PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No.: 2016016402300885

Seat No: 3037887

Roll No: 34

1. Name of the Student

   KARAN PATIL

2. Title of the Project

   SMART IRRIGATION USING IOT

3. Name of the Guide

   Mr. HARESH PAWAR

4. Teaching experience of the Guide
   3 Years

5. Is this your first submission? [✓ Yes] No ☐

Signature of the Student

Date: …………………

Signature of the Guide

Date: …………………….

Signature of the Coordinator

Date: …………………

SMART IRRIGATION USING IOT

A Project Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Tejshree Dalvi
3037586
&
Karan Patil
3037887

Under the esteemed guidance of

Mr. HARESH PAWAR

DEPARTMENT OF INFORMATION TECHNOLOGY

VIVEKANAND EDUCATION SOCIETY'S COLLEGE OF ARTS, SCIENCE AND COMMERCE

(Affiliated to University of Mumbai)

MUMBAI, 400071

MAHARASHTRA

2018-2019

VIVEKANAND EDUCATION SOCIETY'S COLLEGE OF ARTS, SCIENCE AND COMMERCE

(Affiliated to University of Mumbai)

MUMBAI, 400071

MAHARASHTRA

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, "**Smart Irrigation using IoT**", is bonafide work of **KARAN PATIL** bearing Seat. No: (**3037887**) submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date **College Seal**

ABSTRACT

Water is the most essential contribution for upgrading agricultural productivity and therefore expansion of water system has been a key format in the improvement of farming in the nation. An Automated Sprinkler irrigation method distributes water to crops/plants by spraying it over the crops/plants like a natural rainfall. In this thesis we will develop an automated sprinkle system that will help a farmer/people to know about his field, and the status of his plant at his home or he may be residing in any part of the world. This work will helps the farmers to irrigate the farmland in a very efficient manner with automated irrigation system based on soil, humidity, weather .This sprinkler system will provide control for soil temperature, moisture sensing to ensure plants is watered when there is demand, live streaming and also provide the temperature, humidity sensing, forecast lookup from other weather services. Whenever there is a change in temperature, humidity and current status of rain of the surroundings these sensors senses the change in temperature and humidity and gives an interrupt signal to the raspberry pi. Water excess irrigation not only reduces plants production but also damages soil fertility and also causes ecological hazards like water wasting and salinity. In recent years the awareness of water and energy conversation has resulted in the greater use of sprinkler system .Currently the automation is one of the important roles in the human life. It not only provides comfort but also reduce energy, efficiency and time saving. Now a day the industries are using an automation and control machines which are high in cost and not suitable for using in a farm & garden field. So in this work we will design a smart irrigation technology based on IoT using Raspberry pi. Raspberry pi is the main heart of the overall system. The proposed sprinkler system will be low in cost and usable by the Indian farmers.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the following people whose ceaseless co-operation made it possible, whose constant guidance and encouragement crown all the efforts with success.

Firstly, we would like to thank **Dr. (Mrs.) J.K. Phadnis**, Principal of V.E.S. College of Arts, Science and Commerce for providing the facilities and infrastructure for the project.

We thank our I.T Department Coordinator **Prof. (Mrs.) Jayalakshmi Srinivasan** for her valuable guidance at every stage of project.

We wish to express our profound thanks who helped us in making this project a reality. Much needed a moral support and encouragement was bestowed on numerous occasions by our project guide **Mr. Naveen Gupta**, who not only guided us but also paved the way for and through understanding of developed system. Without full support and cheerful encouragement of **Mr. Naveen Gupta** project would not have shaped the way it is.

We would also like to appreciate **Prof. Mrs. Prajisha Jitesh**, **Mr. Digvijay S. Parab, Mr. Ganesh Anandraj** and **Mr. Kishor Jagtap** for their valuable support whenever it was needed.

Our sincere thanks and apologies to all those who deserves credit but fail to appear in this list.

Thankyou

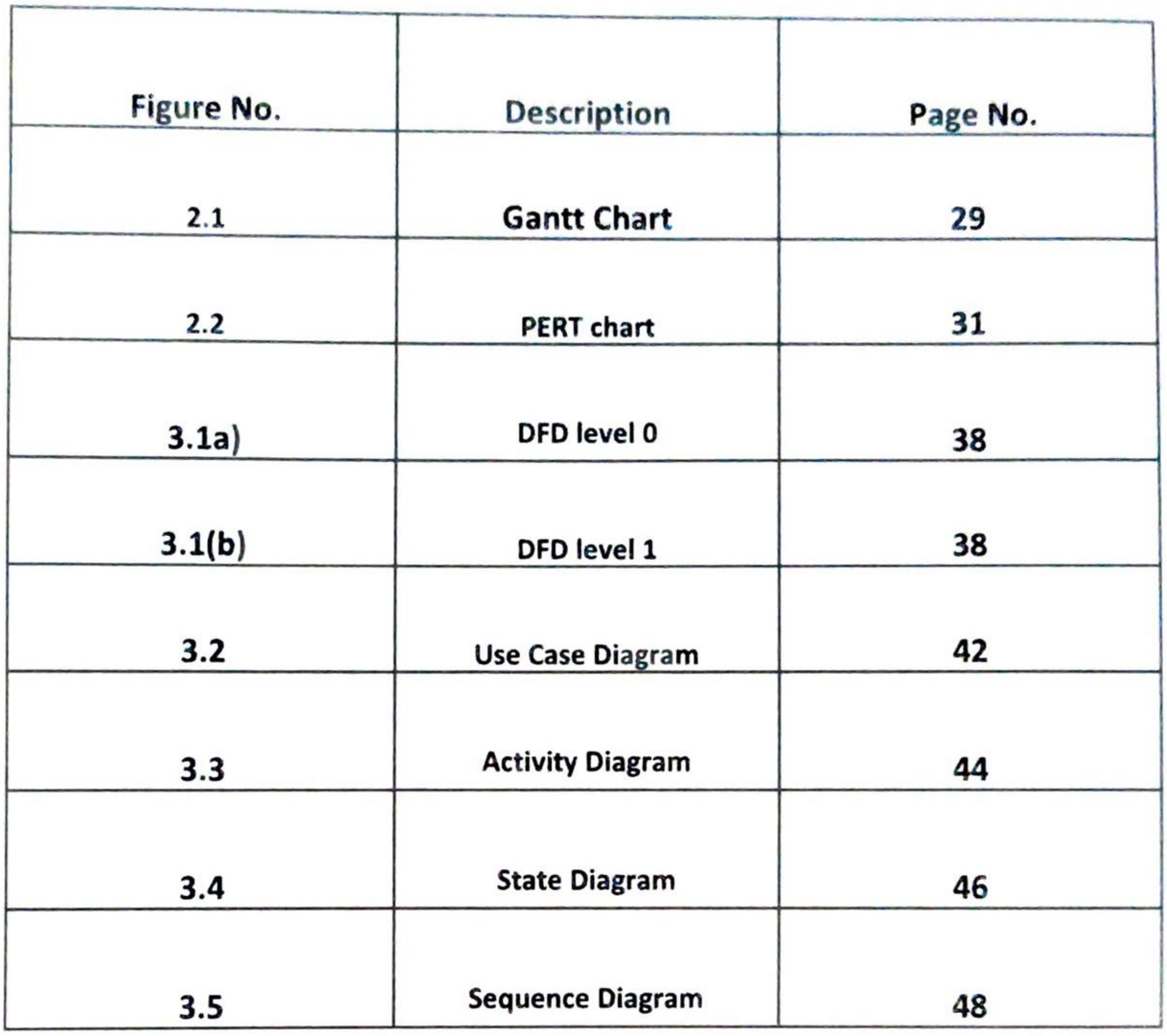
DECLARATION

I hereby declare that the project entitled, “**Smart Irrigation using IoT**”, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

TABLE OF CONTENTS

LIST OF FIGURES



CHAPTER 1
INTRODUCTION

BACKGROUND

With how busy our lives are, it's sometimes easy to forget to pay a little attention to your thirsty indoor plants until it's too late and you are left with a crusty pile of yellow carcasses. Instead of constantly replacing those plants, here is a compact, automated, raspberry pi powered gardener to water and light your plants using IoT. This gardener's memory is impeccable, and assists you in watering your plant in a smarter way. The importance of building an automation system for an office, home or field is increasing day- by-day. Automation makes an efficient use of the electricity, water and reduces much of the wastage. Smart irrigation system makes an efficient use of water for the growth of plants.

Raspberry Pi is the heart of the overall existing system. The Raspberry Pi Model incorporates a number of enhancements and new features. Improved power consumption, enlarged connectivity and greater IO are among the improvements to this powerful, small and lightweight GPIO (General Purpose Input Output) pins. Raspberry Pi model has dedicated general purpose input outputs (GPIO) pins. These all GPIO pins can be accessed for controlling hardware such as LEDs, sensors, and relays, which are examples of outputs. In this work, sensor is interfaced with Raspberry Pi via Wi-Fi Module. This project presents an automatic irrigation system using the Raspberry Pi with soil moisture sensor and water flow management through a simple user friendly website. The system works on IoT based services by which objects or things collect and exchange data when provided with a network. Here, the network communication will be established using the Wi-Fi connectivity and controlled by the webpage controls. The system allows more integration of physical world into computer-based systems resulting in improved economic benefits.

OBJECTIVE

- Smart irrigation is a key component of precision agriculture. This project uses a Raspberry Pi to run the software. It is designed to develop an automatic irrigation system which switches the pump motor ON/OFF on sensing the moisture content of the soil. In the field of agriculture or even at home, use of proper method of irrigation is important. It helps us to **avoid water wastage and improve the quality of crop growth** in their fields by:
  - ✓ Irrigating at the correct times,
  - ✓ Minimizing runoffs and other wastages
  - ✓ Determining the soil moisture accurately
  - ✓ Consumption of water as well as electricity is reduced to an significant amount
- Replacing manual irrigation with automatic valves and systems also **does away with the human error element** (e.g. forgetting to turn off a valve after watering the field), and is instrumental in **saving energy, time as well as resources**.
- Determining the soil moisture levels accurately thereby helps in finding the irrigation requirements at that place.
- The installation and configuration of smart irrigation systems is fairly straightforward which helps the average user in its maintenance also.

PURPOSE

The importance of building an automation system for an office, home or field is increasing day-by-day. Automation makes an efficient use of the electricity, water and reduces much of the wastage. Smart water sprinkler irrigation system makes an efficient use of water for the growth of plants by which your indoor plant may never suffer the lack of water problems.

Problems caused due to Lack of Water in Plants

Plants require varying amounts of supplemental water, depending on the type of plant, growing conditions and amount of rainfall. Young plants need more irrigation than older plants that have developed an established root system. Monitoring the rainfall and amount of water you supply through irrigation helps determine if the plant's water needs are met. Plants also provide physical signs if they are running low on water.

Leaf Changes:

The plant's leaves often show the first signs of too little water. They begin to wilt or droop from lack of moisture. Grass blades wilt and fail to spring back up when stepped on. You may notice shiny plant leaves turning dull. Over time the wilting becomes more pronounced and sometimes becomes permanent. Leaves and stems may turn yellowish or brown with prolonged lack of water. Some plants with drought stress appear scorched along the leaf edges.

Slowed Growth:

Plants that don't receive enough water show a slowdown in growth. If the plant experiences a temporary decrease in water supply, the growth may just slow for a short period. When watering resumes, the plant will likely resume growth. A long-term lack of water may cause the plant to stop growing completely. Leaves that do grow may appear smaller than usual. On trees and shrubs, some branches may die or drop.

Decreased Production:

Because the plant doesn't have enough water to grow properly, a drought-stressed plant often displays a decrease in production. Fruits and vegetables produce fewer crops, affecting your overall yield for the growing season. Flowering plants, trees and shrubs grow fewer buds, resulting in fewer flowers.

Plant Damage:

Extended periods without enough water can cause more severe damage to plants. Instead of just wilting or drooping, the leaves eventually begin to die and drop off of the plant. The weakened root system and stem of the plant becomes more susceptible to damage from insects and diseases. The plant can eventually die if the damage is severe enough. In some cases, supplemental irrigation brings back a plant that appears dead.

Scope:

The main issue here was to design a smart irrigation technology in low cost which is usable anyone in the home and the auto watering feature helps to irrigate the plant by itself when needed. It will help to know his plant status sitting inside his home. The advantage of using this method is to reduce human intervention and still ensure proper irrigation. For the rectification of problems in manual irrigation like wastage of water and human interference the automatic irrigation is used. This system refers to operating systems with no or minimum manual intervention. The introduction of automatic irrigation system has increased application efficiency and reduced requirement. In automatic irrigation 'Raspberry Pi controller' is used to control sensors and for communication purposes. Heart of the system is Raspberry Pi. This project presents an automatic irrigation system using the Raspberry Pi with soil moisture sensor and water flow management. The system works on IoT based services by which objects or things collect and exchange data when provided with a network. Here, the network communication will be established using the Wi-Fi connectivity. The system allows more integration of physical world into computer-based systems resulting in improved economic benefits. In this proposed system Raspberry Pi controller helps to know the conditions of the soil for the irrigation. If the value of sensed data does not match with the threshold values which are coded in the Raspberry pi controller required for the proper crop production and through the controller the relay which act as a switch, motor is turned on then the irrigation will start automatically.

Applicability:

The proposed system is a prototype for automatic controlling an irrigation system. Here prototype includes sensor node and control node. The sensor node is deployed in irrigation field for sensing soil moisture value and the sensed data is sent to controller node on receiving sensor value the controller node checks it with required soil moisture value. When soil moisture in the soil is not up to the required level then the motor is switched on to irrigate associated plant and alert message is send to registered mobile phone. The experimental results show that the prototype is capable for automatic controlling the irrigation motor based on the feedback of soil moisture sensor. This system is used in a remote area and there are various benefits for the farmers. By using the automatic irrigation system it optimizes the usage of water by reducing wastage and reduce the human intervention. It saves energy also as it automatic controlling the system. So there are the system is OFF when the soil is wet and automatically start when the soil is dry. And we present also less number of sensor nodes to use in a large area of field so the cost of the system also decreased. And power consumption of the wireless network devices are also less and the system perform a long time function

Chapter 2
System Analysis

2.1 Existing system

From last decade, few existing systems working for reducing the irrigation water consumption, but these systems have some limitations. So, the modern technology is necessary to resolve this problem and support better irrigation management.
This, in turn, brings to light the importance of smart irrigation powered by the internet of things (IoT) that can go a long way in managing the rising levels of water stress worldwide.

Bennis, H. Fouchal, O. Zytoune, D. Aboutajdine, "Drip Irrigation System using Wireless Sensor
Networks" The Model includes soil moisture, temperature and pressure sensors to monitor the irrigation operations. Specifically, we take into account the case where a system malfunction occurs, as when the pipes burst or the emitters block. Also, we differentiate two main traffic levels for the information transmitted by the WSAN, and we use an adequate priority-based routing protocol to achieve high QoS performance. Simulations conducted over the NS-2 simulator show promising results in terms of delay and Packet Delivery Ratio (PDR), mainly for priority traffic.

