com.google.android.gms"
        app:fontProviderQuery="Acme"
        app:fontProviderCerts="@array/com_google_android_gms_fonts_certs">
</font-family>
```

MAIN MENU.xml CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <group>
   <item
android:id="@+id/it1"
android:title="HOME"
android:icon="@drawable/ic_baseline_home_24" />
   <item
android:id="@+id/it2"
android:title="WATER LEVEL"
android:icon="@drawable/ic_drop" />
   <item
android:id="@+id/it3"
android:title="TYPES"
android:icon="@drawable/type" />
   <item
android:id="@+id/it4"
android:title="CUSTOM"
android:icon="@drawable/custom" />
   </group>
</menu>
```

JAVA CODES:

MAIN ACTIVITY.java CODE:

```
package com.example.smartirrigation;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

public class MainActivity extends AppCompatActivity {
private  static int time=4000;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
       new Handler().postDelayed(new Runnable() {
          @Override
          public void run() {

             Intent home = new Intent(MainActivity.this,HomeActivity.class);
             startActivity(home);
             finish();
          }
       },time);
   }
}
```

HOME ACTIVITY.java CODE:

```
package com.example.smartirrigation;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.google.android.material.navigation.NavigationView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class HomeActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
```

```
    static final float end = 0.7f;
    TextView t, h, m, statust, statust_r,sstemp,sshumi,sssoil;
    Double td, hd, md;
    int ri;
    ImageView menuopen, statusi, statusi_r;
    DrawerLayout dl;
    RelativeLayout home_data;
    NavigationView nv;
    String ss;
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef;

    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        statusi = findViewById(R.id.img_status);
        sstemp = findViewById(R.id.stemp);
        sshumi = findViewById(R.id.shumi);
        sssoil = findViewById(R.id.ssoil);
        statusi_r = findViewById(R.id.img_status_r);
        statust = findViewById(R.id.status);
        statust_r = findViewById(R.id.status_r);
        t = findViewById(R.id.t);
        h = findViewById(R.id.h);
        m = findViewById(R.id.m);
        dl = findViewById(R.id.dra);
        NotificationChannel channel = new NotificationChannel("notifyChan", "notufyChan",
NotificationManager.IMPORTANCE_HIGH);
        NotificationManager manager = getSystemService(NotificationManager.class);
```

```
    manager.createNotificationChannel(channel);
    home_data = findViewById(R.id.home_data);
    menuopen = findViewById(R.id.menuopen);
    nv = findViewById(R.id.nav);
    menuopener();
    database_conn();
    status_view();
    alarm_setter();

}

private void alarm_setter(){
    myRef=database.getReference("dataS");
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            Double stemp=snapshot.child("temp").getValue(Double.class);
            Double shumi=snapshot.child("humi").getValue(Double.class);
            Double ssoil=snapshot.child("soil").getValue(Double.class);
            String permission=snapshot.child("setter").getValue(String.class);
            if(permission.equals("1")) {
                if (
                        stemp > Double.parseDouble(sstemp.getText().toString()) &&
                        shumi > Double.parseDouble(sshumi.getText().toString()) &&
                        ssoil > Double.parseDouble(sssoil.getText().toString())
                ) {
                    myRef.child("alarm").setValue("0");



                } else {
```

```
                myRef.child("alarm").setValue("1");
            }
        }

    }

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

    }
  });
}


private void database_conn() {
  myRef = database.getReference("dataR");
  myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
      td = snapshot.child("temprature").getValue(Double.class);
      hd = snapshot.child("humidity").getValue(Double.class);
      md = snapshot.child("soil").getValue(Double.class);
      t.setText(String.valueOf(td) + "°C");
      h.setText(String.valueOf(hd) + "%");
      m.setText(String.valueOf(md));
      sstemp.setText(String.valueOf(td));
      sshumi.setText(String.valueOf(hd));
      sssoil.setText(String.valueOf(md));
    }
```

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

        }
    });

