A

Final year project report

On

“Smart Water Tank Management System”

IN FULFILLMENT OF DIPLOMA IN COMPUTER ENGINEERING OF MSBTE MUMBAI

Submitted by:

Bhavya M. Oswal	(3317)
Prathamesh P. Jadhav	(3318)
Omkar G. Kajarekar	(3319)
Atharva M. Joshi	(3320)

Program Name: Computer Engineering

Department of Computer Engineering

Shree Swami Vivekanand Shikshan Sanstha’s

Dr. BAPUJI SALUNKHE INSTITUTE OF ENGINNERING AND TECHNOLOGY

KOLHAPUR

Academic year 2019-2020

Acknowledgments:

It is our privilege to express our sincerest regards to our caption project guide Mr.A.R.Sawant for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our caption project co-coordinator Mr.P.K.Shinde sir for encouraging and valuable suggestions for our caption project report.

We are extremely grateful to entire faculty of the Computer Department for their valuable suggestion. Our sincere thanks for providing invaluable and uninterrupted Computing facility support in computer labs. Our gratitude to HOD of Department as well as all the staff members who helped in successful completion of this caption project.

Abstract:

Water is one of the essential parts of life. Wastage of Water is one of the big problems to the world. In order to ensure the safe supply of the drinking and useful water for different purposes like agricultural, the water should be monitored. This report presents a design of a low cost system for real time monitoring of the water quality and quantity of water in IOT (internet of things). The system having of several sensors is used to measuring physical level of the water. The parameters flow sensor of the water can be measured. The Arduino model can be used as a controller. Finally, the sensor data can be shown on internet using WI-FI system. This data can be used in future research and development.

Content Page

Sr. no.	Name of Topic	Page no.
1	Introduction	1
2	Literature Survey	2
3	Scope and problem definition	3
4	Detailed Methodology	5
5	Details of Design, Working, And Processes	14
6	Results and Applications	24
7	Future Scope and Conclusion	30
8	Appendix	32
9	References and Bibliography	59

1.0 Introduction:

There are so many problems related to water which affects the overall the percentage of water present. Among all problems overflow of overhead tanks is most observed problem. So, there is a need for continuous monitoring, water supply scheduling, and proper distribution. Currently drinking water is very prized for all the humans. In recent times water levels are very low and water in the lakes is going down. So it's too important to find the solution for water monitoring & control system. There is a need for continuous real time remote monitoring of water quality parameters within the water system as the concentrations of the pollutants lead to serious health consequences.

The system is focusing on one of the problems i.e. overflow of a water tank where a lot of water gets wasted and fulfills the need of low-cost system for real-time monitoring of water using the Internet of things (IoT) platform for minimizing the overflow of tanks. IoT is a solution. In recent days, development in computing and electronics technologies has triggered Internet of Things technology. Internet of Things can be described as the network of electronics devices communicating among them by the help of a controller. The IoT is a collection of devices that work together in order to serve human tasks in an efficient manner.

This report presents a low cost water monitoring system, which is a solution for the water wastage and water quality. Microcontroller and different sensors are used for the system. Ultrasonic Sensor is used to measuring water level. The other parameters like pH, flow of water and turbidity of the water can be calculated using different corresponding sensors.

2.0 Literature Survey:

Chinta Rohith Reddy, Saransh Shrivastava (2018) [1] Water level measurement data is very important in some water-related fields. An automatic water level measurement system is needed to prevent the difficulties when one does the measurement manually. In this paper, a prototype water monitoring system using IoT is presented. For this some sensors are used. The collected data from the all the sensors are used for analysis purpose for better solution of water problems. The data is send to the cloud server via Wi-Fi module ESP8266. So this application will be the best challenger in real time monitoring & control system and use to solve all the water related problems.

M. Ramakrishna, G. Dayanandam (2018) [2] In this paper, a prototype water monitoring system using IoT is presented. For this some sensors are used. The collected data from the all the sensors are used for analysis purpose for better solution of water problems. The data is sent to the cloud server via Wi-Fi module ESP8266. Only the data about water flow and water quantity is displayed.

K. Ramyadevi, M. Ramya,(2018) [3] the paper presents an IOT device which help to manage and plan the usage of water including the quality of water. This system can be easily installed in residential societies. Sensors placed in the tank which continuously informs the water level at the current time. This information will be updated on the website and using this website, user can visualize the water level on a Smartphone anywhere that is connected to Internet. According to the level of water in the tank the motor functioning will be automatically controlled, at low level of water motor will automatically turn on and when tank is about to fill up it will cut off.

3.0 Scope of project:

3.1 Problem Statement:

Some of the automated water levels monitoring systems are already present, but most of the methods have so far some shortcomings in practice. The proposed system will try to overcome these problems and implement an efficient automated water level monitoring and controlling. The system will focus mainly on the following problems of existing system.

