

# QATra Irrigation System

## Final Report



Texas A&M University at Qatar  
ECEN 404: Electrical Design Lab II

Group Members:

Maryam Al-Emadi

Roqayya AlYousef

Fatima Al-Janahi

Noof Al-Sayed

Mentor: Dr. Hazem Nounou

Course instructor: Dr. Ali Ghrayeb

Date: 26 April 2019

A handwritten signature in black ink, consisting of a long, sweeping horizontal line that curves upwards at the end, with a few smaller loops and a dot below it.

Texas A&M at Qatar University, April 2020  
“An aggie doesn’t lie, cheat or steal, or tolerate those who do.”

# TABLE OF CONTENTS

	Page
ABSTRACT .....	1
<b>CHAPTER I: LITERATURE REVIEW, MOTIVATION, PROBLEM STATEMENT AND PROPOSED SOLUTION</b>	
1.1 LITERATURE REVIEW .....	2
1.2 MOTIVATION .....	2-5
1.3 PROBLEM STATEMENT .....	5
1.4 PROPOSED SOLUTION .....	5
<b>CHAPTER II: BENCHMARKING</b>	
2.1 EXISTING SOLUTIONS .....	6-8
2.1.1 EXISTING PRODUCTS .....	6-8
2.1.2 THE CONVENTIONAL METHODS OF IRRIGATION SYSTEMS .....	8
2.2 PERFORMANCE CRITERIA .....	8-9
2.3 BENCHMARKING TABLE .....	10
2.4 BENCHMARKING STUDY ANALYSIS .....	11-12
<b>CHAPTER III: FUNCTIONAL MODELING</b>	
3.1 DETAILED BLOCK DIAGRAM REPRESENTATION .....	13
3.2 DECOMPOSITION OF BLOCK DIAGRAM FUNCTIONS .....	13-14
<b>CHAPTER IV: PROPOSED DESIGN</b>	
4.1 DETAILED SYSTEM DESIGN .....	15-17
4.1.1 OVERVIEW OF OUR DESIGN .....	15-16
4.1.1.1 HARDWARE COMPONENT DESIGN .....	15-16
4.1.1.2 SOFTWARE COMPONENT DESIGN .....	16
4.1.2 DESIGN COMPONENTS .....	16-17
4.1.2.1 HARDWARE COMPONENTS .....	16-17
4.1.2.2 SOFTWARE COMPONENTS .....	17
4.2 EXISTING STANDARDS, DESIGN CONSTRAINTS AND RISKS .....	18-20
4.2.1 EXISTING STANDARDS .....	18
4.2.2 DESIGN CONSTRAINTS .....	18-19
4.2.3 DESIGN RISKS .....	19-20
4.3 LINK BETWEEN OUR PROJECT AND ECEN COURSES .....	20
<b>CHAPTER V: RESULTS</b>	
5.1 VISUAL PROTOTYPING AND PROGRAM CODE .....	21-25
5.2 FUNCTIONAL PROTOTYPING, TESTING, TROUBLESHOOTING, AND EXPERIMENTAL RESULTS .....	25-36
<b>CHAPTER VI: CONCLUSION</b>	
6.1 CONCLUSIONS, DISCUSSION, VERIFICATION, FUTURE RECOMMENDATIONS, IMPROVEMENTS AND OPTIMIZATIONS .....	37-38
REFERENCES .....	39
APPENDIX .....	40-91

## **Abstract**

Qatar has limited natural water resources, as it has no rivers or lakes. In addition, rain is very rare in Qatar with an annual average of 80 mm of rain water [1]. Groundwater in Qatar supplies the country with 62 million m<sup>3</sup> of water per year, but it does not cover the water needs of the country. Therefore, desalination is being used for domestic usage and treating used water is mainly being used for agriculture. Seawater desalination process is costly and uses natural gas as fuel (a non-renewable energy source), which increases the emission of carbon dioxide in Qatar and accelerates global warming [2]. Cultivation can reduce the impact of global warming and managing the amount of water being used for cultivation can preserve our natural resources. Hence, QATra irrigation system was designed. QATra is an advanced irrigation system that is able to reduce water consumption by checking the soil moisture and controlling the amount of water dispensed. A mobile application, where the user can view the daily water content of the soil is used in our system. The user is informed in case of a fault in the system and can control the irrigation system using the application. The project aims to address environmental issues, cut labor costs, encourage smart innovations and boost cultivation.

## **Chapter I: Literature Review, Motivation, Problem Statement, and Proposed Solution**

### **1.1: Literature Review**

Various research has been conducted regarding existing smart irrigation systems. In this literature review, we will be highlighting the most significant papers.

Ogidan, Onile, Adegboro [3] proposed the idea of a smart irrigation system in their paper. The system has two major components. First, the moisture sensor is used to take readings of the moisture level in the soil. Next, the readings are sent to the microcontroller that analyzes the readings and takes action. If the soil was underwatered the microcontroller sends a signal to the pump to start watering the plant until it reaches the normal moisture level.

Abba, Namkusong, Lee, and Crespo [4] had a similar idea in their research to the previous paper. They proposed the idea of a smart irrigation system that consists of a controller, which is connected to a moisture sensor and a water pump. The controller is taking the measurement from the moisture sensor and deciding whether the plants should be watered or not. The main difference is that they were able to send the data to an analytical and visualization website that views the readings.

Shruthi, Kumari, Rani, and Preyadharan [5] used the Arduino software to design the project. The automated sprinkler system will be activated based on the moisture level readings. The output of the humidity sensors will be displayed on a screen. It minimizes human interference and water runoff over the saturated soil.

Tarange, Mevekari, and Shinde [6] have built an automated wireless irrigation system using Wireless Sensor Network (WSN) and embedded Linux board to collect information from sensor nodes continuously, store it in a database and then provide a web interface to the user. It helps to analyze the soil parameters and reduce water consumption. Web interface and automation helps the user to monitor the system and minimize human intervention.

All of the researchers had done significant work regarding smart irrigation systems. However, some missing pieces of the puzzle need to be put into place to get a more efficient smart irrigation system. QATra irrigation system aims to include these missing pieces by combining an automated irrigation system with control through a mobile application with a fault detection feature.

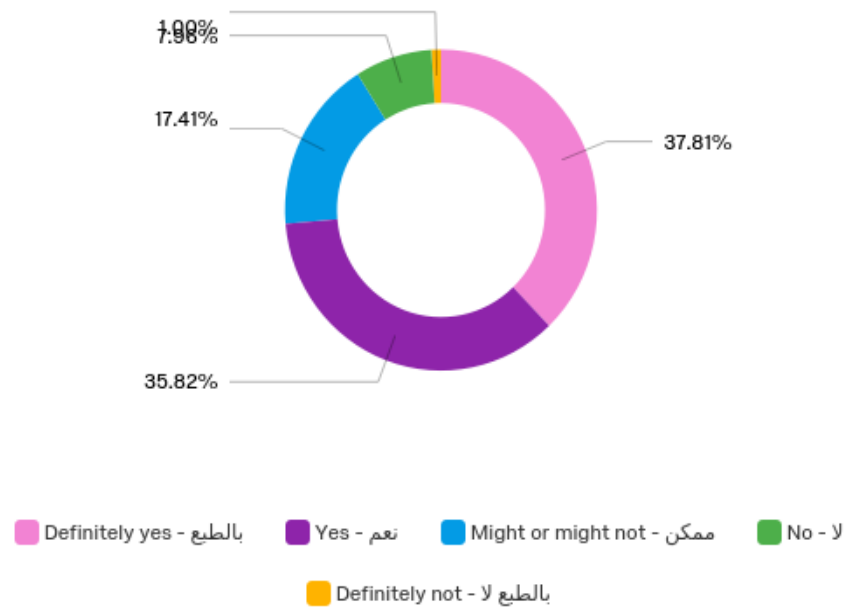
### **1.2: Motivation**

Qatar is a peninsula Arab country that has a total area of 11,586 km<sup>2</sup> which is located on the west line of the Arab Gulf [1]. Qatar is classified as a desert country, where it reaches very high temperatures in summer and have warm winters. As a result, rain is very rare in Qatar with an annual average of 80 mm of rainwater [1]. Also, Qatar has limited natural water resources, as it has no rivers or lakes. However, it has groundwater that supplies the country with 62 million m<sup>3</sup> of water per year [1]. Groundwater does not cover the water needs of the country which is an estimate of 750 million m<sup>3</sup> of water per year. So, Qatar started to desalinate around 540 million m<sup>3</sup> of seawater per year which is mainly for domestic usage and treats 330 million m<sup>3</sup> of used water which is mainly used for agriculture [1].

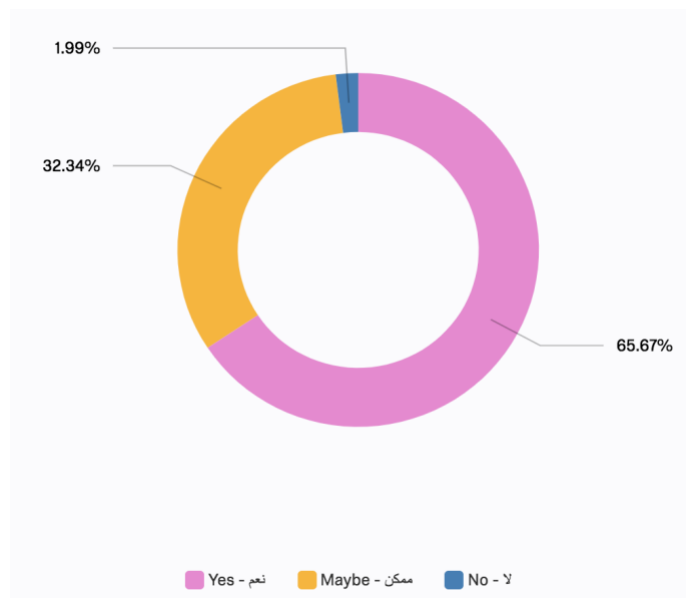
Although the countries' water demands are being supplied by groundwater, seawater desalination and treating used water, Qatar is still facing water scarcity. First, the rate of extracting groundwater is much higher than the refilling rate of accumulated rainfall drained in groundwater [2]. Second, the seawater desalination process is costly and uses natural gas as fuel, which is a non-renewable energy source that increases the emission of carbon dioxide (CO<sub>2</sub>) in Qatar and contributes to global warming [2]. Finally, the consumption of water per person in Qatar is 500 liter per day, which is one of the highest consumption rates in the world [2]. Water scarcity is a heavily contributing factor to the limitation of agriculture and food production. Simultaneously, global warming as a result of high CO<sub>2</sub> emissions is causing an environmental threat to the whole world. Global warming effects can be minimized through cultivating agricultural lands. Therefore, there is an obvious link between water scarcity, agriculture, and global warming [7].

Moreover, the customer needs study, which consists of specialist interviews and an online questionnaire, illustrates that a smart irrigation system is needed to reduce the amount of water being used for cultivation. The questions for the interviews and the questionnaire can be found in **Appendix 1.1 & 1.2**. Engineer Ayman Mashali and Engineer Mohamed Ben Aicha from Kahramaa (Qatar General Electricity and Water Corporation), expressed the fact that the agricultural sector consumes the most amount of water in Qatar, and a smart irrigation system can reduce the amount of water dispensed. Mr. Osman Ahmed Abdalla, an agricultural affairs consultant and an agricultural specialist from the Ministry of Municipality and Environment, revealed that there were no smart irrigation systems in Qatar and having a fault detection feature is more efficient for monitoring than physically checking the system. None of the experts operated a smart irrigation system in their respective companies.

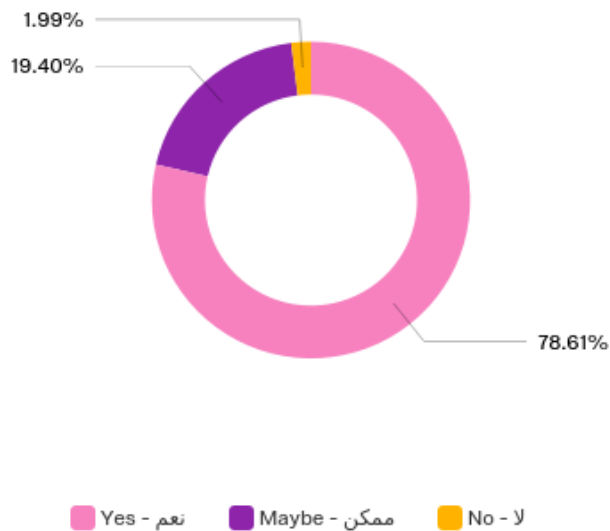
Adding to that, the online questionnaire that was targeting the public, showed that 91.04% of the respondents (total of 202 respondents) were interested or might be interested in a smart irrigation system, as shown in **Figure 1.1**. On the other hand, 61.11% of the 8.96% who were not interested in a smart irrigation system, did not own a farm or a garden. This shows that most of the people who are planting are interested in a smart irrigation system. Moreover, 65.67% of the respondents think that an application feature makes it more convenient to monitor and control the system and 32.34% of them said that it might help, as shown in **Figure 1.2**. Lastly, 78.61% of the respondents agreed that a smart irrigation system helps in reducing water consumption and raises awareness about environmental issues and 19.40% said that it might help, as shown in **Figure 1.3**.



**Figure 1.1:** Respondents' answer to whether they are interested in a smart irrigation system shown as a pie chart.



**Figure 1.2:** Respondents' answers to whether the application feature makes it more convenient for them to control and monitor the system.



**Figure 1.3:** Respondents' answers to does our project helps in reducing water wastage and advocates environmental issues such as global warming

### **1.3: Problem Statement**

QATra is an advanced irrigation system that is able to reduce water consumption by checking the soil moisture and controlling the amount of water dispensed. A mobile application, where the user can view the daily water content of the soil, is used in our system. The user is informed in case of a fault in the system and can control the irrigation system using the application. The project aims to address environmental issues, cut labor costs, encourage smart innovations and boost cultivation.

### **1.4: Proposed Solution**

Our proposed solution is to design an advanced irrigation system, that can avoid overwatering of crops by checking the soil moisture levels. It consists of hardware and software parts. The hardware parts are the microcontrollers, sprinklers, pumps, and relays. The software parts are the mobile application and the server. Using the mobile application, the user will be able to check the moisture levels of the plants, check for faults in the system, and control the irrigation system both automatically and manually.

## **Chapter II: Benchmarking**

Chapter II will focus on the existing commonly used products that will be compared with QATra. Also, how our design meets specific needs with consideration of environmental, public health, welfare, economic, safety, risks, social, global, cultural, political, and ethical factors will be highlighted. Then, based on the benchmarking criteria, the similarities and differences will be examined.

### **2.1: Existing Solutions**

There are different products and papers written about irrigation systems. The functionality of some products is discussed here. This section is divided into two parts. The first part will include products which are currently existing in the market. The second part of this section focuses on the conventional methods of irrigation.

#### **2.1.1: Existing products**

- **Rachio 3:**

The Rachio 3 smart sprinkler system [8] is the most popular product among smart irrigation system devices. The Rachio 3 is a device that replaces the controller of a current automated sprinkler system and can fit up to 16 zones of sprinklers (prices differ depending on the number of zones). The sprinkler cannot be controlled through the device itself instead full control happens through the mobile application of Rachio 3. The mobile application is available for both iOS and Android where Rachio 3 can be connected using the user Wi-Fi network (supports both 2.4GHz and 5GHz). The Rachio 3 mobile application has useful features such as weather intelligence which connects to many different sources of satellite, radar and weather station data, when the user enters the location, in order to check for any rain or wind that might affect the irrigation. If it was found that on a certain day it might rain, then it will stop the irrigation system. Moreover, it allows the user to input a detailed plan for the system such as the water scheduling days, scheduling times, and the duration of watering the plants. The application allows the user to alter the system plan according to the season. The app notifies users when the system is watering the plants.

Although Rachio 3 is a great product, it has some constraints. First, it has a limited number of zones that can be covered. Also, the fault detection feature is not available in Rachio 3. However, a flow meter can be added with additional cost to detect water leakage. Sensors are not supported in this product. Thus, plants can be affected by underwatering or overwatering.

- **Orbit B-hyve:**

Orbit B-hyve Smart Wi-Fi Indoor Timer [9] is a smart watering system. This product cover from 4 up to zones and price varies with the increase in the number of zones. Some of Orbit B-hyve features include weather sensing technology and smart scheduling which identifies watering options based on gathered information on the slope of a crop, type of soil, amount of sunlight and live weather updates. The app works with iOS, Android, or web browser and can be connected using Wi-Fi network (supports 2.4Ghz). The mobile application can create a program for water



scheduling, or the user can do it manually. The user also has a built-in manual watering override which allows to start and stop watering without using the app.

Orbit B-hyve is a very powerful product. However, there are some barriers that can be overcome. For instance, users may not take full advantage of the smart scheduling feature as they might not have enough knowledge and information on the crop. This product does not support any type of sensors so the plant may compromise for overwatering or underwatering.

- **Hydrawise Hunter HC:**

Hydrawise Hunter HC is a smart sprinkler controller [10] which gives detailed reports about the amount of water used, rainfall amount, and also gives alerts about broken pipes, spray heads, faulty wiring, and valves. It's available from 6 to 12 zones and the price varies depending on the number of zones. It also stores past watering history so that the user can view the report. It can be used only indoors. It can connect to a maximum of two sensors which include the Hunter rain sensor. It's available for both iOS and Android devices and connects using Wi-Fi network of bandwidth 2.4GHz and is not compatible with 5GHz.

Despite the fact the Hydrawise Hunter HC is a successful product, it can be further improved and developed. Hydrawise Hunter HC can cover only a limited number of zones. Also, fault detection is supported in this product unless flow meter was installed with additional cost to detect leaking. The rain sensor can be implemented with additional costs.

- **BlueSpray:**

BlueSpray [11] is a smart sprinkler which controls the watering schedule by using the website rather than a mobile application. It uses weather forecasting to schedule the watering time. The number of zones for this product can reach up to 24 zones and the price increases with the increase in the number of zones. It uses weather forecasting, but the controller does not stop the sprinkler if it rains and the rain sensor is required to stop watering the plants during rainy days. History reports of watering schedules can be viewed by using the website. The types of sensors that can be added to Bluespray are rain and flow sensors. However, these sensors do not come with the controller and the user has to buy them separately. It is compatible with both 2.4GHz and 5GHz Wi-Fi networks.

BlueSpray is a very good product. However, it can cover a limited number of zones. Also, the user interface is a web browser, which makes it harder for the user to control and monitor the system. Fault detection feature is not available in this product. However, leakage may be detected if a flow meter was added with additional cost. Moreover, a rain sensor might be added to the controller with extra cost.

### **FCC electrical specification standards:**

All of the four products comply with the FCC rules. Federal Communications Commission (FCC) [12] are in charge of different responsibilities. One of them is setting a number of rules and

regulations in the United States to ensure that devices are at a reasonable price and that devices does not cause harm.