}

private void status_view() {
    myRef = database.getReference("dataR");
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            ss = snapshot.child("status").getValue(String.class);
            ri = snapshot.child("rain").getValue(Integer.class);
            if (ss.equals("0")) {
                statust.setText("ONLINE");
                statusi.setImageResource(R.drawable.on);
            } else {
                statust.setText("OFFLINE");
                statusi.setImageResource(R.drawable.off);
            }
            if (ri == 0) {
                statust_r.setText("RAINY");
                statusi_r.setImageResource(R.drawable.off);
                NotificationCompat.Builder alert = new
NotificationCompat.Builder(HomeActivity.this, "notifyChan");
                alert.setContentTitle("ALERT");
                alert.setContentText("IT'S RAINING");
                alert.setSmallIcon(R.drawable.ic_launcher_foreground);
```

```
                alert.setAutoCancel(true);
                NotificationManagerCompat mnotg = NotificationManagerCompat.from(HomeActivity.this);
                mnotg.notify(1, alert.build());
            } else {
                statust_r.setText("NOT RAINY");
                statusi_r.setImageResource(R.drawable.on);

            }
        }

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

        }
    });


}



  private void menuopener() {
    nv.bringToFront();

    nv.setNavigationItemSelectedListener(this);
    nv.setCheckedItem(R.id.it1);
    menuopen.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
```

```
            if (dl.isDrawerVisible(GravityCompat.START)) {
                dl.closeDrawer(GravityCompat.START);
            } else {
                dl.openDrawer(GravityCompat.START);
            }
        }
    });
    animateNav();

}

private void animateNav() {
    dl.addDrawerListener(new DrawerLayout.SimpleDrawerListener() {
        @Override
        public void onDrawerSlide(View drawerView, float slideOffset) {
            final float diffScaledOffset = slideOffset * (1 - end);
            final float offsetScale = 1 - diffScaledOffset;
            home_data.setScaleX(offsetScale);
            home_data.setScaleY(offsetScale);
            final float xOffset = drawerView.getWidth() * slideOffset;
            final float xOffsetDiff = home_data.getWidth() * diffScaledOffset / 2;
            final float xTranslation = xOffset - xOffsetDiff;
            home_data.setTranslationX(xTranslation);
        }
    });
}

@Override
public void onBackPressed() {
    if (dl.isDrawerVisible(GravityCompat.START)) {
```

```
            dl.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

        switch (menuItem.getItemId()) {
            case R.id.it1:

                Intent intent = new Intent(getApplicationContext(), HomeActivity.class);
                startActivity(intent);
                break;
            case R.id.it2:
                Intent intent2 = new Intent(getApplicationContext(), WaterActivity.class);
                startActivity(intent2);
                break;
            case R.id.it3:
                startActivity(new Intent(getApplicationContext(), TypesActivity.class));
                break;
            case R.id.it4:
                startActivity(new Intent(getApplicationContext(), AlarmActivity.class));
                break;
        }
        return true;
    }
}
```

WATER ACTIVITY.java CODE:

```
package com.example.smartirrigation;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class WaterActivity extends AppCompatActivity {

ImageView back;
ProgressBar prog_bar;
TextView prog_txt,prog_det;
EditText lenSet;
Button cmSet;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_water);
    back = findViewById(R.id.goback_w);
    cmSet=findViewById(R.id.ccmid);
    lenSet=findViewById(R.id.containerCm);
    prog_txt = findViewById(R.id.prog_txt);
    prog_bar = findViewById(R.id.prog_bar);
    prog_det = findViewById(R.id.prog_det);
    backGo();
    cmSet.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            lengthSetter();
        }
    });
    percentageSetter();
}
private void lengthSetter(){
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference("dataS");

    myRef.child("length").setValue(lenSet.getText().toString());

}

private void percentageSetter() {
    final FirebaseDatabase database=FirebaseDatabase.getInstance();
    DatabaseReference myRef=database.getReference("dataR");
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
```

```
            Double wlvl=snapshot.child("waterlvl").getValue(Double.class);
            prog_det.setText(String.valueOf(wlvl+" CM"));
            barO(wlvl,database);
        }