1 Calculating only water quantity – To overcome this problem proposed system will calculate the water quality information with water quantity information with the help of sensors. The water quality information will be measured with the Ph sensor, turbidity sensor and flow sensor.

2 Power failure while starting motor – This problem will be resolved with an alert messages to the system owner as there is no power supply to turn on the motor. In any case of power failure system will send appropriate alert messages to the system owner.

3 No water supply from incoming water lines – This problem will be detected by flow sensor as there will be no flow of water through supplying lines and in such case an alert message about water shortage will be sent to system owner.

Problems in the existing system

1 According to [1] what happens if there is no network coverage in the installation area?
2 According to [2] we can calculate only the level of water in the tank. No signs of discussion about the quality of water.
3 According to [3] what happens there is no power to turn on motor.
4 And none of the above references have addressed what happens during the water filling, if the incoming water lines have no water supply?

3.2 Objectives:

The main purpose of the system is to reduce the amount of overflowing of water. The objective of the project is to control the water motor manually as well as automatically. This system can be used in buildings as well as in industries where the amount of overflowing of water is very high and can be reducible. This system can also be useful to measure the current level of water, the pH value of water, as well as turbidity and speed of falling water. And at the last this system will generate the weekly and monthly report about the usage of water.

3.3 Scope:

- This system can be used where water motor needs to start manually. Using this system there is no need to start motor manually. The motor will be start and stop automatically when water level in water tank reaches at certain level by using automatic option.
- This system is very useful in buildings, industries, where a lot of amount of overflowing water can be reducible.
- The system is very useful where there is need of real time water level monitoring to avoid the overflow of water.

1. To measure the usage of water
2. To check quality of water
3. Access the motor automatically as well as manually
4. Alert messaging system for low water level and overflow of water in tank
5. Generating the weekly analysis of water usage and quality of water
6. Real time water level monitoring through application
7. Displaying the water flow speed

4.0 Methodology:

4.1Requirement Analysis:

Hardware Requirement:

1. Arduino Uno R3-

   Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

2. Wi-Fi module ESP8266-

   The ESP8266 Wi-Fi module is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Wi-Fi significantly expand on the geographic range of your phone and will be able to have in controlling the Arduino.

3. NodeMCU-

   Integrated support for Wi-Fi network.

4. Ultrasonic sensor-

   To measure the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal.

5. Relay-

   A relay is used to turn on and turn off a circuit by a low power signal. Circuits that operate at high voltages or at high currents cannot be controlled directly by an Arduino. Instead, you use a low-voltage control signal from the Arduino to control a relay, which is capable of handling and switching high-voltage or high-power circuits.

6. Turbidity Sensor-

   Turbidity sensors measure the amount of light that is scattered by the suspended solids in water.

7. Flow Sensor-

   Flow sensors are devices which are used to measure a flow rate of Fluid.

Software Requirement-

1. Arduino IDE-

   The Arduino integrated development environment (IDE) is a cross platform application (For Windows, Mac OS, and Linux) that is written in the programming language Java. It is used to write and upload program to Arduino board.

   The source code for IDE is released under the GNU (General Public License) version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the writing project, which provides many common input and output procedures.

2. ThingSpeak-

   ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network.

3. MIT App Inventor-

   MIT App Inventor is a web application integrated development environment. MIT App Inventor is an intuitive, visual programming environment that allows everyone to build fully functional apps for smartphones and tablets.