### **2.1.2: The conventional methods of irrigation systems**

The conventional methods of irrigation systems use hoses are pipes that pressurized water flows through it and is used in watering plants. Water is dispensed from the pipes or houses using sprinklers or drip irrigation. During our interview, we asked them Mr. Osman Ahmed Abdalla, an agricultural specialist about current fault detection techniques. He said to detect any faults in the system a person should physically monitor and check the system. The system is controlled using a timer to activate sprinklers for a set amount of time in order to water the plants. This method does not take into account the threshold values of plants and overwater may occur.

## **2.2: Performance Criteria**

After looking at the different solutions whether they were research or products, we found that there were still some missing pieces of the puzzle that need to be put into place in order to get a more efficient smart irrigation system. QATra irrigation system aims to include these missing pieces by combining an automated irrigation system with control through a mobile application with a fault detection feature.

Our design meets specific needs with consideration of the following factors:

- **Environmental:**  
Our project has a direct link to environmental factors where it helps in preserving water resources and contributes to increasing agriculture.
- **Public Health:**  
Our project has an indirect link to public health benefits, where increasing the level of agriculture can increase levels of oxygen and decrease the level of carbon dioxide in the air. This leads to a better quality of living due to cleaner air. Increasing agricultural productivity can also mean better quality of food.
- **Welfare:**  
Our project has an indirect link to welfare. As public health improves, welfare will increase. Moreover, the project promotes smart innovations, which can indirectly help in increasing welfare.
- **Economic:**  
Our project has a direct link to economic factors. Less water will be used for agriculture which will improve the economic benefits. Also, more agriculture will cause a better economy and less dependency on imports. Statistics from the World Bank showed that Agriculture accounted for one-third of the global gross-domestic-product (GDP) in 2014 [13].

- Safety and Risks:  
Our project has a direct link to safety measures since the water used to irrigate the plants might interact with the electrical components. We initially planned to put the electrical components in waterproof, transparent boxes. However, due to the pandemic circumstances, we could not use the 3D printer in the university to print these boxes. Moreover, there might be potential risks that might affect the flow of data that should be kept in mind. Those risks include the stopping of the main microcontroller and losing the Wi-Fi connection. Both the risks and safety constraints are elaborated in Section 4.2.
- Social and Global:  
Our project has an indirect link to social and global factors, since it aims to raise awareness about global warming and water scarcity in the world.
- Cultural:  
Our project has an indirect link to cultural factors, since we aim to create a culture of preserving natural resources.
- Political:  
Our project has an indirect link to political factors, since one of our aims is to preserve natural resources, which is a big part of many political movements worldwide.
- Ethical:  
Our project has direct link to ethical factors, since as we want to be ethical engineers, we have to “disclose any factors that might endanger the public or the environment.” (IEEE code #1)

Therefore, in this benchmarking assignment, we will be comparing products based on technical criteria and the certain considerations that we mentioned.







Technical criteria:

- fault detection
- zones
- user interface
- weather intelligence
- sensors availability

The considerations that will be used in the comparison are:

- cost
- water preserving
- safety
- promoting smart innovations
- Raising awareness (global issues such as: water scarcity and global warming)

### 2.3: Benchmarking Table

Product	Rachio 3	Hydrawise Hunter HC	Orbit B-hyve	BlueSpray	Conventional Irrigation System	QATra
Product Photo						
Cost	\$228 - \$232	\$160 - \$167	\$50.50 - \$62.19	\$294.99 - \$349.99	≈ \$3000 - \$4000 [15]	\$541
Number of Zones	8 or 16	6 or 12	4 or 8	8, 16 or 24	Unlimited	Unlimited (additional cost will be added) *
Sensor Availability	None	Rain Sensor (add on) **	None	None	Rain Sensor (add on) **	Moisture Sensor
Weather Intelligence	Yes	Yes	Yes	Yes	No	No
Fault Detection	No (leakage only if flow meter was added) **	No (leakage only if flow meter was added) **	No	No	No	Yes
User Interface	IOS, Android and web browser	IOS, Android and web browser	IOS, Android and web browser	web browser	None	IOS Application
Preserving Water Resources	Yes, through Weather Intelligence	Yes, through Weather Intelligence or Rain Moisture level if Purchased	Yes, through Weather Intelligence	Yes, through Weather Intelligence	No	Yes, through Soil Moisture level Detection
Safety	Safe for Indoor and Outdoor if Purchased with a Waterproof Enclosure Case	Safe for Indoor	Safe for Indoor and Outdoor if Purchased with a Weather resistant cabinet	Safe for Indoor	Safe for Indoor and Outdoor (depending if the controller is water resistant or not)	Safe for Indoor and Outdoor
Promoting Smart Innovation	Yes	Yes	Yes	Yes	No	Yes
Raising Awareness	Yes	Yes	Yes	Yes	No	Yes

\*Our prototype will contain only 4 zones. However, due to star network topology unlimited number of zones can be added.

\*\*Those items are sold separately from the main product with additional cost.

## **2.4: Benchmarking Study Analysis**

The benchmarking table will be further discussed in this section. It highlights the similarities and differences with the other products.

- Rachio3 VS. QATra

### **Similarities:**

Some of the common features that the Rachio 3 and QATra share is that they both can navigate the irrigation system through the mobile application. The user is able to turn off the system and have real-time control.

### **Differences:**

When comparing the Rachio 3 with QATra, we can distinguish that for QATra irrigation system, the user does not need to specify the water scheduling times through the mobile application since the system contains soil moisture sensors that takes readings in time intervals to determine when the system is required to water the crop automatically. A feature of QATra irrigation system it that it notifies the user in case of potential fault through its mobile application and allows the user to view data of the soil moisture level from the sensors. Moreover, a unique feature that the Rachio 3 controller has is that it uses weather intelligence to determine priorly if any rain might occur in order to stop the system from irrigating. The Rachio 3 mobile application is supported for both iOS and android where QATra's mobile application is only supported for iOS. Furthermore, the Rachio 3 controller can navigate only up to 16 zones where as QATra irrigation system can navigate unlimited number zones due to its star network topology. For both systems prices differ depending on the number of zones.

- Hydrawise VS. QATra

### **Similarities:**

Both Hydrawise and QATra enable the user to remotely control the irrigation system. Also, both systems allow the user to monitor the irrigation process through a mobile application.

### **Differences:**

Comparing the Hydrawise and QATra irrigation system we can analyze that Hydrawise is an irrigation controller that waters the plants due to a watering schedule and watering duration provided by the user. On the other hand, QATra is a smart irrigation system that waters the plants according to their water needs due to the installation of soil moisture sensors. Weather intelligence is a feature included in Hydrawise, but it is not included in QATra. In Hydrawise, user can choose between web browser, IOS or Android application to interfere with the irrigation system, or even manually using the screen on the controller but in QATra there is only an IOS mobile application. In addition, the Hydrawise controller can navigate only up to 12 zones whereas QATra irrigation system can navigate unlimited number zones due to its star network topology. The price of each product differs due to different number of zones.

- **Orbit B-hyve Smart Wi-Fi Indoor Timer VS. QATra**

**Similarities:**

Some of the features that Orbit B-hyve Smart Wi-Fi Indoor Timer and QATra share are that they both can control the watering of the crops through the mobile application.

**Differences:**

Comparing the Orbit B-hyve and QATra irrigation system we can differentiate that QATra uses soil moisture sensors to measure the soil moisture level in order to determine whether the crops need watering according to a water level threshold. Hence, the system irrigates the plant automatically without a requirement from the user through the application but, the user is still able to interfere with the system. On the other hand, the Orbit B-hyve Smart Wi-Fi Indoor Timer has a smart watering feature where the user is required to input detailed information about the crops in order for the system to figure out the right amount of water for irrigation. Moreover, the Orbit B-hyve uses weather sense technology that receives local weather data and alter the controller's water needs based on the data to deliver the right amount of water to crops. Furthermore, the Orbit B-hyve does not take into consideration any fault detection of the system whereas QATra irrigation system notifies the user about potential fault in the system. In addition, the Orbit B-hyve controller can navigate only up to 16 zones whereas QATra irrigation system can navigate unlimited number zones due to its star network topology (for both systems prices differ depending on the number of zones). The Orbit B-hyve mobile application is supported for both iOS and android whereas QATra's mobile application is only supported for iOS.

- **BlueSpray VS. QATra**

**Similarities:**

Both BlueSpray and QATra provide the users with the ability to monitor and control the irrigation system through a user interface such as a website or mobile application.

**Differences:**

Comparing BlueSpray and QATra irrigation systems, we can examine that BlueSpray irrigation system controller depends on the user's input of watering schedule and duration of crops. However, QATra uses sensors to identify the water needs of the plants and water it accordingly. Weather intelligence feature that is available in BlueSpray but not in QATra. Also, QATra irrigation system has fault detection feature which is not available in BlueSpray controller. Coming to the user interface, the user can control and monitor the irrigation system through the website in BlueSpray controller, and through IOS mobile application in QATra irrigation system. The two products have different prices due to different features and number of zones.

After comparing different products to our design, it was concluded that our design fills the gaps of those products through implementing smart watering using moisture sensors, fault detection feature, and monitor the soil content and irrigation process through the mobile application.

## Chapter III: Functional Modeling

### 3.1: Detailed Block Diagram Representation

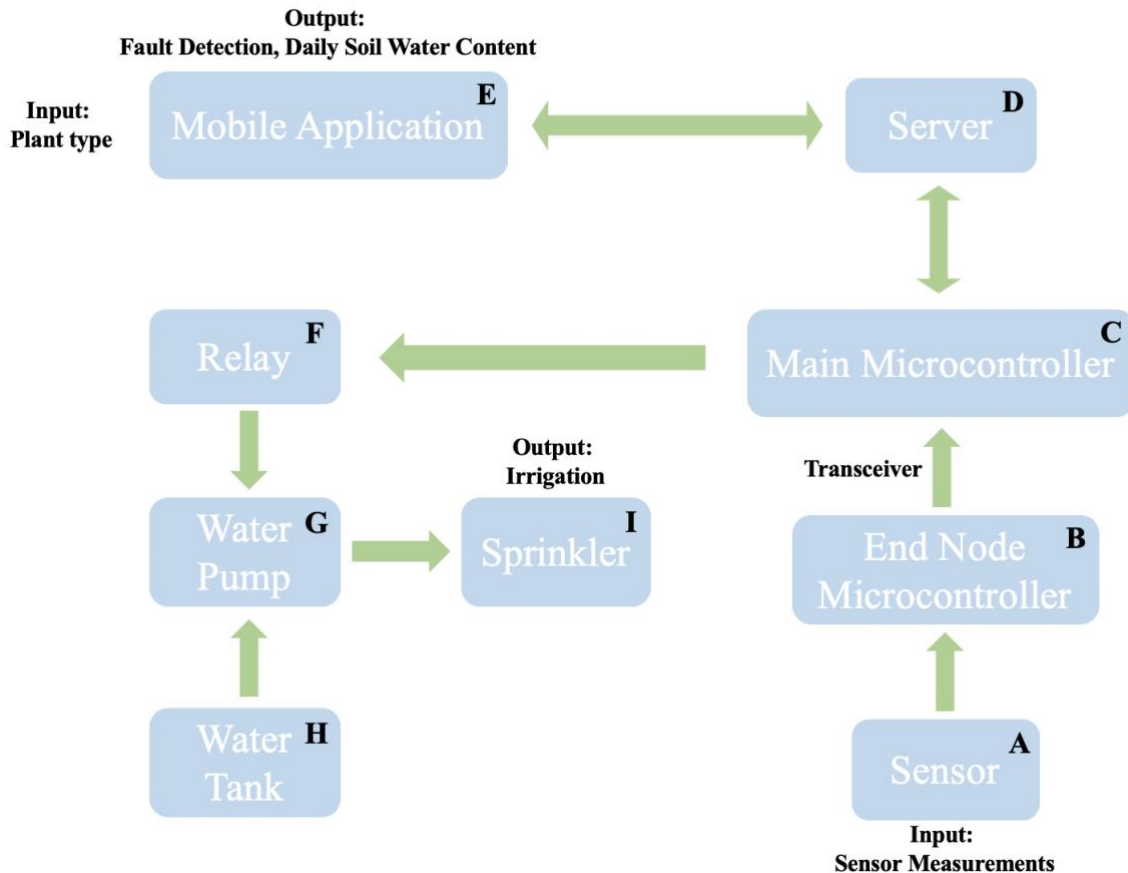


Figure 3.1: Detailed Block Diagram of Our System

### 3.2: Decomposition of Block Diagram Functions

#### Sensor (A) - End Node Microcontroller (B) Connection:

Each soil moisture sensor is paired with a small end node microcontroller. The sensor has a wired connection with the microcontroller. Each microcontroller-sensor pair (we have a total of 4 pairs in our prototype) is placed near the root zone of the plant. Each sensor will be responsible for measuring the soil moisture of a specific portion of the agricultural land (where it will have certain sprinklers responsible for it). This is done to have control over individual sections of the whole agricultural land. The job of the end node microcontroller is to take the soil moisture readings that are measured from the sensor to the main microcontroller.

#### End Node Microcontroller (B) - Main Microcontroller (C) Connection:

The end node microcontroller transmits the data gathered from sensor to the main microcontroller. The data is transmitted using a transceiver which follows the Enhanced ShockBurst protocol. Enhanced ShockBurst used to establish a wireless sensor network. This standard will be further

discussed below. The main microcontroller will receive moisture level readings from the different zones through the end node microcontroller and will receive irrigation mode and plant type from the server. Using this information, the main microcontroller will set the maximum and minimum moisture levels threshold for each zone. Then, the main microcontroller will check if there is a fault in the system, if there is a fault, it will send a notification to the server. After that, the main microcontroller will check the irrigation mode. If it is automatic, and if the moisture level is below the minimum water threshold level, then the main microcontroller will run the watering system (connected to the sprinkler) to irrigate the plants. However, if the water level was above the maximum water threshold level then the main microcontroller will not allow the irrigation of plants. If the irrigation mode was manual, the user will control the irrigation system through the mobile application. The details of the communication between the main microcontroller and the water system will be further discussed below.

#### Main Microcontroller (C) - Server (D) Connection:

The main microcontroller communicates with the server using WIFI connection. The server is a software platform or hardware device that organizes the networking between the two devices. The purpose of the server is to continuously update the mobile application with the data collected as well as receive instruction from the application (user) to interfere with the system in which the server will communicate back to the main microcontroller. This will be further discussed below.

#### Server (D) - Mobile Application (E) Connection:

The server sends data and receives instructions from the mobile application continuously. This includes soil moisture readings or notification related to fault detection (where it informs the user about a potential problem). Also, the user can control the irrigation system through the mobile application and input the plant type, which will communicate back to the server, and to the main microcontroller.

#### Main Microcontroller (C) - Relay (F) Connection:

The main microcontroller will turn the relay switch on or off depending on the irrigation mode and soil moisture level. The relay is an electrical on/off switch used to control the water pump. Once the relay is turned on, the pump will supply water to the system. The input of the relay switch can come from end nodes (automation) or from the user through the mobile application (control). This will result in controlling the flow of water.

#### Relay (F) - Water Pump (G) Connection:

The relay will switch the water pump on or off depending on the sensor readings or user's control using the mobile application. In our prototype, there are four relays and four water pumps. This was done due to the feature that allows each water sprinkler to be controlled individually.

#### Water tank (H) - Water Pump (G) - Water Sprinkler (I) Connection:

The water pump will be connected through a pipe network to the water tank on one side, and the water sprinkler on the other side. The water pump will receive the signal from the relay when it needs to be on or off. If it has to be on, it will pump water from the tank to the sprinkler. As mentioned above, there are four water pumps, each of them is connected to a sprinkler.



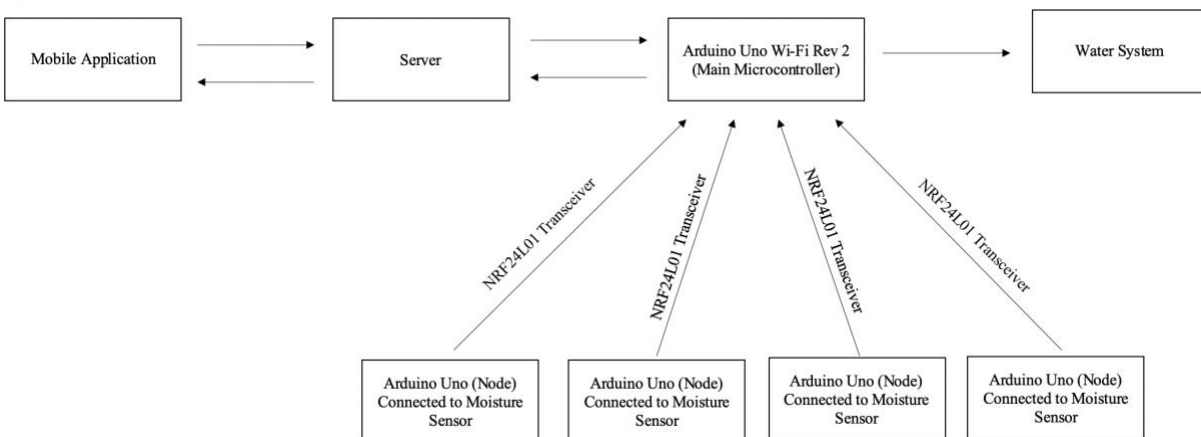
## Chapter IV: Proposed Design

### 4.1: Detailed System Design

QATra is an advanced irrigation system that aims to reduce water consumption through controlling the amount of water being used in agriculture. A mobile application, where the user can view the daily soil water content, control the irrigation system, and get notified in case of a potential fault in the system, is used in QATra. By using the mathematical, scientific and engineering concepts we were able to come up with the proposed design.

#### 4.1.1: Overview of our design

Our proposed design is summarized below in **Figure 4.1**. Soil moisture level readings will be taken from the end nodes (sensor-microcontroller pair) and will be transmitted to the central node (main microcontroller). The water system (water pump, relay and sprinkler) will turn on and off according to the minimum and maximum threshold values or user's input. Data from end nodes will be transmitted wirelessly using a transceiver to the main microcontroller. Then the main microcontroller will transmit the readings of the system to the server. Finally, daily moisture level data will be available in the mobile application for the user to monitor and control.



**Figure 4.1: Wireless Sensor Network Proposed for our Project**

#### 4.1.1.1: Hardware Component Design

This irrigation system consists of microcontrollers, water pumps, relay switches, sprinklers, moisture sensors, transceivers and batteries. The scientific principle behind the moisture sensor is the soil resistivity measurements, which measures the conductivity of the soil. Different moisture sensors are placed near the root zone, to get accurate readings of soil moisture levels as suggested by Mr. Osman due to his experience in agriculture. The moisture sensor that is connected to a microcontroller will be taking data frequently. When the moisture level hits a minimum threshold, the sprinklers will turn on and when it reaches a maximum threshold, the sprinklers will turn off. If the moisture sensor detects a value that is higher than the threshold maximum, the sprinklers

will not turn on (the case of a rainy day). Those threshold values will be taken from scientific sources such as Food and Agriculture Organization (FAO).