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

        }
    });
}

private void barO(Double wlvl, FirebaseDatabase database) {
    final Double watlvl=wlvl;
    DatabaseReference myRef=database.getReference("dataS");
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            String lengthOc=snapshot.child("length").getValue(String.class);
            Double perc = watlvl/Double.parseDouble(lengthOc);
            Double totperc =perc * 100;
            int percent = (int) Math.round( totperc);
            int percentage=100-percent;

            prog_det.setText(String.valueOf(lengthOc+" CM"));
            prog_txt.setText(String.valueOf(percentage+"%"));
            prog_bar.setProgress(percentage);
            setwaterPerc(percentage);
        }
```

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

        }
      });
  }

  private void setwaterPerc(int percentage) {
      FirebaseDatabase database = FirebaseDatabase.getInstance();
      DatabaseReference myRef = database.getReference("dataM");
      myRef.child("waterPerc").setValue(percentage);
  }

 /*  private barO(final Double wlvl, FirebaseDatabase database) {
      final
  }*/


  private void backGo() {
      back.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
              WaterActivity.super.onBackPressed();
          }
      });

  }
}
```

TYPES ACTIVITY.java CODE:

```
package com.example.smartirrigation;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.Timer;
import java.util.TimerTask;

public class TypesActivity extends AppCompatActivity {
    ImageView goBack;
    TextView typetemp;
    TextView typehumi;
    TextView typesoil;
    ToggleButton dripTB,otherTB;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_types);
```

```
    goBack = findViewById(R.id.goback2);
    typehumi = findViewById(R.id.typeshumi);
    typesoil = findViewById(R.id.typesoil);
    typetemp = findViewById(R.id.typestemp);
    dripTB=findViewById(R.id.dripirr);
    otherTB=findViewById(R.id.otherirr);
    dripTB.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            onclickIrr(dripTB,"dripOn");
        }
    });
    otherTB.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            onclickIrr(otherTB,"other");
        }
    });
    goBack.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            TypesActivity.super.onBackPressed();
        }
    });
    dataOfSensor();
}
private void onclickIrr(ToggleButton mc,String tocol) {
    if(mc.isChecked()){
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("dataM");
```

```
            myRef.child(tocol).setValue("0");
        }
        else{
            FirebaseDatabase database = FirebaseDatabase.getInstance();
            DatabaseReference myRef = database.getReference("dataM");
            myRef.child(tocol).setValue("1");
            if (tocol.equals("dripOn")){
                myRef.child("drip").setValue("1");
            }
        }
    }
    private void dataOfSensor() {
        DatabaseReference dataOfsensor =
FirebaseDatabase.getInstance().getReference("dataR");
        dataOfsensor.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                Double temp = snapshot.child("temprature").getValue(Double.class);
                Double humi = snapshot.child("humidity").getValue(Double.class);
                Double soilmoist = snapshot.child("soil").getValue(Double.class);
                typehumi.setText(String.valueOf(humi));
                typesoil.setText(String.valueOf(soilmoist));
                typetemp.setText(String.valueOf(temp));
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
            }
        });
    }
}
```