4.2 Architecture Diagram:

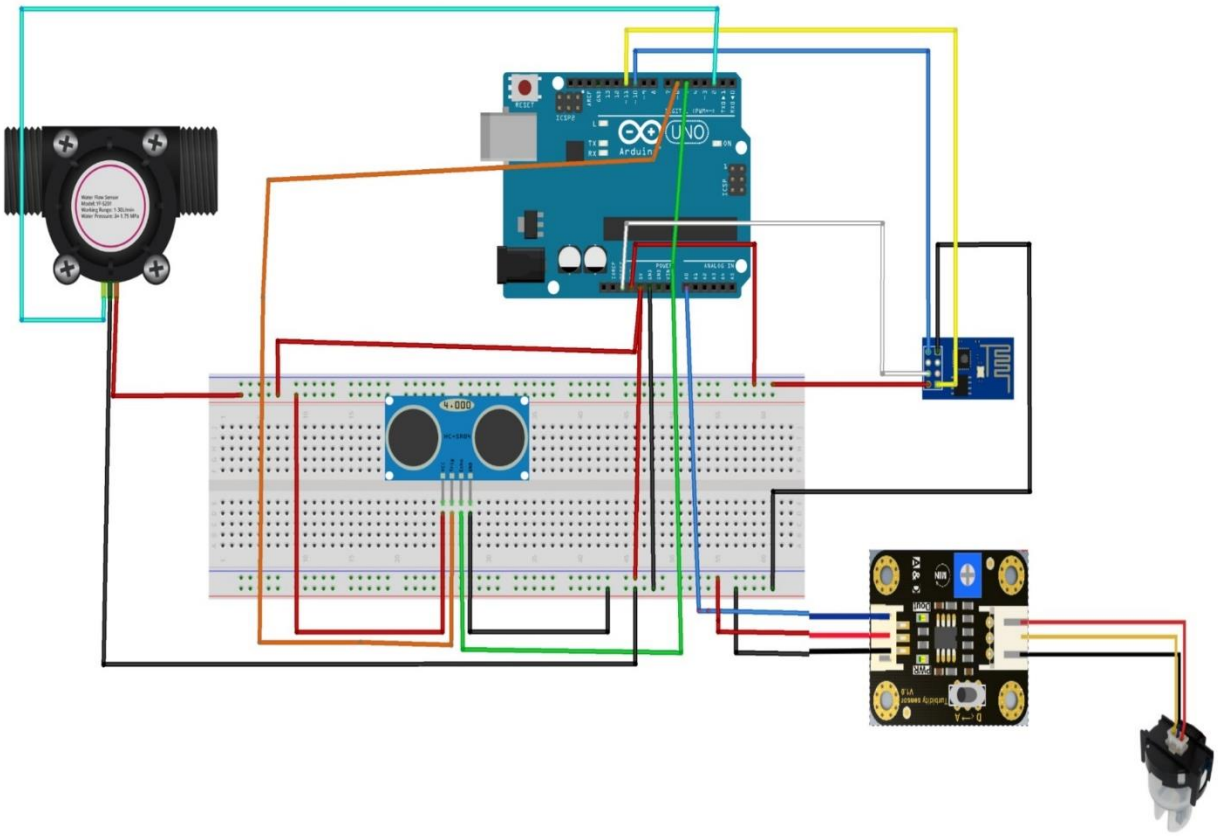

Fig. Circuit diagram of Arduino

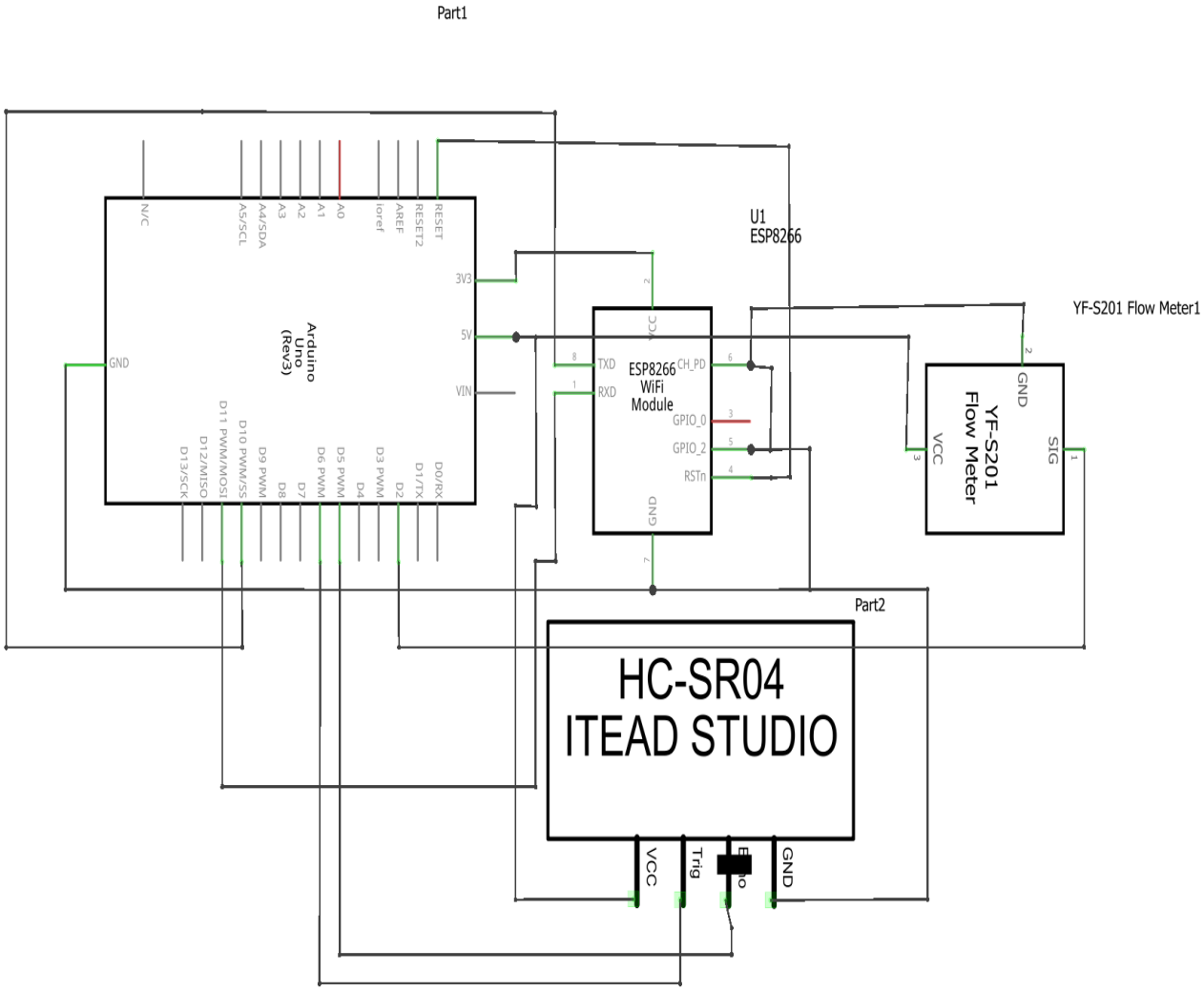

Fig. Schematic diagram of Arduino

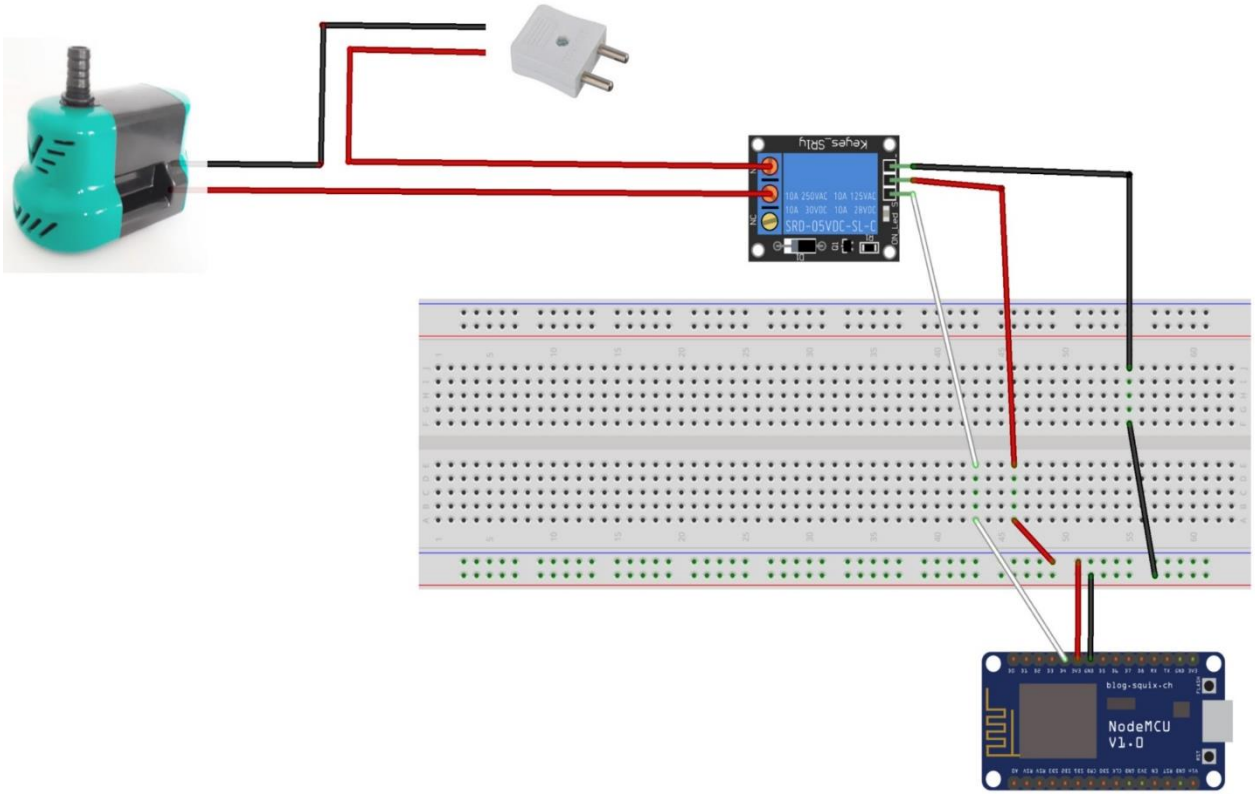

Fig. Circuit diagram of NodeMCU

4.3 Technology used:

4.3.1 Arduino Uno R3:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. The Arduino Uno is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features an ATmega16U2 programmed as a USB-to-serial converter. This auxiliary microcontroller has its own USB boot loader, which allows advanced users to reprogram it.

Why Arduino Uno R3?

Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

Open source and extensible hardware.

4.3.2 ThingSpeak:

ThingSpeak is an Internet of Things (IoT) platform that lets you collect and store sensor data in the cloud and develop IoT applications. The ThingSpeak IoT platform provides apps that let you analyze and visualize your data in MATLAB, and then act on the data.

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

Why ThingSpeak?