#### **4.1.1.2: Software Component Design**

With the fast growth of agriculture and the use of IoT, introducing an application software that accompanies the advanced irrigation system can help tremendously in real-time monitoring and controlling the irrigation system.

Using the moisture sensors, physical quantities will be measured from the soil and transferred as electrical signals connected to the end node microcontroller. Our suggested and preferable topology is the star network topology as shown previously in **Figure 4.1** which is supported by the Enhanced Shockburst standard. The system uses star network topology so that additional end nodes can be added without complicating the system.

Each end node will have a transceiver which follows the Enhanced Shockburst protocol and sends data to the main microcontroller. Enhanced Shockburst uses wireless communication which is an engineering principle to transmit and receive data from end nodes to the main microcontroller without a physical connection.

The main microcontroller has the ability to connect to the Wi-Fi due to its built-in crypto chip accelerator. The main microcontroller is connected to a server (Firebase) that can store data of the soil moisture readings. The server is also connected to the mobile application and so it provides the user with real time soil moisture readings.

This mobile application is used to keep records of soil conditions such as soil water levels and controls the irrigation system. As mentioned earlier, this application allows the user to interact and interfere with the irrigation system. Moreover, a vital feature of this application is that it notifies the crop owner of a potential fault by sending an alert. This can be done by monitoring the measured soil moisture level every hour. It checks to see if these monitored values are higher than the maximum threshold or below the minimum threshold. These readings are compared to see if the moisture value above the maximum keeps on increasing or the value below the minimum keeps on decreasing. In case that the moisture level keeps increasing or decreasing for a long period of time, for different reasons i.e. water leakage, the user will be notified that there might be a potential fault in the system through the mobile application. Programming will be used to establish this algorithm which is an Engineering principle.

#### **4.1.2: Design Components**

##### **4.1.2.1: Hardware components**

The hardware components in our system include the physical equipment needed to establish the communication system between components of the advanced irrigation system. The communication system used in our design is WSN which includes end nodes (sensor-microcontroller pair) and a main microcontroller that is connected to the water system (i.e. water

pump, relay and sprinklers). A transceiver is used to transmit moisture level data from the end node wirelessly. These are the specific hardware components used:

- **Grove Moisture Sensor**  
Sensors convert the measured physical quantities to electrical signals that are then interpreted. Each end node will include a soil moisture level sensor used to measure the volumetric water levels in the soil.
- **Arduino Uno Rev 3**  
The Arduino Uno microcontroller will be placed in each node to digitize physical parameters and send them to the main microcontroller using the transceiver.
- **Arduino Uno Wi-Fi Rev 2**  
The Uno Wi-Fi Rev 2 is used as the main microcontroller where it receives data from the end nodes, analyze the readings, and transmits these data to the mobile application through the Firebase server.
- **NRF24L01 Transceiver**  
This transceiver used to transmit data from the end node to the main microcontroller wirelessly.
- **Relay (electrical switch)**  
The relay switch is used to control the water pump.
- **Source/Battery**
- **Small water tank, water pumps, pipes and sprinklers**
- **iPhone 8**

#### **4.1.2.2: Software components**

The software will allow the hardware components to communicate wirelessly with each other and/or with the server. In the advanced irrigation system, the system is programmed to allow two-way communication which will allow controlling and monitoring the soil moisture level manually (by the user through the mobile application) or automatically. These are the specific software systems used:

- **Arduino Integrated Development Environment (IDE)**  
The Arduino Uno is a programmable microcontroller, Integrated Development Environment (IDE) software is used to program the microcontroller.
- **XCode**  
XCode is used for IOS application development.
- **Firebase**  
Firebase is a backend server which stores data and connects the mobile application to the irrigation system.

The total price of our proposed design is \$520. This price is not the final product price since building a prototype usually costs more than bulk production.

## **4.2: Existing Standards, Design Constraints and Risks**

### **4.2.1: Existing standards**

The two standards that are used in our system are the Enhanced Shockburst [14],[15] standard that is used to establish the wireless sensor network and the IEEE 802.11b/g/n [16] that is used for the Wi-Fi connection of the main microcontroller.

#### 1. Enhanced Shockburst

- The standard we are complying with is Enhanced Shockburst. It is a protocol that is used to establish the WSN. The network topologies that it supports are star, tree and mesh. Our suggested topology and the one we implemented is the star network topology. The end nodes can be added in the star network topology, without complicating the system. However, if there is a problem in the central node, the server will not be able to receive the soil moisture readings as it will be mentioned in Section 4.2.3. The bandwidth in Shockburst is 2.4GHz. The transceiver is used to transmit and receive data up to a range of 100 meters between end nodes and the main microcontroller.

#### 2. IEEE 802.11b/g/n

- This standard is used for the main microcontroller's Wi-Fi, this is because the main microcontroller should be connected to the server using Wi-Fi.

### **4.2.2: Design Constraint**

Resources constraint:

- We had an issue with determining where the sensors will be placed. We interviewed Mr. Osman Ahmed Abdalla (Agricultural specialist) to determine which placement gives the optimal soil moisture reading in order to achieve the best watering conditions. Mr. Osman explained that sensors must be placed near the root zone and that the root zone differs for different plant species, so it affects the level of soil depth for sensor placement. This constraint was resolved.
- Moreover, the lack of knowledge and inexperience in farming increases the possibility of misinterpreting readings which can affect the whole irrigation system. This constraint was explored and tackled during the sensor testing phase.

Technical constraints:

- Ensuring well functionality of the application (user-friendly, free of defects and app complication) is crucial, especially if we consider that it was developed by beginners in application development. Therefore, testing and simulating the application thoroughly (with advice from specialists and general people) on iPhone 8 to make sure it works properly.

Environmental constraint:

- Depending on the geographical locations, weather and climate change can be very unpredictable. Heavy rainfall or dry weather can have an extreme effect on the soil moisture water level, where it can drive water level to be under soil water threshold or over the threshold. This constraint was hard to tackle, since nature is unpredictable.

Health and Safety constraint:

- Part of the irrigation system consists of a water source (sprinkler) connected to electrical components (sensors and controllers). The combination of electricity and water can cause a dangerous hazard. Therefore, all electrical appliances should be in waterproof, transparent boxes. We already planned and designed the waterproof cases to use for our design but unfortunately the timing of the COVID-19 pandemic made it difficult to execute it since the university's doors were closed.

Social constraint

- Many consumers or even farmers may not accept smart and new innovative technology to be used in life's necessity such as agriculture. Therefore, it is important to raise awareness of how new technologies can be used for good (i.e. reducing water wastage of irrigation).

Ethical constraint

- Ensuring that every process of our proposed design is complied with the IEEE code of ethics [19].

Financial\Economics constraint

- The cost of this advanced irrigation system is a constraint since our design is sold as a whole system (i.e. with pipes, sprinklers, water pump, etc.) rather than just an add-on controller to current irrigation systems. Therefore, the initial price is higher. With mass production, the cost of the system will be reduced.

Political constraint

- There is no direct political constraint that is related to our project.

### **4.2.3: Design Risks**

There are two main risks to our proposed design:

#### 1. The main microcontroller

- The main microcontroller is the main link between the end nodes, water system and the mobile application. If the main microcontroller stopped working then it will not be able to receive moisture readings from end nodes, thus, the automatic control of sprinklers will stop working. Also, the main microcontroller will not be able to receive instructions from the mobile application.

## 2. The Wi-Fi connection

- Wi-Fi connection is required in the system in order to send data or receive data from the mobile application. If the Wi-Fi connection was lost, then the user will not receive any data and will not have the ability to control the system.

### **4.3: Link Between Our Project and ECEN Courses**

#### **ECEN 420: Linear Control Systems**

In ECEN 420 course, Linear Control Systems, we looked at control systems; specifically, automatic control systems. A control system consists of a process/system and a controller. A system can be open-loop or closed-loop. A closed-loop system takes measurements of the output and has a feedback that compares those values with a desired output. An open-loop system has no feedback (the process is controlled directly). Our project will be a closed loop system, since measurements from the soil (moisture level) will either start the irrigation or stop the irrigation of the plants. The controller of the system is the main microcontroller and the process is the irrigation of the plants.

#### **ECEN 449: Microprocessor System Design**

This course taught us how to work with microcontrollers and how there are many types of processors used in the microcontroller intended for different jobs. We learned different types of serial communications used in order to transmit or receive data and how to use and operate sensors. This is all very beneficial to our project because we have some background on how to connect our system together (i.e. sensors and microcontrollers) and are able to determine the flow of data transmitted. Also, we can get familiar with the data the sensor generates by looking through the datasheets.

#### **ECEN489: Cyber Security**

In ECEN 489: Cyber Security we studied eight different network topologies with their advantages and disadvantages. Since Enhanced Shockburst only supports three of the studied topologies which are: star, tree and mesh, only those three were being considered. This helped us in the decision making of the network topology of our design and the risk that might accompany that topology. This is helpful for the wireless connection between the main microcontroller (central node) and the end nodes.

#### **ECEN 210: Computer Programming and Algorithms**

There is an obvious link between our project and ECEN 210: Computer Programming and Algorithms. ECEN 210 course taught us the basics of computer programming in C language and how to build an algorithm to solve a specific problem. Thus, it is easy for us to program using C or any other language; since the basics of all programming languages are similar.

To build our prototype, we will have to implement knowledge gained from ECEN 210 to program different components of our design. For instance, the Arduino will be programmed using the C language and the mobile application will be programmed using Swift language. Also, to implement our fault detection feature, a well-developed algorithm should be written.

## Chapter V: Results

### 5.1: Visual Prototyping and Program Code

Our system consists of various components that are linked together. To develop a better understanding of our system and illustrate how each component should work, program flows and diagrams were created for our three main components.

- **Mobile Application**

The mobile application was developed using XCode, an integrated development environment for IOS applications. The purpose of the mobile application is to give the user the ability to control and monitor the irrigation system. The friendliness and ease of the mobile application was considered while developing the mobile application for our targeted audience. The following **Figure 5.1** shows the launching screen of our application. The launching screen shows our logo for a few seconds and then automatically moves to the main screen. The main screen of the mobile application shows several options (**Figure 5.2**). The options are: control the irrigation system, fault detection, check the soil moisture levels in different zones, and learn about QATra.



Figure 5.1: Launching screen.

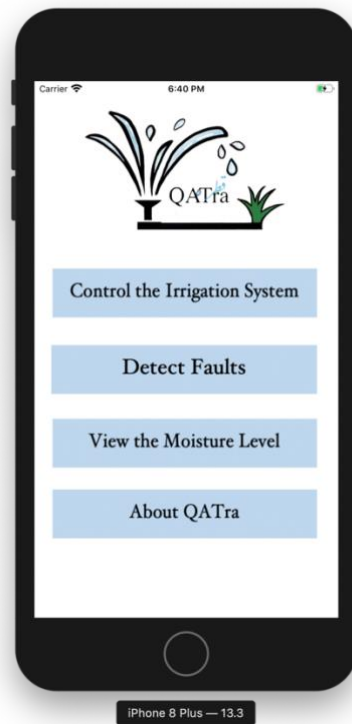


Figure 5.2: Main screen.

Control irrigation system page (**Figure 5.3**) will give the user the choice to choose the irrigation mode and the plant type for each of the four zones. While the fault detection page (**Figure 5.4**), informs the user of the possible faults in the system. The moisture level page (**Figure 5.5**) is capable to view the soil moisture level for the four zones. Finally, the last page (**Figure 5.6**) shows a small paragraph about QATra Irrigation System.

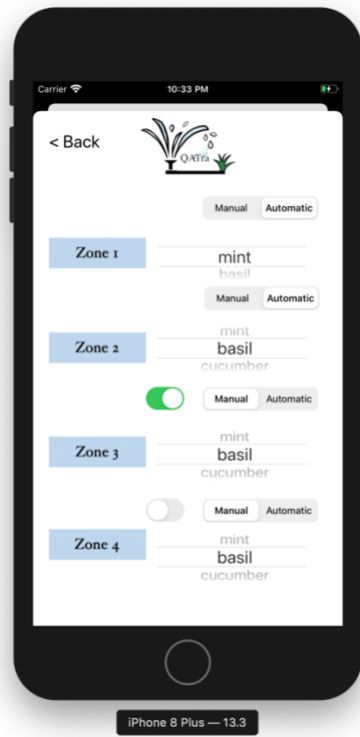


Figure 5.3: Control the irrigation system page.

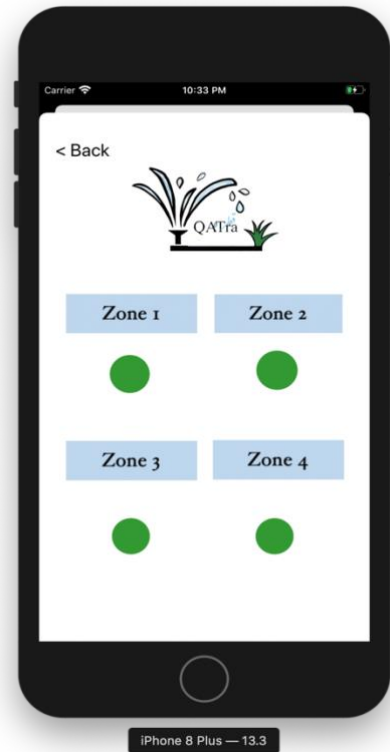


Figure 5.4: Fault detection page.

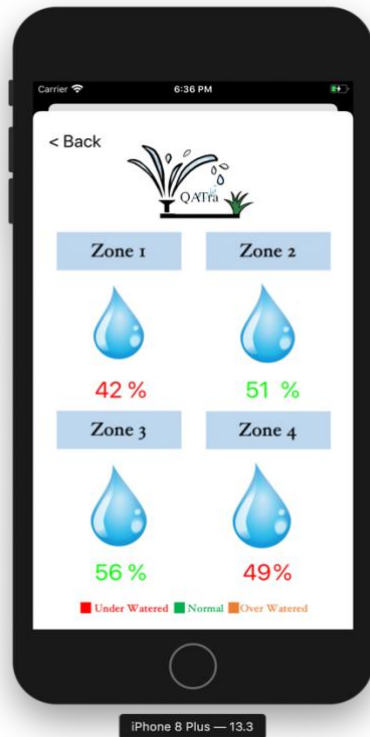


Figure 5.5: Soil moisture level page.

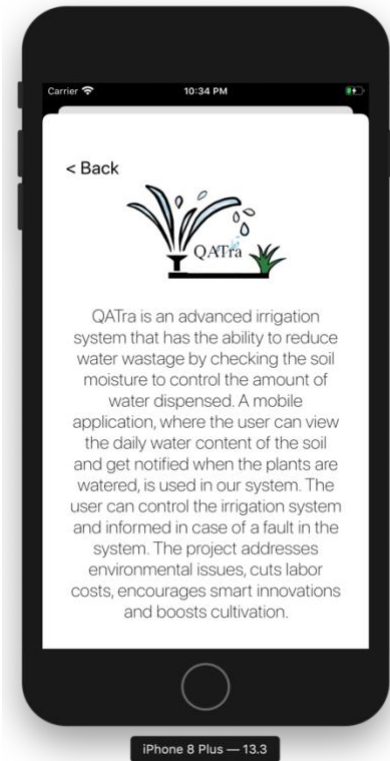


Figure 5.6: About QATra page.



The mobile application is ready to be deployed, as it was successfully tested on a real iPhone. Also, it is connected to the firebase server, which will allow the communication between the mobile application and the main microcontroller.

- **End Node**

The main function of the end node microcontroller is to take measurements of the soil moisture level and transmit it to the main microcontroller. **Figure 5.7** shows the program code flowchart for the end node. The end node consists of a microcontroller that is connected to the soil moisture sensor and to a transceiver. As shown in the figure, the grove soil moisture sensor takes the readings of the water content in the soil and then it converts the data from integers to characters. This step is important because the NRF24L01 transceiver only operates by transmitting characters from one chip to another. Once the conversion is done, the sensor reading gets transmitted to the main microcontroller using the NRF24L01 transceiver.

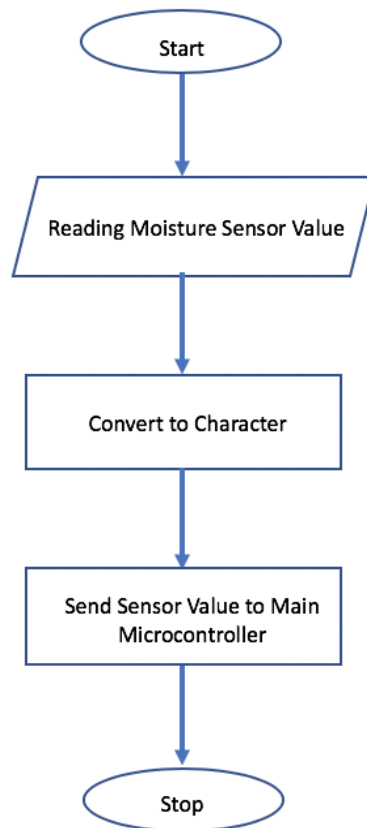


Figure 5.7: Flowchart of program code of end node.

- **Main Microcontroller**

The main microcontroller is the main link between the mobile application, the water pump/sprinklers and the end nodes. **Figure 5.8** shows the program code flowchart for the main microcontroller with only one end node for simplicity. As shown in **Figure 5.8**, the user can input

through the mobile application the type of plant for each end zone. The main microcontroller then receives the soil moisture level data from the end node through the transceiver. Next, it checks if there is a potential fault in the system and if there is, it notifies the user through the mobile application that there is a potential fault in the system. If there is no fault, it checks if the system is controlled automatically or manually by the user. If it is controlled manually, then it'll be controlled by the user through the mobile application. If it is controlled automatically, then the system will run by itself. This is done by first converting back the moisture level data from character to integer. Then the code will check if the soil moisture level is below a certain threshold and if this condition is met, the main microcontroller will instruct to turn the water pump on. This process will continue through a loop until the condition is no longer valid.

