National College of Ireland
BSc in Computing
2017/2018

Sam Conneely
x14517817
x14517817@student.ncirl.ie

Smart Irrigation System

Technical Report

Table of Contents

Contents

Executive Summary

The reason behind the creation of the smart irrigation system is to address the issue of over watering plants and to remove the possibility of forgetting about watering your plants entirely. The smart irrigation system that I will be implementing removes the need to water your plants by hand. The accompanying app will also show you just how much water your plants have, and the system will also let you know when your water storage levels are running low.

By using a plethora of sensors, the system can automatically keep a watch over your favourite plants/crops and display the details directly to your phone whilst you're out and about living your life. The smart irrigation system uses a raspberry pi to control all these sensors and has the power to trigger the self-watering system to that you never have to. The system uses three solenoid valves which are connected to smart relays that in turn activate when the soil samples become a dry undesirable condition for the plants.

1 Introduction

This report has been created to give a in depth analysis of the project and how it was designed and implemented from start to finish. There are many parts to this report, there will be a detailed overview of the technologies that have been used throughout the creation process, the main goals of the project and what it aims to achieve, the requirements of the system and how they should function as well as an implementation section outlining the process that was followed when building the smart irrigation system.

1.1 Background

The idea came very soon after I began looking into what actually had to be done for the project. The main reason for choosing to design, build and implement a smart irrigation system is down to my choice to reduce the amount of water that I waste on a given thing. The need for plants to be watered is known to everybody so where better to save water. By designing a system that only waters the plants when the soil they are in becomes dry was the perfect way of reducing wasted water.

My interest in the Great Green Wall project was also an inspiration for this project. The project is a huge undertaking across Africa that aims to slow and eventually stop the spread of northern deserts. To see an entire continent trying to protect their climate against environmental changes urged me to create a project that could potentially make a difference. I have worked with a Landscaping company for many years and I have an interest in gardening and growing plants as a whole. Recently I have been looking into ways of tackling climate change in small ways. This I believe is another step towards making changes.

1.2 Aims

My aims for this project are to design, build and implement a simple self-watering system that removes the need for owners of plants to keep a watch on their plants and tend to their needs. This irrigation system will automatically water the plants when needed as to keep them healthy.

The goal of this project is to reduce the wasting of water and to implement a system that remove the need for any user input from watering your plants and vegetables. I have an interest in growing plants and have recently began growing vegetables in an allotment near my home. This system would be ideal for the tending of these crops. I intend to in the future to install the system in a garden situation or perhaps a greenhouse as this would provide much needed cover for the electrical components of the project.

For the final presentation of the project I aim to have three separate soil samples with very different conditions. This will allow for each of the moisture meters to be under different stresses. To set these different environment conditions for the soils samples I plan to create:

- The first moisture sensor (moisture_sensor_A) will be located in the first soil sample. This soil sample will contain very small seedlings that have not taken hold yet. These young seedlings will have little to no effect on the soil as they will not require much water to grow. The less water required from the seedlings the slow the soil should dry out. This in turn means that this soil sample should be watered less often. The main loss of moisture from the soil in this sample would most likely be through evaporation.

- An established plant will be present for the second sample (moisture_sensor_B) which will absorb water at a much faster rate from the soil than the sample with the seedlings. The larger plant in this example will have had time to get accustomed to the soil and spread some small roots. These roots will take up more water from the soils, drying the soil in the

process. This established plant should require more water that the previous sample as it is a larger organism.

- The third example I hope to implement will be an enclosed environment. This would be to simulate a greenhouse or a hotter climate. I plan to simulate the hot greenhouse effect using a lamp and a plastic container. Any water that would be evaporated from the soil should collect on the plastic box around the sample and create a very different environment from the others. In this environment, although the moisture will be mostly contained within a container the water content of the soil will differ from the rest of the samples as the humidity and temperature will be vastly different.

1.3 Technologies

Due to the project size and scope there are many things used in order to build this system.

Function	**Resource**
Coding Languages	Pi - Python Android - Java
Development Platforms	Android Studio VS Code Geany IDE
Data Transfer	CloudMqtt Github
Computer Boards	Raspberry Pi GrovePi Kit
Sensors	Ultrasonic Sensor Moisture Sensor Temperature and Humidity Sensor Relay Modules
Other Hardware	Solenoid Valves 9v Batteries 1M Extra Copper Cabling Irrigation Piping/Equipment Electrical Tape
Other Software	VNC Viewer Draw.io

Throughout this project, one of the major issues I had was working around hardware constraints. Due to the use of many different sensors and external components, connecting all of these together was difficult to overcome.

Hardware

The main components contained in the system will be the moisture sensors, the solenoid valves, the relay modules and the GrovePi and RaspberryPi. Each of these parts will have a major function in the system. The moisture sensors are very small and act as tiny probes that are stuck into the soil for gathering the moisture content of the sample. These modules return a data reading to the GrovePI/RaspberryPi allowing for the computations to take place. These computations will be done in a python program that will use these values to gain the moisture content within the soil samples. When a desired value is reached, the RaspberryPi will send an electrical current to the appropriate relay, in turn completing the circuit that the valves are on. This triggering of the relay will activate the solenoid valve in which the 9v battery is powering.

There were difficulties in learning how all of these components will function together and how to implement a system that allows them to work with one another without any issues.

Software

There were two languages used when coding this project, Python and Java. The main functionality of the code was written in Python on the RaspberryPi and Java was used for the creation of the android mobile application. Python was used as the language for coding on the Pi as it is a very popular and language that includes a vast array of libraries that can be used, allowing for easier coding of projects for IoT related topics. Most of this code was written to the Pi using VNC Viewer. This program allowed me to connect to the Pi remotely and program while away from the Pi. When not coding directly onto the Pi a platform called VS Code was used on a windows machine to code in Python. Once any code had been written it would be pushed from the windows machine to Github where it could be pulled to the Raspberry Pi and tested on another IDE of choice.

One of the main goals of the project was to send the moisture sensor data, and other sensor data alike from the RaspberryPi to the android application. This connecting of the two devices was a major concern throughout the planning of the project but thankfully during our Internet of Things module we were thought about MQTT (Message Queuing Telemetry Transport) and this allowed me to publish the data from the RaspberryPi quickly and easily to the cloud using CloudMqtt. This data could then be pulled from the cloud back down to the android application where it could be displayed for user consumption.

Using Android Studio, I designed and built an application from scratch. The application is built using Java and connects to the CloudMqtt server using some simple code. The application was created by following the design diagram I made a few months ago.

2 System

2.1 Requirements

When broken down into its most basic elements, the main requirements of this project are as follows:

1. All sensors must be connected to the RaspberryPi.
2. Data must be sent from Pi to the android Application.
3. Application has to display the received data.
4. Python code must activate the valve system.
5. Water is required to flow through the system and water the soil samples.

These requirements are the most important features of this system. Without the use of a single requirement the system will not work as intended and the project will be unsuccessful.

The need to have all sensors transmitting data is necessary as the other elements of the system require certain values from these sensors in order to function correctly. The moisture sensors are the main components of this system as the readings they return will dictate when the solenoid valves are triggered and subsequently when and which soil sample is watered.

When creating the application, the final layout and design must be taken into account. The finished application must be simple to use and is required to serve its purpose of clearly displaying the data gathered from the sensors connected to the Pi. Although there is no functionality available to user from the app itself the layout should be well made and clear enough to read easily.

2.1.1 Functional requirements

Moisture Sensor Data Collection

There are three moisture sensors included with this project. These sensors are collecting analog data about the moisture content of the soil they are placed into. This data will be represented on the android application where it will be labelled and display the info for the user to see.

Ultrasonic Sensor Data Collection

In order to gauge how much water there is available to the system the ultrasonic sensor is required. This acts as a sonar device and will display the amount of water in the storage tank.

Temperature and Humidity Sensor Data Collection

This sensor is being added to the system in order to collect data about the enclosed environment surrounding the soil sample (moisture_sensor_C). This specific data is being collected to see if there is a difference in how the moisture content of the soil changes whilst being set at unique temperature and humidity to the other soil samples.

RaspberryPi Data Retrieval

The Raspberry Pi will continually run a Python file that will allow the sensors to collect the data. The method that the file runs will pass through each of the sensors code executing it and looping again and again until commanded to stop.

Any data gathered by the sensors will be pushed to CloudMqtt using a publish method on the Python side of things. When this data reaches CloudMqtt it will be retrieved by the android end and pulled down to the app in turn being displayed there.

Solenoid Valves Opening

For the solenoid valves to work they must only trigger when the moisture sensors return a specific precoded value. For the valves to work properly, the accompanying relay must be activated at the correct time. Each valve will have a corresponding relay and each relay a moisture sensor. For this to work seamlessly the cycle must trigger correctly.

Mobile Application Notifications

The RaspberryPi must send a push notification to the mobile device of the user once the values being collected from the moisture sensors are at a specific value. The mobile app will alert the user to this event.