Joaquín Gutiérrez, Juan Francisco Villa-Medina et al. "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module", In this paper the System has a distributed wireless network of soil-moisture & temperature sensors placed in root zone of plants. Gateway unit handles sensor information, triggers actuators, and transmits data to a web application. An algorithm was developed with threshold values of sensors that was programmed into a microcontroller-based gateway to control water quantity.

Sangamesh Malge, Kalyani Bhole, "Novel, Low cost Remotely operated smart Irrigationsystem" In this paper a Small embedded system device (ESD) which takes care of a whole irrigation process. The PIC18F4550 microcontroller interfaced with GSM module works as a brain and several sensors like temperature, level and rain works as eyes of this ESD. If and only if eyes of the ESD see all parameters are within a safe range, the PIC18F4550 starts irrigation process by starting the irrigation pump. The farmer gets time to time feedback from ESD through SMS about the action that has taken place by PIC18F4550.

Nikhil Agrawal, Smita Singhal, "Smart Drip Irrigation System using Raspberry pi and Arduino" The commands from the user are processed at raspberry pi using python programming language. Arduino microcontrollers are used to receive the on/off commands from the raspberry pi using zigbee protocol. Star zigbee topology serves as backbone for the communication between raspberry pi and end devices. Raspberry pi acts a central coordinator and end devices act as various routers.

Pravina B. Chikankar, Deepak Mehetre, Soumitra Das, "An Automatic Irrigation System using
ZigBee in Wireless Sensor Network" In the research field of wireless sensor network power efficient time is major issue which can be overcome by using ZigBee technology. The main idea is to understand how data travels through wireless medium transmission using WSN and monitoring system. Design of an irrigation system which is automated by using controllable parameter such as temperature, soil moisture and air humidity because they are the important factors to be controlled in PA(Precision Agriculture).

Sneha Angal "Raspberry pi and Arduino Based Automated Irrigation System" The paper presents a home automation system which is based on Raspberry pi, Arduino microcontrollers, and zigbee and relay boards to water plants. Raspberry pi acts as the control block in the automatic irrigation system to control the flow of motor.
The commands from the Arduino are processed at raspberry pi. Zigbee module is used for communication between the Raspberry pi and Arduino. This paper presents an efficient and fairly cheap automation irrigation system. By using moisture sensor we will make the irrigation system smart and automated. System once installed has no maintenance cost and is easy to use.

Bhagyashree K.Chate , Prof.J.G.Rana ,"Smart irrigation system using Raspberry pi" The aim of this paper is to develop a smart irrigation monitoring system using raspberry pi.Focus area will be parameters such as temperature and soil moisture. This system will be a substitute to traditional farming method .We will develop such a system that will help a farmer to know his field status in his home or he may be residing in any part of the world. It proposes a automatic irrigation system for the agricultural lands. Currently the automation is one of the important role in the human life.

It not only provide comfort but also reduce energy, efficiency and time saving. Now the industries are use automation and control machine which is high in cost and not suitable for using in a farm field. So here it also design a smart irrigation technology in low cost which is usable by Indian farmers. Raspberry pi is the main heart of the whole system.

An automated irrigation system was developed to optimize water use for agricultural crops. Automation allows us to control appliances autom-atically. The objectives of this paper were to control the water motor automatically, monitor the plant growth using webcam and we can also watch live streaming of farm on android mobiles by using wi-fi.

Suprabha Jadhav, Shailesh Hambarde,"Android based Automated Irrigation System using

Raspberry Pi" Nowadays, adopting an optimized irrigation system has become a necessity due to the lack of the world water resource. The system has a distributed wireless network of soil-moisture and temperature sensors.

This project focuses on a smart irrigation system which is cost effective. As the technology is growing and changing rapidly, Wireless sensing Network (WSN) helps to upgrade the technology where automation is playing important role in human life. Automation allows us to control various appliances automatically.

DC motor based vehicle is designed for irrigation purpose. The objectives of this paper were to control the water supply to each plant automatically depending on values of temperature and soil moisture sensors. Mechanism is done such that soil moisture sensor electrodes are inserted in front of each soil. It also monitors the plant growth using various parameters like height and width. Android app.[8]

Nikhil Agrawal , Smita Singhal "Smart Drip Irrigation System using Raspberry pi and Arduino"This paper proposes a design for home automation system using ready-to-use, cost effective and energy efficient devices including raspberry pi, arduino microcontrollers, xbee modules and relay boards. Use of these components results in overall cost effective, scalable and robust implementation of system.

The commands from the user are processed at raspberry pi using python programming language. Arduino microcontrollers are used to receive the on/off commands from the rasperry pi using zigbee protocol.Star zigbee topology serves as backbone for thecommunication between raspberry pi and end devices.Raspberry pi acts a central coordinator and end devices act as various routers.

Low-cost and energy efficient drip irrigation system serves as a proof of concept. The design can be used in big agriculture fields as well as in small gardens via just sending an email to the system to water plants. The use of ultrasound sensors and solenoid valves make a smart drip irrigation system. The paper explains the complete installation of the system including hardware and software aspects.

Gajjala Ashok, Gogada Rajasekar, "Smart Drip Irrigation System using Raspberry Pi and Arduino" This paper proposes a design for home automation system using ready-to-use, cost effective and energy efficient devices including raspberry pi, arduino microcontrollers, xbee modules and relay boards. Use of these components results in overall cost effective, scalable and robust implementation of system.

The sensor data were uploaded in to cloud by raspberry pi using python programming language. Arduino microcontrollers used to transmit the sensor data to the raspberry pi using zigbee protocol. Star zigbee topology serves as backbone for the communication between raspberry pi and end devices. Raspberry pi acts a central coordinator and end devices act as various routers. Low-cost and energy efficient drip irrigation system serves as a proof of concept.

The design can be used in big agriculture fields as well as in small gardens and water plants. The use of ultrasound sensors and solenoid valves make a smart drip irrigation system. The paper explains the complete installation of the system including hardware and software aspects.

2.2 Proposed system

An IoT based irrigation system aims to utilize the features of IoT to make agriculture simple, smart, interesting and time saving. Having sensors connected with controller, the system reads the soil moisture, temperature and electrical conductivity of the soil and then the sensed data are processed in the controller. Raspberry Pi is the heart of the overall existing system. The Raspberry Pi incorporates a number of enhancements and new features. Improved power consumption, enlarged connectivity and greater IO are among the improvements to this powerful, small and lightweight GPIO (General Purpose Input Output) pins. We are using here relay to switch on Water motor, sprinkler. Soil moisture sensor and temperature detection sensor are connected to Raspberry Pi board. Soil moisture sensor gives a resistance variation at the output. That signal is given to the raspberry pi board. It checks for moisture value and the temperature. If the soil moisture value is above the moisture level and humidity is high at the given value and also if the temperature is high then the water motor will be on, whereas if the moisture level and temperature is low the motor will be off through the relay. Initially the threshold moisture and temperature value must be defined. When the sensed moisture value goes above the threshold value, the controller checks for the temperature. Only if the sensed temperature value is higher than the threshold value, irrigation is done and the user is acknowledged. This is because all crops can withstand in the dry soil moisture condition if the temperature is moderate. This would conserve the water used for irrigation and remotely monitor the agriculture area. Sending notification to the user about the field enables the user to remotely monitor the agriculture area. The notification include the warning and suggestion to the affected system. You can use a web browser on a computer, or on a smartphone if it is connected to your Wi-Fi network. For this reason, the Raspberry Pi needs to have a network connection, which also allows the Pi to synchronize with an internet time server. Raspberry Pi is used as an embedded Linux board which is designed based on the ARM V8 microcontroller architecture. The board has an Ethernet interface and runs the simple data web server. The present work focus on automatic control of water motor, monitor the plant growth using mobile application and control of farm on mobile by using Wi-Fi.

2.3 Requirement Analysis

2.3.1 Functional Requirements

2.3.1.1 Central Control Unit (Raspberry Pi Zero W)

Description: The central control unit is responsible for all communication between the web application, soil moisture sensors, and water valves. The control unit will transmit the readings from the soil moisture sensors and report the data to the web application.

The control unit also controls the water valves, which sets the flow of water to the sprinkler system.

We will test the Raspberry Pi for all physical and functional tests. We will connect all the ports like USB, Ethernet, power, audio, and HDMI and test it to make sure it works correctly. We will also test it for its communication with the web server and the Hardware I/O Layer to ensure its functioning as well.

2.3.1.2 Soil Moisture Sensors

Description: In-ground sensors that monitor and report soil moisture levels.

We will provide different soil moisture environments to the sensor and measure its accuracy. It will also be tested against the different physical conditions like pressure and temperature such that it won't break or malfunction.

2.3.1.3 Web Application

Description: The web application will be used to interface with the central control unit. The application is responsible for scheduling watering times as well as interfacing with the central control unit to control the water valves. The web application will be built with a scalable interface to enable the same functionality on a mobile device as it would on a desktop computer while maintaining a similar look and feel.

We will open our web browser on smartphone, tablet and computer. We must be able to see the moisture readings and current schedule for watering on the web application. We will give command such as watering and schedule creation; we must be able to see the updated information.

2.3.1.4 Watering Scheduler

Description: The web application can schedule watering times and durations for the sprinkler system on a soil moisture percentage basis. These schedules are determined by other environmental conditions as well such as temperature and time of year. The scheduler will display upcoming watering schedules and allow users to manually create, edit, or delete them.

We will create, edit, and delete watering schedules and test the respond time and efficiency. When we schedule a watering against the city's schedule, the system must warn the user and suggest user to choose different time available.

2.3.1.5 Soil Moisture Reports

Description: The web application will generate visual reports of the current soil moisture levels. The reports will be generated as graphic representations of the data and will be filterable by zone, year, month, and day.

Readings collected from each soil moisture sensor will be saved on the database. To test this, we will note down some previous data and compare it with the one we see on the soil moisture report. We will test if the sensor reading corresponds with the specific zone, time, and day.

2.3.1.6 Web Hosting Service

Description: There will be a web service that hosts our web application with an associated URL for domain user access.

We will test our application on the web. After we have hosted our web application, we must be able to run our application from the internet.

2.3.1.7 Auto ON/OFF

Description: The user can set an auto on/off timer via the web application that terminates/starts watering if the duration lasts longer than the set time.

For testing, we will put few seconds, e.g. 10 seconds as maximum time to water soil. We won't dip the soil moisture sensor on water and let the watering continue, the watering must stop at 10 seconds.

2.3.1.8 Temperature Sensor

Description: A temperature sensor will monitor the temperature conditions at the central control unit's location. If the sensor detects temperatures around freezing (34° F), the APPLICATION will shut off the watering valves and notify the user to prevent freeze hazards.

We will test the temperature for freezing conditions. We will put ice on the temperature sensor, in which the watering should stop automatically.

2.3.2 Performance Requirements

This section describes all the performance requirements listed on our System Requirement Specification document and gives a brief description of how each feature will be tested to meet the acceptance criteria of the requirement.

2.3.2.1 Sensor Accuracy

Description: Proper sensor readings of soil moisture levels must be captured to ensure accurate and efficient watering schedules. This requires solid construction of the sensor modules and proper software analysis of the data provided by the sensors. All sensors will be tested for their accuracy. To get the accurate reading from the soil moisture sensor, the sensors will be placed on the right depth of soil. Rain sensor and temperature sensor will be on the rooftop. While coding, we will make sure the calibration will be exact.

2.3.2.2 Device Power Malfunction

Description: If the web application fails to receive communication from the central control unit for a specified period of time, it will notify the user that the system is offline. We will test each device and make sure there won’t be any power malfunction under our specified criteria. In case of damage to device, we will test the device to make sure it will go to fail-safe mode.

2.3.2.3 Response Time To Watering

Description: Watering grass when dry is important. But delay of watering for few seconds does no harm to the grass or anyone. After getting command to start watering, APPLICATION may take few seconds to actually have the sprinkles dispensing water. So, the response time is nothing to worry about. We will initiate a web request for watering instantly. We will note the time difference between the service call done from the web and the actual watering. We will also schedule a watering and note the time for the watering to happen by itself. We will test this multiple times and under different environments like varying temperature, changing moisture and different internet speed to make sure our readings are consistent. It should not take more than 10 seconds for our system to respond.

2.3.3 Safety Requirements

This section describes all the safety requirements listed on our System Requirement Specification document and gives a brief description of how each feature will be tested to meet the acceptance criteria of the requirement.

2.3.3.1 Soil Moisture Sensor Insulation

Description: The electronic components of the soil moisture sensors must be properly insulated against external environmental conditions. This is to ensure that they do not malfunction while in use.
The wires connecting the sensor must not touch any conductor or water.
We will test that the insulation is waterproof on the sensor.

2.3.3.2 Central Control Unit Temperature

Description: Efficient airflow and low temperatures must be established while the central control unit is in operation. Computer fans and proper ventilation in the housing container will be used to

cool the device and prevent it from overheating and possibly starting a fire.
We will test the APPLICATION central control unit under different temperatures.
We will put it in different environment temperatures ranging from 10-130 degrees
Fahrenheit and make sure each device is functioning correctly.

2.3.3.3 Proper Wiring of Central Control Unit

Description: The central control unit will be powered from an external wall plug. The source will be 120v and improper wiring could leave the user exposed to currents in the range of 15-20 amps. It is crucial that the control unit is wired properly to accept this input and cause no harm to the device or the user.

We will test that the wires coming and going to the central control unit are not bare anywhere and are connected properly.

2.3.4 Maintenance and Support Requirements

This section details the requirements for ongoing use of application after its final delivery to the user. Achieving maximum long-term performance is the key to reducing product maintenance. These requirements aim to minimize system errors and hardware failures.

2.3.4.1 Sensor Maintenance

Description: The soil and rain sensors will be replaceable in the event of hardware failure.

We will add replace a sensor and test if our system recognizes the new sensor. We will also add and remove few sensors to make sure that sensor mapping works correctly.

2.3.4.2 Scalability

Description: The central control unit will support adding additional valves or soil sensors.
To test the scalability, we will add one more relay module and control the valves. If we can take moisture reading and control the valve for one set, we can do it for multiple set of valves and sensors.

2.3.5 Other Requirements

This section describes all the other requirements listed on our System Requirement Specification document and gives a brief description of how each feature will be tested to meet the acceptance criteria of the requirement.

2.3.5.1 Mapping the Soil Moisture Sensors to Irrigation Valves

Description: Application requires that the user properly maps the soil moisture sensors to the irrigation valves. The central control unit will feature a diagram showing the one-to-one relationship between soil moisture sensor inputs and irrigation valve inputs.
To test the mapping, we will take the reading of the specific soil sensor and turn on the valve associated with it. We will also make sure it's in the right zone by turning on the valves of specific zones individually.