ALARM ACTIVITY.java CODE:

```
package com.example.smartirrigation;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
public class AlarmActivity extends AppCompatActivity {
    LinearLayout formforfix;
    LinearLayout layAlarmcancel;
    ImageView tooption;
    EditText fixTemp;
    EditText fixHumi;
    EditText fixSoil;
    TextView msgview;
    TextView anmsg;
    Button fixAlarm;
    Button cancelalarm;
    int flag=0;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_alarm);
    tooption = findViewById(R.id.tooptions);
    layAlarmcancel = findViewById(R.id.msgofala);
    formforfix = findViewById(R.id.formforfix);
    fixTemp = findViewById(R.id.fixTemp);
    fixHumi = findViewById(R.id.fixH);
    fixSoil = findViewById(R.id.fixSoil);
    fixAlarm = findViewById(R.id.fixAlarm);
    cancelalarm = findViewById(R.id.cancelalarm);
    msgview = findViewById(R.id.msgalar);
    anmsg = findViewById(R.id.anmsg);
    tooption.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            AlarmActivity.super.onBackPressed();
        }
    });
    visibilityOfLayout();
    cancelalarm.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            deleteAlarm();
        }
    });
    fixAlarm.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
```

```
            fixAlarmForIrr();
        }
    });
}
private void deleteAlarm() {
    DatabaseReference deleteAlarm = FirebaseDatabase.getInstance().getReference("dataS");
    deleteAlarm.child("setter").setValue("0");
    deleteAlarm.child("alarm").setValue("0");
    deleteAlarm.child("temp").setValue(0);
    deleteAlarm.child("humi").setValue(0);
    deleteAlarm.child("soil").setValue(0);
    AlarmActivity.super.onBackPressed();
}
private void visibilityOfLayout() {
    DatabaseReference visibilityOFL = FirebaseDatabase.getInstance().getReference("dataS");
    visibilityOFL.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            String setter = snapshot.child("setter").getValue(String.class);
            Double t = snapshot.child("temp").getValue(Double.class);
            Double h = snapshot.child("humi").getValue(Double.class);
            Double m = snapshot.child("soil").getValue(Double.class);
            if(setter.equals("1")){
                formforfix.setVisibility(View.GONE);
                layAlarmcancel.setVisibility(View.VISIBLE);
                anmsg.setText("Alarm will on when temprature reaches "+fixTemp.getText().toString()+" ,Humidity reaches "+fixHumi.getText().toString()+" ,soil moisture reaches "+fixSoil.getText().toString());
            }
        }
```

```
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
  }
  private void fixAlarmForIrr() {
    DatabaseReference fixAlarmForIrr = FirebaseDatabase.getInstance().getReference("dataS");
    if (fixTemp.getText().toString().isEmpty() || fixHumi.getText().toString().isEmpty() || fixSoil.getText().toString().isEmpty()) {
        startActivity(new Intent(getApplicationContext(), HomeActivity.class));
        Toast.makeText(this, "Please Enter the values properly", Toast.LENGTH_LONG).show();
    } else {

fixAlarmForIrr.child("temp").setValue(Double.parseDouble(fixTemp.getText().toString()));

fixAlarmForIrr.child("humi").setValue(Double.parseDouble(fixHumi.getText().toString()));

fixAlarmForIrr.child("soil").setValue(Double.parseDouble(fixSoil.getText().toString()));
        fixAlarmForIrr.child("setter").setValue("1");
        msgview.setVisibility(View.VISIBLE);
        msgview.setText("Alarm will on when temprature reaches "+fixTemp.getText().toString()+" ,Humidity reaches "+fixHumi.getText().toString()+" ,soil moisture reaches "+fixSoil.getText().toString());
    }
  }
}
```

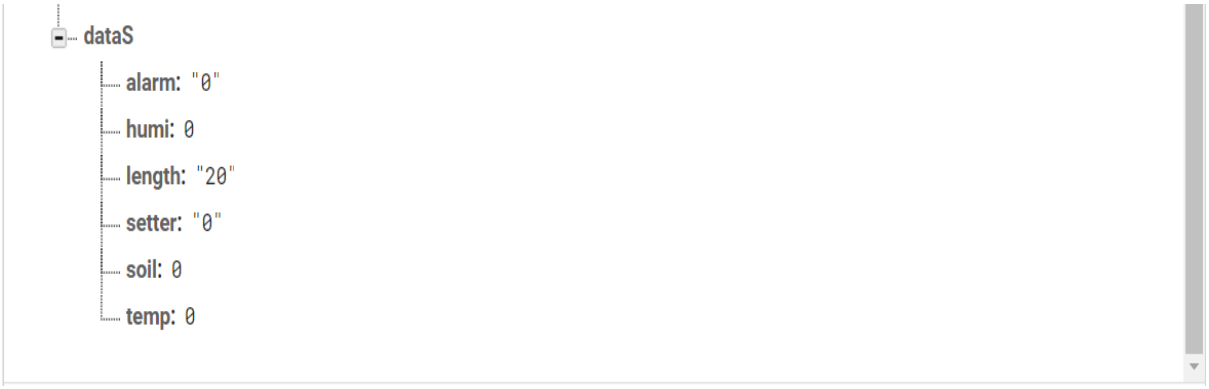
GRADLE SCRIPT CODE:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
android
{
   compileSdkVersion 30
   buildToolsVersion "30.0.2"
   defaultConfig
{
      applicationId "com.example.smartirrigation"
      minSdkVersion 16
      targetSdkVersion 30
      versionCode 1
      versionName "1.0"
      testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
   }
   buildTypes
{
      release
{
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules.pro'
      }
   }
}
dependencies
{
   implementation platform('com.google.firebase:firebase-bom:27.1.0')
   implementation 'com.google.firebase:firebase-analytics'
   implementation fileTree(dir: "libs", include: ["*.jar"])
```