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.
- Automatically act on your data and communicate using third-party services

4.3.3 MIT App Inventor2:

MIT App Inventor is a great initiative. Basically, it is a visual blocks programming atmosphere. Its easy interface allows users to construct a functioning app without prior knowledge of any programming languages. It’s easy enough that even the children can

use it to build apps. The most considerable feature is that you can use these apps on all smartphones and tablets. Its unique features include blocks terminology that helps to build an App. Its user-friendly interface makes it simple to use while its terminologies make it unique which is a powerful combo. All in all, MIT App Inventor can have a simple first app build and running in less than 30 minutes.

Why MIT APP INVENTOR?

- It is very less time-consuming; you can develop an App in less than one hour.
- It helps in promoting creativity through the blocks and eliminates annoying factors of failure.
- The Apps build in this inventor can be easily shared.
- It is ideal for teaching perspectives. One can use it to teach the students to start with the very basics.
- It has an ability to invoke other apps. This is possible with one of its programming component which is the Activity Starter.
- It also provides access to many other functions of a phone like phone calls, SMS texting, sensors for location, sound, video, and much more.
- You can also save your data along its web database among other phones.
- You can also import the data from different sites. These sites maybe social or online stores.

4.3.4 IOT (Internet of Things):

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analysed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

Why IOT?

Communication: IoT encourages the communication between devices, also famously known as Machine-to-Machine (M2M) communication. Because of this, the physical devices are able to stay connected and hence the total transparency is available with lesser inefficiencies and greater quality.

Automation and Control: Due to physical objects getting connected and controlled digitally and centrally with wireless infrastructure, there is a large amount of automation and control in the workings. Without human intervention, the machines are able to communicate with each other leading to faster and timely output.

Efficient and Saves Time: The machine-to-machine interaction provides better efficiency, hence; accurate results can be obtained fast. This results in saving valuable time. Instead of repeating the same tasks every day, it enables people to do other creative jobs.

Monitor: The second most obvious advantage of IoT is monitoring. Knowing the exact quantity of supplies or the air quality in your home, can further provide more information that could not have previously been collected easily.