2.3.5.2 Browser Support

Description: The web application should be scalable and responsive in all of the latest versions of Mozilla Firefox, Google Chrome, and Internet Explorer.

We will open our web application on all supporting browsers and make sure we can use all features without any trouble. We will walk through every process that our system allows through web.

2.3.6 Non functional requirements:

This section will detail the requirements of application that will not be tested during the testing phase. Most of the requirements are not to be tested either cannot be tested or their functionality has no impact on how the system operates.

2.3.6.1 Customer Requirements

All customer requirements to be taken care of.

2.3.6.2 Performance Requirements

All functionalities should be responsive and working properly.

2.3.6.3 Safety Requirements

Taking care of hardware components as they can get damaged by the water especially Raspberry Pi.

2.3.6.4 Maintenance and Support Requirements

Software Maintenance

Description: All source code files and relevant documentation must be available for software maintenance and troubleshooting. The required files and documentation will be available via the GitHub repository. The software will be split into loosely coupled parts so it would be easier to make changes and improvements to the system over time.

Reasoning**:** Time constraint. Our software is made from one of the modern and compatible technologies, so there will likely be no issues with our software for a number of years. Also this is a year term project so there is no need to maintain the software.

2.3.6.5 Other Requirements

Central Control Unit Mounting

Description: The central control unit must be mounted in a location where it can interface with wires from the water valves and soil moisture sensors. It must also be within range of Ethernet and power outlets.
Reasoning: People may mount their central control unit anywhere they want.

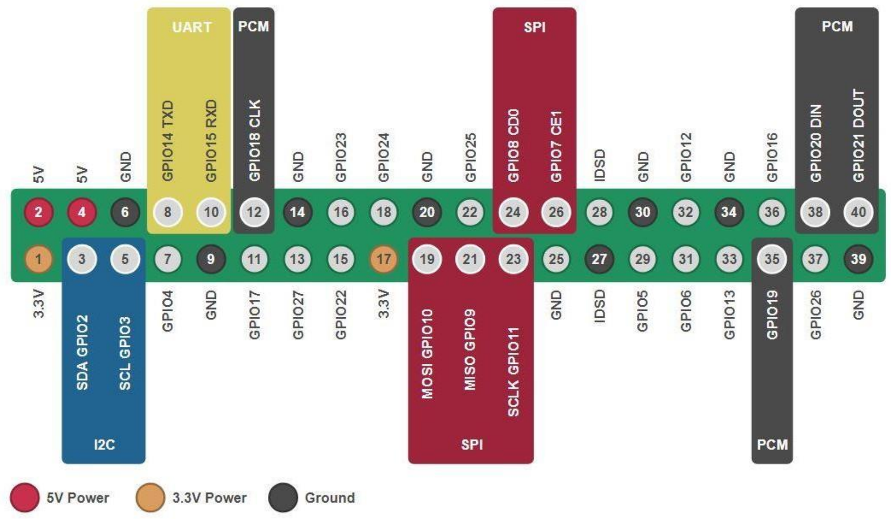
2.4 Hardware Requirements

2.4.1 Raspberry Pi

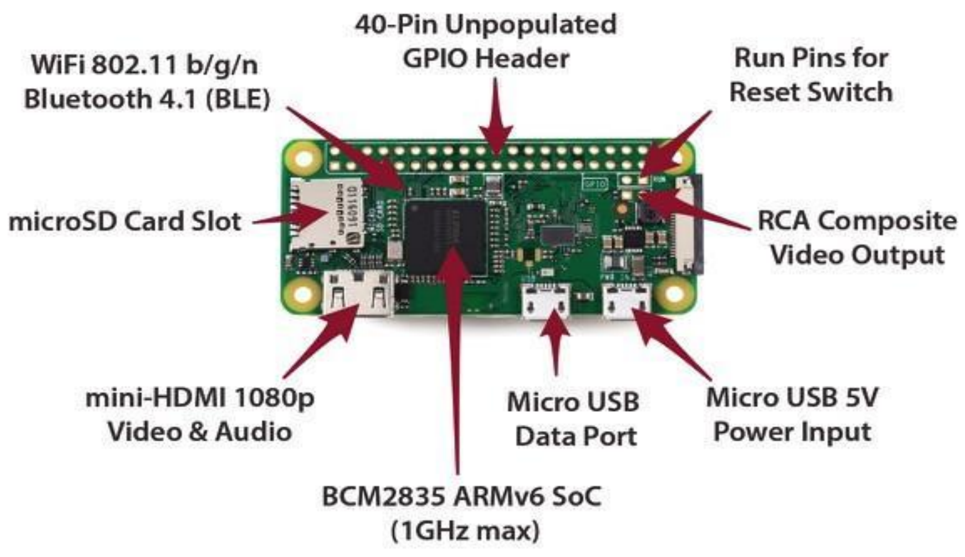
The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is small, powerful and lightweight ARM based computer which can do many of the things a desktop PC can do. The powerful grapapplication capabilities and HDMI video output make it ideal for multimedia applications such as media centers and narrowcasting solutions. The Raspberry Pi is based on a Broadcom BCM2835 chip. It does not feature a built-in hard disk or solid-state drive, instead relying on an SD card for booting and long-term storage. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word- processing, and playing games. Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. Raspberry Pi is being used all over the world to learn to program and understand how computers work.

Raspberry Pi Zero w

GPIO Pin-Out



Raspberry Pi Zero W (Wireless)



40-Pin Unpopulated GPIO Header
WiFi 802.11 b/g/n Bluetooth 4.1 (BLE)
Run Pins for Reset Switch
microSD Card Slot
RCA Composite Video Output
mini-HDMI 1080p Video & Audio
Micro USB Data Port
Micro USB 5V Power Input
BCM2835 ARMv6 SoC (1GHz max)

piaustralia.com.au/raspberry-pi-zero-w

Jumper wires: For connecting components to raspberry pi

2.4.2 Soil moisture sensor

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Measuring soil moisture is important for agricultural applications to help farmers manage their irrigation systems more efficiently. Knowing the exact soil moisture conditions on their fields, not only are farmers able to generally use less water to grow a crop, they are also able to increase yields and the quality of the crop by improved management of soil moisture during critical plant growth stages.

Soil moisture sensor includes comparator which converts analog data to discrete. Two soil probes consist of two thin copper wires each of 5 cm length which can be immersed into the soil under test. The circuit gives a voltage output corresponding to the conductivity of soil. The soil between the probes acts as a variable resistance whose value depends upon moisture content in soil. The resistance across soil probes can vary from infinity (for completely dry soil) to a very little resistance (for 100% moisture in soil) his variation in resistance across the probes (RS) leads to variation in forward-bias voltage which leads to corresponding variation in input base current (Ib).

2.4.3 Temperature sensor:

A **temperature sensor** is a device, typically, a thermocouple or RTD, that provides

for **temperature** measurement through an electrical signal. A thermocouple (T/C) is made from two dissimilar metals that generate electrical voltage in direct proportion to changes in **temperature**.

2.4.4 Relay:

A relay is an electrically operated switch. The relay module is an electrically operated switch that allows you to turn on or off a circuit using voltage and/or current much higher than a microcontroller could handle. The relay protects each circuit from each other. The each channel in the module has three connections named NC, COM, and NO. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal.

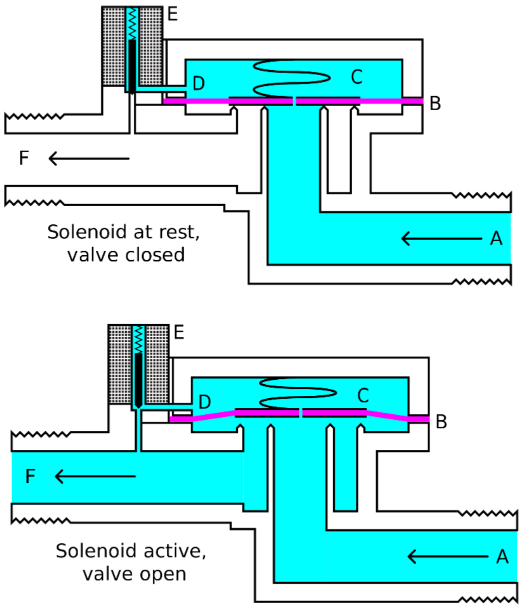
Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect.

2.4.5 Solenoid valve:

A Solenoid valve is an electromechanical device in which the solenoid uses an electric current to generate a magnetic field and thereby operate a mechanism which regulates the opening of fluid flow in a valve.

Solenoid valves differ in the characteristics of the electric current they use, the strength of the magnetic field they generate, the mechanism they use to regulate the fluid, and the type and characteristics of fluid they control. The mechanism varies from linear action, plunger-type actuators to pivoted-armature actuators and rocker actuators. The valve can use a two- port design to regulate a flow or use a three or more port design to switch flows between ports. Multiple solenoid valves can be placed together on a manifold.

Solenoid valves are the most frequently used control elements in fluidics. Their tasks are to shut off, release, dose, distribute or mix fluids. They are found in many application areas. Solenoids offer fast and safe switching, high reliability, long service life, good medium compatibility of the materials used, low control power and compact design.



2.5Software Requirements

2.5.1 Operating System Raspbian

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Stretch and Raspbian Jessie. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012.The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs. Raspbian uses PIXEL, Pi Improved X windows Environment, Lightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Open box stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecraft called Minecraft Pi as well as a lightweight version of Chromium as of the latest version.

2.5.2 Python for basic coding

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.[Van Rossum led the language community until July 2018.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python features a comprehensive standard library, and is referred to as "batteries included".

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model. Python and CPython are managed by the non-profit Python Software Foundation.

2.5.3 Flask

Here, we have created a web server using Flask, which provides a way to send the commands from webpage to Raspberry Pi to control the Robot over the network. Flask allows us to run our python scripts through a webpage and we can send & receive data from Raspberry Pi to web browser and vice versa. Flask is a microframework for Python. This tool is Unicode based having built-in development server and debugger, integrated unit testing support, support for secure cookies and it’s easy to use, these things make it useful for the hobbyist

2.5.4 Internet Explorer or any supported web browser.

Browser to open and operate the web application.

2.5.5 Putty for SSH

PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port. The name "PuTTY" has no official meaning.

PuTTY was originally written for Microsoft Windows, but it has been ported to various other operating systems. Official ports are available for some Unix-like platforms, with work-in-progress ports to Classic Mac OS and macOS, and unofficial ports have been contributed to platforms such as Symbian, Windows Mobile and Windows Phone.

PuTTY was written and is maintained primarily by Simon Tatham.

2.5.6 Thingspeak Account (A server side platform for collecting IoT device data.)

According to its developers, "**ThingSpeak** is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates".

ThingSpeak was originally launched by ioBridge in 2010 as a service in support of IoT applications. ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyze and visualize uploaded data using Matlab without requiring the purchase of a Matlab license from Mathworks. ThingSpeak has a close relationship with Mathworks, Inc. In fact, all of the ThingSpeak documentation is incorporated into the Mathworks' Matlab documentation site and even enabling registered Mathworks user accounts as valid login credentials on the ThingSpeak website. The terms of service and privacy policy of ThingSpeak.com are between the agreeing user and Mathworks, Inc. ThingSpeak has been the subject of articles in specialized "Maker" websites like Instructables, Codeproject, and Channel 9.

2.5.7 VNC Viewer

In computing, Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical-screen updates back in the other direction, over a network. VNC is platform-independent – there are clients and servers for many GUIbased operating systems and for Java. Multiple clients may connect to a VNC server at the same time. Popular uses for this technology include remote technical support and accessing files on one's work computer from one's home computer, or vice versa. VNC was originally developed at the Olivetti & Oracle Research Lab in Cambridge, United Kingdom. The original VNC source code and many modern derivatives are open source under the GNU General Public License. There are a number of variants of VNC which offer their own particular functionality; e.g., some optimised for Microsoft Windows, or offering file transfer (not part of VNC proper), etc. Many are compatible (without their added features) with VNC proper in the sense that a viewer of one flavour can connect with a server of another; others are based on VNC code but not compatible with standard VNC.

2.5.8 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets . Browsers do not display the HTML tags but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

2.6 Justification of selection of Technology

Why Raspberry pi zero w an and why not Arduino?

What is the difference between the two?

An Raspberry Pi is a microcontroller motherboard. A microcontroller is a simple computer that can run one program at a time, over and over again. It is very easy to use.

A Raspberry Pi is a general-purpose computer, usually with a Linux operating system, and the ability to run multiple programs. It is more complicated to use than an Raspberry Pi.

What would I use each for?

An Raspberry Pi board is best used for simple repetitive tasks: opening and closing a garage door, reading the outside temperature and reporting it to Twitter, driving a simple robot.

Raspberry Pi is best used when you need a full-fledged computer: driving a more complicated robot, performing multiple tasks, doing intense calculations (as for Bitcoin or encryption)

Is there a simple rule of thumb to help me decide?

Yes, there is! Think about what you want your project to do. If you can describe it with less than two ‘and’s, get an Raspberry Pi. If you need more than two ‘and’s, get a Raspberry Pi.

Examples:
“I want to monitor my plants and have them Tweet me when they need water.” That can best be done by an Raspberry Pi.

“I want to monitor my plants and have them Tweet me when they need water and check the National Weather Service, and if the forecast is for fair weather, turn on the irrigation system and if the forecast is for rain, do nothing.” That would best be handled by a Raspberry Pi.

CHAPTER 3

SYSTEM DESIGN

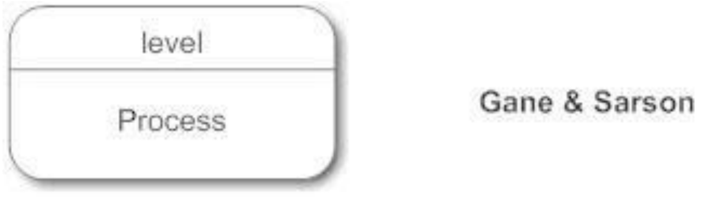
Data Flow Diagram:

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

Data Flow Diagrams Symbols:

There are essentially two different types of notations for data flow diagrams (Yourdon & Coad or Gane & Sarson) defining different visual representations for processes, data stores, data flow and external entities.
Yourdon and Coad type data flow diagrams are usually used for system analysis and design, while Gane and Sarson type DFDs are more common for visualizing information systems.
Visually, the biggest difference between the two ways of drawing data flow diagrams is how processes look. In the Yourdon and Coad way, processes are depicted as circles, while in the Gane and Sarson diagram the processes are squares with rounded corners.
Process Notations:

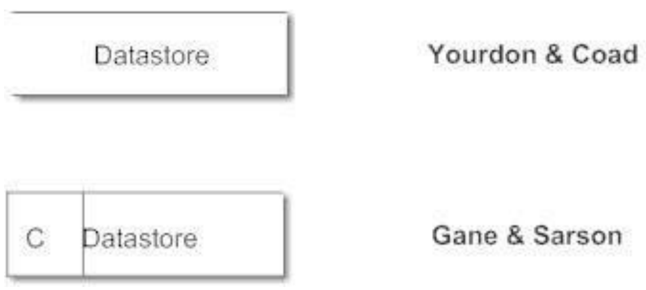
A process transforms incoming data flow into outgoing data flow.