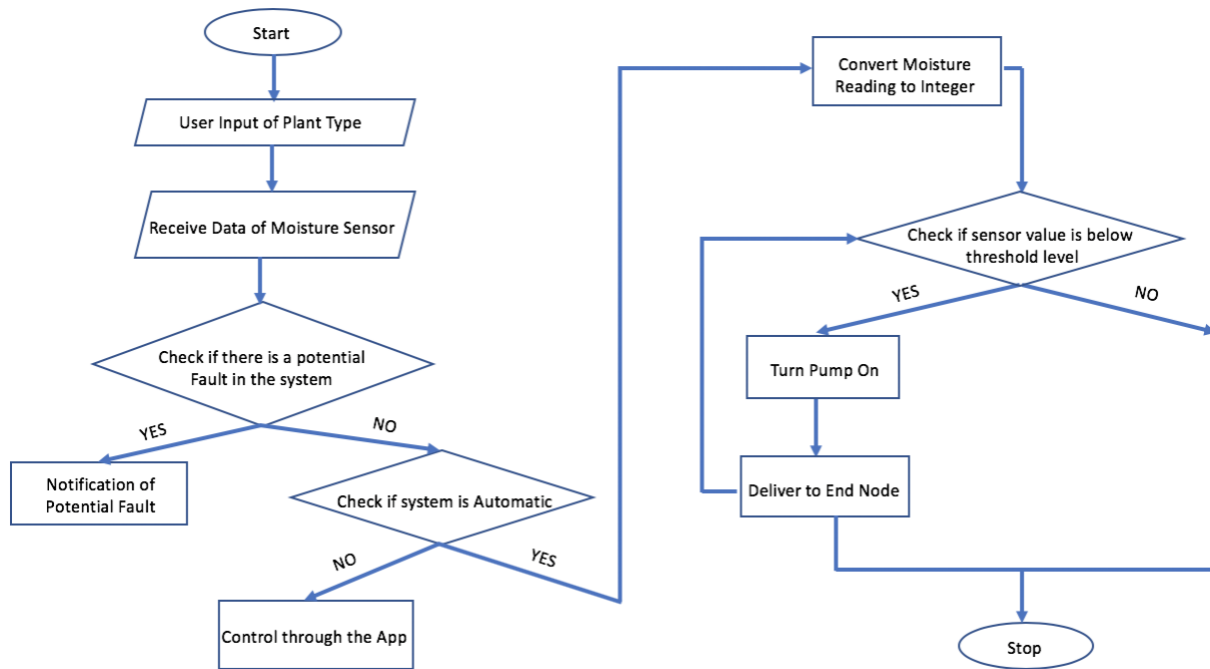


Figure 5.8: Flowchart of program code of main microcontroller.

- **Fault Detection Algorithm**

The fault detection algorithm, shown in **Figure 5.9**, is implemented in the main microcontroller code to detect any potential faults in any zone. The main microcontroller stores 24 soil moisture level readings in four different arrays for the four different zones. For each zone, the values in the array will be compared to maximum and minimum moisture level threshold. If there is no moisture level value that is lower than the minimum moisture level threshold or that is higher than the maximum moisture level threshold, there will be no fault. However, if there is a value which is lower than the minimum moisture level threshold or which is higher than the maximum moisture level threshold, then it will check the second condition. The main microcontroller will check if the value higher than the maximum moisture level threshold will keep on increasing or the value lower than the minimum moisture level threshold will keep on decreasing through scanning the remaining elements in the array. If the second condition is met, there will be a potential fault in

the system, and the user will be notified through the mobile application. However, if the second condition was not met, there will be no potential fault in the system.

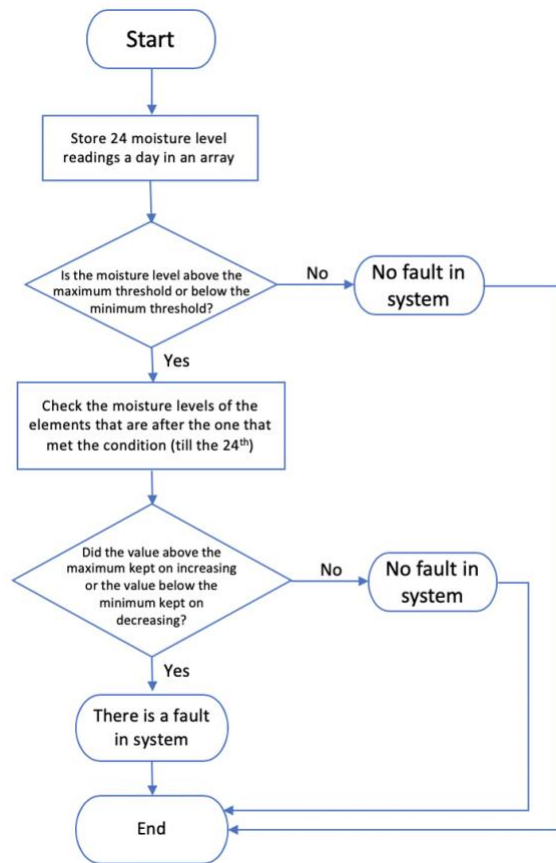


Figure 5.9: Fault detection algorithm flowchart.

## 5.2: Functional Prototyping, Testing, Troubleshooting, and Experimental Results

- **Functional Prototype**

**Figure 5.10 and Figure 5.11** show the functional prototype of the smart irrigation system. The system consists of many parts that includes microcontrollers, water pumps, relay switches, sprinklers, moisture sensors, transceivers, batteries and a mobile application. As shown in **Figure 5.12**, the system has two main inputs which are the soil moisture sensor readings and the type of plant the user is inputting to each end zone. The main function of the system is that it takes the soil moisture readings for each zone and checks if the soil water content is below the minimum threshold or above a maximum threshold. Whenever it is at or under a minimum threshold it automatically pumps water to irrigate the plants in order to prevent it from falling below this minimum threshold and it stops the irrigation if it reaches the maximum threshold. What makes the system more sophisticated is that the user, through the mobile application, can turn the system on or off manually and can monitor the soil moisture level for each zone. Moreover, if the system detects a potential fault, it informs the user by sending a notification to the mobile application.

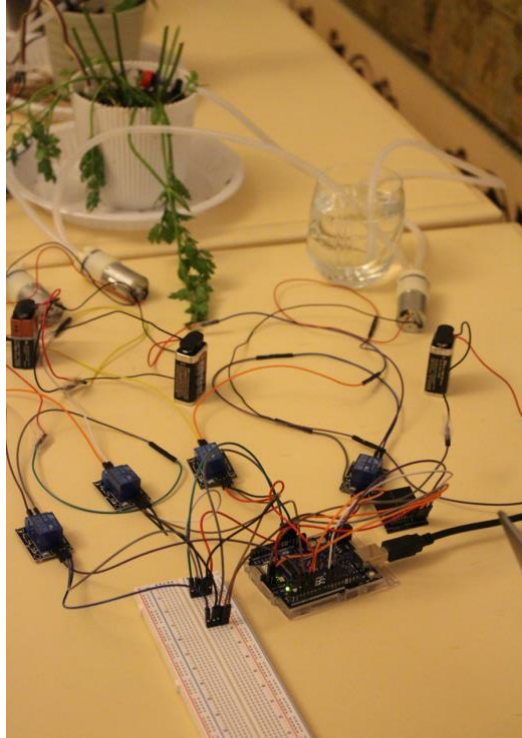


Figure 5.10: Main microcontroller.

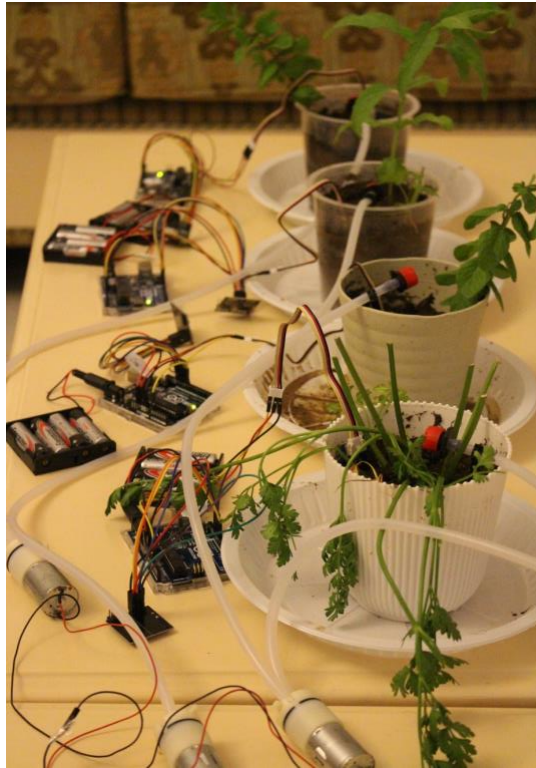


Figure 5.11: End nodes.



Figure 5.12: System high level representation.

- **Testing and Experimental Results**

After we gained understanding on how each component should work; the system was implemented, the codes were written, and the mobile application was built. Meanwhile, each part of the system was tested to ensure a smooth operation of the whole system.

- **End Node**

As mentioned earlier, the end node is responsible for taking soil moisture measurements and sending them to the main microcontroller. Thus, each end node, shown in **Figure 5.13**, is connected to a moisture sensor (to take soil moisture level) and a NRF24I01 transceiver (to send data to the main microcontroller).

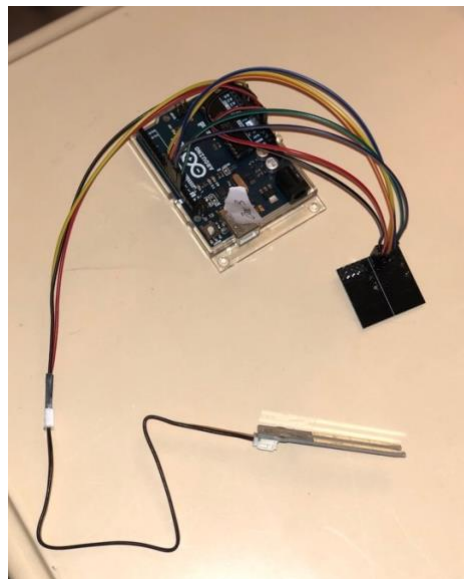


Figure 5.13: End node microcontroller connections.

First, a program was written to ensure a proper moisture level reading from the end node moisture sensor. The output of the program is shown in **Figure 5.14**.

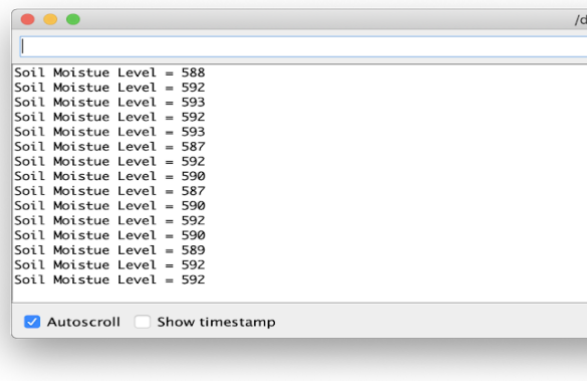


Figure 5.14: Testing of soil moisture sensor.

- Main Microcontroller

As previously stated, the main microcontroller is the link between the mobile application, irrigation system, and end nodes. Thus, the main microcontroller (**Figure 5.15**) is connected to the mobile application using firebase server (to send and receive data), to the relay, pumps, and sprinklers (to irrigate the plants based on user's input), and NRF24L01 transceiver (to receive soil moisture sensor reading from the end node microcontrollers).

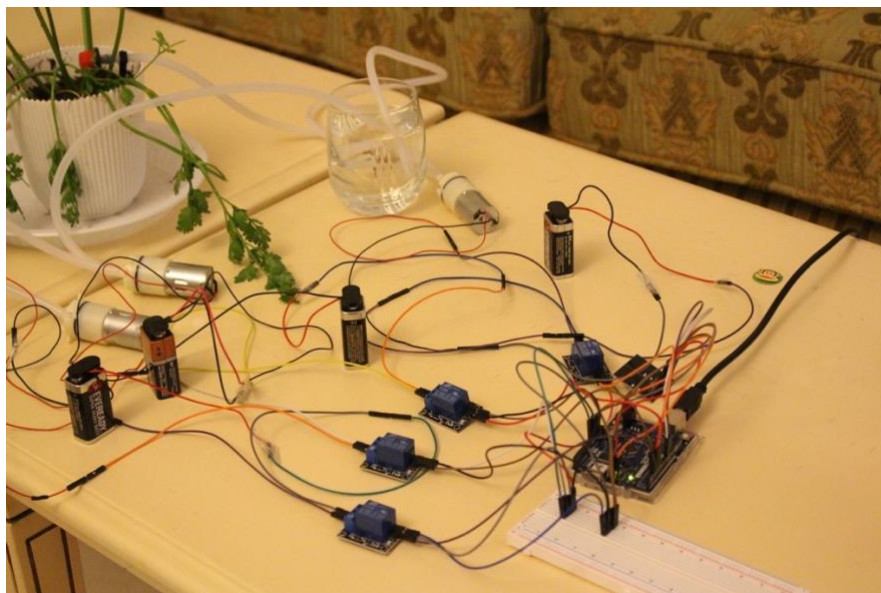


Figure 5.15: Main microcontroller connections.

To test the data transfer between the end node microcontroller and the main microcontroller, an Arduino code was developed. The main aim of this Arduino code was to ensure that the main microcontroller is successfully receiving the soil moisture readings from the end node microcontroller. The result of this code is shown in **Figure 5.16**.

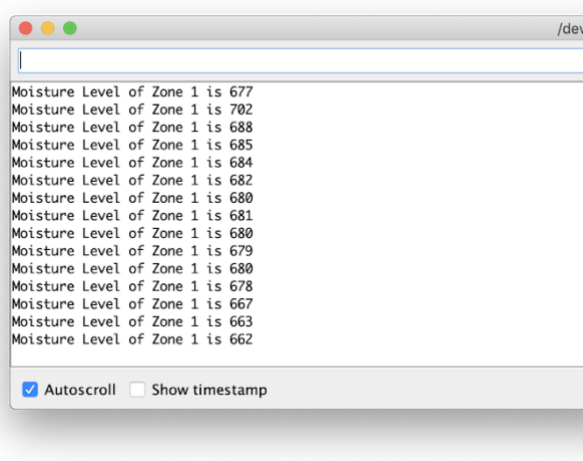


Figure 5.16: Main microcontroller receiving soil moisture level from zone.

After confirming that the wireless connection between the main microcontroller and zone 1 end node, the code was applied for the three other end nodes. The communication between the main microcontroller and the four end nodes was successful, as the main microcontroller was able to receive the soil moisture level for the four end nodes (**Figure 5.17**).

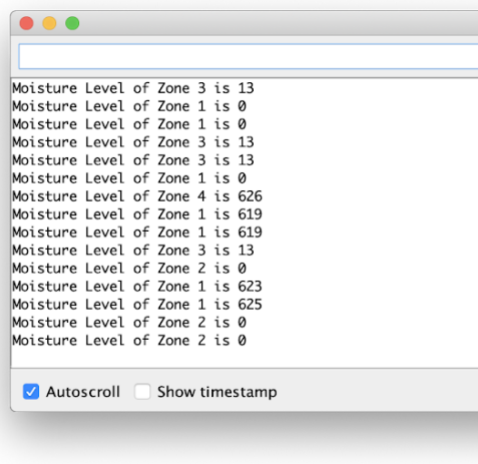


Figure 5.17: Main microcontroller receiving soil moisture levels from four zones.

Then, the Arduino code was developed to activate the pump and irrigate the plants based on the soil moisture levels reading and the user input from the mobile application. Since the user is able to choose the irrigation mode (automatic or manual) and to choose the plant type (to set minimum and maximum moisture level threshold).

After we wrote the codes and built the system, each feature was tested to ensure a smooth operation of the whole system.

- Monitor Moisture Level

After the microcontroller reads the moisture level values from the four end nodes, it sends the readings to the server (Firebase) to allow the user to monitor the moisture levels through the mobile application. To check the feature and ensure that the user gets the right moisture level readings, the moisture levels reading from the main microcontroller (**Figure 5.18**) and from the mobile application (**Figure 5.19**) were compared. It was verified that the values are the same.

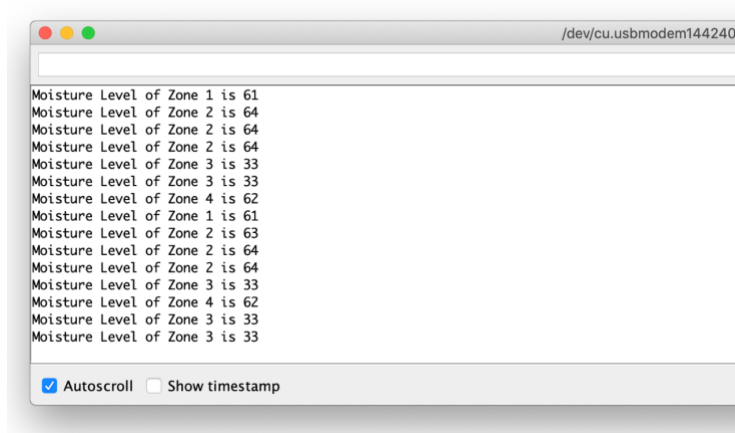


Figure 5.18: Moisture level readings from main microcontroller.

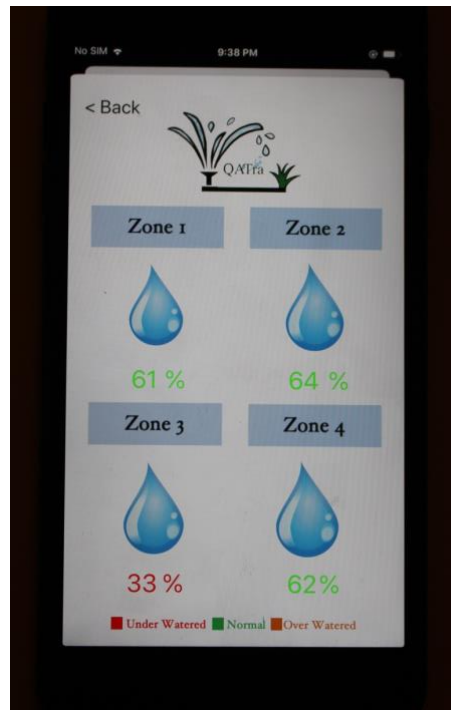


Figure 5.19: Moisture level readings from mobile application.



- Fault Detection

To check the operation of the fault detection feature, and whether the user is informed or not of a potential fault, we disconnected the moisture sensor as shown in **Figure 5.20** for testing purposes. After certain time, the main microcontroller detected the fault (**Figure 5.21**) and sent the information to the mobile application using the server (Firebase). Then, the mobile application was able to view and inform the user about a potential fault as shown in **Figure 5.22**.