5.0 Details of Design, Working, And Processes:

5.1 Diagrams:

5.1.1 Activity Diagram

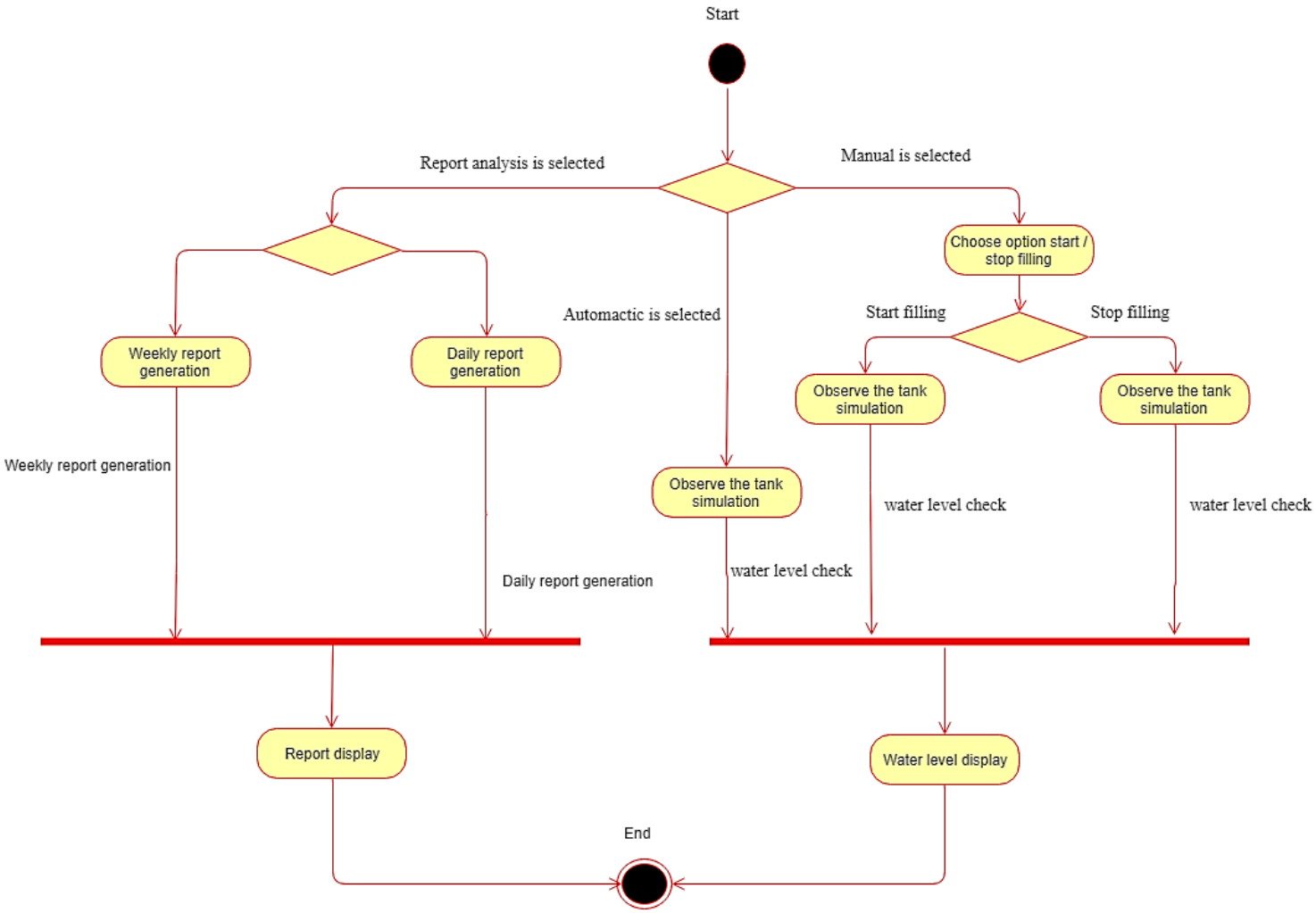

Fig. Activity Diagram

5.1.2 Flow Diagram:

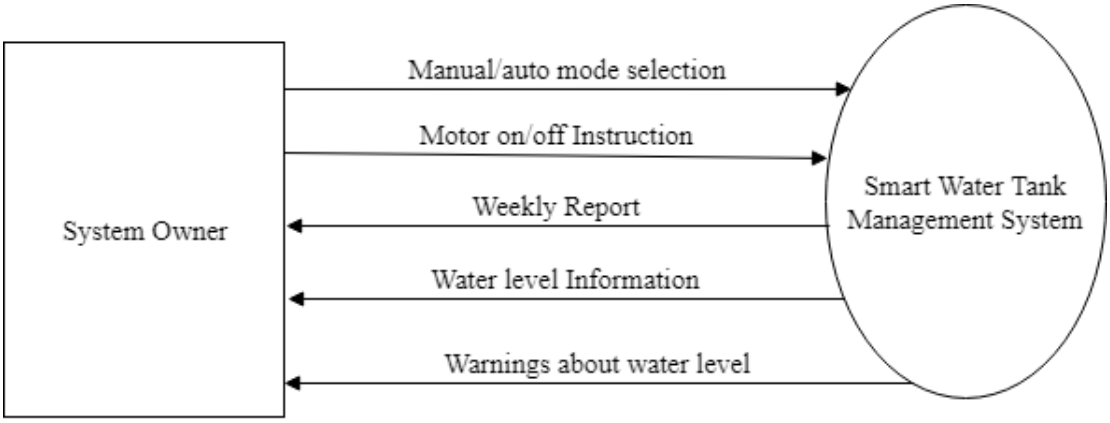

Fig. DFD level-0

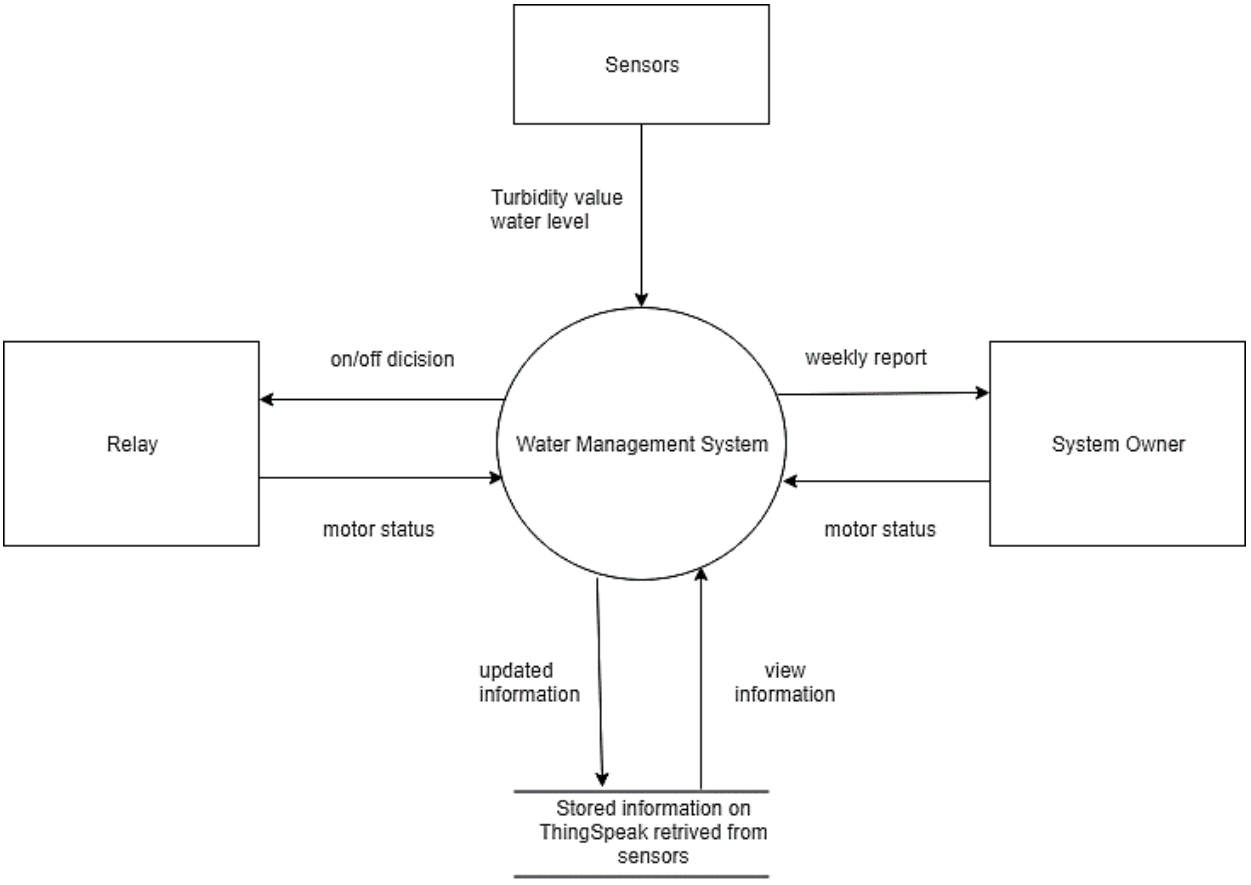

Fig. DFD level-1

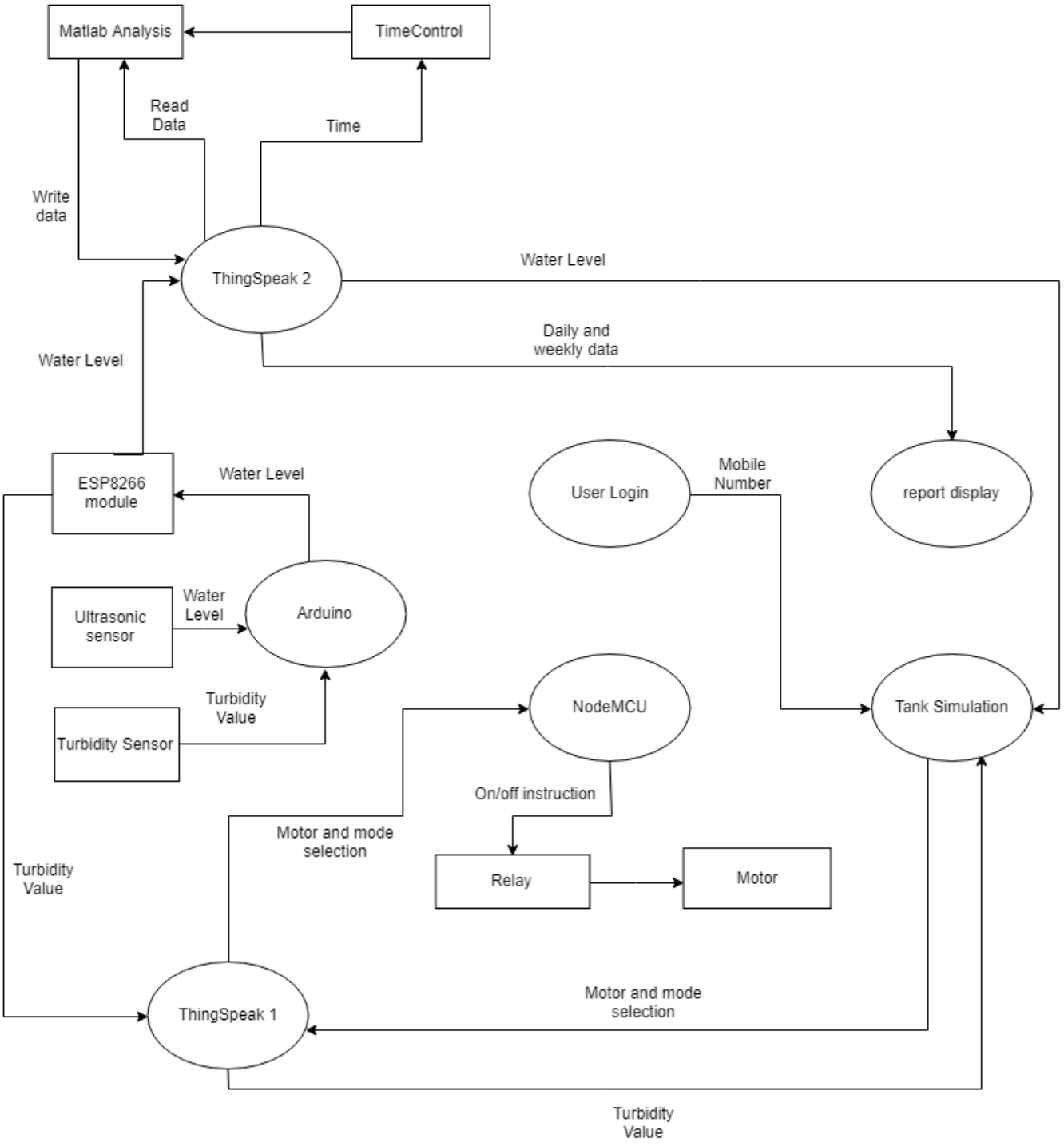