Process Notations

Datastore Notations:

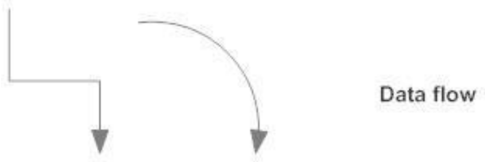
Datastores are repositories of data in the system. They are sometimes also referred to as files.



Datastore Notations

Dataflow Notations:

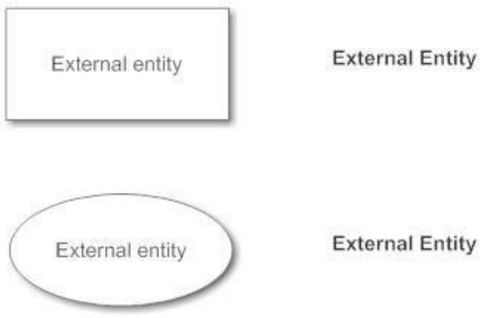
Dataflows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.



Dataflow Notations

External Entity Notations:

External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs.



External Entity Notations

DATA FLOW DIAGRAM FOR SMART IRRIGATION SYSTEM

LEVEL 0

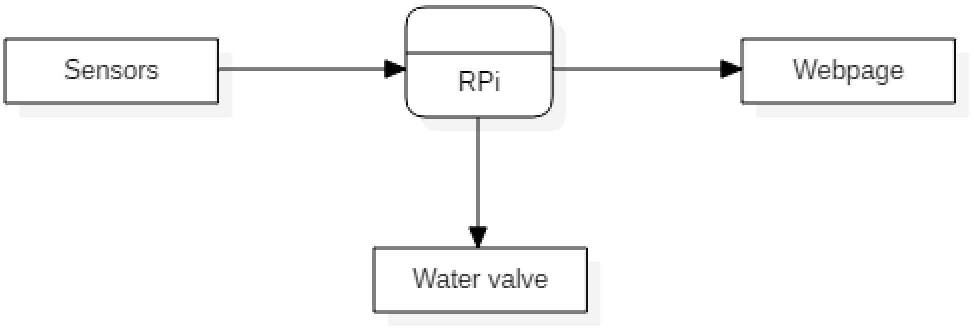


Fig. 4.1(a)

LEVEL 1

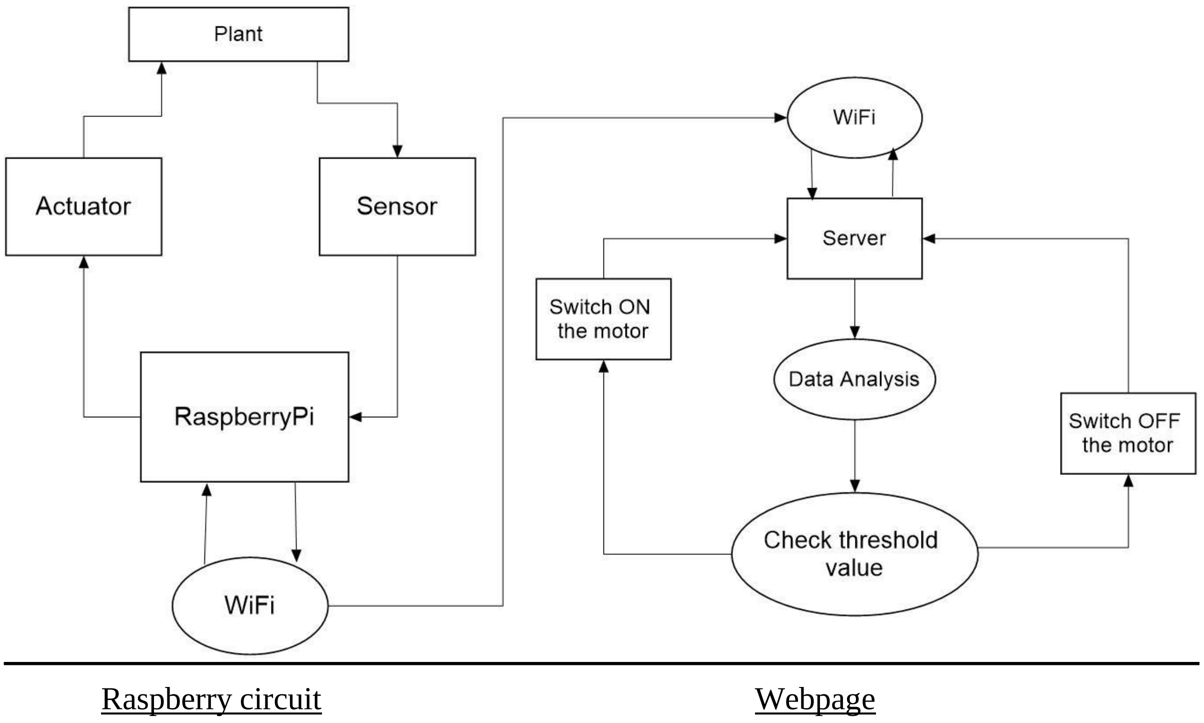

Plant
Actuator
Sensor
RaspberryPi
WiFi
WiFi
Server
Switch ON the motor
Data Analysis
Switch OFF the motor
Check threshold value
Raspberry circuit
Webpage

Fig. 4.1(b)

USE CASE DIAGRAM

A use case diagram depicts the various operations that a system performs. It contains use cases, actors, and their relationships. Use cases are the sequence of actions that form a single unit of work for an actor. An actor represents a user who is external to the system and interacts with the use case. Elements of Use Case Diagram.

Actors

An actor portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system. An actor in a use case diagram interacts with a use case. For example, for modelling a banking application, a customer entity represents an actor in the application. Similarly, the person who provides service at the counter is also an actor. But it is up to you to consider what actors make an impact on the functionality that you want to model. If an entity does not affect a certain piece of functionality that you are modelling, it makes no sense to represent it as an actor. An actor is shown as a stick figure in a use case diagram depicted outside the system boundary.

Use Cases

A use case in a use case diagram is a visual representation of distinct business functionality in a system. The key term here is business functionality. To choose a business process as a likely candidate for modelling as a use case, you need to ensure that the business process is discrete in nature. As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident. A use case is shown as an ellipse in a use case diagram.

System Boundary

A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases also need to have definitive limits defined.

A system boundary of a use case diagram defines the limits of the system. The system boundary is shown as a rectangle spanning all the use cases in the system.

Relationships: The following relationships can be established among use cases

- **Extends:** A use case may extend another. This relationship indicates that the behaviour of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label “«extend»”.

- **Includes:** A use case may include another. Include is a Directed Relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviour from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label “«include»”.

USE CASE OF SMART IRRIGATION SYSTEM:

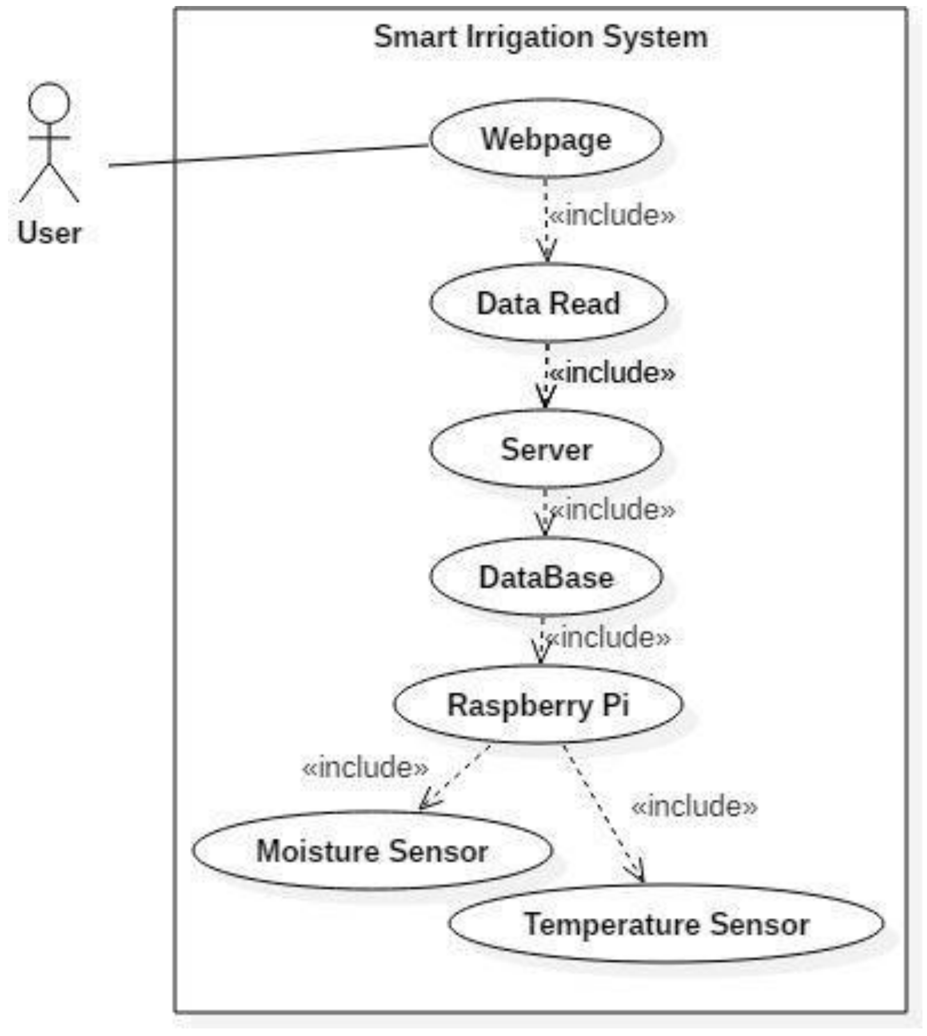


Fig. 4.2

Data dictionary

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

The data dictionary is a crucial component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary. It is a collection of descriptions of the objects or items in a data model for the benefit of programmers and others who need to refer to them. A first step in analysing a system of objects with which users interact is to identify each object and its relationship to other objects. This process is called data modelling and results in a picture of object relationships. After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms. For example, a bank or group of banks could model the data objects involved in consumer banking. They could then provide a data dictionary for a bank's programmers.

Data Dictionary for Smart Irrigation System

User

Controls website through the website.

Webpage

- Shows moisture content.
- Shows temperature value.
- Water valve control.

ACTIVITY DIAGRAM

Activities are a representation of the various operations performed by a class. An activity diagram depicts the flow of control from one activity to another. An activity diagram uses activities to model objects, classes, interfaces, components and nodes. An activity represents a set of actions such as call to a method of class, send or receive a signal, create or destroy an object, and evaluate an expression. The basic elements of activity diagram are

- ❖ Action state: Represents the state of the system in terms of actions
- ❖ Activity state: Represents an activity and therefore this state can further expand into another activity state or action state.
- ❖ Transition: Represents the control flow that performs a particular operation
- ❖ Decision: Represents the if-else or branch condition that decides the path of control flow
- ❖ Initial State: A Filled circle followed by an arrow represents the initial action state.
- ❖ Final State: An arrow pointing to a filled circle nested inside another circle represents the final action state.
- ❖ Synchronization: A synchronization bar helps illustrate parallel transition. It is also called fork and joining.
- ❖ Swim lanes: Swim lanes group related activities into one column.

ACTIVITY DIAGRAM FOR SMART IRRIGATION SYSTEM

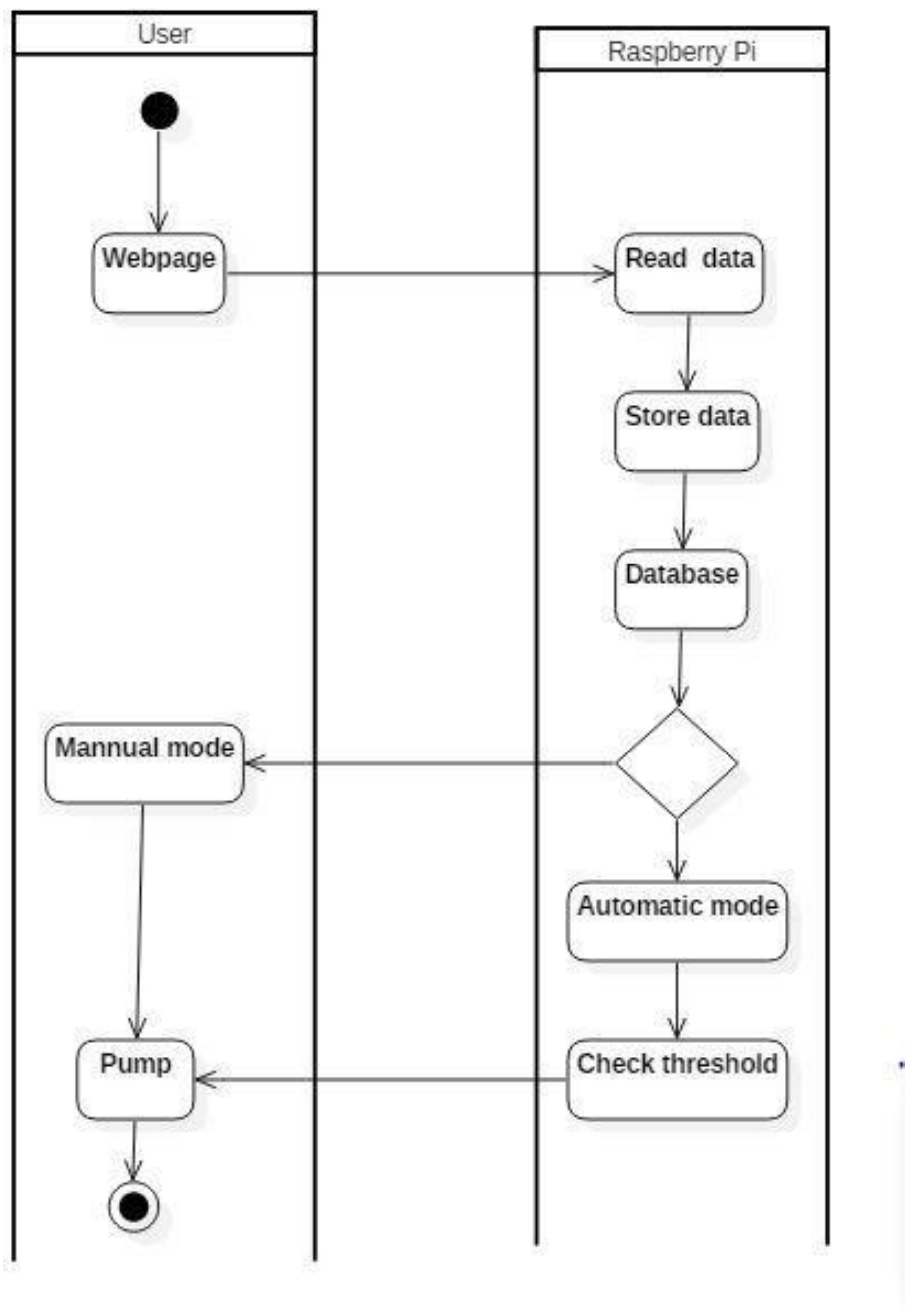


Fig. 4.3:

STATE DIAGRAM

A state chart diagram is a behaviour which specifies the sequence of states an object visits during its lifetime in response to events, together with its responses to those events.