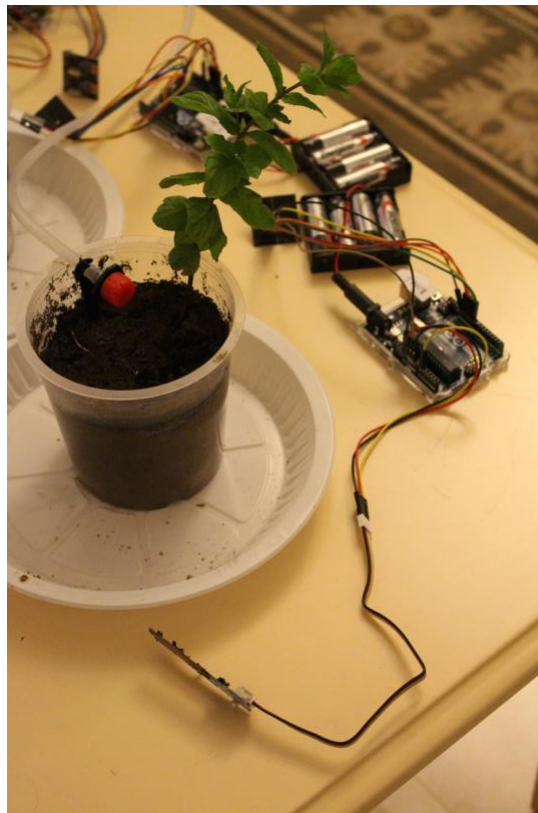


Figure 5.20: The moisture sensor was disconnected to test the fault detection feature.

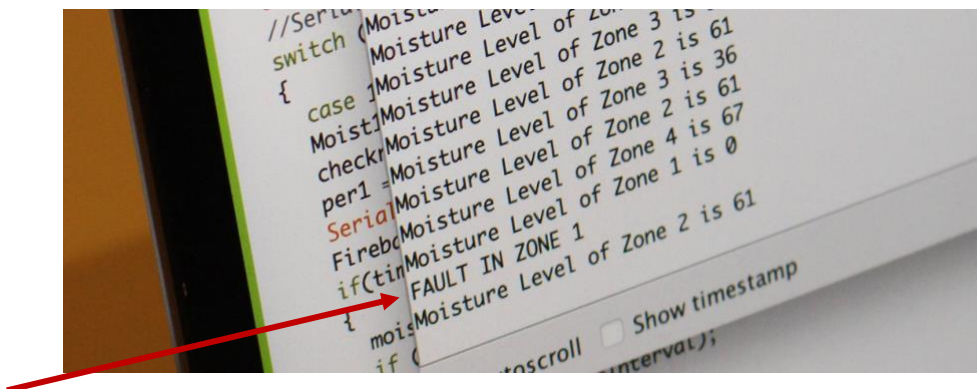


Figure 5.21: Main microcontroller was able to detect the potential fault.

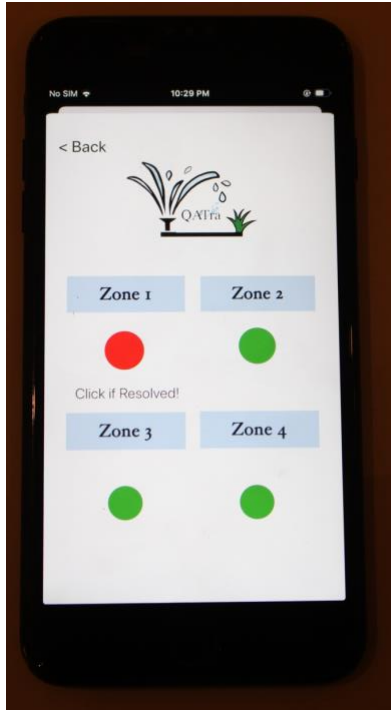


Figure 5.22: The user is informed about a potential fault in a specific zone using the mobile application.

- Automatic Irrigation Feature

Using the soil moisture level monitoring feature, we were able to identify that Zone 2, as shown in **Figure 5.23**, is underwatered. Thus, to test the automatic irrigation feature, the irrigation mode was changed to automatic (**Figure 5.24**). Once the irrigation mode changed to automatic, the water system started to water the plant in Zone 2, as shown in **Figure 5.25**.

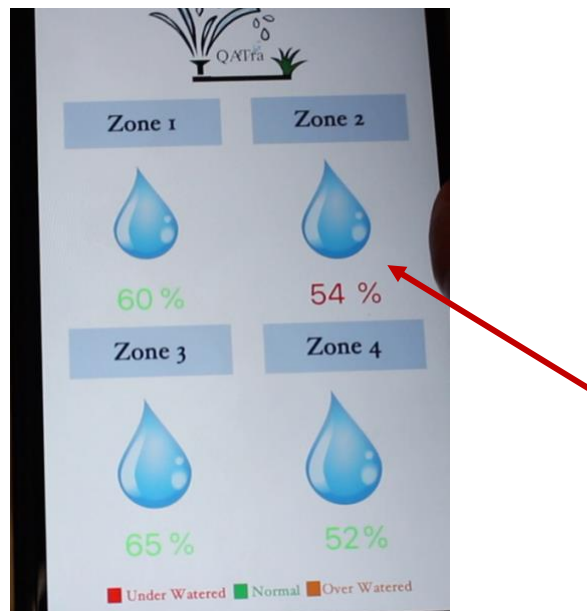


Figure 5.23: Moisture level monitoring feature enable the user to notice Zone 2 is underwatered.

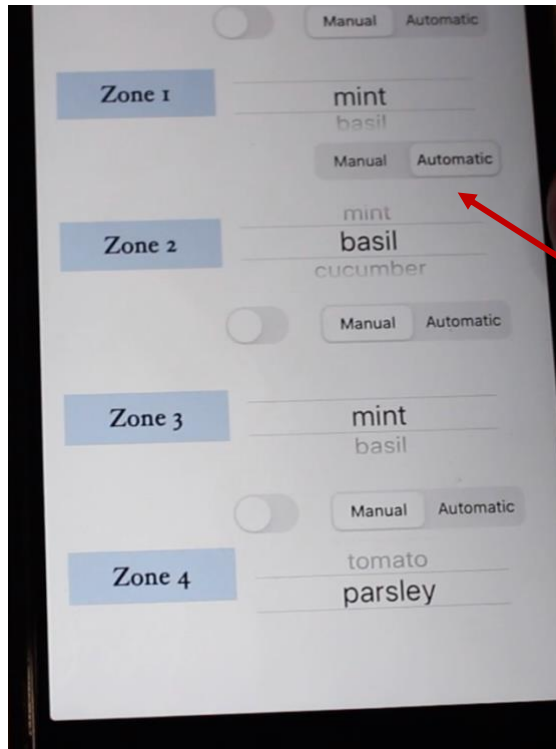


Figure 5.24: The irrigation mode of Zone 2 is changed into automatic.

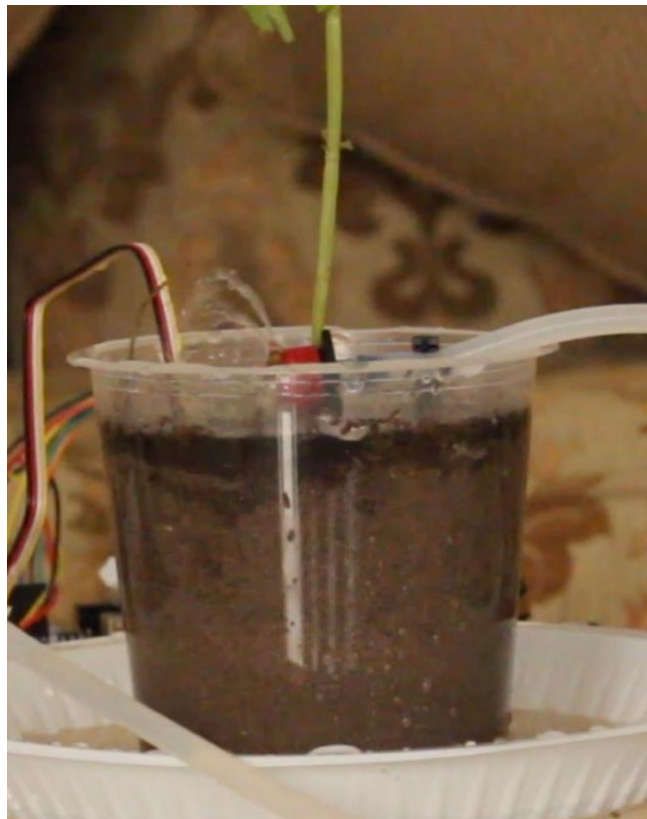


Figure 5.25: Once the irrigation mode was changed to automatic, the watering system turned on.

- Manual Irrigation Feature

One of the system main feature is that the user can control the irrigation system manually through the mobile application. To do so, the user should change the irrigation mode of the zone (Zone 1 was taken as an example) to manual (Figure 5.26). Then, the user can turn on or turn off the irrigation system. Once the user turns on the system (Figure 5.27), the irrigation system will start watering (Figure 5.28).

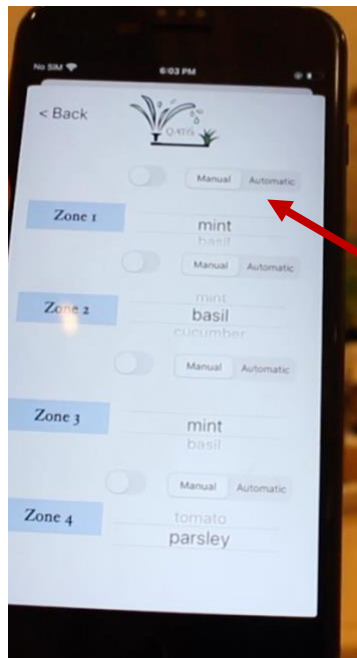


Figure 5.26: Zone 1 irrigation mode was changed to manual.

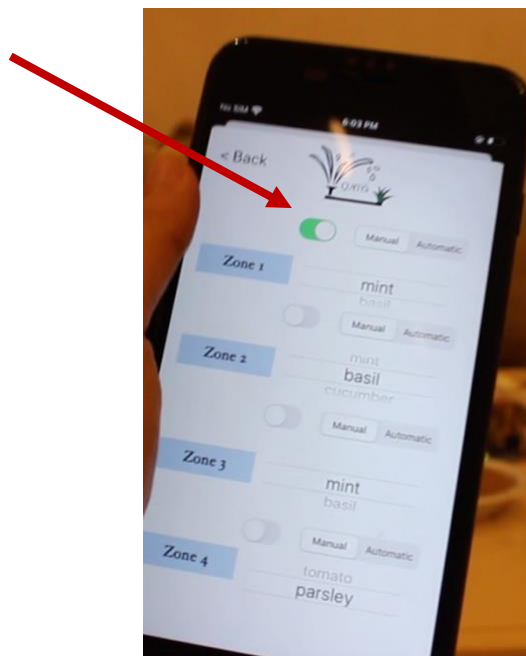


Figure 5.27: Zone 1 irrigation system was turned on.



Figure 5.28: Once the irrigation system is turned on, the irrigation system will start watering the plant.

- **Troubleshooting**

During our process of building our system, we faced some problems. However, these problems were solved.

- Transceiver NRF24L01

After connecting NRF24L01 transceivers to two Arduinos (microcontrollers), we tried to send strings as a starting point to check whether the connection was secured. Strings were received at the receiver node Arduino. However, when we connected the soil moisture level sensor to the transmitter node Arduino and tried sending integers, the receiver node Arduino was receiving either zeros or question marks. We overcame that problem by changing the type of the parameters that are transmitted and received by the NRF24L01 transceiver to integers and unsigned characters.

- Raspberry pi (Gateway)

In our proposed design, the gateway was the link between the main microcontroller (central node) and the server. During the programming and testing stage of the Raspberry Pi with Arduino, NRF24L01 transceivers were used to send and receive data. However, 255 was being received all of the time no matter what we sent. We decided to eliminate the Raspberry pi from our system and make the central node Arduino as the link between the system and the server.

- The Wi-Fi connection of Central node Arduino after eliminating the Raspberry pi (Gateway)

The Arduino is a microcontroller, unlike the Raspberry pi which is a computer that can enable Wi-fi easily. After researching, we found a Wi-fi micro-chip that can be used to connect the Arduino to the internet. This chip is ESP8266. We talked to Mr. Wesam regarding this and he ordered the Wi-fi micro-chip for us. When we received the ESP8266, we started programming it but kept getting errors. Mr. Wesam provided another type of Arduino called Arduino Uno WIFI Rev 2. The Wi-fi connection worked because the new Arduino has a WIFI library that can be used. We scanned the networks nearby then connected to the Wi-Fi successfully by providing the network name and password in the code.

## **Chapter VI: Conclusion**

### **6.1: Conclusions, discussion, verification, future recommendations, improvements and optimizations**

Our project is finalized. For the demo video, we showed 4 zones/end nodes (4 different plants) working with the system. One of the pipes of the water pumps was removed, so that the system will detect a fault and show it on the application. We showed how one end node can be turned on and controlled manually. For the last two end nodes, we showed one of the plants' moisture level being above the maximum threshold (to show how the system works automatically) and the other plants' moisture level being under the minimum threshold to show how the sprinkler will be turned on.

For verification, it is important to note that separate parts of the system were verified instead of the system as a whole. First, the mobile application was tested on the virtual device, which showed us the layout of the application, whether the transitions between the pages were working, and the application as a whole was running. Later, the mobile application was installed on an iPhone, and we verified that it can be deployed. Second, the moisture sensor manual said that the moisture sensor level is less than 300 for dry soil and between 300 and 700 for moist soil. Therefore, we tested the moisture sensor with different types of soil and measured the soil moisture level to verify whether the moisture sensors were working properly or not.

For improvements, we believe that the system can be cheaper. Now, the system costs 520 dollars. This price includes the irrigation system (pumps, relays, sprinklers and pipes) as well as the controller part. Therefore, it is more expensive than other smart irrigation systems, since they include the controllers only. If the system was to be fabricated all at once, instead of buying the different components in our system from different companies (i.e. if we manufactured those components) and the system was mass produced, then the system will be cheaper. We realized that the price can be a financial constraint, and therefore we believe an improvement to it is necessary. Moreover, we believe that the fault detection method can be optimized. Since currently it can only notify the user in case of a potential fault, without specifying where the exact fault is. The reason for that is because fault detection is a whole field on its own, and we decided that we will partially implement it in our system. For future, we or other groups that wish to improve our project can enhance the fault detection algorithm.

For future recommendations, either if we continue developing this project or other people continue with it, we believe there are additional features that can be added. Those features are weather intelligence and multi-language. Our project is specifically designed for Qatar. Since rain is very rare in Qatar, we believed that a weather sensor is not necessary. However, if the app was to be developed for different countries, a weather sensor can enhance the performance of the system. Also, the app can be linked to a weather forecast, in order to plan irrigation schedules accordingly. Moreover, the language used in our mobile application is English, since it is the most widely spoken language around the world. Other languages can be implemented like Arabic, Urdu, etc. to make the system more accessible and universal.

Qatar National Vision 2030 aims to achieve sustainable development to ensure a better life for future generations. QATra hopes to contribute to this vision.



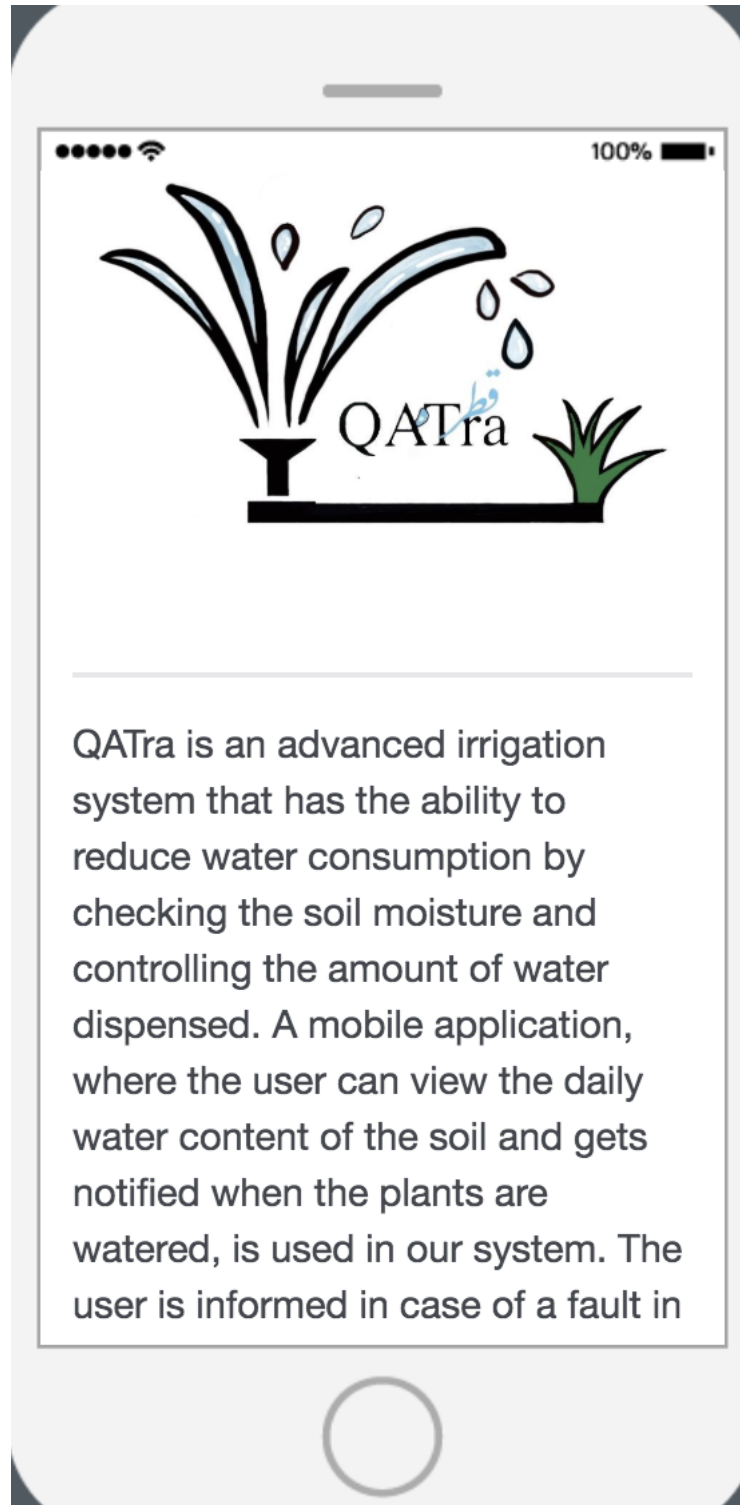
## References:

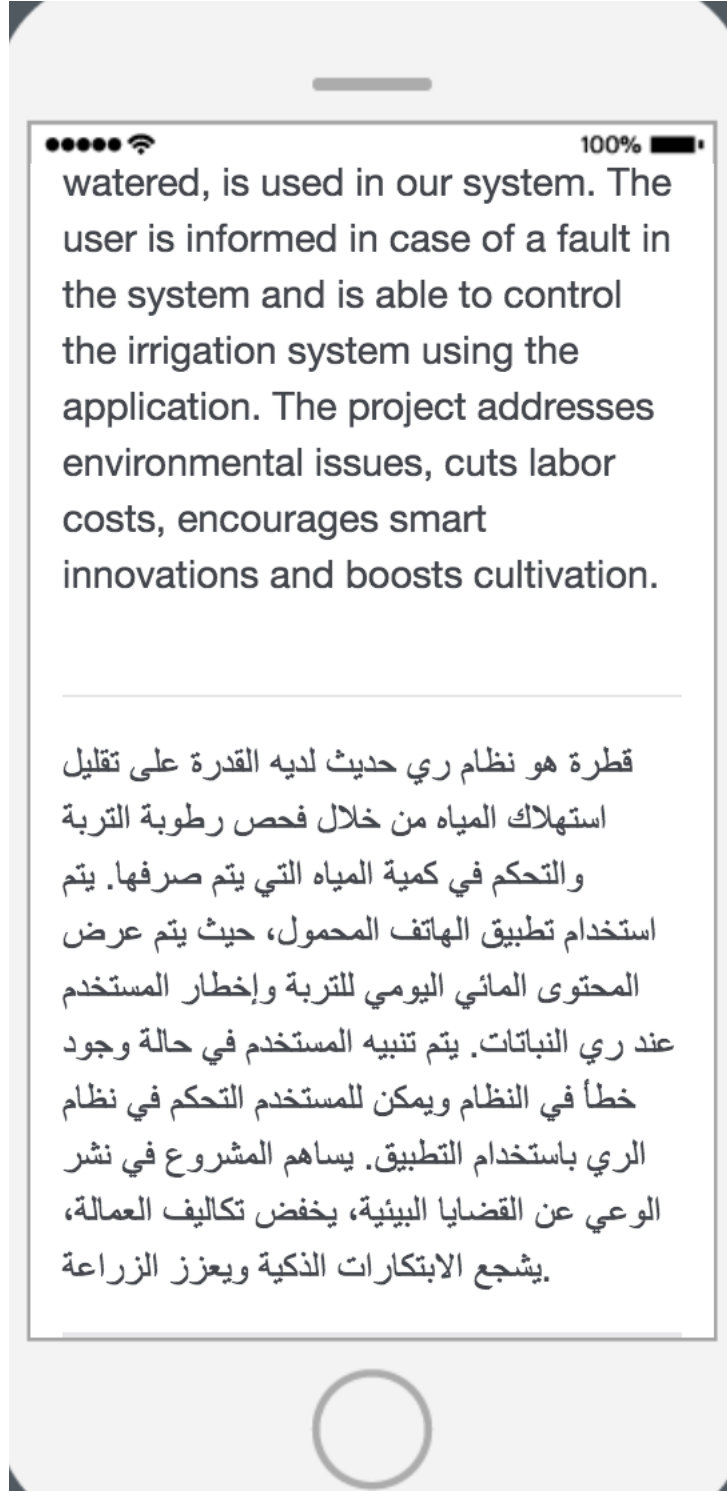
- [1] H. M. Baalousha and O. K. M. Ouda, "Domestic water demand challenges in Qatar," *Arabian Journal of Geosciences*, vol. 10, no. 24, Dec. 2017.
- [2] M. Darwish and R. Mohtar, "Qatar water challenges," *Desalination and Water Treatment*, vol. 51, no. 1-3, Jun. 2012.
- [3] O. K. Ogidan, A. E. Onile, and O. G. Adegboro, "Smart Irrigation System: A Water Management Procedure," *Agricultural Sciences*, vol. 10, no. 01, pp. 25–31, Jan. 2019.
- [4] Abba, Namkusong, Lee, and Crespo, "Design and Performance Evaluation of a Low-Cost Autonomous Sensor Interface for a Smart IoT-Based Irrigation Monitoring and Control System," *Sensors*, vol. 19, no. 17, Aug. 2019.
- [5] G. Shruthi, B. S. Kumari, R. P. Rani, and R. Preyadharan, "A-real time smart sprinkler irrigation control system," *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, 2017.
- [6] P. H. Tarange, R. G. Mevekari, and P. A. Shinde, "Web based automatic irrigation system using wireless sensor network and embedded Linux board," *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, 2015.
- [7] D. S., *Strategies for Combating Climate Change in Drylands Agriculture*. Nov. 2012.
- [8] "Raise the Bar on Smart Watering," *Rachio*. [Online]. Available: <https://www.rachio.com/rachio-3/>. [Accessed: 11-Nov-2019].
- [9] "Welcome to B-hyve," *b*. [Online]. Available: <https://bhyve.orbitonline.com/indoor-timer/>. [Accessed: 11-Nov-2019].
- [10] "HC," *Hunter Industries*, 06-Nov-2019. [Online]. Available: <https://www.hunterindustries.com/irrigation-product/controllers/hc>. [Accessed: 11-Nov-2019].
- [11] "Web Based, Wireless (Wifi) Irrigation Controller," *BlueSpray*. [Online]. Available: <https://www.bluespray.net/>. [Accessed: 11-Nov-2019].
- [12] "Equipment Authorization Procedures," *Federal Communications Commission*, 02-Apr-2018. [Online]. Available: <https://www.fcc.gov/general/equipment-authorization-procedures>. [Accessed: 11-Nov-2019].
- [13] "Overview," *World Bank*. [Online]. Available: <https://www.worldbank.org/en/topic/agriculture/overview>. [Accessed: 11-Nov-2019].
- [14] H. Saha, S. Mandal, S. Mitra, S. Banerjee, and U. Saha, "Comparative Performance Analysis between nRF24L01 and XBEE ZB Module Based Wireless Ad-hoc Networks," *International Journal of Computer Network and Information Security*, vol. 9, no. 7, pp. 36–44, Aug. 2017.
- [15] "nRF24L01 Single Chip 2.4GHz Transceiver," Mar-2008. [Online]. Available: [https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss\\_Preliminary\\_Product\\_Specification\\_v1\\_0.pdf](https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf). [Accessed: 15-Mar-2020].
- [16] F. Copes, "The Arduino Uno WiFi rev 2 board," Flavio Copes, 23-Jan-2020. [Online]. Available: <https://flaviocopes.com/arduino-board-uno-wifi-rev-2/>. [Accessed: 12-Apr-2020].

## Appendix:

### 1. Customer Needs Analysis:

#### 1.1 Survey 1: Questionnaire





Age Range - الفئة العمرية

Below 18 - اصغر من ١٨ سنة

18-25

26-35

Above 35 - اكبر من ٣٥ سنة

Gender - الجنس

Male - ذكر

Female - انثى

Will you be interested in a Smart Irrigation System? - هل انت/ي مهتم/ة في نظام ري حديث؟

Definitely yes - بالطبع

Yes - نعم

Might or might not - ممكن

No - لا

Definitely not - بالطبع لا

---

Are you familiar with such system?  
هل سمعت عن هذا النظام من قبل؟

Yes - نعم

Maybe - ممكن

No - لا

Do you have a farm/garden that needs to be watered regularly? - هل لديك مزرعة / حديقة تحتاج الى الري بانتظام؟

Yes - نعم

No - لا

---

Does the application feature makes it more convenient for you to control and monitor the system? - هل ميزة التطبيق في مشروعنا يسهل عليك المراقبة و التحكم في نظام الري؟

Yes - نعم

Maybe - ممكن

No - لا

Any additional suggestions? - أي  
اقتراحات إضافية؟

Do you think our project is reducing  
water consumption and advocating  
environmental issues such as  
global warming? - هل تعتقد بأن مشروعنا  
يقلل من استهلاك المياه ويساهم في نشر الوعي عن  
القضايا البيئية مثل الاحتباس الحراري؟

Yes - نعم

Maybe - ممكن

No - لا

••••• 100%

Would you like to see this technology with more features (Any suggestions)? - هل ترغب برؤية هذه التقنية - مع مميزات إضافية (أي اقتراح)؟

What price range do you think is suitable for such product? - برأيك، ما هو السعر المناسب لهذا النظام؟

1000-1500 QR

1500 - 2000 QR

+2000 QR

Additional suggestions? - أي اقتراحات إضافية؟



## 1.2 Survey 2: Interview Questions



### QATra Irrigation System Interview:

#### General Questions:

1. Are you familiar with a smart irrigation system? Will you be interested in it?
2. What is the feasibility/convenient/practicality of our proposed irrigation system?
3. What constraints do you think will be faced with our proposed irrigation system?
4. Do you think the application feature makes it more convenient for the user to control and monitor the system? Would you like to see the same technology with more features?
5. Do you think fault detection in the irrigation system is an important feature? Why?
6. Water is a scarcity in Qatar, do you think that our project will help in reducing wastage of water?
7. Do you think that there is scope for large scale implementation of our project in Qatar?

#### Specific questions for both:

1. What is the average cost of watering per square meter of land?
2. What do you think a suitable price for our project would be?

#### Specific questions for Kahrama:

1. Do you think that by implementing water threshold for various species of plants in our system will help in saving water/consuming less water? (If this system was implemented in small scale or large scale)
2. If this system is implemented will the cost of water reduce for its consumers.
3. Is the farming/agricultural industry consuming the most water?
4. Do you think that if our system was implemented and today's world adapted to this change of water control, then this will benefit the future generations? (In terms of water scarcity)
5. Will this project encourage the production of more ecofriendly systems that help save our environment?

#### Specific questions for Department of Agricultural research:

1. Are you familiar with a similar product? What feature does it have? Do you want to add any new features to it? Does it have any problems? Do you face problem while using it?
2. How much water does this product consume (per km<sup>2</sup>)?
3. How much you are willing to spend on such system (per km<sup>2</sup>)?
4. Where should the sensor be placed to get the most accurate results?
5. How often should we check for the moisture level in the soil?
6. Does the moisture level of the soil depend on the type of the plant? If yes, how should we know the desired moisture level for every plant?


### **1.3: Consent Forms of Interviewees**

#### **Consent form of Eng. Ahmed Shaker:**

#### CONSENT FORM

**Project:** QATra Irrigation System  
**Texas A&M University at Qatar**  
**Course:** ECEN 403-Electrical Design Lab I  
**Group Members:** Maryam Al-Emadi  
Roqayya AlYousef  
Fatima Al-Janahi  
Noof Al-Sayed  
**Mentor:** Dr. Hazem Nounou

I agree to participate in this interview for QATra irrigation system project. I also understand that I will be photographed/recorded/videotaped as part of the project and I agree to it without any pressure.

**Name:** Ahmed  
**Date:** 3/10/2019  
**Signature:** 

**Consent form of Eng. Ayman Mashali:**



**CONSENT FORM**

**Project:** QATra Irrigation System  
**Texas A&M University at Qatar**  
**Course:** ECEN 403-Electrical Design Lab I  
**Group Members:** Maryam Al-Emadi  
Roqayya AlYousef  
Fatima Al-Janahi  
Noof Al-Sayed  
**Mentor:** Dr. Hazem Nounou

I agree to participate in this interview for QATra irrigation system project. I also understand that I will be photographed/recorded/videotaped as part of the project and aware that this material will be used for assignments like the market analysis study, ethnographic study video and any upcoming assignment that might need this material.

**Name:** *Ayman Mashali*

**Date:** *14-10-2019*

**Signature:** *[Handwritten Signature]*

Consent form of Eng. Mohamed Ben Aicha:



CONSENT FORM

**Project:** QATra Irrigation System  
**Texas A&M University at Qatar**  
**Course:** ECEN 403-Electrical Design Lab I  
**Group Members:** Maryam Al-Emadi  
Roqayya AlYousef  
Fatima Al-Janahi  
Noof Al-Sayed  
**Mentor:** Dr. Hazem Nounou

I agree to participate in this interview for QATra irrigation system project. I also understand that I will be photographed/recorded/videotaped as part of the project and aware that this material will be used for assignments like the market analysis study, ethnographic study video and any upcoming assignment that might need this material.

**Name:** Mohamed Salah Ben Aicha // Project Engineer.

**Date:** 14/10 2019

**Signature:** 

**Consent form of Mr. Osman Ahmed Abdalla:**



**CONSENT FORM**

**Project:** QATra Irrigation System  
**Texas A&M University at Qatar**  
**Course:** ECEN 403-Electrical Design Lab I  
**Group Members:** Maryam Al-Emadi  
Roqayya AlYousef  
Fatima Al-Janahi  
Noof Al-Sayed  
**Mentor:** Dr. Hazem Nounou

I agree to participate in this interview for QATra irrigation system project. I also understand that I will be photographed/recorded/videotaped as part of the project and aware that this material will be used for assignments like the market analysis study, ethnographic study video and any upcoming assignment that might need this material.

**Name:** Osman Ahmed El Shavief Abdalla

**Date:** 14 Oct 2019

**Signature:**

## 2. Arduino Codes:

### 2.1: End Nodes Code

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Transmitting (Transceiver)
RF24 radio(9, 10); // CE, CSN
const uint64_t address1 = 0xE8E8F0F0A1LL;

int SensorPin = A0; // Connect the Moisture Sensor through Analog In Pin A0
unsigned int Moist1 = 0; // Creating a Variable to Store the Sensor Value and Initialize it to Zero
int value1;

void setup() {
  radio.begin();
  radio.openWritingPipe(address1);
  radio.stopListening();
}

void loop() {
  value1 = analogRead(SensorPin);
  Moist1 = value1;
  radio.write(&Moist1, sizeof(Moist1));
}
```

## 2.2: Main Microcontroller Code

```
#include <Firebase_Arduino_WiFiNINA_HTTPClient.h>
#include <Firebase_Arduino_WiFiNINA.h>
#include <SandTimer.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <WiFiNINA.h>

#define basilmin 600
#define basilmax 760
#define mintmin 600
#define mintmax 760
#define tomatomin 475
#define tomatomax 710
#define cucumbermin 475
#define cucumbermax 665
#define parsleymin 475
#define parsleymax 760
#define TimeInterval 100
#define NumRead 2
#define FIREBASE_HOST "arduino-test-28572.firebaseio.com"
#define FIREBASE_AUTH "oruqSjcdPmdYOP2urYxkfd5DkHEjS3O97BjHJute"

char ssid[] = "Alico"; // network SSID (name)
char pass[] = "*****"; // network password
int status = WL_IDLE_STATUS; // the Wifi radio's status

FirebaseData firebaseData;

uint8_t pipeNum;
unsigned int Moist1;
unsigned int Moist2;
unsigned int Moist3;
unsigned int Moist4;
unsigned int data;

int per1;
int per2;
int per3;
int per4;
int perc;

int moistarray1[NumRead];
int moistarray2[24];
int moistarray3[24];
int moistarray4[24];

int zone1 = 0;
int zone2 = 0;
int zone3 = 0;
int zone4 = 0;
```

```

//Minimum and Maximum Threshold
int min1 = 0;
int max1 = 0;
int min2 = 0;
int max2 = 0;
int min3 = 0;
int max3 = 0;
int min4 = 0;
int max4 = 0;

int PumpPin1 = 2;
int PumpPin2 = 3;
int PumpPin3 = 4;
int PumpPin4 = 5;

//zero is manual on, one is manual off
int zone1On = 0;
int zone2On = 0;
int zone3On = 0;
int zone4On = 0;

//Vegetables type for each zone
String Veg1 = "basil";
String Veg2 = "basil";
String Veg3 = "basil";
String Veg4 = "basil";

//zero is no fault, one is fault
int fault1 = 0;
int fault2 = 0;
int fault3 = 0;
int fault4 = 0;

//timers
SandTimer timer1;
SandTimer timer2;
SandTimer timer3;
SandTimer timer4;

int rgy1;
int rgy2;
int rgy3;
int rgy4;

int i1 = 0, i2 = 0, i3 = 0, i4 = 0;

RF24 radio(9,10); // CE, CSN

const uint64_t address1 = 0xE8E8F0F0A1LL;
const uint64_t address2 = 0xE8E8F0F0A2LL;
const uint64_t address3 = 0xE8E8F0F0A3LL;
const uint64_t address4 = 0xE8E8F0F0A4LL;

```



```

void setup() {
  Serial.begin(9600);
  pinMode(PumpPin1, OUTPUT);
  pinMode(PumpPin2, OUTPUT);
  pinMode(PumpPin3, OUTPUT);
  pinMode(PumpPin4, OUTPUT);
  timer1.start(TimeInterval);
  timer2.start(TimeInterval);
  timer3.start(TimeInterval);
  timer4.start(TimeInterval);

  while (status != WL_CONNECTED)
  {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);
  }
  delay(10000);
  Serial.println("Arduino is now connected to the network");
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH, ssid, pass);
  delay(2000);
  Serial.println("Arduino is now connected to Firebase");
  getParameters();
  radio.begin();
  //radio.setPALevel(RF24_PA_MIN);
  radio.openReadingPipe(1, address1);
  radio.openReadingPipe(2, address2);
  radio.openReadingPipe(3, address3);
  radio.openReadingPipe(4, address4);
  radio.startListening();
}

void loop() {
  while(zone1 == 0 && zone2 == 0 && zone3 == 0 && zone4 == 0)
  {
    Vegetables();
    readMicrocontrollers();

    if(Moist1 < min1)
    {
      digitalWrite(PumpPin1,HIGH);
    }
    else
    {
      digitalWrite(PumpPin1,LOW);
    }
    if(Moist2 < min2)
    {
      digitalWrite(PumpPin2,HIGH);
    }
    else
    {

```

```

    digitalWrite(PumpPin2,LOW);
}
if(Moist3 < min3)
{
    digitalWrite(PumpPin3,HIGH);
}
else
{
    digitalWrite(PumpPin3,LOW);
}
if(Moist4 < min4)
{
    digitalWrite(PumpPin4,HIGH);
}
else
{
    digitalWrite(PumpPin4,LOW);
}
getParameters();
}

while(zone1 == 1 && zone2 == 0 && zone3 == 0 && zone4 == 0)
{
    Vegetables();
    readMicrocontrollers();

    if(zone1On == 0)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else if (zone1On == 1)
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(Moist2 < min2)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else
    {
        digitalWrite(PumpPin2,LOW);
    }
    if(Moist3 < min3)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(Moist4 < min4)
    {
        digitalWrite(PumpPin4,HIGH);
    }
}

```

```

}
else
{
    digitalWrite(PumpPin4,LOW);
}
getParameters();
}

while(zone1 == 1 && zone2 == 1 && zone3 == 0 && zone4 == 0)
{
    Vegetables();
    readMicrocontrollers();

    if(zone1On == 0)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else if (zone1On == 1)
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(zone2On == 0)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else if (zone2On == 1)
    {
        digitalWrite(PumpPin2,LOW);
    }
    if(Moist3 < min3)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(Moist4 < min4)
    {
        digitalWrite(PumpPin4,HIGH);
    }
    else
    {
        digitalWrite(PumpPin4,LOW);
    }
    getParameters();
}

while(zone1 == 1 && zone2 == 1 && zone3 == 1 && zone4 == 0)
{
    Vegetables();
    readMicrocontrollers();
}

```

```

if(zone1On == 0)
{
    digitalWrite(PumpPin1,HIGH);
}
else if (zone1On == 1)
{
    digitalWrite(PumpPin1,LOW);
}
if(zone2On == 0)
{
    digitalWrite(PumpPin2,HIGH);
}
else if (zone2On == 1)
{
    digitalWrite(PumpPin2,LOW);
}
if(zone3On == 0)
{
    digitalWrite(PumpPin3,HIGH);
}
else if (zone3On == 1)
{
    digitalWrite(PumpPin3,LOW);
}
if(Moist4 < min4)
{
    digitalWrite(PumpPin4,HIGH);
}
else
{
    digitalWrite(PumpPin4,LOW);
}
getParameters();
}

```