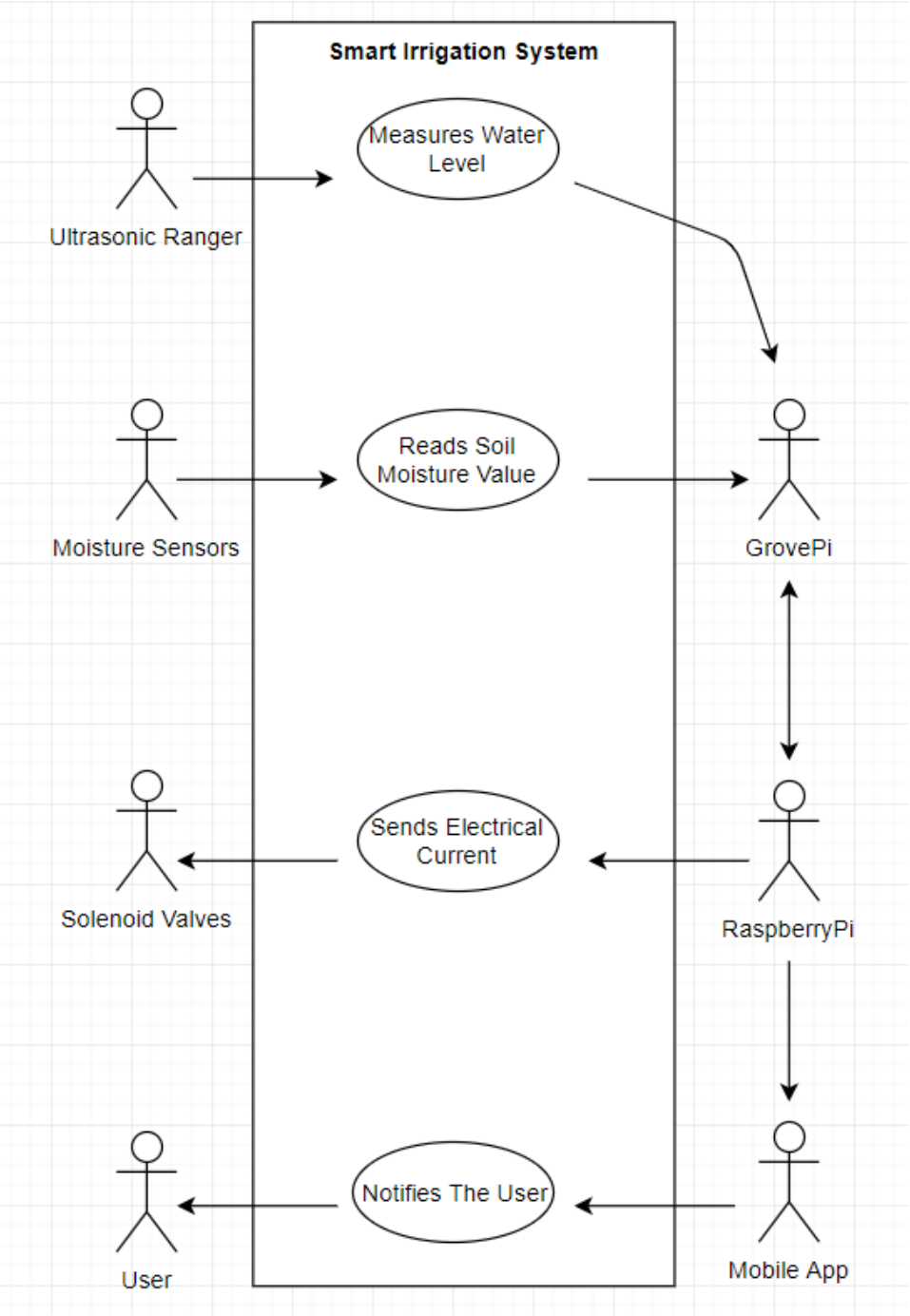
This was going to be an interesting feature available to the user from the android application, but the app does not send the user any notifications when certain sensor values are reached or when watering is taking place as this proved too difficult to implement.

Mobile Application Display

The mobile application that will be accompanying the irrigation system will act as a display hub for all the information gathered by the sensors throughout the build. The function of the app will be a display for users and will also include the ability to manually open and close the solenoid valves.

Due to the nature of the project there is a very minimal need for the application to have any functionality. The purpose of this system is to remove the need for users to look after their plants, therefore creating a system that implements too much functionality would be pointless and the system is meant to be installed once and it should run on its own without any interaction.

Use Case Diagram



Requirement 1 <Moisture Sensor Readings>

Description & Priority

Each moisture sensor will be continuously getting readings from the soil sample that it is to be placed into. The value of the readings are percentages of moisture content within the soil around the plant. The three different moisture modules will be sending the current readings to the GrovePi board which is attached to the RaspberryPi board. These two processing units are connected via GPIO pins. Each moisture sensor is also connected to the GrovePi via the same pins.

This function is extremely important as the rest of the elements within the project are based on and function around the data values collected from the sensors.

Priority: Very High

Use Case

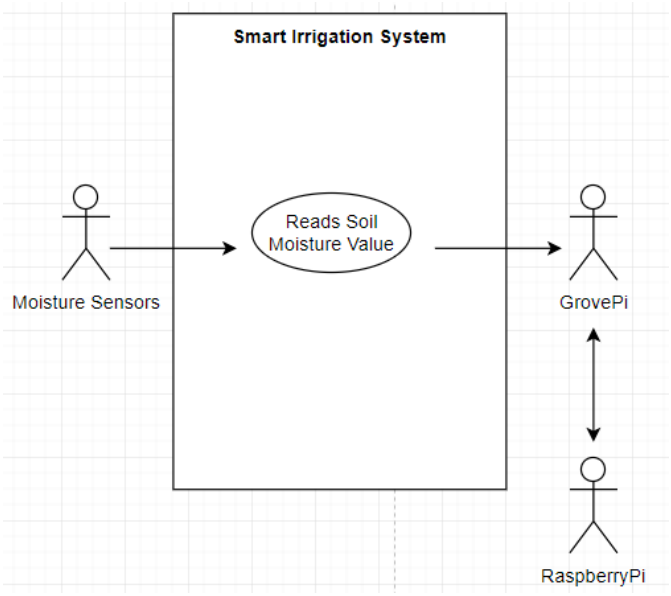
Scope

The scope of this use case is to collect the moisture readings from the three different soil samples that will be sent to the connected computing boards.

Description

This use case covers the collecting and sending of moisture data from three separate moisture sensor modules through the GrovePi sensor kit and into the RaspberryPi computing board.

Use Case Diagram



Precondition

The system sits at stand by waiting for the moisture sensors to gauge a specific value from the soil.

Activation

This use case starts when a moisture sensor reads a certain value for the soil data and sends it to the GrovePi.

Requirement 2 <Ultrasonic Sensor Measurements>

Description & Priority

The ultrasonic ranger is the sensor module that will collect the water levels within the water storage tank. The ultrasonic sensor uses a sonar like functionality to measure the distance between itself and the surface of the water. This module collects the level of the water in the storage tank and sends feeds this data to the GrovePi module where it is then pass on to the RaspberryPi board.

Priority: Medium

Use Case

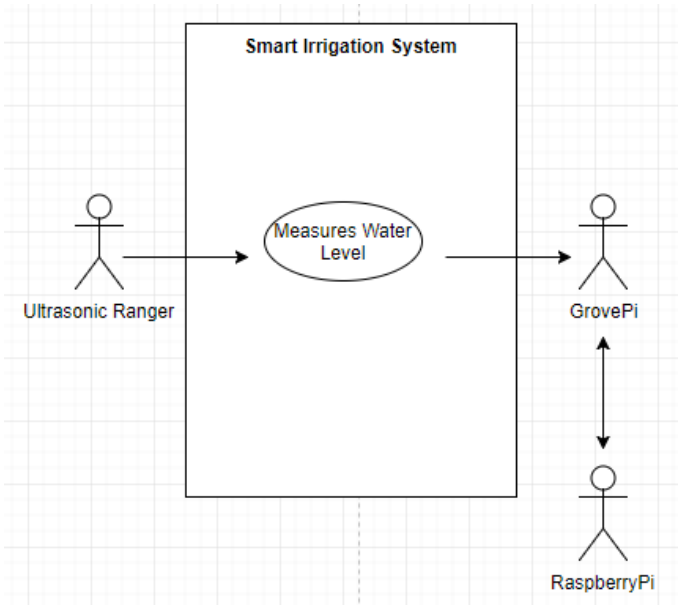
Scope

The scope of this use case is to collect the distance between the top of the water storage container and the sensor itself. This data will be passed to the computer to find the level of water in the tank.

Description

This use case describes the functionality of the Ultrasonic Ranger in conjunction with the GrovePi and RaspberryPi modules. The ranger module will be connected to the GrovePi and situated at the top of the water tank. This sensor is important as the need to know how much water there is in the storage tank will let the user know when it requires refilling.

Use Case Diagram



Precondition

The sensor will continuously measure the distance between the water surface and the ultrasonic ranger module.

Activation

The sensor is always active as the distance readings are being sent to the GrovePi module at a specific constant pace.