ELEMENTS OF A STATE DIAGRAM

- **Initial State:** This shows the starting point or a first activity of the flow, denoted by a solid circle. This is also called as a “pseudo state,” where the state has no variables describing it further and no activities.
- **State:** Represents the state of object at an instant of time. In a state diagram, there will be multiple of such symbols, one for each state of the Object we are discussing. Denoted by a rectangle with rounded corners and compartments (such as a class with rounded corners to denote and Object). We will describe this symbol in detail a little later.
- **Event and Action**: A trigger that causes a transition to occur is called as an event or action. Every transition need not occur due to the occurrence of an event or action directly related to the state that transitioned from one state to another. As described above, an event/action is written above a transition that it causes.
- **Final State:** The end of the state diagram is shown by a bull’s eye symbol, also called a final state. A final state is another example of a pseudo state because it does not have any variable or action described.

STATE DIAGRAM FOR SMART IRRIGATION SYSTEM:

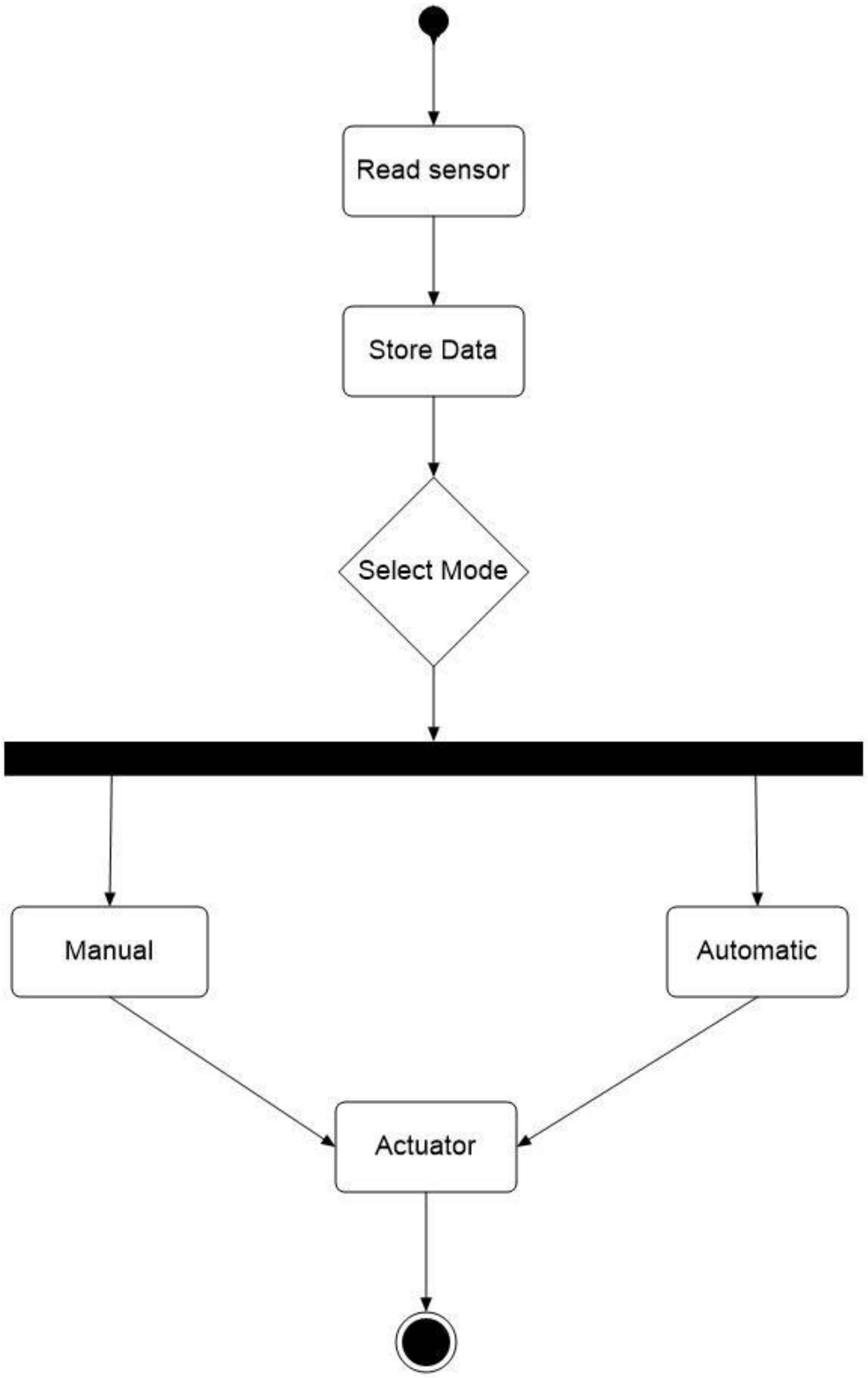


Fig. 4.4:

SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Diagram building blocks:

If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn't have to wait for a response. Asynchronous calls are present in multithreaded applications, event-driven applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message.

SEQUENCE DIAGRAM OF SMART IRRIGATION SYSTEM

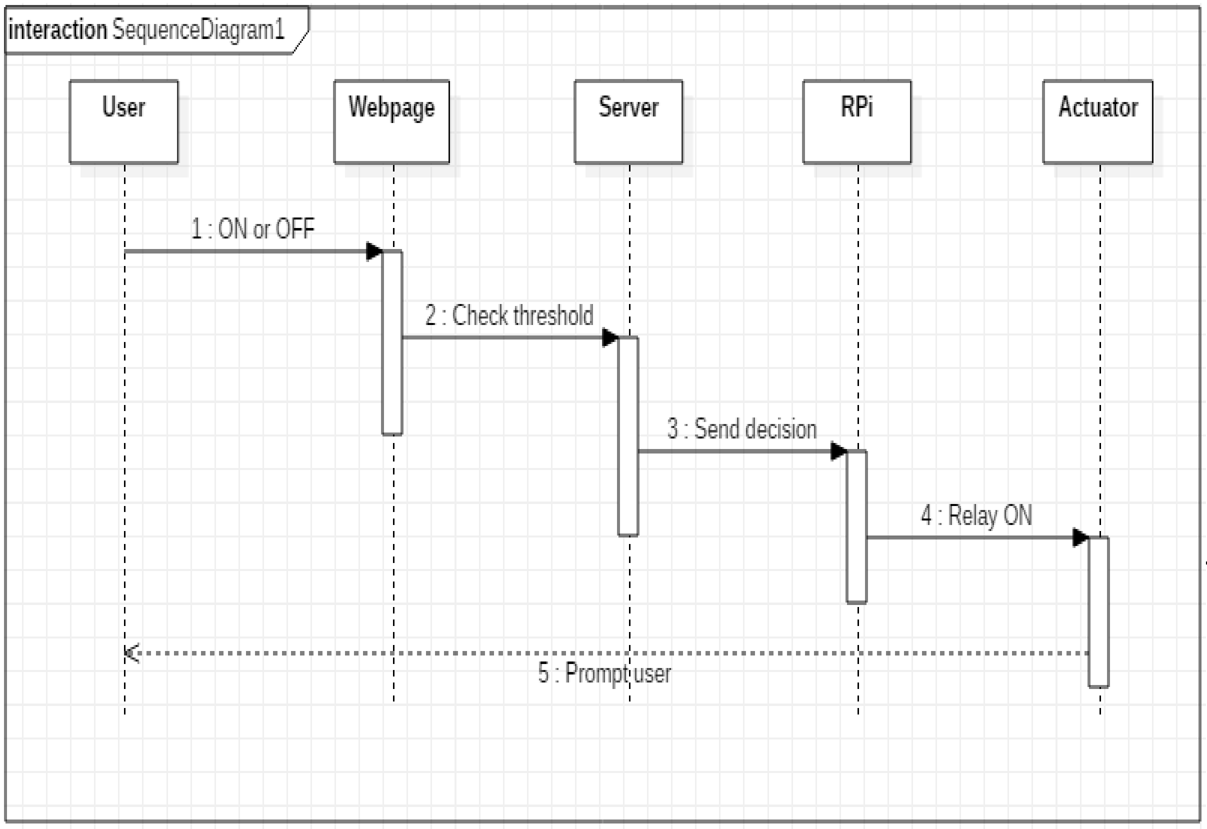


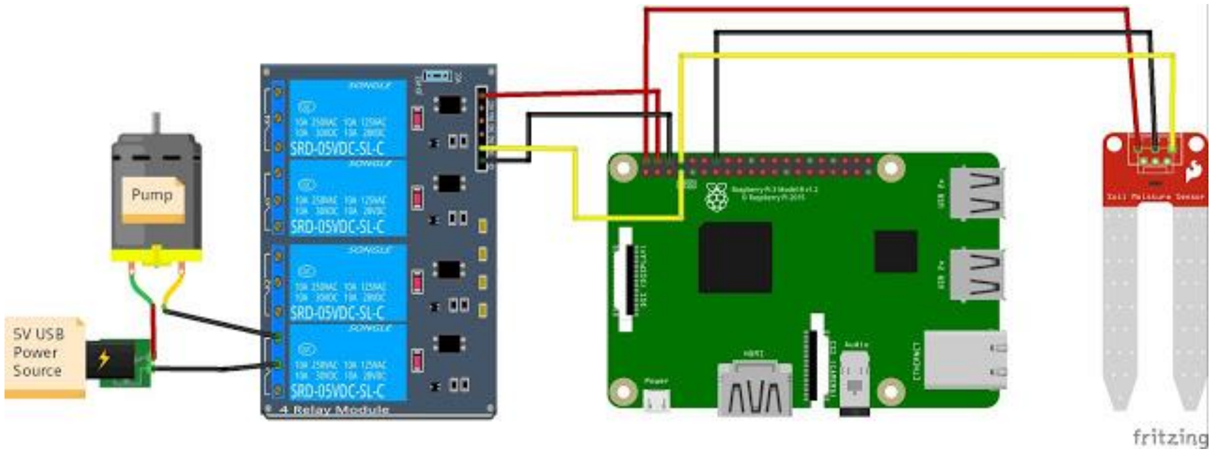
Fig. 4.5:

CHAPTER 4

IMPLEMENTATION AND TESTING

Moisture sensor Circuit

RPi Wiring:



Following this GPIO layout:

Water Sensor - plug the positive lead from the water sensor to pin 2, and the negative lead to pin 6. Plug the signal wire (yellow) to pin 8.

Relay - Plug the positive lead from pin 7 to IN1 on the Relay Board. Also connect Pin 2 to VCC, and Pin 5 to GND on the Relay board.

Pump - Connect your pump to a power source, run the black ground wire between slots B and C of relay module 1 (when the RPi sends a LOW signal of 0v to pin 1, this will close the circuit turning on the pump).

This diagram should capture the correct GPIO so long as you are using Raspberry Pi 3. Not shown is another power source to the RPi.

There are two parts to this setup. One file controls all the GPIO and circuit logic, and the other runs a local web server.

All files:

water.py

```
# External module imp
import RPi.GPIO as GPIO
import datetime
import time

init = False

GPIO.setmode(GPIO.BOARD) # Broadcom pin-numbering scheme

def get_last_watered():
    try:
        f = open("last_watered.txt", "r")
        return f.readline()
    except:
        return "NEVER!"

def get_status(pin = 8):
    GPIO.setup(pin, GPIO.IN)
    return GPIO.input(pin)

def init_output(pin):
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)
    GPIO.output(pin, GPIO.HIGH)

def auto_water(delay = 5, pump_pin = 7, water_sensor_pin = 8):
    consecutive_water_count = 0
    init_output(pump_pin)
    print("Here we go! Press CTRL+C to exit")
    try:
        while 1 and consecutive_water_count < 10:
            time.sleep(delay)
            wet = get_status(pin = water_sensor_pin) == 0
            if not wet:
                if consecutive_water_count < 5:
                    pump_on(pump_pin, 1)
                consecutive_water_count += 1
            else:
                consecutive_water_count = 0
    except KeyboardInterrupt: # If CTRL+C is pressed, exit cleanly:
        GPIO.cleanup() # cleanup all GPI

def pump_on(pump_pin = 7, delay = 1):
    init_output(pump_pin)
    f = open("last_watered.txt", "w")
    f.write("Last watered {}".format(datetime.datetime.now()))
    f.close()
    GPIO.output(pump_pin, GPIO.LOW)
    time.sleep(1)
    GPIO.output(pump_pin, GPIO.HIGH)
```

web_plants.py

```
from flask import Flask, render_template, redirect, url_for
```

```
import psutil
import datetime
import water
import os

app = Flask(__name__)

def template(title = "HELLO!", text = ""):
    now = datetime.datetime.now()
    timeString = now
    templateDate = {
        'title' : title,
        'time' : timeString,
        'text' : text
        }
    return templateDate

@app.route("/")
def hello():
    templateData = template()
    return render_template('main.html', **templateData)

@app.route("/last_watered")
def check_last_watered():
    templateData = template(text = water.get_last_watered())
    return render_template('main.html', **templateData)

@app.route("/sensor")
def action():
    status = water.get_status()
    message = ""
    if (status == 1):
        message = "Water me please!"
    else:
        message = "I'm a happy plant"

    templateData = template(text = message)
    return render_template('main.html', **templateData)

@app.route("/water")
def action2():
    water.pump_on()
    templateData = template(text = "Watered Once")
    return render_template('main.html', **templateData)

@app.route("/auto/water/<toggle>")
def auto_water(toggle):
    running = False
    if toggle == "ON":
        templateData = template(text = "Auto Watering On")
        for process in psutil.process_iter():
            try:
                if process.cmdline()[1] == 'auto_water.py':
                    templateData = template(text = "Already running")
                    running = True
            except:
                pass
        if not running:
            os.system("python3.5 auto_water.py&")
    else:
        templateData = template(text = "Auto Watering Off")
        os.system("pkill -f water.py")
    return render_template('main.html', **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

auto_water.py

```
import water

if __name__ == "__index__":
    water.auto_water()
```

main.html

```
<html><head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
<style>
.btn {
  background-color: #ddd;
  border: none;
  color: black;
  padding: 16px 32px;
  text-align: center;
  font-size: 16px;
  margin: 4px 2px;
  transition: 0.3s;
}

.btn:hover {
  background-color: #3e8e41;
  color: white;
}
.container {
  position: relative;
}