```
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.firebase:firebase-database:19.7.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Appendix 5:

- Firebase Realtime Database



REFERENCES

1. Akyildiz. I.F, W. Su, Y. Sankarasubramaniam, E. Cayirci Wireless sensor networks: a survey Computer Networks, 38 (4) (2019), pp. 393-422.

In-text reference: (Akyildiz. I.F, W. Su, Y. Sankarasubramaniam & E. Cayirci, 2019).

2. Amarendra Goap, Deepak Sharmab, A.K. Shuklab and C. Rama Krishnaa "An IoT based smart irrigation management system using Machine learning and open-source technologies" Computers and Electronics in Agriculture vol. 155 pp. 41-49 2018.

In-text reference: (Amarendra Goap, Deepak Sharmab, A.K. Shuklab & C. Rama Krishnaa, 2018).

3. Andaluz V. H., A. Y. Tovar, K. D. Bedson, J. S. Ortiz and E. Pruna, "Automatic control of drip irrigation on hydroponic agriculture: Daniela tomato production", 2016 IEEE International Conference on Automatic (ICA-ACCA), Curico, 2017, pp. 1-6.

In-text reference: (Andaluz V.H., A.Y. Tovar, K.D. Bedson, J.S. Ortiz & E. Pruna, 2017).

4. "Archana. S, G. Nishanth, E. S. Praveen Kumar, G. Nandha Kumar", Automated Plant Watering System using the Internet of Things Technology in India in "Recent Trends in Microelectronics and Nanoelectronics Volume 2 Issue 3 in Mantech Publications-2017".

In-text reference: (Archana. S, G. Nishanth, E.S. Praveen Kumar & G. Nandha Kumar, 2017).

5. Arvindan. A.N and D. Keerthika, "Experimental investigation of remote control via Android smart phone of Arduino-based automated irrigation system using moisture sensor," 2018 3rd International Conference on Electrical Energy Systems (ICEES), Chennai, 2018, pp. 168-175.

In-text reference: (Arvindan. A.N & D. Keerthika, 2018).

6. Barcelo-Ordinas. J.M, J.P. Chanet, K.M. Hou, J. García-Vidal.A survey of wireless sensor technologies applied to precision agriculture Precision Agriculture'13, Springer (2017), pp. 801-808.

In-text reference: (Barcelo-Ordinas. J.M, J.P. Chanet, K.M. Hou & J. Garcia-Vidal, 2017).

7. Bishnu Deo Kumar, Prachi Srivastava, Reetika Agrawal, Vanya Tiwari, “Microcontroller based automatic plant Irrigation system, “International research journal of engineering and technology (IRJET) on May 2017.

In-text reference: (Bishnu Deo Kumar, Prachi Srivastava, Reetika Agarwal & Vanya Tiwari, 2017).

8. Boursianis M. S. Papadopoulou P. Diamantoulakis A. Liopa-Tsakalidi P. Barouchas G. Salahas et al. "Internet of things (iot) and agricultural unmanned aerial vehicles (uavs) in smart farming: A comprehensive review" Internet of Things pp. 100187 2020.

In-text reference: (Boursianis, M.S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas & G. Salahas, 2020).

9. Devkar. P, Kumari. A, Choudhari. S and Dhanawate. H, 2017. Automated Irrigation System using loT. System, 159(8).

In-text reference: (Devkar. P, Kumari. A, Choudhari. S & Dhanawate. H, 2017).

10. Lopez-Iturri. P, M. Celaya-Echarri, L. Azpilicueta, E. Aguirre, J. Astrain, J. Villadangos, F. Falcone, Integration of autonomous wireless sensor networks in academic school gardens Sensors, 18 (11) (2018), p. 3621.

In-text reference: (Lopez-Iturri. P, M. Celaya-Echarri, L. Azpilicueta, E. Aguirre, J. Astrain, J. Villadangos & F. Falcone, 2018).