Requirement 3 <Solenoid Valve Functionality>

Description & Priority

The solenoid valve is an important feature in the irrigation section of this project as it will control when the water flows through the pipe system and which plant will be getting water as needed. The need for solenoid valves is great as the system would not be able to regulate the water flow otherwise.

Priority: High

Use Case

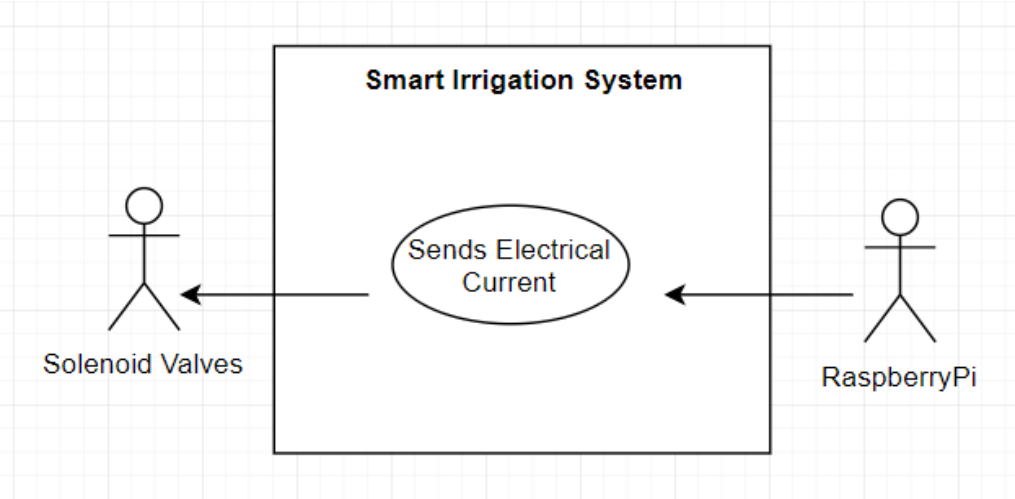
Scope

The scope of this use case is to allow the valves to open and close when required. These valves will also regulate the flow direction of the system.

Description

The use case shows that the valves are opened when an electrical current is passed from the RaspberryPi to the solenoids which in turn allow water to flow to the plant required.

Use Case Diagram



Precondition

The solenoid valves are by default set to be closed. This ensures that the water does not pass through the valves and water the plants when it is not required.

Activation

The water is passed through the valve when an electrical current is passed into the solenoid by the RaspberryPi. This current releases the pressure and allows the flow of water to the plants.

Requirement 4 <Mobile Application>

Description & Priority

The mobile application that will be implemented will allow for users to receive data from the RaspberryPi. This data will display the sensor readings in real time. The mobile application will receive push notifications about certain sensor readings such as when the water tank requires refilling. This is a minor priority as the main irrigation system will function without the sensor data being displayed to the user.

Priority: Low

Use Case

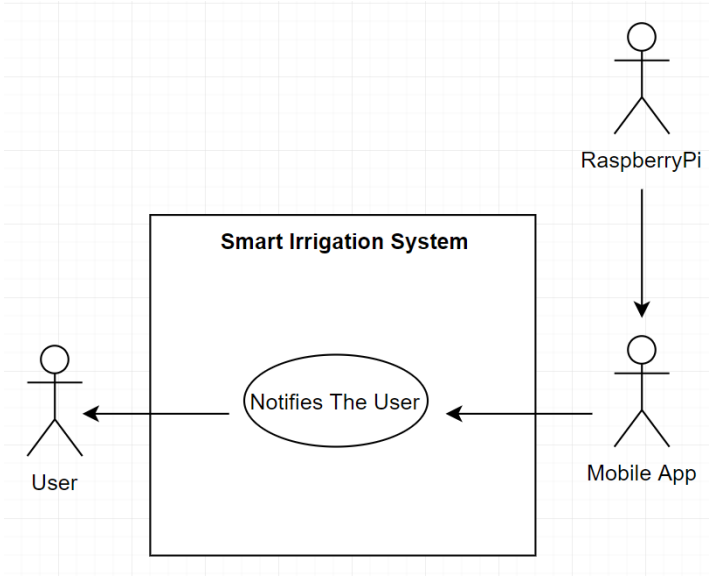
Scope

The data that will be collected from the sensors and sent to the RaspberryPi. These readings will then be displayed on the mobile app in a clear and easy to read manner.

Description

This use case describes the communication between the mobile app and the user. The application will mainly be used for displaying the sensor data in a user-friendly manner. The user will be given notifications with certain sensor data readings.

Use Case Diagram



Precondition

The app will await notifications from the software on the RaspberryPi.

Activation

Once certain data from the sensors is sent to the RaspberryPi dependent on the value of the readings the mobile app will receive notifications

List further functional requirements here, using the same structure as for Requirements 1 & 2. Most systems would have at least five main functional requirements.

Non-Functional Requirements

Specifies any other particular non-functional attributes required by the system. Examples are provided below.

1.1.1 Performance/Response time requirement

The project should update the application with the relevant information quickly but there is no need for extremely fast data transfer. CloudMqtt data transfers seem nearly instantaneous to the user so this should be sufficient.

1.1.2 Availability requirement

The data gathered from the RaspberryPI sensors will be available to the project and the application as long as the Python code is running.

1.1.3 Recover requirement

All of the code will be saved using Github. Here the project files can be stored without the fear of losing them through hardware or software malfunction.

1.1.4 Robustness requirement

The project is certainly robust but has got some smaller parts that could be easily broken or damaged. The over functionality of the system should continue throw some wear and tear.

1.1.5 Security requirement

There is no deploying of this application and the only instance of the app will be the single local android app that we used for developing.

1.1.6 Reliability requirement

Thankfully due to the robustness of the hardware and the simple nature of the software, the project is unlikely to fail. There may be some cable damage over time but that would be a simple fix.

1.1.7 Maintainability requirement

This project should be very simple to maintain due to the cheap sensors and other easily replaceable parts. There are some customised wiring fixes that would take more time to maintain but still these components are very cheap to replace.

1.1.8 Portability requirement

Most elements of this project are very portable and can be quickly and easily moved around from one place to another. Once the components have been wired up and set in place it will be less portable, but the system can be moved with relative ease.

1.1.9 Extendibility requirement

This project uses many different sensors and has the potential the be repurposed or expanded upon. The system could be added to and more sensors connected for larger area coverage if needs be. The project would work perfectly if scaled up for bigger projects.

1.1.10 Reusability requirement

The code of this project would be very simple to reuse for other projects or even when creating another automatic irrigation system. Some of the hardware elements of this project have been fitted purposefully for this exact project but there are sensors that could be reused.

2.1.2 Data requirements

The data requirements for this project are very minimal. The RaspberryPi requires internet if the mobile application is to work. The app will not feed the user with live and current data if there is no internet connection established. The moisture sensors must be continuously collecting live data for the project to function also, without the data collection from the moisture sensors the irrigation system will stop entirely. If the ultrasonic sensor does not send any data, the user will not know how full the water storage tank is and will have to check it manually.

2.1.3 User requirements

The users of this system will require a simple yet effective and fully functioning automatic system. The mobile application should be simple and easy to use allowing users to navigate all of the sensor data with ease. Users will expect a system that once installed will actually work and water their plants when needed. This is the main requirement for users as the project will be a failure if it does not water the plants when they need to be watered.

2.1.4 Environmental requirements

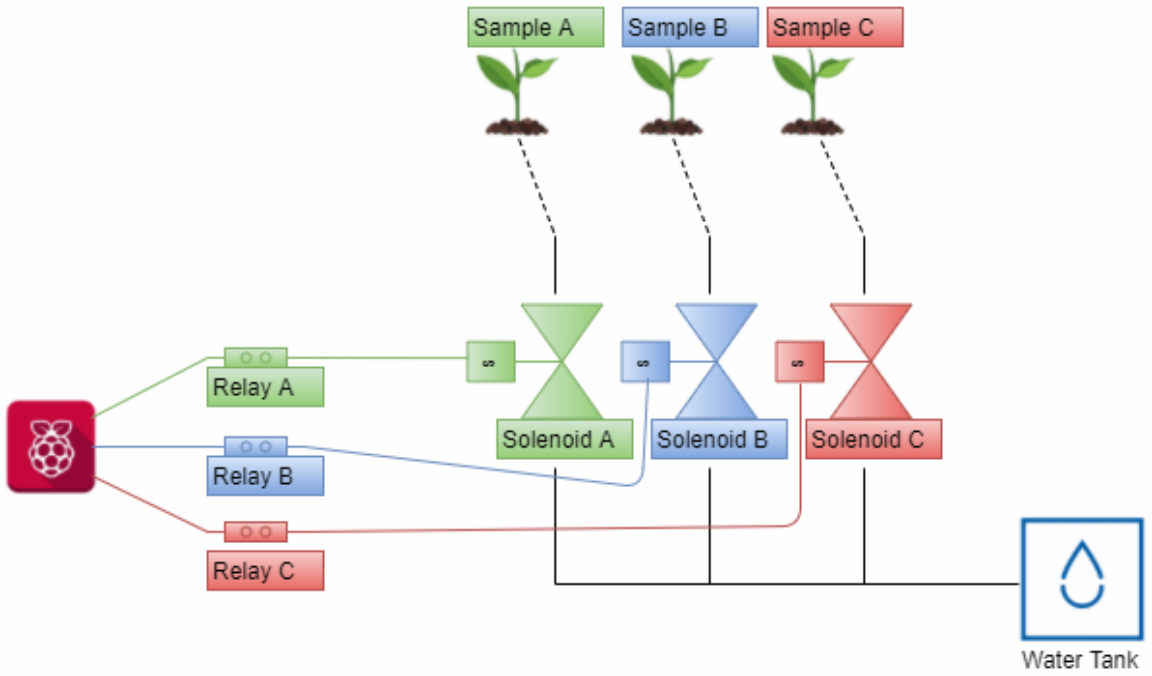
The moisture sensors will require a certain amount of moisture within the soil to work so water is required. The project may expand to include a rain collection system as the main way of storing water, that way it removes the need for users to fill the tank and also removes the need for external water sources to be used, only natural rain water.