.topright {
  position: absolute;
  top: 8px;
  right: 16px;
  font-size: 18px;
}
</style>
</head>
<body>

<div class="container">

  <a href="https://www.w3schools.com" target="_blank"><div class="topright">How to
use?</div></a>
</div>

<h1><center><p><font size="100" color="#009933">PLANT CARE</font></p></center></h1>

  <h3><center>The date and time on the server is: 2019-04-20 00:57:52.335201</center></h3>

  <a href="/sensor"></a><center><a href="/sensor"><button class="btn">Check Soil
```

```
Status</button></a>
  <a href="/last_watered"><button class="btn">Check Time Last Watered</button></a>
  <a href="/water"><button class="btn">Water Once</button></a>
  <h3>Auto Watering:<center>
  <a href="/auto/water/ON"><button class="btn">Turn ON</button></a>
  <a href="/auto/water/OFF"><button class="btn">Turn OFF</button></a><br>
  <button class="alert alert-info"><center>
          Already running
</center></button>
</center></h3>
<iframe width="450" height="250" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/746903/charts/1?bgcolor=%23ffffff&amp;color=%23d62020&am
p;dynamic=true&amp;results=60&amp;type=line&amp;update=15"><center></iframe>

<iframe width="450" height="250" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/746903/charts/3?bgcolor=%23ffffff&amp;color=%23d62020&am
p;dynamic=true&amp;results=60&amp;type=line&amp;update=15"><center></iframe>

</center></body></html>
```

GPIO Script

Let's start with the code for controlling the GPIO. This requires the RPi.GPIO python library which can be installed on your Raspberry Pi as follows:

```
$> python3.5 -m pip install RPi.GPIO
```

With that installed, you should be able to use the water.py script.

You can test this is working correctly by running an interactive python session as follows:

```
$> python3.5
>>> import water
>>> water.get_status()
>>> water.pump_on()
```

This should print a statement about whether your sensor is wet or dry (get_status()), and also turn on the pump for 1s. If these work as expected, you're in good shape.

At this point you can also calibrate your water sensor. If your plant status is incorrect, try turning the small screw (potentiometer) on the sensor while it is in moist soil until the 2nd light comes on.

Flask Webserver

The next aspect of this project is to setup the web server. This code can be found here in a file called web_plants.py. This python script runs a web server enabling various actions from the script described above.

You will need to keep web_plants.py in the same directory as water.py described above, as well as auto_water.py. You will also need a sub-directory called "templates" containing the html file here called main.html.

You will need to install flask, and psutil as follows:

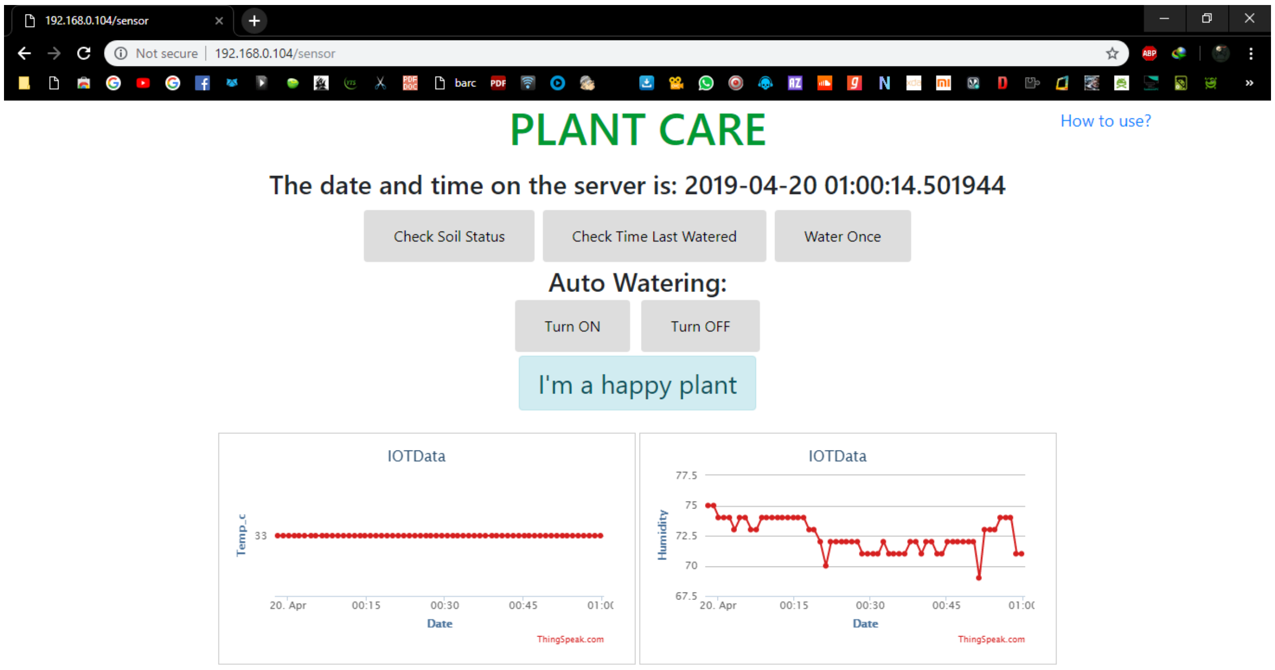
```
$> python3.5 -m pip install flask
$> python3.5 -m pip install psutil
```

You will also need to create a sub-directory called templates, and place main.html in the templates directory.

Now run the following command command to start your web server:

```
$> sudo python3.5 web_plants.py
```

Now if you navigate to the ip address of your RPi, you should see a web dashboard something like this:



192.168.0.104/sensor

Not secure | 192.168.0.104/sensor

PLANT CARE

How to use?

The date and time on the server is: 2019-04-20 01:00:14.501944

Check Soil Status | Check Time Last Watered | Water Once

Auto Watering:

Turn ON | Turn OFF

I'm a happy plant

Try clicking the buttons to make sure everything works as expected! If so, you're off to the races. here's another great tutorial I followed on flask + GPIO

Run Website Automatically

Finally, you probably want the website to auto start when the RPi gets turned on. This can be done using a tool called cronjob, which registers your website as a startup command.

To do so, type:

```
$> sudo crontab -e
```

This will bring up a text editor. Add a single line that reads (and make sure to leave one empty line below):

```
@reboot cd <your path to web_plants>; sudo python3.4 web_plants.py
```

Now when you reboot your pi, it should auto start the server.

Temperature and humidity

Connecting Sensor DHT11 to Raspberry PI

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

DHT11 can have three or four pins. If it is a 4 pin version, then a pin is unused[NC]. PIN 1 is VCC, PIN 2 is data, if you have 4 PIN version then PIN3 is unused[NC], PIN4 is ground. [In case of 3 PIN version PIN 3 is ground]

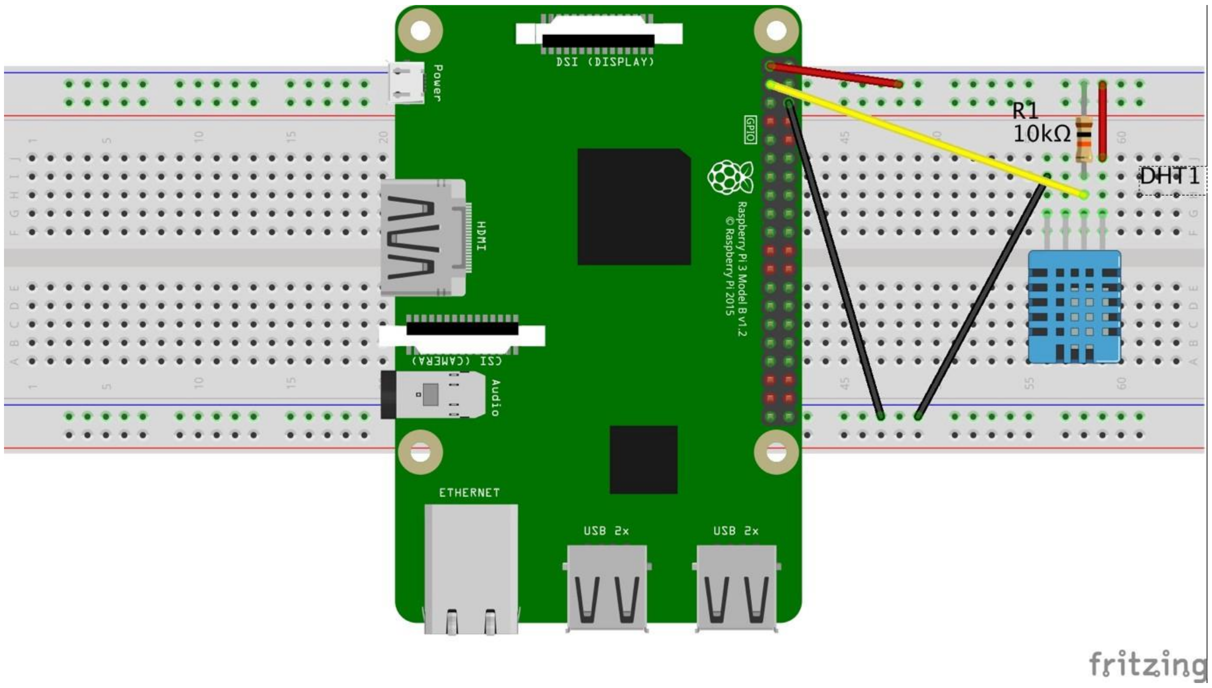
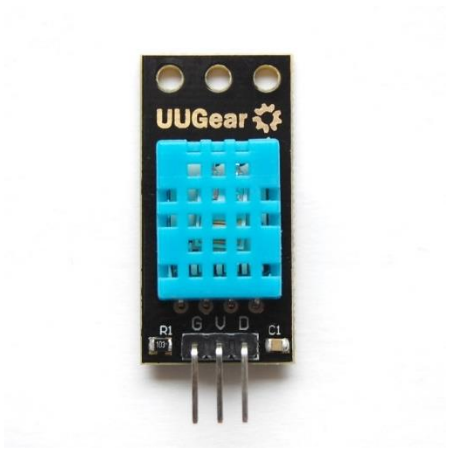


Diagram 2—Connection Diagram

Note—I used 10K 0hm resistor as shown in the Diagram 2—Connection diagram above but it didn't worked for me and once I removed the resistor my sensor started reporting data. Some tutorials will include instructions for using resistor but it didn't worked for me.

Step 1—Insert DHT11 in the breadboard



DHT11 Sensor

1. VCC connected to +3.3V~5V
2. DATA connected to the microcontroller IO port
3. GND connected to ground

Step 2—Now make below connections using Jumper Wire

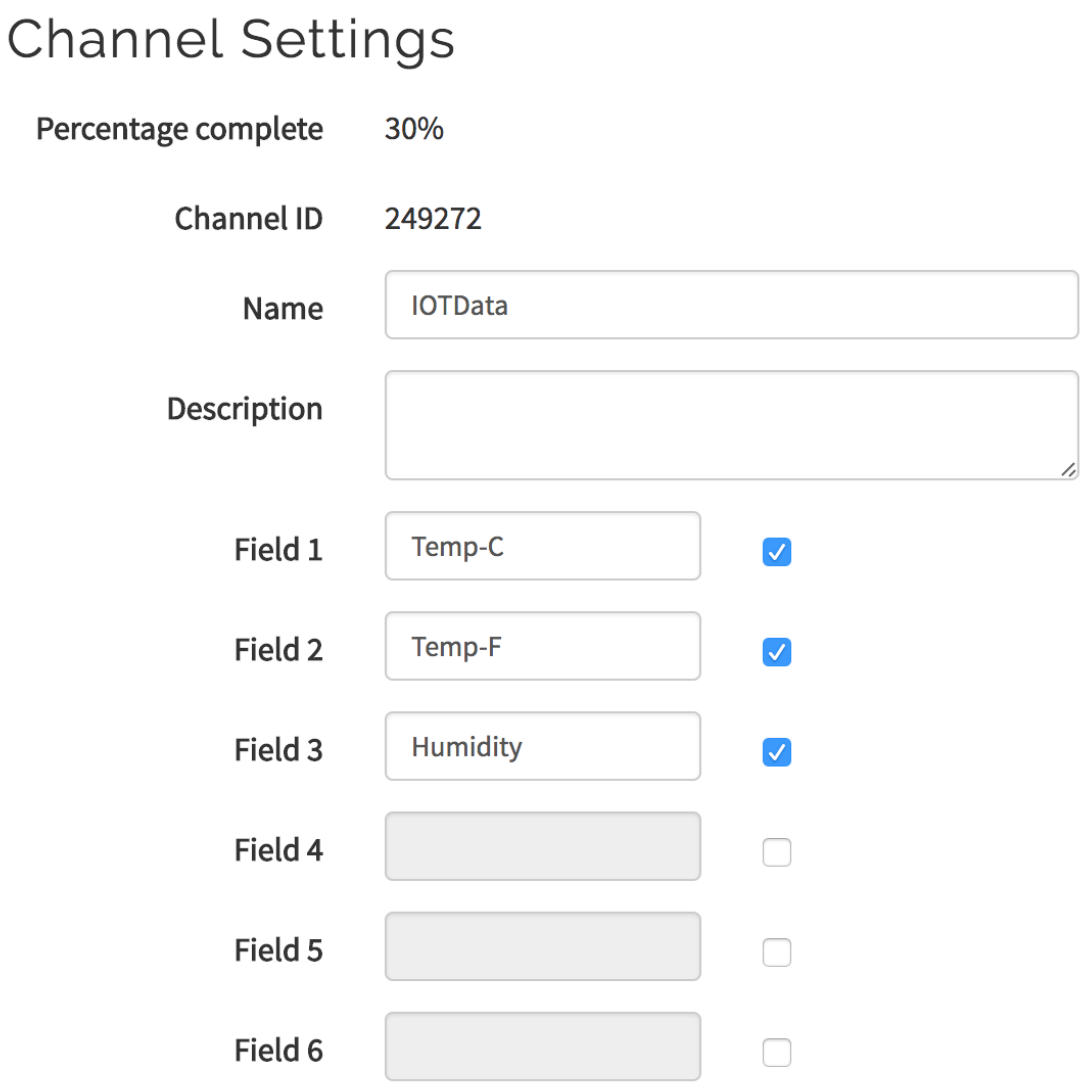
1. Connect 3.3v P1 pin from Raspberry Pi to VCC (V) of DHT11 using Jumper wire.
2. Connect Pi ground P6(Black dot from Diagram 1 above) to PIN 3 which is GND pin in DHT11 (Or PIN4 if that exists)
3. Connect Pi GPIO2 P3 to DHT11 Data pin

Server Side Setup

Now lets setup server side to receive the data. We will use ThingSpeak as it provides server side infrastructure to collect the data and process it but you can use any other service as well.

1. Create a free account on ThingSpeak.

2. Create a channel as shown below with 3 fields for Temp-C (in celsius), Temp-F (in Fahrenheit), Humidity



Channel Settings

Percentage complete	30%	
Channel ID	249272	
Name	IOTData	
Description		
Field 1	Temp-C	✓
Field 2	Temp-F	✓
Field 3	Humidity	✓
Field 4		☐
Field 5		☐
Field 6		☐

3. Once you save the Channel, go to API Key section of channel & note the Write API key. This key will be needed to send data to ThingSpeak channel from device.

Making Raspberry Pi ready to read data from sensor

1. If Install updates