```

while(zone1 == 1 && zone2 == 1 && zone3 == 1 && zone4 == 1)
{
    Vegetables();
    readMicrocontrollers();

    if(zone1On == 0)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else if (zone1On == 1)
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(zone2On == 0)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else if (zone2On == 1)

```

```

    {
        digitalWrite(PumpPin2,LOW);
    }
    if(zone3On == 0)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else if (zone3On == 1)
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(zone4On == 0)
    {
        digitalWrite(PumpPin4,HIGH);
    }
    else if (zone4On == 1)
    {
        digitalWrite(PumpPin4,LOW);
    }
    getParameters();
}

while(zone1 == 1 && zone2 == 1 && zone3 == 0 && zone4 == 1)
{
    Vegetables();
    readMicrocontrollers();

    if(zone1On == 0)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else if (zone1On == 1)
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(zone2On == 0)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else if (zone2On == 1)
    {
        digitalWrite(PumpPin2,LOW);
    }
    if(Moist3 < min3)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(zone4On == 0)
    {

```

```

    digitalWrite(PumpPin4,HIGH);
  }
  else if (zone4On == 1)
  {
    digitalWrite(PumpPin4,LOW);
  }
  getParameters();
}

while(zone1 == 1 && zone2 == 0 && zone3 == 1 && zone4 == 1)
{
  Vegetables();
  readMicrocontrollers();

  if(zone1On == 0)
  {
    digitalWrite(PumpPin1,HIGH);
  }
  else if (zone1On == 1)
  {
    digitalWrite(PumpPin1,LOW);
  }
  if(Moist2 < min2)
  {
    digitalWrite(PumpPin2,HIGH);
  }
  else
  {
    digitalWrite(PumpPin2,LOW);
  }
  if(zone3On == 0)
  {
    digitalWrite(PumpPin3,HIGH);
  }
  else if (zone3On == 1)
  {
    digitalWrite(PumpPin3,LOW);
  }
  if(zone4On == 0)
  {
    digitalWrite(PumpPin4,HIGH);
  }
  else if (zone4On == 1)
  {
    digitalWrite(PumpPin4,LOW);
  }
  getParameters();
}

while(zone1 == 1 && zone2 == 0 && zone3 == 1 && zone4 == 0)
{
  Vegetables();
  readMicrocontrollers();
}

```

```

if(zone1On == 0)
{
    digitalWrite(PumpPin1,HIGH);
}
else if (zone1On == 1)
{
    digitalWrite(PumpPin1,LOW);
}
if(Moist2 < min2)
{
    digitalWrite(PumpPin2,HIGH);
}
else
{
    digitalWrite(PumpPin2,LOW);
}
if(zone3On == 0)
{
    digitalWrite(PumpPin3,HIGH);
}
else if (zone3On == 1)
{
    digitalWrite(PumpPin3,LOW);
}
if(Moist4 < min4)
{
    digitalWrite(PumpPin4,HIGH);
}
else
{
    digitalWrite(PumpPin4,LOW);
}
getParameters();
}

```

```

while(zone1 == 1 && zone2 == 0 && zone3 == 0 && zone4 == 1)
{
    Vegetables();
    readMicrocontrollers();

    if(zone1On == 0)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else if (zone1On == 1)
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(Moist2 < min2)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else

```

```

    {
        digitalWrite(PumpPin2,LOW);
    }
    if(Moist3 < min3)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(zone4On == 0)
    {
        digitalWrite(PumpPin4,HIGH);
    }
    else if (zone4On == 1)
    {
        digitalWrite(PumpPin4,LOW);
    }
    getParameters();
}

while(zone1 == 0 && zone2 == 1 && zone3 == 1 && zone4 == 1)
{
    Vegetables();
    readMicrocontrollers();

    if(Moist1 < min1)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(zone2On == 0)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else if (zone2On == 1)
    {
        digitalWrite(PumpPin2,LOW);
    }
    if(zone3On == 0)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else if (zone3On == 1)
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(zone4On == 0)
    {

```



```

    digitalWrite(PumpPin4,HIGH);
  }
  else if (zone4On == 1)
  {
    digitalWrite(PumpPin4,LOW);
  }
  getParameters();
}

while(zone1 == 0 && zone2 == 1 && zone3 == 1 && zone4 == 0)
{
  Vegetables();
  readMicrocontrollers();

  if(Moist1 < min1)
  {
    digitalWrite(PumpPin1,HIGH);
  }
  else
  {
    digitalWrite(PumpPin1,LOW);
  }
  if(zone2On == 0)
  {
    digitalWrite(PumpPin2,HIGH);
  }
  else if (zone2On == 1)
  {
    digitalWrite(PumpPin2,LOW);
  }
  if(zone3On == 0)
  {
    digitalWrite(PumpPin3,HIGH);
  }
  else if (zone3On == 1)
  {
    digitalWrite(PumpPin3,LOW);
  }
  if(Moist4 < min4)
  {
    digitalWrite(PumpPin4,HIGH);
  }
  else
  {
    digitalWrite(PumpPin4,LOW);
  }
  getParameters();
}

while(zone1 == 0 && zone2 == 1 && zone3 == 0 && zone4 == 1)
{
  Vegetables();
  readMicrocontrollers();
}

```

```

if(Moist1 < min1)
{
  digitalWrite(PumpPin1,HIGH);
}
else
{
  digitalWrite(PumpPin1,LOW);
}
if(zone2On == 0)
{
  digitalWrite(PumpPin2,HIGH);
}
else if (zone2On == 1)
{
  digitalWrite(PumpPin2,LOW);
}
if(Moist3 < min3)
{
  digitalWrite(PumpPin3,HIGH);
}
else
{
  digitalWrite(PumpPin3,LOW);
}
if(zone4On == 0)
{
  digitalWrite(PumpPin4,HIGH);
}
else if (zone4On == 1)
{
  digitalWrite(PumpPin4,LOW);
}
getParameters();
}

```

```

while(zone1 == 0 && zone2 == 1 && zone3 == 0 && zone4 == 0)
{
  Vegetables();
  readMicrocontrollers();

  if(Moist1 < min1)
  {
    digitalWrite(PumpPin1,HIGH);
  }
  else
  {
    digitalWrite(PumpPin1,LOW);
  }
  if(zone2On == 0)
  {
    digitalWrite(PumpPin2,HIGH);
  }
  else if (zone2On == 1)

```

```

    {
        digitalWrite(PumpPin2,LOW);
    }
    if(Moist3 < min3)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(Moist4 < min4)
    {
        digitalWrite(PumpPin4,HIGH);
    }
    else
    {
        digitalWrite(PumpPin4,LOW);
    }
    getParameters();
}

while(zone1 == 0 && zone2 == 0 && zone3 == 1 && zone4 == 1)
{
    Vegetables();
    readMicrocontrollers();

    if(Moist1 < min1)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(Moist2 < min2)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else
    {
        digitalWrite(PumpPin2,LOW);
    }
    if(zone3On == 0)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else if (zone3On == 1)
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(zone4On == 0)
    {

```

```

    digitalWrite(PumpPin4,HIGH);
}
else if (zone4On == 1)
{
    digitalWrite(PumpPin4,LOW);
}
getParameters();
}

while(zone1 == 0 && zone2 == 0 && zone3 == 1 && zone4 == 0)
{
    Vegetables();
    readMicrocontrollers();

    if(Moist1 < min1)
    {
        digitalWrite(PumpPin1,HIGH);
    }
    else
    {
        digitalWrite(PumpPin1,LOW);
    }
    if(Moist2 < min2)
    {
        digitalWrite(PumpPin2,HIGH);
    }
    else
    {
        digitalWrite(PumpPin2,LOW);
    }
    if(zone3On == 0)
    {
        digitalWrite(PumpPin3,HIGH);
    }
    else if (zone3On == 1)
    {
        digitalWrite(PumpPin3,LOW);
    }
    if(Moist4 < min4)
    {
        digitalWrite(PumpPin4,HIGH);
    }
    else
    {
        digitalWrite(PumpPin4,LOW);
    }
    getParameters();
}

while(zone1 == 0 && zone2 == 0 && zone3 == 0 && zone4 == 1)
{
    Vegetables();
    readMicrocontrollers();
}

```

```

if(Moist1 < min1)
{
    digitalWrite(PumpPin1,HIGH);
}
else
{
    digitalWrite(PumpPin1,LOW);
}
if(Moist2 < min2)
{
    digitalWrite(PumpPin2,HIGH);
}
else
{
    digitalWrite(PumpPin2,LOW);
}
if(Moist3 < min3)
{
    digitalWrite(PumpPin3,HIGH);
}
else
{
    digitalWrite(PumpPin3,LOW);
}
if(zone4On == 0)
{
    digitalWrite(PumpPin4,HIGH);
}
else if (zone4On == 1)
{
    digitalWrite(PumpPin4,LOW);
}
getParameters();
}
}

void Vegetables(){
if (Firebase.getString(firebaseData, "/Veg1"))
{

    if (firebaseData.dataType() == "string")
    {
        Veg1 = firebaseData.stringData();
    }

}

if (Firebase.getString(firebaseData, "/Veg2"))
{

    if (firebaseData.dataType() == "string")
    {
        Veg2 = firebaseData.stringData();
    }

}
}

```

```

    }
}

if (Firebase.getString(firebaseData, "/Veg3"))
{
    if (firebaseData.dataType() == "string")
    {
        Veg3 = firebaseData.stringData();
    }
}

if (Firebase.getString(firebaseData, "/Veg4"))
{
    if (firebaseData.dataType() == "string")
    {
        Veg4 = firebaseData.stringData();
    }
}

if (Veg1 == "basil")
{
    min1 = basilmin;
    max1 = basilmax;
}
else if (Veg1 == "mint")
{
    min1 = mintmin;
    max1 = mintmax;
}
else if (Veg1 == "tomato")
{
    min1 = tomatomin;
    max1 = tomatomax;
}
else if (Veg1 == "cucumber")
{
    min1 = cucumbermin;
    max1 = cucumbermax;
}
else if (Veg1 == "parsley")
{
    min1 = parsleymin;
    max1 = parsleymax;
}

if (Veg2 == "basil")
{
    min2 = basilmin;

```

```

    max2 = basilmax;
}
else if (Veg2 == "mint")
{
    min2 = mintmin;
    max2 = mintmax;
}
else if (Veg2 == "tomato")
{
    min2 = tomatomin;
    max2 = tomatomax;
}
else if (Veg2 == "cucumber")
{
    min2 = cucumbermin;
    max2 = cucumbermax;
}
else if (Veg2 == "parsley")
{
    min2 = parsleymin;
    max2 = parsleymax;
}

if (Veg3 == "basil")
{
    min3 = basilmin;
    max3 = basilmax;
}
else if (Veg3 == "mint")
{
    min3 = mintmin;
    max3 = mintmax;
}
else if (Veg3 == "tomato")
{
    min3 = tomatomin;
    max3 = tomatomax;
}
else if (Veg3 == "cucumber")
{
    min3 = cucumbermin;
    max3 = cucumbermax;
}
else if (Veg3 == "parsley")
{
    min3 = parsleymin;
    max3 = parsleymax;
}

if (Veg4 == "basil")
{
    min4 = basilmin;
    max4 = basilmax;
}

```

```

}
else if (Veg4 == "mint")
{
  min4 = mintmin;
  max4 = mintmax;
}
else if (Veg4 == "tomato")
{
  min4 = tomatomin;
  max4 = tomatomax;
}
else if (Veg4 == "cucumber")
{
  min4 = cucumbermin;
  max4 = cucumbermax;
}
else if (Veg4 == "parsley")
{
  min4 = parsleymin;
  max4 = parsleymax;
}
}

void readMicrocontrollers() {
  if(radio.available(&pipeNum))
  {
    radio.read(&data,sizeof(data));
    Serial.print("Moisture Level of Zone ");
    Serial.print(pipeNum);
    Serial.print(" is ");
    //Serial.println(data);
    switch (pipeNum)
    {
      case 1:
        Moist1 = data;
        checkrgy1();
        per1 = (Moist1*100)/950;
        Serial.println(per1);
        Firebase.setInt(firebaseData,"MoistZone1",per1);
        if(timer1.finished())
        {
          moistarray1[i1] = Moist1;
          if (i1 == NumRead-1)
          {
            i1 = 0;
            timer1.start(TimeInterval);
            if(zone1 == 0)
            {
              faultdetection1();
            }
          }
        }
        else
        {

```



```

    i1++;
  }
}
/*for(int f = 0; f < 24; f++)
{
  Serial.println(moistarray1[f]);
}
delay(5000);*/
break;
case 2:
Moist2 = data;
checkrgy2();
per2 = (Moist2*100)/950;
Serial.println(per2);
Firebase.setInt(firebaseData,"MoistZone2",per2);
if(timer2 .finished())
{
  moistarray2[i2] = Moist2;
  if (i2 == 23)
  {
    i2 = 0;
    timer2.start(TimeInterval);
    if(zone2 == 0)
    {
      faultdetection2();
    }
  }
  else
  {
    i2++;
  }
}
break;
case 3:
Moist3 = data;
checkrgy3();
per3 = (Moist3*100)/950;
Serial.println(per3);
Firebase.setInt(firebaseData,"MoistZone3",per3);
if(timer3.finished())
{
  moistarray3[i3] = Moist3;
  if (i3 == 23)
  {
    i3 = 0;
    timer3.start(TimeInterval);
    if(zone3 == 0)
    {
      faultdetection3();
    }
  }
  else
  {

```

```

        i3++;
    }
}
break;
case 4:
Moist4 = data;
checkrgy4();
per4 = (Moist4*100)/950;
Serial.println(per4);
Firebase.setInt(firebaseData,"MoistZone4",per4);
if(timer4.finished())
{
    moistarray4[i4] = Moist4;
    if (i4 == 23)
    {
        i4 = 0;
        timer4.start(TimeInterval);
        if(zone4 == 0)
        {
            faultdetection4();
        }
    }
    else
    {
        i4++;
    }
}
break;
}
}
}

```

```

void faultdetection1() {

    int minormax = 0;
    int minormaxLOC = 0;
    for(int i = 0; i < NumRead; i++)
    {
        if (moistarray1[i] < min1 | moistarray1[i] > max1)
        {
            minormaxLOC = i;
            minormax = 1;
            break;
        }
    }

    if (minormax == 1)
    {
        for(int j = minormaxLOC+1; j < NumRead; j++)
        {
            if(moistarray1[j] > max1 | moistarray1[j] < min1)
            {
                if(j == NumRead-1)

```

```

        {
            fault1 = 1;
            Firebase.setInt(firebaseData, "fault1", 1);
            Serial.println("FAULT IN ZONE 1");
            delay(2000);
        }
    }
    else
    {
        fault1 = 0;
        Firebase.setInt(firebaseData, "fault1", 0);
        break;
    }
}
else
{
    fault1 = 0;
    Firebase.setInt(firebaseData, "fault1", 0);
}
}

void faultdetection2() {

    int minormax = 0;
    int minormaxLOC = 0;
    for(int i = 0; i < NumRead; i++)
    {
        if (moistarray2[i] < min2 | moistarray2[i] > max2)
        {
            minormaxLOC = i;
            minormax = 1;
            break;
        }
    }

    if (minormax == 1)
    {
        for(int j = minormaxLOC+1; j < NumRead; j++)
        {
            if(moistarray2[j] > max2 | moistarray1[j] < min2)
            {
                if(j == NumRead-1)
                {
                    fault2 = 1;
                    Firebase.setInt(firebaseData, "fault2", 1);
                    Serial.println("FAULT IN ZONE 2");
                    delay(2000);
                }
            }
        }
    }
    else
    {

```

```

        fault2 = 0;
        Firebase.setInt(firebaseData,"fault2",0);
        break;
    }
}
else
{
    fault2 = 0;
    Firebase.setInt(firebaseData,"fault2",0);
}
}

void faultdetection3() {

    int minormax = 0;
    int minormaxLOC = 0;
    for(int i = 0; i < NumRead; i++)
    {
        if (moistarray3[i] < min3 | moistarray3[i] > max3)
        {
            minormaxLOC = i;
            minormax = 1;
            break;
        }
    }

    if (minormax == 1)
    {
        for(int j = minormaxLOC+1; j < NumRead; j++)
        {
            if(moistarray3[j] > max3 | moistarray3[j] < min3)
            {
                if(j == NumRead-1)
                {
                    fault3 = 1;
                    Firebase.setInt(firebaseData,"fault3",1);
                    Serial.println("FAULT IN ZONE 3");
                    delay(2000);
                }
            }
            else
            {
                fault3 = 0;
                Firebase.setInt(firebaseData,"fault3",0);
                break;
            }
        }
    }
    else
    {
        fault3 = 0;
    }
}

```

```

    Firebase.setInt(firebaseData,"fault3",0);
  }
}

void faultdetection4() {

  int minormax = 0;
  int minormaxLOC = 0;
  for(int i = 0; i < NumRead; i++)
  {
    if (moistarray4[i] < min4 | moistarray4[i] > max4)
    {
      minormaxLOC = i;
      minormax = 1;
      break;
    }
  }

  if (minormax == 1)
  {
    for(int j = minormaxLOC+1; j < NumRead; j++)
    {
      if(moistarray4[j] > max4 | moistarray4[j] < min4)
      {
        if(j == NumRead-1)
        {
          fault4 = 1;
          Firebase.setInt(firebaseData,"fault4",1);
          Serial.println("FAULT IN ZONE 4");
          delay(2000);
        }
      }
      else
      {
        fault4 = 0;
        Firebase.setInt(firebaseData,"fault4",0);
        break;
      }
    }
  }
  else
  {
    fault4 = 0;
    Firebase.setInt(firebaseData,"fault4",0);
  }
}

void getParameters()
{
  if (Firebase.getInt(firebaseData, "/zone1"))
  {

```

```

if (firebaseData.dataType() == "int")
{
    zone1 = firebaseData.intData();
}
}

if (Firebase.getInt(firebaseData, "/zone2"))
{

    if (firebaseData.dataType() == "int")
    {
        zone2 = firebaseData.intData();
    }

}

if (Firebase.getInt(firebaseData, "/zone3"))
{

    if (firebaseData.dataType() == "int")
    {
        zone3 = firebaseData.intData();
    }

}

if (Firebase.getInt(firebaseData, "/zone4"))
{

    if (firebaseData.dataType() == "int")
    {
        zone4 = firebaseData.intData();
    }

}

if (Firebase.getInt(firebaseData, "/zone1On"))
{

    if (firebaseData.dataType() == "int")
    {
        zone1On = firebaseData.intData();
    }