2.1.5 Usability requirements

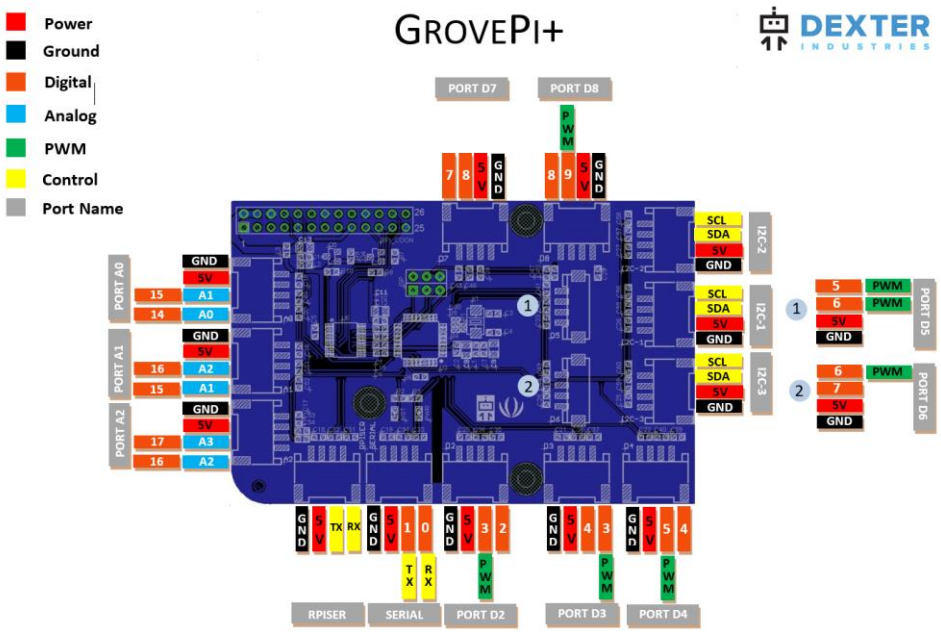
The usability of the project must be kept to a minimum as there must be no need to learn how to use the system or the mobile application. For this reason, the system and app must be very simple and continue functioning long after they have been installed.

2.2 Design and Architecture

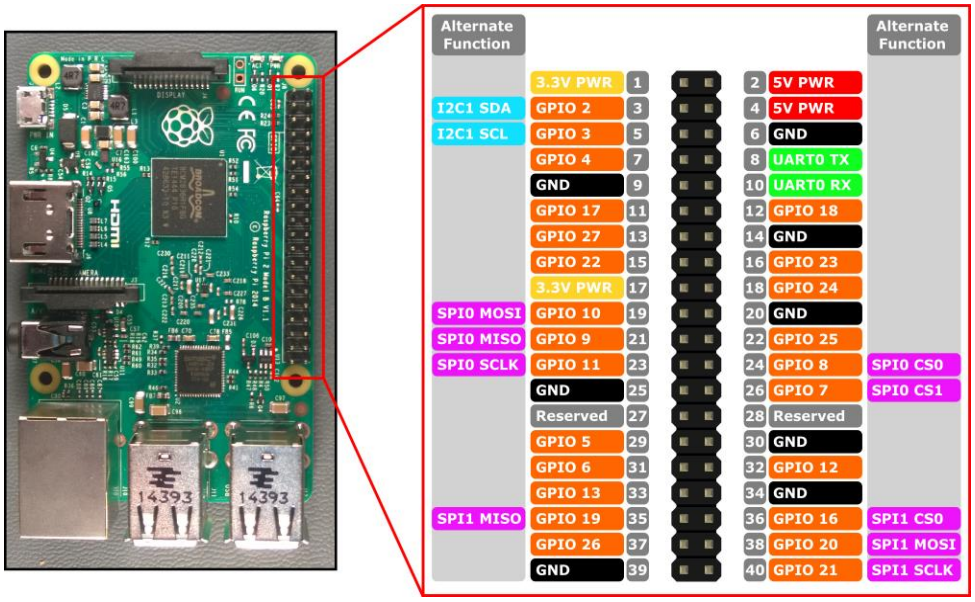
Throughout the planning of this project one of the main considerations has been how to connect the entire project together for the final system. From the beginning of the project simple diagrams were designed to try and get a clear layout of the full and completed project.



Above you can see the proposed design layout for the project once all the parts were connected. In theory this type of layout would be ideal for the project but upon closer inspection and a further understanding about how some of the elements of the project work, there would need to be some alterations to the parts. When designing the layout of the project the lengths of certain cables did not cross my mind. Nearing the end of the project some of the cables required extending. This was not a skill that I had before undertaking this project. The elements of system required they be placed on a board of some sort, this was another task that required the learning of some new skills. The mounting of the pieces to a wooden board was not difficult be required some thought before actually putting the parts in their final places. When deciding on the final location for all the parts of the project, cable management and general tidiness was a concern. The two guides that I followed helped with cable management and layout design in general.



This GrovePi port diagram was important for choosing where each sensor would be located



The GPIO pin layout was extremely important as I need to know which pins corresponded to each part of the board

2.3 Implementation

Where to Begin

When the coding began I was unsure what the best method of coding might be. I had yet to decide whether or not I was going to code directly onto the Pi using Geany IDE or was I to code on my own windows machine with the comfort of VS Code. Initially I chose to code onto the Pi using a Bluetooth keyboard and a 10-inch monitor, but this proved difficult and slow. The best way of doing it was to remotely access the Pi using VNC Viewer and code with Geany IDE which is a program that comes installed on the operating system of the raspberry pi.

Researching the Sensors

In order to get the code that is used for this project to work many test files were created. Every type of sensor used in the system had a file dedicated to testing. Each sensor was tested and the code perfected. The files would begin as long messing python files but as I learned about the sensors and what code was required to execute each sensor the files would become far tidier and the amount of lines of code was greatly reduced. This type of code testing continued until I was happy I had a precise and efficient python file. Once all the code for each individual sensor was written the process began to splice them all together into one simple yet functional file.

When using the GrovePi sensors you have to add your variables and set them to a corresponding analog or digital port on the board. This setting of the sensors to a port can be done very easily at the start of your code. To distinguish between

moisture_sensor_A = 0

moisture_sensor_B = 1

moisture_sensor_C = 2

relay_A = 4

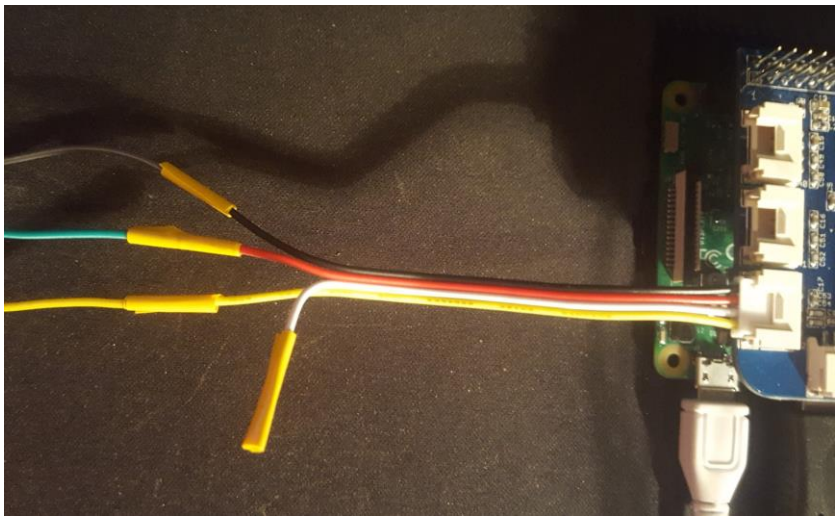
relay_B = 3

relay_C = 2

ultrasonic_ranger = 6

Rewiring the Sensors

The moisture sensors used in this project were not ones from the GrovePi starter Kit and therefore required a little bit more work to get them to function. The moisture sensors came with only jumper lead connections that were to be attached to the GPIO pins on the Pi, so these cables had to be modified for a better connection option. Originally the code for the moisture sensors was hard to understand and there were some parts of the python files that did not look nice and did not read well. The output of values required from the moisture pins was supposed to be a number that would translate well into a percentage but unfortunately when the sensors arrived they were not calibrated correctly for what I wished to do. The initial output of the sensors was a digital one and the values being return were 1's and 0's. In order to achieve the goal of analog output the cables had to be cut open and reconnected to new plugs that would fit into one of the three analog ports on the GrovePi Kit.