```
sudo apt-get update
```

2. Install Python if not already there. You can check the python version by running command

```
python -V
```

3. If Python doesn’t exists. Install it using

```
sudo apt-get install python-dev
```

4. Install Rpi.GPIO (Library used to interact with GPIO pins)

```
sudo apt-get install python-rpi.gpio
```

5. Install library for ADH11 sensor—Download library

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
```

6. Install library

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-openssl
sudo python setup.py install
```

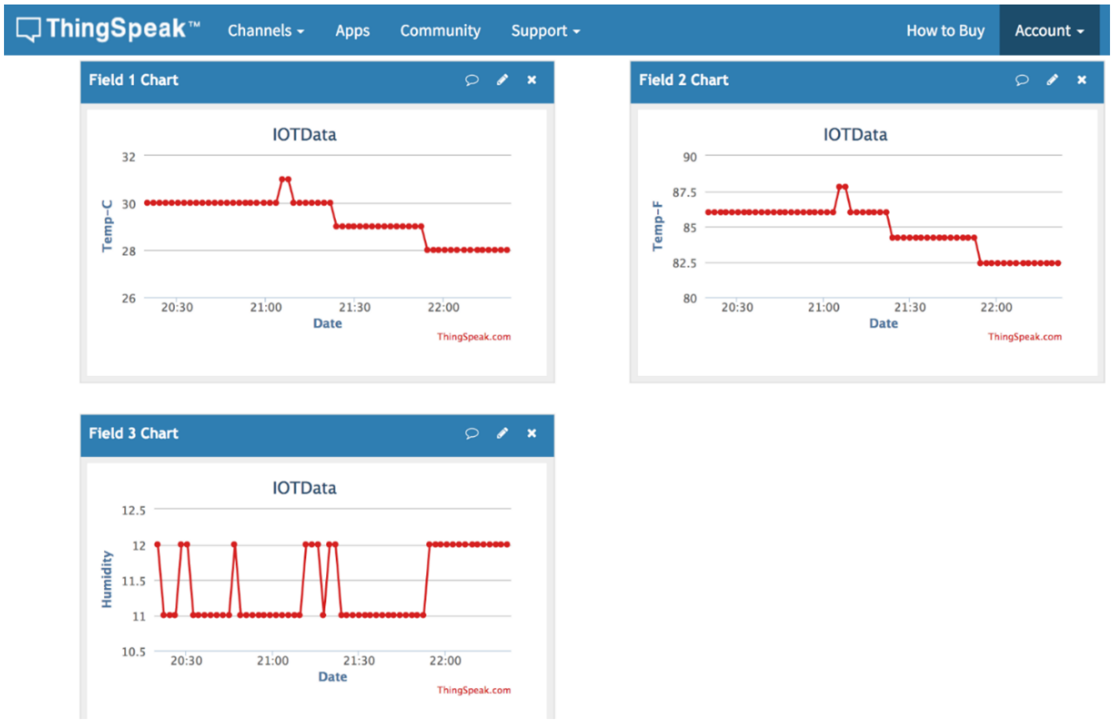
Update your API key here in the code

```
# Define GPIO pin to which DHT11 is connected
DHTpin = 2
#Setup our API and delay
myAPI = "GET_YOUR_KEY"  # API Key from thingSpeak.com
myDelay = 60 #number of seconds between posting data
```

Run Application

```
sudo python temperaturesensor.py
```

Once you run the python script, you should start seeing the data on ThingSpeak as shown in the below image.