}

if (Firebase.getInt(firebaseData, "/zone2On"))
{

    if (firebaseData.dataType() == "int")
    {

```

```

    zone2On = firebaseData.intData();
}

}

if (Firebase.getInt(firebaseData, "/zone3On"))
{

    if (firebaseData.dataType() == "int")
    {
        zone3On = firebaseData.intData();
    }

}

if (Firebase.getInt(firebaseData, "/zone4On"))
{

    if (firebaseData.dataType() == "int")
    {
        zone4On = firebaseData.intData();
    }

}

}

}

void checkrgy1()
{
    if (Moist1 < min1)
    {
        rgy1 = 1;
        Firebase.setInt(firebaseData,"rgy1",1);
    }
    else if (Moist1 > max1)
    {
        rgy1 = 3;
        Firebase.setInt(firebaseData,"rgy1",3);
    }
    else
    {
        rgy1 = 2;
        Firebase.setInt(firebaseData,"rgy1",2);
    }
}

void checkrgy2()
{
    if (Moist2 < min2)
    {
        rgy2 = 1;
        Firebase.setInt(firebaseData,"rgy2",1);
    }
    else if (Moist2 > max2)

```

```

    {
    rgy2 = 3;
    Firebase.setInt(firebaseData,"rgy2",3);
    }
else
    {
    rgy2 = 2;
    Firebase.setInt(firebaseData,"rgy2",2);
    }
}

```

```

void checkrgy3()
{
if (Moist3 < min3)
{
    rgy3 = 1;
    Firebase.setInt(firebaseData,"rgy3",1);
}
else if (Moist3 > max3)
{
    rgy3 = 3;
    Firebase.setInt(firebaseData,"rgy3",3);
}
else
{
    rgy3 = 2;
    Firebase.setInt(firebaseData,"rgy3",2);
}
}

```

```

void checkrgy4()
{
if (Moist4 < min4)
{
    rgy4 = 1;
    Firebase.setInt(firebaseData,"rgy4",1);
}
else if (Moist4 > max4)
{
    rgy4 = 3;
    Firebase.setInt(firebaseData,"rgy4",3);
}
else
{
    rgy4 = 2;
    Firebase.setInt(firebaseData,"rgy4",2);
}
}

```



### 3. Mobile Application Codes:

#### 3.1: Control the Irrigation System Page

```
//
// ViewController.swift
// QATra Irrigation System
//
// Created by Fatima Al-Janahi on 2/6/20.
// Copyright © 2020 Fatima Al-Janahi. All rights reserved.
//

import UIKit
import Firebase

class ViewController: UIViewController, UIPickerViewDelegate, UIPickerViewDataSource {
    /**
     * Zone 1
     */

    @IBOutlet weak var pickerView1: UIPickerView!
    @IBOutlet weak var control1: UISegmentedControl!
    @IBOutlet weak var switch1: UISwitch!

    var ref: DatabaseReference!

    override func viewDidLoad() {
        super.viewDidLoad()

        ref = Database.database().reference()

        pickerView1.dataSource = self
        pickerView1.delegate = self
        pickerView2.dataSource = self
        pickerView2.delegate = self
        pickerView3.dataSource = self
        pickerView3.delegate = self
        pickerView4.dataSource = self
        pickerView4.delegate = self

        let chosen1 = UserDefaults.standard.integer(forKey: "chosen1")
        let chosen2 = UserDefaults.standard.integer(forKey: "chosen2")
        let chosen3 = UserDefaults.standard.integer(forKey: "chosen3")
        let chosen4 = UserDefaults.standard.integer(forKey: "chosen4")
        pickerView1.selectRow(chosen1, inComponent: 0, animated: false)
        pickerView2.selectRow(chosen2, inComponent: 0, animated: false)
        pickerView3.selectRow(chosen3, inComponent: 0, animated: false)
        pickerView4.selectRow(chosen4, inComponent: 0, animated: false)

        let zone1 = UserDefaults.standard.integer(forKey: "zone1")
        let zone1On = UserDefaults.standard.integer(forKey: "zone1On")
        if zone1 == 0
        {

```

```

    control1.selectedSegmentIndex = 1
    switch1.isHidden = true
}
else
{
    control1.selectedSegmentIndex = 0
    switch1.isHidden = false
}

if zone1On == 0
{
    switch1.isOn = false
}
else
{
    switch1.isOn = true
}

let zone2 = UserDefaults.standard.integer(forKey: "zone2")
let zone2On = UserDefaults.standard.integer(forKey: "zone2On")
if zone2 == 0
{
    control2.selectedSegmentIndex = 1
    switch2.isHidden = true
}
else
{
    control2.selectedSegmentIndex = 0
    switch2.isHidden = false
}

if zone2On == 0
{
    switch2.isOn = false
}
else
{
    switch2.isOn = true
}

let zone3 = UserDefaults.standard.integer(forKey: "zone3")
let zone3On = UserDefaults.standard.integer(forKey: "zone3On")
if zone3 == 0
{
    control3.selectedSegmentIndex = 1
    switch3.isHidden = true
}
else
{
    control3.selectedSegmentIndex = 0
    switch3.isHidden = false
}
}

```

```

if zone3On == 0
{
    switch3.isOn = false
}
else
{
    switch3.isOn = true
}

let zone4 = UserDefaults.standard.integer(forKey: "zone4")
let zone4On = UserDefaults.standard.integer(forKey: "zone4On")
if zone4 == 0
{
    control4.selectedSegmentIndex = 1
    switch4.isHidden = true
}
else
{
    control4.selectedSegmentIndex = 0
    switch4.isHidden = false
}

if zone4On == 0
{
    switch4.isOn = false
}
else
{
    switch4.isOn = true
}
}

```

```

@IBAction func index1(_ sender: UISegmentedControl) {
    switch control1.selectedSegmentIndex
    {
        case 0:
            UserDefaults.standard.set(1, forKey: "zone1")
            self.ref.child("zone1").setValue(1)
            switch1.isHidden = false
            print("zone 1 manual")
        case 1:
            UserDefaults.standard.set(0, forKey: "zone1")
            self.ref.child("zone1").setValue(0)
            switch1.isHidden = true
            print("zone 1 automatic")
        default:
            UserDefaults.standard.set(1, forKey: "zone1")
            self.ref.child("zone1").setValue(1)
            switch1.isHidden = false
            print("zone 1 manual")
    }
}

```

```

}

@IBAction func switch1control(_ sender: UISwitch) {
    if switch1.isOn
    {
        UserDefaults.standard.set(1, forKey: "zone1On")
        self.ref.child("zone1On").setValue(0)
        print("zone 1 manual on")
    }
    else
    {
        UserDefaults.standard.set(0, forKey: "zone1On")
        self.ref.child("zone1On").setValue(1)
        print("zone 1 manual off")
    }
}

let veg = ["mint",
"basil",
"cucumber",
"tomato", "parsley"]

//*****
//*****ZONE 2*****

@IBOutlet weak var pickerView2: UIPickerView!
@IBOutlet weak var control2: UISegmentedControl!
@IBOutlet weak var switch2: UISwitch!

@IBAction func index2(_ sender: UISegmentedControl) {
    switch control2.selectedSegmentIndex
    {
    case 0:
        UserDefaults.standard.set(1, forKey: "zone2")
        self.ref.child("zone2").setValue(1)
        switch2.isHidden = false
        print("zone 2 manual")
    case 1:
        UserDefaults.standard.set(0, forKey: "zone2")
        self.ref.child("zone2").setValue(0)
        switch2.isHidden = true
        print("zone 2 automatic")
    default:
        UserDefaults.standard.set(1, forKey: "zone2")
        self.ref.child("zone2").setValue(1)
        switch2.isHidden = false
        print("zone 2 manual")
    }
}

@IBAction func switch2control(_ sender: UISwitch) {

```

```

if switch2.isOn
{
    UserDefaults.standard.set(1, forKey: "zone2On")
    self.ref.child("zone2On").setValue(0)
    print("zone 2 manual on")
}
else
{
    UserDefaults.standard.set(0, forKey: "zone2On")
    self.ref.child("zone2On").setValue(1)
    print("zone 2 manual off")
}
}

//*****
//*****ZONE 3*****
@IBOutlet weak var pickerView3: UIPickerView!
@IBOutlet weak var control3: UISegmentedControl!
@IBOutlet weak var switch3: UISwitch!

@IBAction func index3(_ sender: UISegmentedControl) {
    switch control3.selectedSegmentIndex
    {
    case 0:
        UserDefaults.standard.set(1, forKey: "zone3")
        self.ref.child("zone3").setValue(1)
        switch3.isHidden = false
        print("zone 3 manual")
    case 1:
        UserDefaults.standard.set(0, forKey: "zone3")
        self.ref.child("zone3").setValue(0)
        switch3.isHidden = true
        print("zone 3 automatic")
    default:
        UserDefaults.standard.set(1, forKey: "zone3")
        self.ref.child("zone3").setValue(1)
        switch3.isHidden = false
        print("zone 3 manual")
    }
}

@IBAction func switch3control(_ sender: UISwitch) {
    if switch3.isOn
    {
        UserDefaults.standard.set(1, forKey: "zone3On")
        self.ref.child("zone3On").setValue(0)
        print("zone 3 manual on")
    }
    else
    {
        UserDefaults.standard.set(0, forKey: "zone3On")
        self.ref.child("zone3On").setValue(1)
    }
}

```

```

        print("zone 3 manual off")
    }
}
//*****
//*****ZONE 4*****

@IBOutlet weak var pickerView4: UIPickerView!
@IBOutlet weak var control4: UISegmentedControl!
@IBOutlet weak var switch4: UISwitch!

@IBAction func index4(_ sender: UISegmentedControl) {
    switch control4.selectedSegmentIndex
    {
    case 0:
        UserDefaults.standard.set(1, forKey: "zone4")
        self.ref.child("zone4").setValue(1)
        switch4.isHidden = false
        print("zone 4 manual")
    case 1:
        UserDefaults.standard.set(0, forKey: "zone4")
        self.ref.child("zone4").setValue(0)
        switch4.isHidden = true
        print("zone 4 automatic")
    default:
        UserDefaults.standard.set(1, forKey: "zone4")
        self.ref.child("zone4").setValue(1)
        switch4.isHidden = false
        print("zone 4 manual")
    }
}

@IBAction func switch4control(_ sender: UISwitch) {
    if switch4.isOn
    {
        UserDefaults.standard.set(1, forKey: "zone4On")
        self.ref.child("zone4On").setValue(0)
        print("zone 4 manual on")
    }
    else
    {
        UserDefaults.standard.set(0, forKey: "zone4On")
        self.ref.child("zone4On").setValue(1)
        print("zone 4 manual off")
    }
}

func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {

```

```

    return veg.count
}

func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) ->
String? {
    return veg[row]
}

func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
    let Veg1 = veg[pickerView1.selectedRow(inComponent: 0)]
    let Veg2 = veg[pickerView2.selectedRow(inComponent: 0)]
    let Veg3 = veg[pickerView3.selectedRow(inComponent: 0)]
    let Veg4 = veg[pickerView4.selectedRow(inComponent: 0)]
    UserDefaults.standard.set(pickerView1.selectedRow(inComponent: 0), forKey: "chosen1")
    UserDefaults.standard.set(pickerView2.selectedRow(inComponent: 0), forKey: "chosen2")
    UserDefaults.standard.set(pickerView3.selectedRow(inComponent: 0), forKey: "chosen3")
    UserDefaults.standard.set(pickerView4.selectedRow(inComponent: 0), forKey: "chosen4")
    self.ref.child("Veg1").setValue(Veg1)
    self.ref.child("Veg2").setValue(Veg2)
    self.ref.child("Veg3").setValue(Veg3)
    self.ref.child("Veg4").setValue(Veg4)
    print(Veg1)
    print(Veg2)
    print(Veg3)
    print(Veg4)
}
}

```

### 3.2: Moisture Level Page

```
//  
// MoistViewController.swift  
// QATra Irrigation System  
//  
// Created by Fatima Al-Janahi on 2/20/20.  
// Copyright © 2020 Fatima Al-Janahi. All rights reserved.  
//  
  
import UIKit  
import Firebase  
  
class MoistViewController: UIViewController {  
  
    var ref: DatabaseReference!  
  
    @IBOutlet weak var textMoist1: UILabel!  
    @IBOutlet weak var textMoist2: UILabel!  
    @IBOutlet weak var textMoist3: UILabel!  
    @IBOutlet weak var textMoist4: UILabel!  
    @IBOutlet weak var perc1: UILabel!  
    @IBOutlet weak var perc2: UILabel!  
    @IBOutlet weak var perc3: UILabel!  
    @IBOutlet weak var perc4: UILabel!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        ref = Database.database().reference()  
  
        ref?.child("MoistZone1").observe(.value, with: { (snap1) in  
            let MoistZone1 = snap1.value as? Int  
  
            self.textMoist1.text = MoistZone1?.description  
  
        })  
        ref?.child("MoistZone2").observe(.value, with: { (snap2) in  
            let MoistZone2 = snap2.value as? Int  
  
            self.textMoist2.text = MoistZone2?.description  
  
        })  
  
        ref?.child("MoistZone3").observe(.value, with: { (snap3) in  
            let MoistZone3 = snap3.value as? Int  
  
            self.textMoist3.text = MoistZone3?.description  
  
        })  
  
        ref?.child("MoistZone4").observe(.value, with: { (snap4) in  
            let MoistZone4 = snap4.value as? Int
```



```

        self.textMoist4.text = MoistZone4?.description
    })

    ref?.child("rgy1").observe(.value, with: { (snap11) in
        let rgy1 = snap11.value as? Int

        if rgy1 == 1
        {
            self.textMoist1.textColor = UIColor.red
            self.perc1.textColor = UIColor.red
        }
        if rgy1 == 2
        {
            self.textMoist1.textColor = UIColor.green
            self.perc1.textColor = UIColor.green
        }
        if rgy1 == 3
        {
            self.textMoist1.textColor = UIColor.orange
            self.perc1.textColor = UIColor.orange
        }
    })

    ref?.child("rgy2").observe(.value, with: { (snap22) in
        let rgy2 = snap22.value as? Int

        if rgy2 == 1
        {
            self.textMoist2.textColor = UIColor.red
            self.perc2.textColor = UIColor.red
        }
        if rgy2 == 2
        {
            self.textMoist2.textColor = UIColor.green
            self.perc2.textColor = UIColor.green
        }
        if rgy2 == 3
        {
            self.textMoist2.textColor = UIColor.orange
            self.perc2.textColor = UIColor.orange
        }
    })

    ref?.child("rgy3").observe(.value, with: { (snap33) in
        let rgy3 = snap33.value as? Int

        if rgy3 == 1
        {
            self.textMoist3.textColor = UIColor.red
            self.perc3.textColor = UIColor.red
        }
        if rgy3 == 2

```

```

    {
      self.textMoist3.textColor = UIColor.green
      self.perc3.textColor = UIColor.green
    }
    if rgy3 == 3
    {
      self.textMoist3.textColor = UIColor.orange
      self.perc3.textColor = UIColor.orange
    }
  })

  ref?.child("rgy4").observe(.value, with: { (snap44) in
    let rgy4 = snap44.value as? Int

    if rgy4 == 1
    {
      self.textMoist4.textColor = UIColor.red
      self.perc4.textColor = UIColor.red
    }
    if rgy4 == 2
    {
      self.textMoist4.textColor = UIColor.green
      self.perc4.textColor = UIColor.green
    }
    if rgy4 == 3
    {
      self.textMoist4.textColor = UIColor.orange
      self.perc4.textColor = UIColor.orange
    }
  })

})

}
}

```

### 3.3: Fault Detection Page

```
//  
// FaultDetectionViewController.swift  
// QATra Irrigation System  
//  
// Created by Fatima Al-Janahi on 2/20/20.  
// Copyright © 2020 Fatima Al-Janahi. All rights reserved.  
//
```

```
import UIKit  
import Firebase
```

```
class FaultDetectionViewController: UIViewController {
```

```
    @IBOutlet weak var green1: UIImageView!  
    @IBOutlet weak var green2: UIImageView!  
    @IBOutlet weak var green3: UIImageView!  
    @IBOutlet weak var green4: UIImageView!  
    @IBOutlet weak var red1: UIImageView!  
    @IBOutlet weak var red2: UIImageView!  
    @IBOutlet weak var red3: UIImageView!  
    @IBOutlet weak var red4: UIImageView!
```

```
    @IBOutlet weak var resolved1: UIButton!  
    @IBOutlet weak var resolved2: UIButton!  
    @IBOutlet weak var resolved3: UIButton!  
    @IBOutlet weak var resolved4: UIButton!
```

```
    var ref: DatabaseReference!
```

```
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        ref = Database.database().reference()  
  
        ref?.child("fault1").observe(.value, with: { (snap1) in  
            let fault1 = snap1.value as? Int  
            if fault1 == 0  
            {  
                self.green1.isHidden = false  
                self.red1.isHidden = true  
                self.resolved1.isHidden = true  
            }  
            else  
            {  
                self.green1.isHidden = true  
                self.red1.isHidden = false  
                self.resolved1.isHidden = false  
            }  
        })  
    }
```

```
        ref?.child("fault2").observe(.value, with: { (snap2) in
```

```

let fault2 = snap2.value as? Int
if fault2 == 0
{
    self.green2.isHidden = false
    self.red2.isHidden = true
    self.resolved2.isHidden = true
}
else
{
    self.green2.isHidden = true
    self.red2.isHidden = false
    self.resolved2.isHidden = false
}
})

ref?.child("fault3").observe(.value, with: { (snap3) in
let fault3 = snap3.value as? Int
if fault3 == 0
{
    self.green3.isHidden = false
    self.red3.isHidden = true
    self.resolved3.isHidden = true
}
else
{
    self.green3.isHidden = true
    self.red3.isHidden = false
    self.resolved3.isHidden = false
}
})

ref?.child("fault4").observe(.value, with: { (snap4) in
let fault4 = snap4.value as? Int
if fault4 == 0
{
    self.green4.isHidden = false
    self.red4.isHidden = true
    self.resolved4.isHidden = true
}
else
{
    self.green4.isHidden = true
    self.red4.isHidden = false
    self.resolved4.isHidden = false
}
})
}

@IBAction func faultresolved1(_ sender: Any) {
    self.ref.child("fault1").setValue(0)
}

```

```
@IBAction func faultresolved2(_ sender: Any) {
    self.ref.child("fault2").setValue(0)
}

@IBAction func faultresolved3(_ sender: Any) {
    self.ref.child("fault3").setValue(0)
}

@IBAction func faultresolved4(_ sender: Any) {
    self.ref.child("fault4").setValue(0)
}

let fault1 = UserDefaults.standard.integer(forKey: "fault1")
let fault2 = UserDefaults.standard.integer(forKey: "fault2")
let fault3 = UserDefaults.standard.integer(forKey: "fault3")
let fault4 = UserDefaults.standard.integer(forKey: "fault4")

}
```