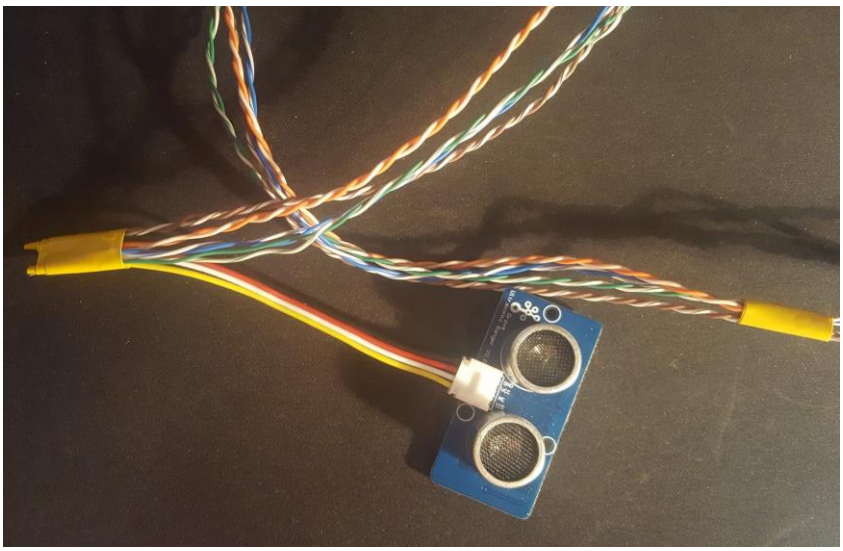
After the modifications were made to the cables, the moisture sensors could be plugged directly into the GrovePi analog ports. The three wires now returned analog reads with values measuring from 0 to 1023. These values could then be converted into percentages using some simple maths and displayed on the application.

```
moisture_A = analogRead(moisture_sensor_A)
moisturePercentA = int(moisture_A * 0.09775)
if moisture_A > 650:
        digitalWrite(relay_A, 1)
        mqttc.publish("relayA", str(100 - moisturePercentA))
        print str(100 - moisturePercentA)
else:
        digitalWrite(relay_A, 0)
        mqttc.publish("relayA", str(100 - moisturePercentA))
        print str(100 - moisturePercentA)

        time.sleep(0.5)
```

Code for getting data from moisture sensor A and activating relay A

Due to the need for the water storage tank to be raised up above the project it was necessary that I extend the length of the cabling for the ultrasonic sensor. The ultrasonic sensor will be placed above the water storage container and will measure the distance between the top of the water and the sensor module. The original cable for the ultrasonic sensor was only a standard 20cm and did not fit the measurements required for the raised water container. More rewiring was done on this cable to extend it and create a new longer cable measured at one meter. The standard GrovePi cable was cut in half and a meter-long section of twisted pair copper wire added to the middle. The GrovePi ports were connected back to each end of the wire and sealed with electrical tape.



The reason that the water container must be placed at a height is down to the need for pressure in the irrigation system. The solenoid valves are made in such a way that the water in the pipes must be at a certain pressure to trigger the valves properly, even if the 9v battery is operating at full capacity. For this reason, the water needs to use its own mass to apply force through the hose that will be connected to the irrigation pipes lower down. With this newly applied pressure the solenoid valves should function as intended.

After converting these values into percentages, the data is ready to be transferred over to the android application via Mqtt.

CloudMqtt Services

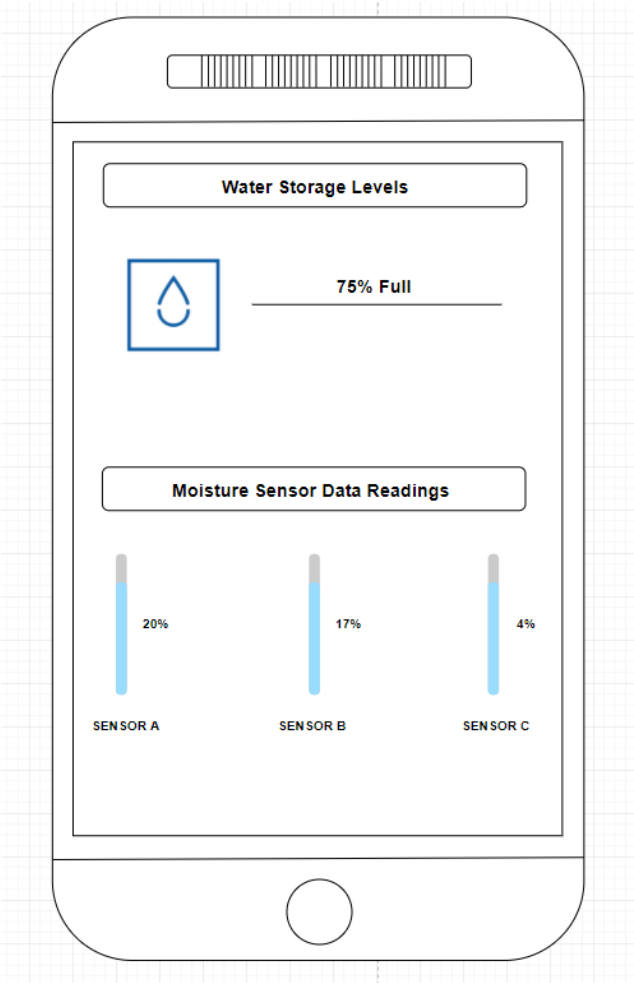
The challenge of connecting both the Raspberry Pi and the Android Application was still a major issue. The answer to this problem came as we learned about Dweet and MQTT in our IoT Principles module at the beginning of the second semester. In order to send sensor data from the Pi to the cloud we used a cloud service call Dweet. This system allowed us to send data from the Pi directly to the cloud and display it on a HTML page. Understanding this was the basics of learning how to send messages across through the cloud using CloudMqtt for the final project. I chose to use CloudMqtt after trying to learn how to adopt Amazon Web Services to my project. This unfortunately did not work as I had hoped as setting up the AWS IoT on the Raspberry Pi was more difficult than I had previously thought. Thankfully CloudMqtt had many tutorials and lots of documentation online for an easy connection of both the Pi and the Application. These tutorials helped me along the way as they were easy to follow and resulted in very quick data transfer.

Setting up a connection to CloudMqtt could be done very quickly and effectively.

```
mqttc = mqtt.Client("ClientA", clean_session = False)
mqttc.username_pw_set(“user", "password")
mqttc.connect("m21.cloudmqtt.com", "port", 60)
```

Code for establishing an Mqtt connection

2.4 Graphical User Interface (GUI) Layout



The mobile application is mainly going to be a display unit. The UI for the mobile app should clearly show the number of sensors you have installed in the system and the ultrasonic ranger data should show the amount of water that is left in the storage tank.

The percentage bars indicate the amount of moisture in the soil for each soil sample. When this value reaches a specific range the watering system will trigger.

A simple GUI design idea is shown, this does not include a temperature and humidity section on the application. This design was created before the idea of an enclosed environment was thought of. The feature will be added to the final version of the application as it will help distinguish between the different soils samples and it also returns great information to the user about how this environment (greenhouse effect) can have a profound effect on the moisture content of soils.

The water storage levels can be seen as a percentage on the application along with the moisture sensor readings.