```
@reboot cd <your path to temp.py>; sudo python3.5 temp.py
```

Now when you reboot your pi, it should auto start the server.

4.2 Testing Approach

Hardware Tests

Test ID	Hardware	Input	Output	Test	Priority
H1	Raspberry Pi B+	JSON response after sending HTTP request, sensor data from Arduino	Commands to the Arduino through USB interface, sensor data through an API call	We will test the functionality of the operating system to ensure that the program on the Raspberry Pi can process all communication between the web services and Arduino.	Critical
H2	Arduino	Commands from Raspberry Pi, environment readings from sensors, valve control statuses	Sensor data to Raspberry Pi, control signal to turn on and turn off the irrigation valves	We will test that we can send the sensor data to Raspberry Pi. We will also test that the Arduino can control all the valve switches.	Critical
H3	Soil moisture sensor	Soil moisture levels from the environment	Soil moisture reading to Arduino	We will test that the data received from the soil moisture sensors reflects the changes in the environment.	High
H4	Temperature sensor	Temperature reading from environment	Temperature reading to Arduino	We will test that the temperature reading received	High

				from the sensor is correct to the reading in the thermostat.	

Table Hardware Test Table

Unit Tests

– Soil Moisture Sensor Subsystem

Test ID	Module	Input	Output	Test	Priority
USSS1	Soil Moisture Reading Collector	Soil moisture sensor level from environment	Voltage difference to control board processed	We will test that the collector gives a voltage difference.	Moderate
USSS2	Control Board	Voltage difference from collector	Voltage level responding to the moisture reading	We will test that control board can power on and is giving out the voltage difference.	Moderate

Table Soil Moisture Sensor Subsystem Unit Test

–Temperature Sensor Subsystem

Test ID	Module	Input	Output	Test	Priority
USTS1	Temperature Reading Collector	Rain condition from environment	Voltage difference to control board processed	We will test that the collector give a voltage difference	Medium
USTS2	Control Board	Voltage difference from collector	Voltage level responding to the temperature	We will test that control board can power on and giving out the voltage difference	Medium

Table Temperature Sensor Subsystem Unit Test

– Sensor Control Subsystem

Test ID	Module	Input	Output	Test	Priority
USSC1	Analog Digital Converter	Voltage from sensors	Digital values based on the analog signal	We will test that the analog digital converter gives out a numeric value response to the analog signal.	High
USSC2	Sensor Data Package	Digital sensor data from Analog Converter	Meaningful data about environment condition	We will test that the output data is correct to the environment condition.	High
USSC3	Serial Data Sender	Communication rate of a serial connection	A serial connection is ready for transmitting data	We will test that at this point the data will be transmitted to Raspberry Pi.	High

Table Temperature Sensor Subsystem Unit Test

– Valve Control Subsystem

Test ID	Module	Input	Output	Test	Priority
UHVC 1	Serial Data Receiver	Communication rate of a serial connection	A serial connection is ready for receiving data	We will test that at this point the data can be received through a serial connection.	High
UHSC2	Command Executor	Control command being sent through serial connection	Control signals to the digital pins	We will test that this module will set the digital pins on the Arduino.	High
UHVC 3	Relay Module	Command signals from Arduino	High voltage control signal to turn the valve on or off	We will test that we can turn on or off the relay switches by applying the correct signal.	High

Table Valve Control Subsystem Unit Test

– Data Processing Subsystem

Test ID	Module	Input	Output	Test	Priority
UIVDP1	USB/Serial Interface	Communication rate of a serial connection	A serial connection is ready for sending and receiving data	We will test that at this point the data can be sent and received from a serial connection.	High
UIDP2	Valve Command Processor	Control command as an array	Correct control command to single local variables	We will test that this module assigns correct values to local variables based on an input array.	High
UIDP3	JSON Message Builder	Sensor data received through a serial connection	A JSON message to be sent to a web service	We will test that the sensor data is built into the correct JSON format.	High

Table Data Processing Subsystem Unit Test

– Service Caller Subsystem

Test ID	Module	Input	Output	Test	Priority
UISC1	Response Parser	API response from the Server Layer	An array in JSON format	We will test this module will receive a response and parse it into a JSON format after a HTTP request is returned.	High

UISC2	API Caller	JSON message to be sent to web services	An URI format data being sent to web service	We will test that this module will send a HTTP request in a correct URI format to a web service to be processed.	High

Table Service Caller Subsystem Unit Test

– Web Application Subsystem

Test ID	Module	Input	Output	Test	Priority
USWA 1	View	User interaction with the web application	User event object specifying the events the user set off	We will test all page inputs including correct and incorrect information and verify the system outputs the correct output.	High
USWA 2	Controller	Event object from a user interaction	Query on database system based on event object	We will test that this module will be able to handle the data transfer from database and UI.	High
USWA 3	Module	Controller request	A new view model or a model that represents existing data	We will test with different kinds of view models to check against run time errors and null reference exceptions.	High

Table Web Application Subsystem Unit Test

– Web Services Subsystem

Test ID	Module	Input	Output	Test	Priority
USWS1	Response Parser	Command data from data base	A response to send back to the Interface Layer based on the database query and received data.	We will test by sending a HTTP request, and then we verify the change on the database, also to check the response to the Interface layer.	High
USWS2	Web Services	URI request from API caller	A response that contains control commands	We will test by sending a HTTP request with different values in the fields of the JSON body, then we verify the response.	High
USWS3	URI Authenticator	HTTP request	A confirm whether the URI is authenticated	We will test by sending a correct URI and an invalid UIR to see how each is handled.	High
USWS4	JSON Converter	HTTP request	Extracted sensor data from JSON body message	We will test by sending a HTTP request with a JSON body, then verify the	High

				information saved on database.	

Table Web Service Subsystem Unit Test

Component Tests

– Sensor Layer

Test ID	Subsystem	Input	Output	Test	Priority
CSS1	Soil Moisture Sensor Subsystem	Soil moisture levels from the environment	Soil moisture reading to the Sensor Controller subsystem	We will test that the data received from the soil moisture sensors reflects the changes in the environment.	High
CSR1	Rain Sensor Subsystem	Rain condition from the environment	Rain condition reading to the Sensor Controller subsystem	We will test that the data received from the rain sensor reflecting the changes in the environment.	High
CST1	Temperature Sensor Subsystem	Temperature reading from the environment	Temperature reading to the Sensor Controller subsystem	We will test that the temperature reading received from the sensor is correct to the reading on the thermostat.	High

Table Sensor Layer Component Tests

– Hardware I/O Layer

Test ID	Subsystem	Input	Output	Test	Priority
CHS1	Sensor Controller Subsystem	Data from environment conditions	Sensor data as electric signals	We will test this subsystem is able to interface and collect data from all sensors.	High
CHV1	Valve Controller Subsystem	Control Command	Digital signal to turn the valves on or off	We will send some command control through USB connect to this subsystem in order to turn on and turn off the valves.	High

Table Hardware I/O Layer Component Tests

Server Layer

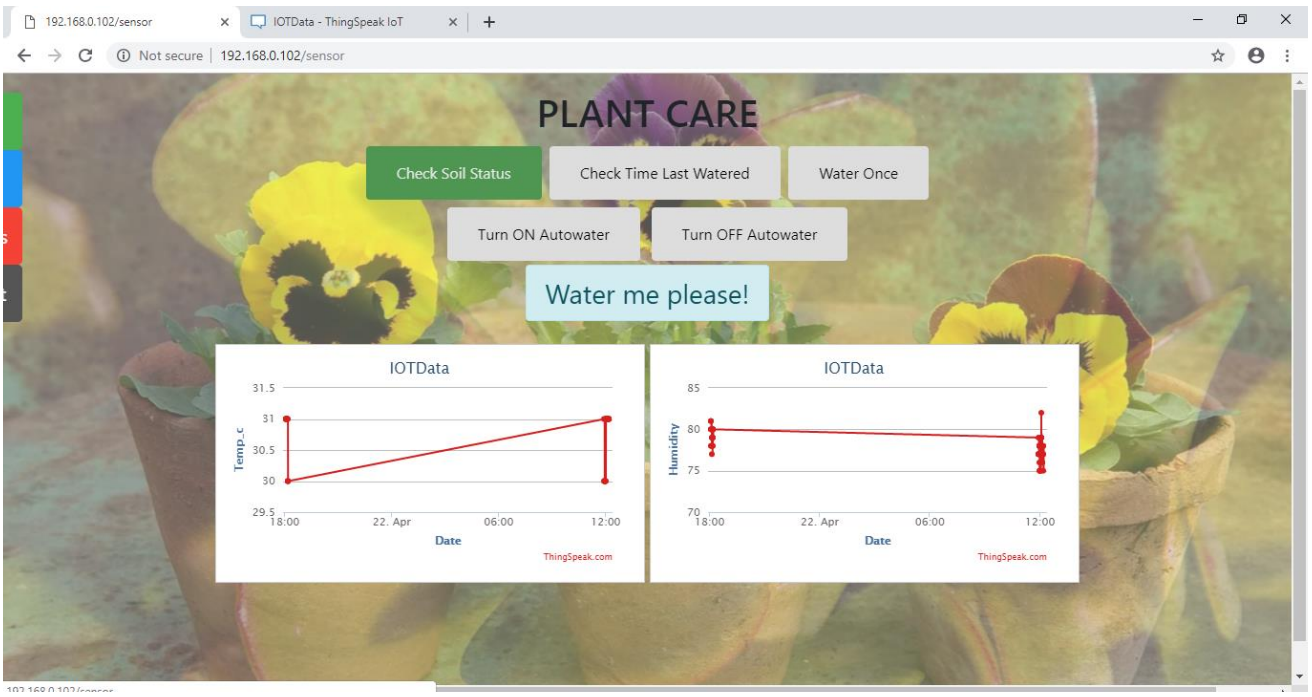
Test ID	Subsystem	Input	Output	Test	Priority
CSA1	Web Application Subsystem	User interaction with the web application	User event object specifying the events the user set off	We will test all pages inputs including correct and incorrect information and verify the system outputs the correct output.	High
CSS1	Web Service Subsystem	URI request from API caller	A response to send back to the Interface Layer based on the database query and received data	We will test by sending a HTTP request, and then we verify the change on database, also to check the response to the Interface Layer.	High

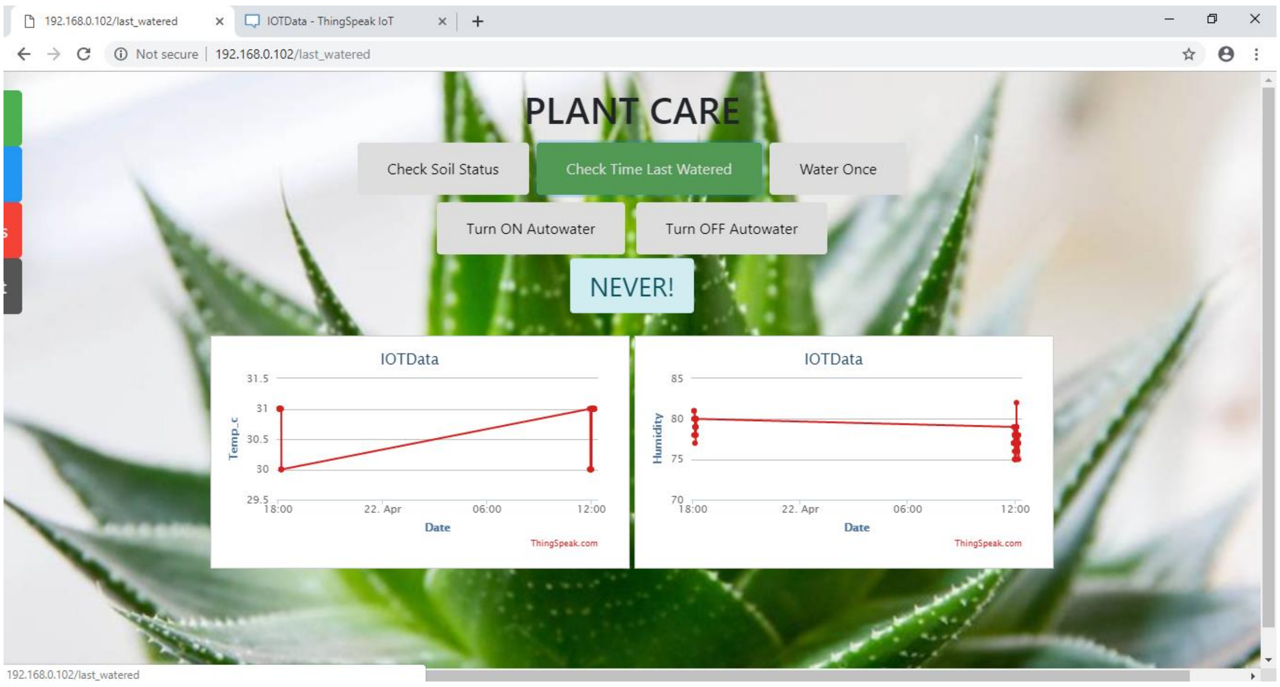
Table Server Layer Component Tests

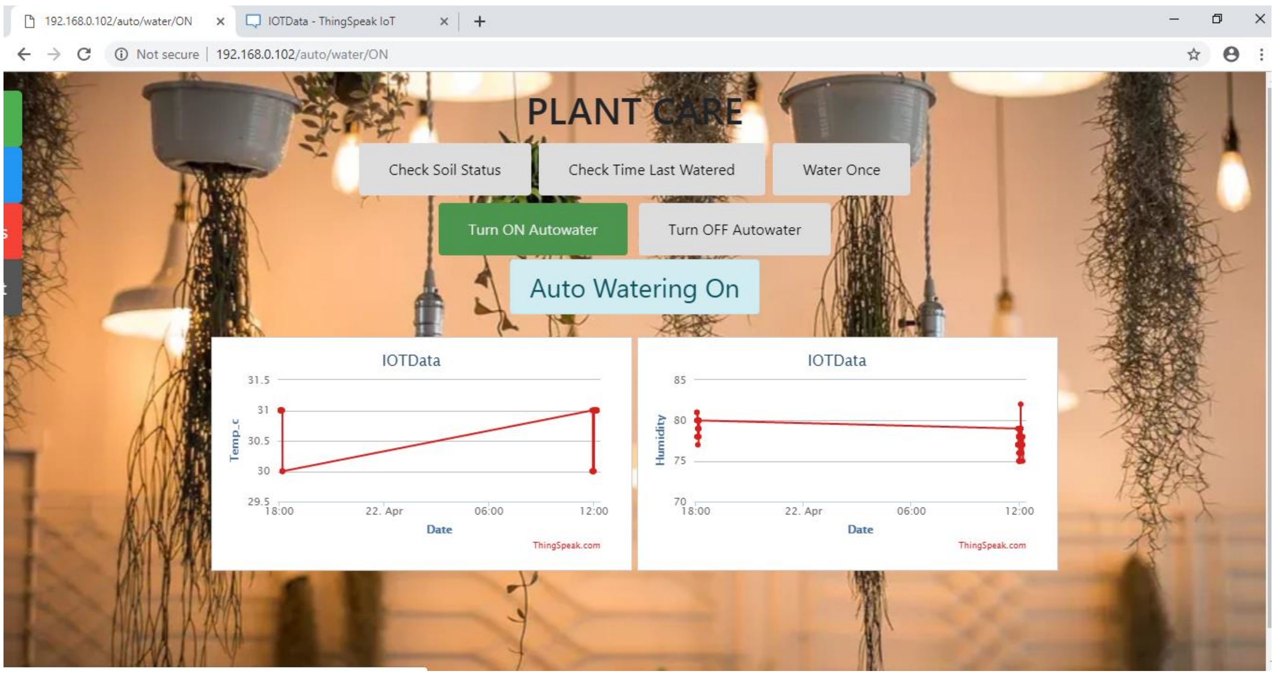
Integration testing

Test ID	Layer	Input	Output	Test	Priority
IS1	Sensor Layer	Changes in environment conditions	Electric signal on the pins of Raspberry pi	We will test by changing the environment conditions by spilling water on the header of the rain sensor, sinking the header of the soil sensor into water, and pull the temperature of thermal Sensor at the same time. Then, we will verify the change of the values of The sensor readings.	High
IH1	Hardware I/O Layer	Control command from Raspberry Pi	Operation of the valves and reading values of sensors	We will use Raspberry Pi to send a command to turn the valves on or off, then read sensor values	High

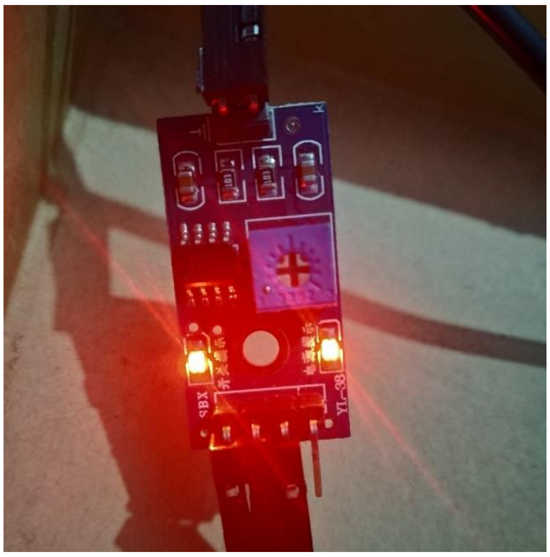
CHAPTER 5
RESULTS AND DISCUSSIONS



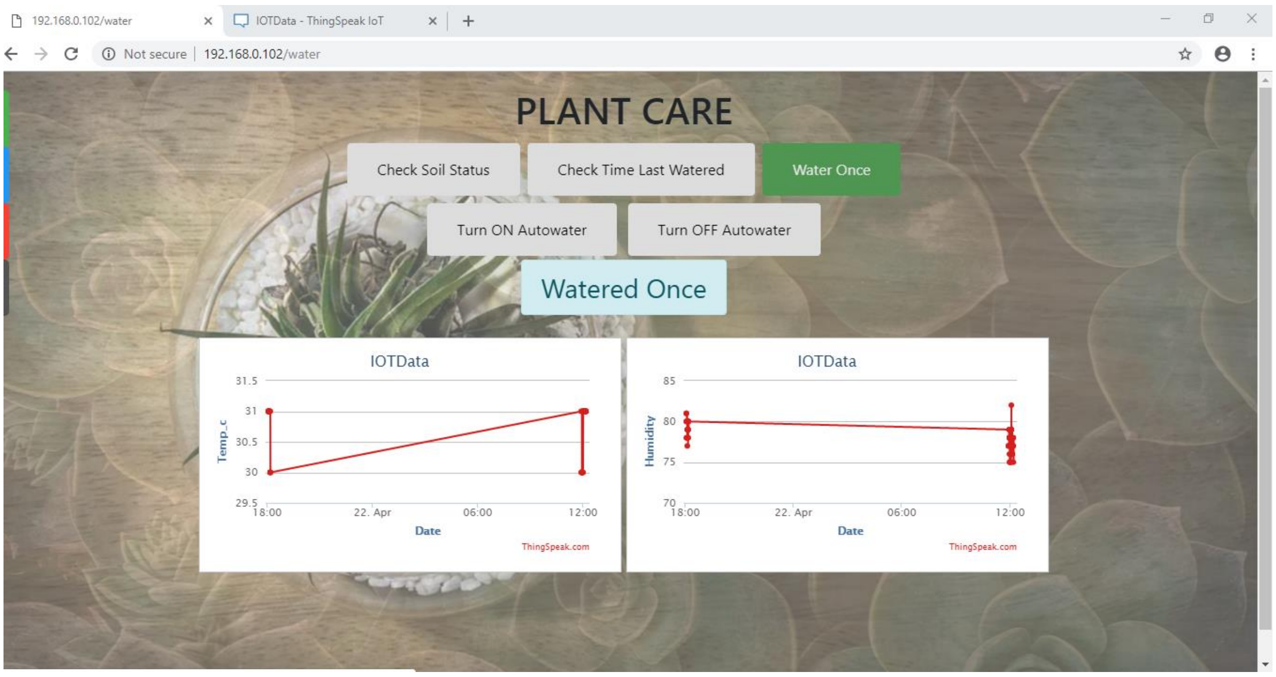


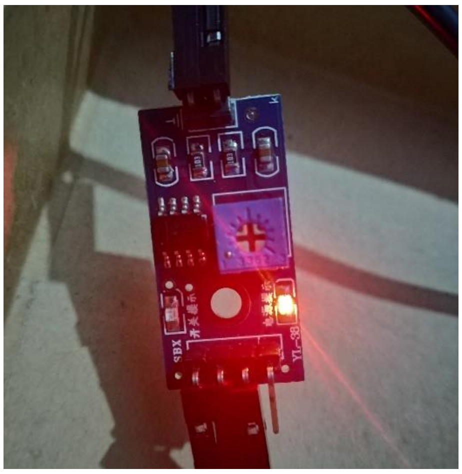


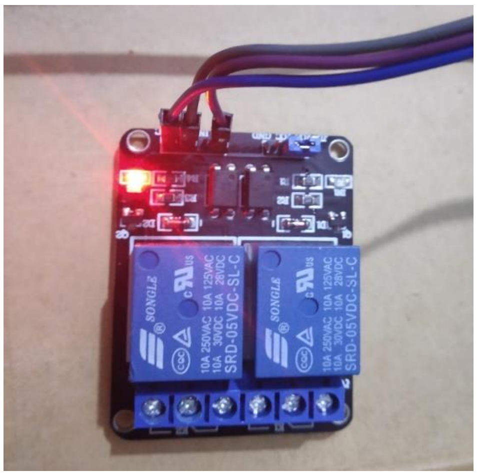
When the auto watering is ON and moisture is detected (i.e. both lights on moisture sensors turn ON)



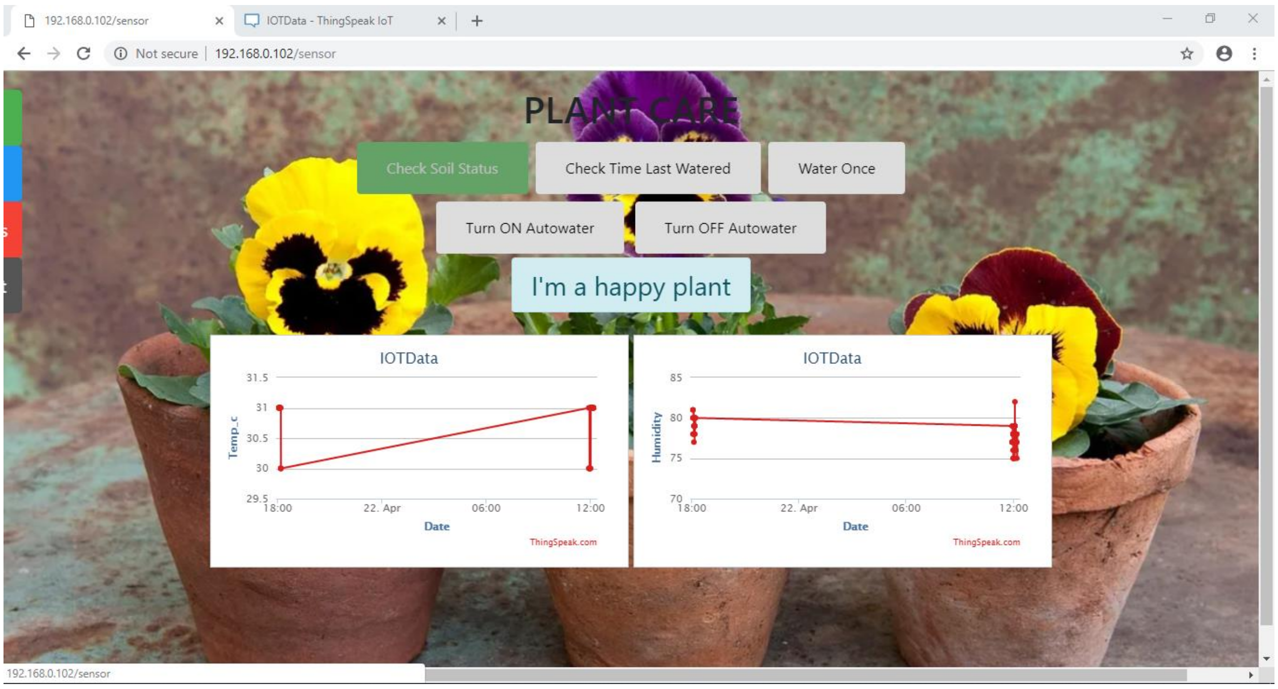
the relay remains off and so no water is supplied.







When only one light of the moisture sensor is ON it indicates that there is no moisture content in the soil, so the relay turns ON the pump and starts watering so that the plant gets watered.



CHAPTER 6

CONCLUSION AND FUTURE WORK

Conclusion

- Using this system, one can save manpower, water to improve production and ultimately increase profit.
- The smart plant system is feasible and cost effective for optimizing water resources for agricultural production.
- The system would provide feedback control system which will monitor and control all the activities of irrigation system efficiently.

Future Work

We can interface LCD screen in order to display the current status of the soil moisture content levels, percentage of water utilized to water the plant, duration of time for which the water pump is ON, etc. To improve the efficiency and effectiveness of the system, the following recommendations can be put into consideration. Option of controlling the water pump can be given to the user. The user may choose to stop the growth of plant or the plant may get damaged due to adverse weather conditions. In such cases user may need to stop the system remotely. The idea of using IOT for irrigation can be extended further to other activities in farming such as cattle management, fire detection and climate control. This would minimize human intervention in farming activities.

CHAPTER 7 REFERENCE

- https://medium.com/@mr_prateekjain/my-first-iot-internet-of-things-project-42acf93fb03d
- https://www.elecfreaks.com/store/blog/post/how-to-send-temperature-threshold-value-alarm-email-via-ifttt.html/
- www.w3school.com
- www.youtube.com