11. Meng. X, W. Cong, H. Liang, J. Li Design and implementation of apple orchard monitoring system based on wireless sensor network 2018 IEEE International Conference on Mechatronics and Automation (ICMA), IEEE (2018), pp. 200-204.

In-text reference: (Meng. X, W. Cong, H. Liang & J. Li, 2018).

12. Nishant Kumar Verma and Adil Usman, “Internet of Things (TOT): A Relief for Indian farmers”, “Global Humanitarian Technology conference” on 2017 IEEE.

In-text reference: (Nishant Kumar Verma & Adil Usman, 2017).

13. Prathibha. S.R, A. Hongal and M. P. Jyothi "IoT Based Monitoring System in Smart Agriculture" 2017 Intl. Conf. on Recent Advances in Electronics and Communication Technology pp. 81-84 2017.

In-text reference: (Prathibha. S.R, A. Hongal & M.P. Jyothi, 2017).

14. Priyanka Padalalu, Sonal Mahajan, Kartikee Dabir, Sushmita Mitkarand Deepali Javale, “Smart Water Dripping System for Agriculture/Farming”,2nd International Conference for Convergence in Technology (I2CT), Mumbai, India, 2017, pp. 659 – 662.

In-text reference: (Priyanka Padalalu, Sonal Mahajan, Kartikee Dabir & Sushmita Mitkarand Deepali Javale, 2017).

15. Raiten G. F. Combs et al. "Nutritional ecology: Understanding the intersection of climate/environmental change food systems and health" Agriculture for Improved Nutrition: Seizing the Momentum pp. 68 2019.

In-text reference: (Raiten & G.F. Combs, 2019).

16. Ram. V.V.H, H. Vishal, S. Dhanalakshmi and P. M. Vidya, "Regulation of water in agriculture field using Internet of Things," 2015 IEEE Technological Innovation in ICT for Agriculture and Rural Development (TIAR), Chennai, 2019, pp. 112-115.

In-text reference: (Ram. V.V.H, H. Vishal, S. Dhanalakshmi & P.M. Vidya, 2019).

17. Reilly.M Economic issues in global climate change: agriculture forestry and natural resources CRC Press 2019.

In-text reference: (Reilly, 2019).

18. Saranya Nair .M and Karan Bhatia "A SOLAR TRACKER ASSISTED AUTOMATIC IRRIGATION SYSTEM FOR AGRICULTURAL FIELDS" International Journal of Civil Engineering and Technology (IJCIET) vol. 8 no. 10 pp. 279-287 October 2017.

In-text reference: (Saranya Nair .M & Karan Bhatia, 2017).

19. Sekaran. K, M.N. Meqdad P. Kumar S. Rajan and S. Kadry "Smart agriculture management system using internet of things" Telkomnika vol. 18 no. 3 2020.

In-text reference: (Sekaran. K, M.N. Meqdad, P. Kumar, S. Rajan & S. Kadry, 2020).

20. Singh. P and S. Saikia, “Arduino-based smart irrigation using water flow sensor, soil moisture sensor, temperature sensor andESP8266 WiFi module”, 2019 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Agra, India, 2019, pp. 1-4.

In-text reference: (Singh. P & S. Saikia, 2019).

21. Vellidis. G, M. Tucker, C. Perry, C. Kevin, C. Bednarz. A real-time wireless smart sensor array for scheduling irrigation Computer Electronics. Agric., 61 (1) (2018), pp. 44-50.

In-text reference: (Vellidis. G, M. Tucker, C. Perry, C. Kevin & C. Bednarz, 2018).

22. Viani. F, Bertolli. M, Salucci. A. Polo Low-cost wireless monitoring and decision support for water saving in agriculture IEEE Sens. J., 17 (13) (2017), pp. 4299-4309.

In-text reference: (Viani. F, Bertolli. M & Salucci. A, 2017).

23. Zhou. Y, Q. Zhou, Q. Kong, and W. Cai, Wireless temperature amp; humidity monitor and control system, in 2019 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), April 2019.

In-text reference: (Zhou. Y, Zhou. Q, Kong. Q & W. Cai, 2019).