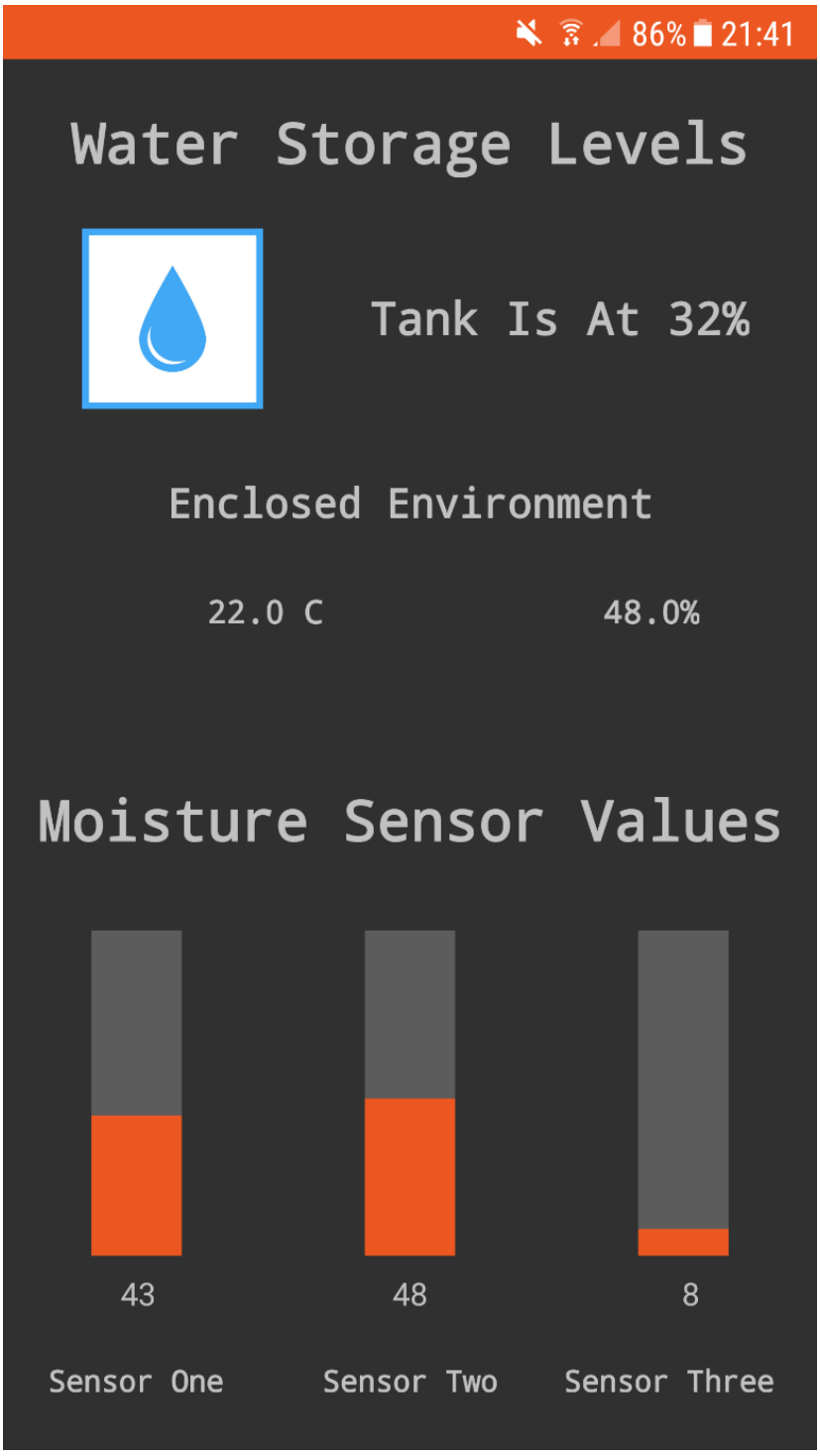
When building the android application, I followed the original design as closely as possible as I thought the original design was very simple, yet the layout was ideal.

The location of all the elements on the application are easy to read and can be clearly understood in my opinion.

The following image is a screenshot of the application running on an android device. The water level in the storage tank is shown at the top of the page as there is only one ultrasonic sensor. I felt that the single sensor data would suffering if put to the bottom of the application and that is why you can clearly see it at the top. Below the water storage levels there is an element called “Enclosed Environment” this is where the temperature and humidity sensor data can be seen. The 22.0 C indicates that the environment in which the sensor in currently located in and the 48.0% is the humidity value which is using the same sensor as the temperature but clearly display the water vapour content of the air (humidity) as a percentage.

Below this again the three moisture sensors are clearly labelled and can be seen in a different colour to the rest of the application. The values returned form the moisture sensors is being displayed on progress bars to give a neat and visually appealing look to the app. At a quick glance the users can clearly see the moisture content of the soils and roughly when they might be getting watered.

Unfortunately, there is no feature that allows the watering of all the plants manually. This is a function I wished to implement but ran out of time nearing the end of the college semester. If I had more time this is one of the first things that I would add to the application to improve the users overall experience on the application.

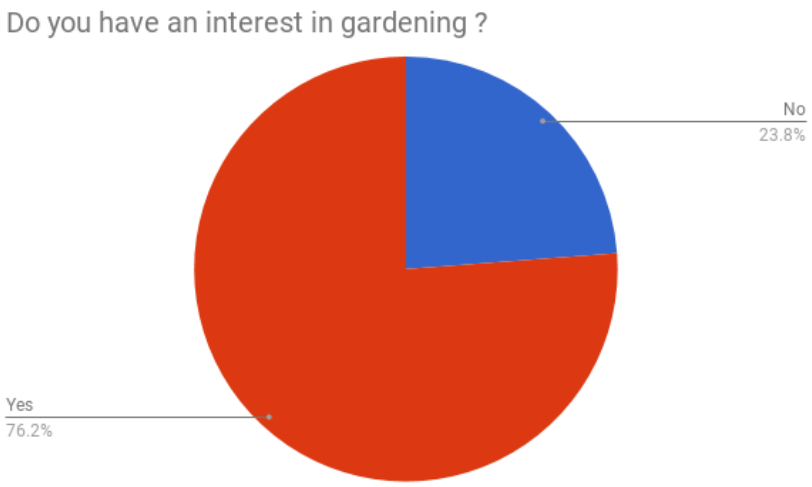

86% 21:41
Water Storage Levels
Tank Is At 32%
Enclosed Environment
22.0 C
48.0%
Moisture Sensor Values
43
48
8
Sensor One
Sensor Two
Sensor Three

2.5 Target Market (Survey Data)

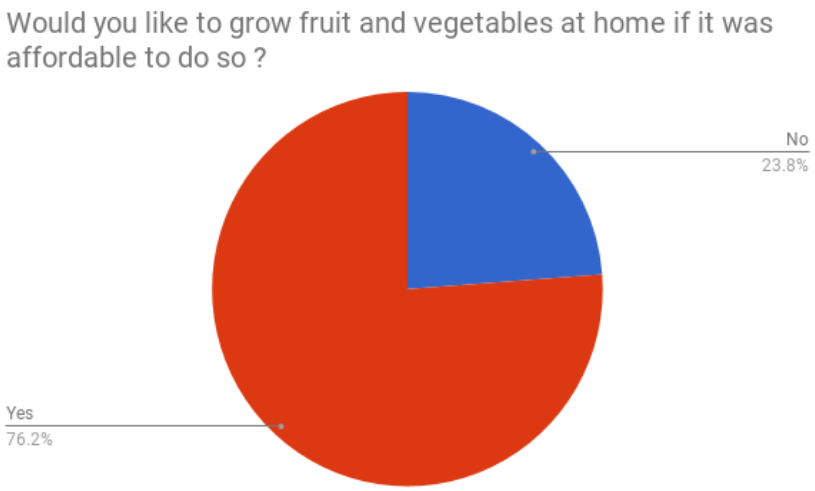
In order to gain a better understanding of whether or not there was a potential market for a Smart Irrigation System or similar products I conducted some user surveys. There was a very short survey created and circulated to possible users. The responses of the survey were gathered and graphed. The data created by the replies helped to build an idea of what the most important factors of the system would be. The survey proposed three simple questions:

- Do you have an interest in gardening?
- Would you like to grow fruit and vegetables at home if it was affordable to do so?
- Why do you think a smart irrigation system would be beneficial?

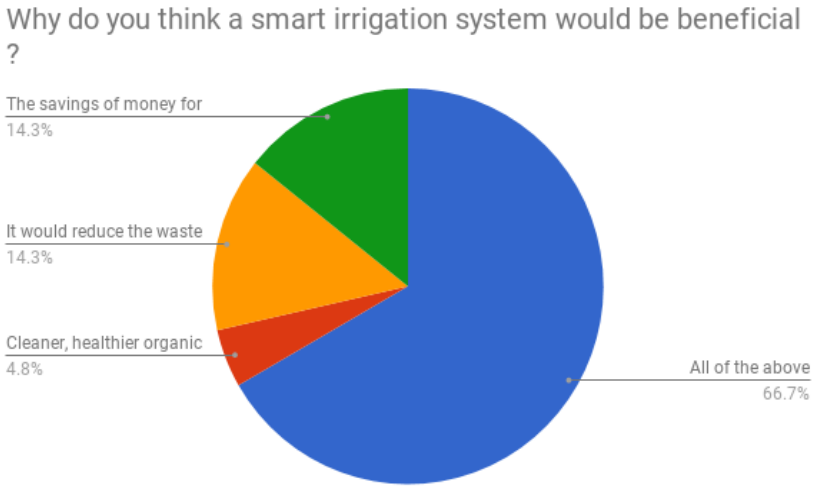
Although very basic, these questions gathered enough data to allow for the creation of graphs that would help forward the potential structure of the system in question.



From this data we can establish that there may in fact be a market for gardening related products as over 75% of the participants said that they do indeed have an interest in gardening. This leads me to believe that people who currently have an interest in gardening may look to purchase a product such as mine.



Once again 76.2% of participants answered “Yes” to this question and from this we can see an extremely clear correlation between people who have an interest in gardening, and people who would consider actually growing sustainable products in it was presented to them in an affordable manner.



From this graph we can see that most people believe that the smart irrigation system would be beneficial in many ways such as savings money for the user, reducing the wastage of water and by producing clean organic fruit and vegetables. I found this data very uplifting as it showed that may be a market for products such as smart irrigation systems.

2.6 Evaluation

The project seems to be success if we consider the main goals were to collect data from a multitude of sensors, publish this data to a cloud service and an accompanying android application, use this data to trigger certain events are specific times according to the return values of the data.

The system is by no means perfect and some elements of the project were not completed to a standard that I had originally intended but all of these are minor features such as mobile application push notifications upon certain results within the data. The loss of some features in the final project may well be negated by the fact that other minor ones were added during the building process that had not been in the first draft of the project. Elements such as the addition of a temperature and humidity sensor to gauge the climate of an enclosed environment and the extra cable extension work that was required. Although the extra work on the wiring was not an added feature it did take a long time and was a major requirement or some of the most functional of elements to work correctly.

3 Conclusion

The project as a whole does function as an automated system. The system is not 100% complete and I honestly don't think it could ever be completely done but I am proud of the standard of project that I have produced through the final year of my college degree. I have learned an uncountable number of things during the process of designing, building and implementing the smart irrigation system.

While the system will water the soil samples, retrieve the necessary data and display this data to a nice user-friendly android application, the project is by no means commercially ready. The system is functional to within a degree of the proposed idea that it is arguably "complete" but there are a thousand and one more tweaks and improvements that could of easily be made if given the time.

4 Further development or research

One of the major resources that I did not consider being an issue at the beginning of this project was time. Other things such as sensors, cables and funding were of course on my mind and plans were put in places to manage these to the best of my ability, but time is one resource that I found far more difficult to ration and use efficiently. For a lot of the time during the year whenever I would do apply myself to getting college work done working on the Software Project was last in the loss of modules that I would work on. This I would think is normal for students as they often prioritise the assignments that due closer to the date. Subsequently this meant that a lot of the work for Software Project was pushed back to “next week” and progress was very much a slow process.

5 References

Cases for your Raspberry Pi. (2015). *Raspberry Pi Plant Pot Moisture Sensor with Email Notification Tutorial*. [online] Available at: https://www.modmypi.com/blog/raspberry-pi-plant-pot-moisture-sensor-with-email-notification-tutorial

Docs.microsoft.com. (2017). *Raspberry Pi 2 & 3 Pin Mappings - Windows IoT*. [online] Available at: https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi

ElectroincTutorials. (2018). *Capacitance and Charge*. [online] Available at: https://www.electronics-tutorials.ws/capacitor/cap_4.html

Fluid Controls. (2018). *What is a solenoid valve?*. [online] Available at: http://www.fluidcontrols.co.uk/what-is-a-solenoid-valve/

GitHub. (2017). *CloudMQTT/python-mqtt-example*. [online] Available at: https://github.com/CloudMQTT/python-mqtt-example/blob/master/app.py

Great Green Wall. (2018). *Great Green Wall*. [online] Available at: http://www.greatgreenwall.org/great-green-wall/#great-green-wall-internal

Industries, D. (2017). *DexterInd/GrovePi*. [online] GitHub. Available at: https://github.com/DexterInd/GrovePi/blob/master/Software/Python/grove_moisture_sensor.py

Maulana Syahidillah, W. (2017). *MQTT Android Client Tutorial*. [online] Wildan's Tech Blog. Available at: https://wildanmsyah.wordpress.com/2017/05/11/mqtt-android-client-tutorial/

Wiki.seeedstudio.com. (2018). *Grove - Temperature&Humidity Sensor Pro*. [online] Available at: http://wiki.seeedstudio.com/Grove-Temperature_and_Humidity_Sensor_Pro/

6 Appendix

6.1 Project Proposal

Objectives
There are many goals that I wish to achieve during the creation and the implementation of my Software Project.

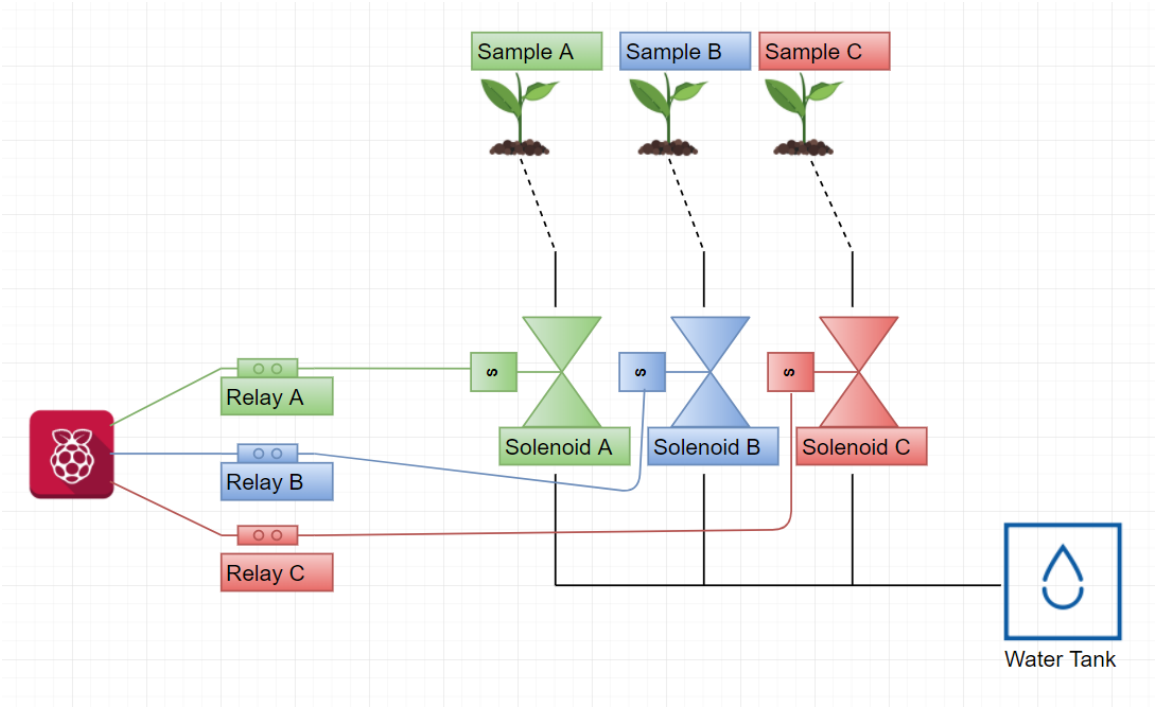
Background (Inspiration)
My Software Project will be to create a smart irrigation system that will allows the watering of plants and crops through the use of a RaspberryPi and many different sensors that will pick up the moisture content of the soil around the plant.
The main reasoning behind the idea of a smart irrigation system was that I have worked with a landscaping company for a couple of years and I have an interest in gardening, but it also extends to the idea of water conservation and reforestation to a small extent. One of the reasons that I thought this project would be a success is that it is a viable option for every household that grows plants or perhaps has a greenhouse and these houses would consume large amounts of water to keeps these plants healthy. This is where the water conservation elements come into play. I believe that with the use of moisture meters and a water tank the amount of water used in keeping plants healthy could be greatly reduced.
I have a huge interest in the Great Green Wall project, this is a large-scale project that is taking place in Africa and it plans to grow a natural barrier consisting of millions of trees across twenty countries. The 8,000km wall is planned to stop the spread of desertification from the north and greatly improves the lives of millions of people by providing a lush green environment for people to grow food and enjoy a better standard of living. This project has inspired me to create a system that will reduce the amount of water wasted in first world countries.
Throughout the summer before I started my fourth year of college the planning of projects was difficult. The process of thinking of different things that could be designed as a final year project was tough. But once I thought of this idea and began planning the system that would be required to build a project like this I was excited to get started on it.

Technical Approach

One of the major difficulties that I will come across during this project is the construction of the hardware for the project to function correctly. This project requires a watering system to be put in place for the water to be taken from the storage tank directly to the plants that will need to be watered. I will have to build a small-scale prototype watering system that will allow the water to travel from source to each plant sample as required. Each soil sample will have an individual moisture sensor and these sensors will allow the RaspberryPi to read the values of each sample and display the amount of water in the soil around each plant. These moisture values are the key to giving the irrigation system a smart element over other watering system.

The system will use a water tank, solenoid valves, hose splitters, GrovePi and a RaspberryPi to run smoothly.

This is an example of how the system will be set up



Special Resources Required
There are many different parts that will be required for the building and completion of my Software Project.
These parts will include: RaspberryPi, Water tank, Length of hose, Solenoid valves, Hose Splitter, GrovePi, Moisture, Ultrasonic, Humidity and Temperature sensors, a breadboard, jumper leads and a relay node.

RaspberryPi: This is the main element of the entire system. The RaspberryPi is the part of the system that will be doing all the work. It is a small credit card sized computer that will controlling the valves, sensors and it is where all the data from the sensors will be pushed to the mobile application. It is the real brain of the project.

GrovePi: This piece of technology is the middleman between the computing elements of this project. The GrovePi acts as the connection where the sensors will be attached to the RaspberryPi. This board sits on top of the RaspberryPi and allows the data from the moisture sensors and ultrasonic sensor to be interpreted and then be acted on.

Water Tank: This will be the storage device that will hold the water before it is distributed out to the individual plant samples.

Ultrasonic Sensor: The ultrasonic sensor is a very useful distance measurement sensor. It uses ultrasonic signal to measure the distance between itself and the object in front of it. This is similar to sonar.

Length of Hose: There will be multiple lengths of hose needed to link all of the systems to one another. The hose must be a certain diameter as to allow a good water flow. As the water will be split into many different hoses after leaving the water tank the hose at the beginning of the system may require a larger diameter as to allow enough water to flow from the tank before being split by the hose splitters.

Solenoid Valves (12v): These are specialised valves that will allow for the stopping of the water flow throughout the irrigation system.

Moisture Sensors: These are small sensors that stick into the soil and return a moisture reading. The moisture level is gathered by the ability of the sensor probes to generate an electric charge. The permittivity of the probe is measured and translated into a percentage which denotes the resistance of the medium that the sensor is located, in this case this is the soil. The resistance measured by the probe is directionally proportional to the amount of water located in the soil around the sensors.

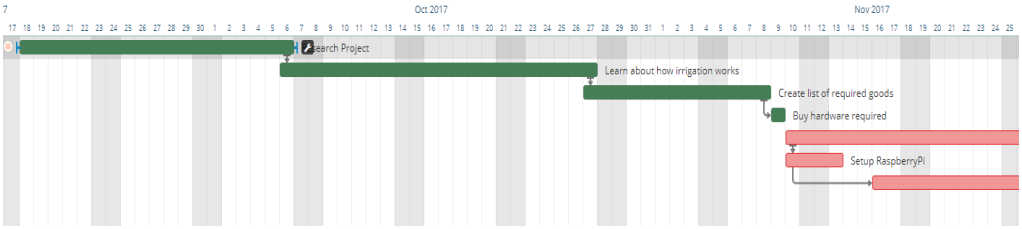
Breadboard: These are very simple connection boards that allows the testing of your electronic circuits. The use of the bread is effective as it gives you the option to plan your circuit before soldering them into place for good.

Jumper Leads: These are cables that connect from the moisture sensors and the RaspberryPi and GrovePi.

Relay Node: Relay modules are very basic switches that you can control with the passing of a certain voltage through the circuit. When the current is passed through the relay it closes the switches

6.2 Project Plan

Below is a Gantt chart that displays the proposed workloads from September through to the end of the year marked as April. Work continued on after the date that I had hoped to finish up by so unfortunately the chart does not extend from April through May as my estimated completion date was nearing the end of April.



7 Monthly Journals

Monthly Journal (September)

Programme: BSHC IoT

Month: September

My Achievements

For the month of September, I was only getting started on my Software Project. I had lots of ideas rolling around my head. I decided to try and narrow down these ideas by writing them down and separating them out based on how realistic they were to complete. This was how I finally ended up with the rough idea for my project. I feel like I had many other good ideas, but I was happy with the project that I chose after eliminating some of the weaker ones.

During the first month there was a lot of research into IoT elements that would require for my project. This lead to the research into the hardware elements that I would need to purchase to put the entire together.

The project pitch was a major part of September, but I feel like it went well. The lecturers seemed to like my idea and it was passed without the need for any revision.

My Reflection

I feel like I should have had a much more precise idea in my head before returning to college after the summer. I am happy at how quickly I decided upon my final project once I returned to college and began considering the project and what could be done.

During the month I would have liked to put more time into getting some hardware together for the build.

Goals for next month

I hope to get some of the sensors and hardware that I will need for the project during October. The main goal of the next month is get begin writing the project proposal and get majority of that out of the way before uploading it about halfway through October.

I will also need to ask about supervisors as I not heard anything about them yet. I will be required to meet with my assigned supervisor and explain my project to them.

Supervisor Meetings

Date of Meeting: N/A

Items discussed: N/A

Action Items: N/A

Monthly Journal (October)

Programme: BSHC IoT

Month: October

My Achievements

During the month of October, I struggled to complete much of the Software Project as there were many other assignments that were required to be finished. I began writing the Project Proposal early during this month but unfortunately did not get to finish it as soon as I had hoped. I did complete the writing of the document during this month.

My Reflection

The month of October came and went very quickly, and I feel like I could have produced much better documents had I been more on top of my time management.

Goals for next month

I hope to get far more on top of things next month and begin working on the Requirements Spec document. I will be purchasing some of the hardware that I will need to begin the construction of my prototype for the midpoint presentation in early December.

Supervisor Meetings

Date of Meeting: 12/10/2017

Items discussed: Spoke about the project. Supervisor helped with some narrowing down of certain items in the project and where I could expand other elements.

Monthly Journal (November)

Programme: BSHC IoT

Month: November

My Achievements

I began working on the Requirements Spec document properly during the month. This month consisted of a lot of research into the hardware I was going to buy. The amount of hardware required was unnerving at first, but I finally settled on buying some sensors from Amazon.co.uk. The process of planning my midpoint presentation was one of my main priorities throughout November.

My Reflection

I honestly believe that the month of November went well, and I was happy with progress this month. Looking forward to the midpoint presentations. Some more work is need on my Requirements Specification document, but I am hopeful that I can get a good result in the midpoint presentation.

Goals for next month

After the midpoint presentations I hope to reduce the amount of work I am currently doing on the Software project and focus on studying for the exams that will be facing me in January. That will most likely be my main goal of December once the midpoint presentation is over.

Supervisor Meetings

Date of Meeting: NA

Items discussed: Unfortunately, I did not meet with my supervisor during this month. If I had of planned a meeting I would of like to show him the list of hardware

that I have accumulated since beginning my research into what would be required for the final system.

Monthly Journal (December)

Programme: BSHC IoT

Month: December

My Achievements

This month began with the midpoint presentations. The presentation went well. I was happy with how I feel it went. I completed the Requirement Spec document to a standard that I was proud of. The document was required for the presentations and had been completed on time. Some of the hardware had arrived in time for the presentations but I did not get to start writing any code beforehand and this may reflect poorly.

My Reflection

The midpoint went well in my opinion and I was happy with how I presented to my supervisor and the second marker. I did not have enough time to get a good start on the project code before the presentations and I feel this may result in lower marks, but I was delighted to be finished and coming up to Christmas I need to focus on the nearing exams.

Goals for next month

The final batch of exams are coming on next month and I hope to be studying for those for most of the next month. Unfortunately, I think this will probably affect my Software Project workflow. No set goals for this project next month.

Supervisor Meetings

Date of Meeting: After Midpoint Presentation

Items discussed: I spoke to my supervisor directly after the midpoint presentation about how the meeting went and what I could expand on from his perspective. He mentioned that perhaps the project could do with another sensor or two and we spoke about the possibility of adding a camera into the project.

Monthly Journal (January)

Programme: BSHC IoT

Month: January

My Achievements

January was a very slow month for this project. The pressure of the exams combined with the laziness of December really didn't help. There was not much done this month. I did some research into how the moisture sensors work but that was about it.

My Reflection

With the timing of the exams this was one of the worst months in terms of progress so far, very little was done but hopefully it was worth it considering I need to pass the exams as well.

Goals for next month

Hopefully I can get back into the rhythm of things next month and start making some progress on the code of the project because as of this point the project still have very little functional code.

Supervisor Meetings

Date of Meeting: NA

Items discussed: There was no meeting with my supervisor this month due to exams and just general lack of urgency to get anything done on this project. The exams took priority.

Monthly Journal (February)

Programme: BSHC IoT

Month: February

My Achievements

February was another slow month in terms of progress. There was a small breakthrough to do with getting some of the sensors working. The ultrasonic sensor code now works perfectly.

My Reflection

Glad to have made some progress on the code of the project but I feel like my other modules require so much time that I cannot focus on the Software Project for any extended period of time.

Goals for next month

I want to get a serious move on with the coding of the project. I am getting worried about how much work still has to be done. I also need to learn more about how Android Studio works and how to make a nice-looking application.

Supervisor Meetings

Date of Meeting: NA

Items discussed: Once again there no proper meeting for this month. I briefly past Arghir by in the corridor and spoke to him about arranging a meeting but I forgot to email him about it, so it didn't happen in the end.

Monthly Journal (March)

Programme: BSHC IoT

Month: March

My Achievements

This was the first month since December that I feel like I made real progress. The moisture sensors are working to some degree. There was a lot of issues with getting them working but they are now at a point where they are returning some values. When the moisture sensors come in contact with water they trigger print 0 to the terminal and when removed from water they return a 1.

My Reflection

The progress that was made with the moisture sensors feels good. For about a week or two during the Feb I was worried about how much I wasn't getting done for Software Project but now at least I feel like something has been done and I can see where the project may be going.

Goals for next month

Continue on the coding of the project and try and finish off the report before my birthday which is on the 22nd of April. Having this done before then would be a serious boost as I feel like I am getting close to the end now. Meet up with supervisor!

Supervisor Meetings

Date of Meeting: NA

Items discussed: I have not met up with Arghir in a few weeks. This is my fault as he has made attempts to schedule meetings, but I have not felt like I have enough done to show to him. This was a mistake on my behalf.