Fig. DFD Level-2

5.1.3 Entity-Relationship Diagram:

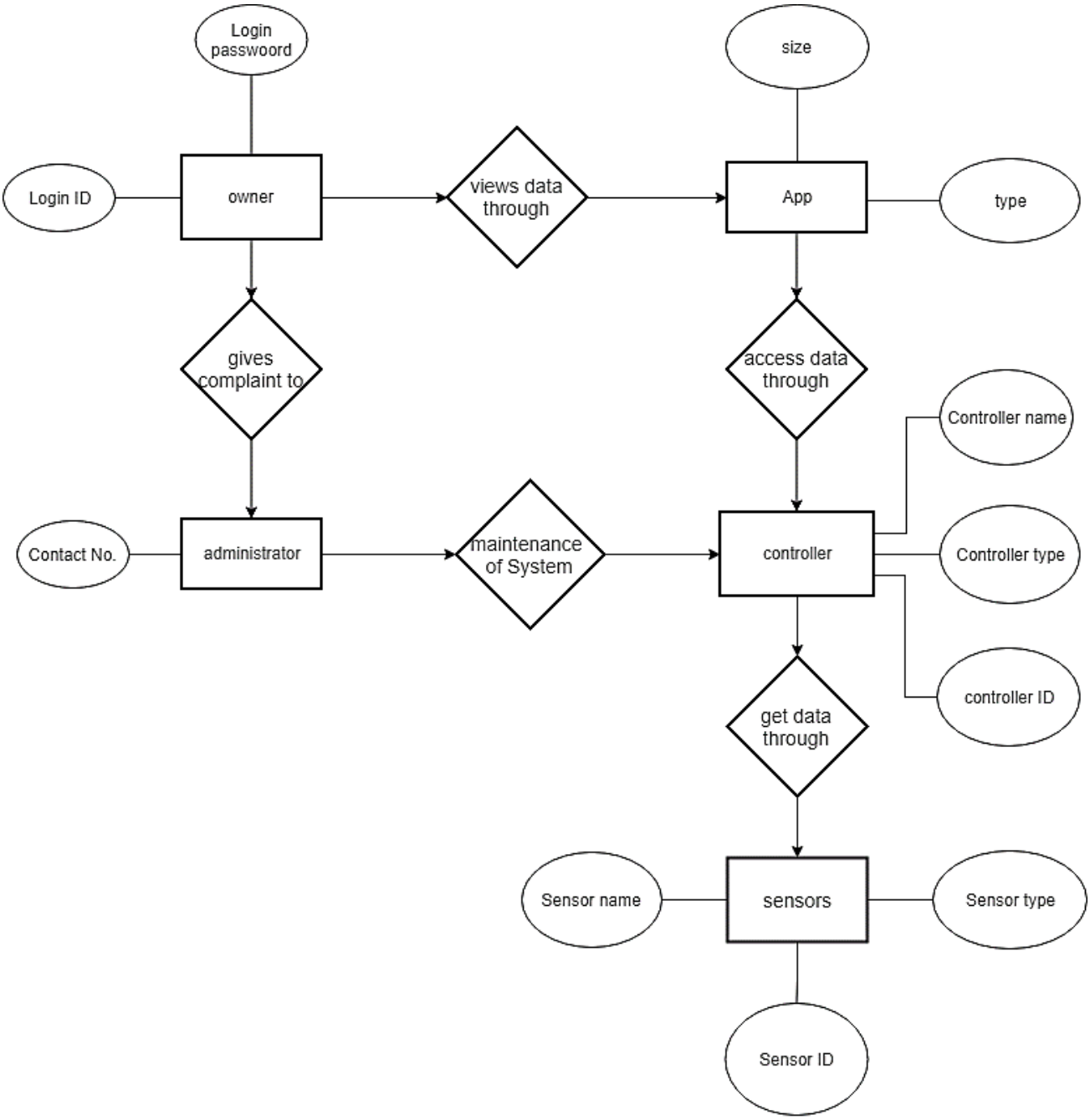

Fig. Entity -Relationship Diagram

5.1.4 Sequence Diagram:

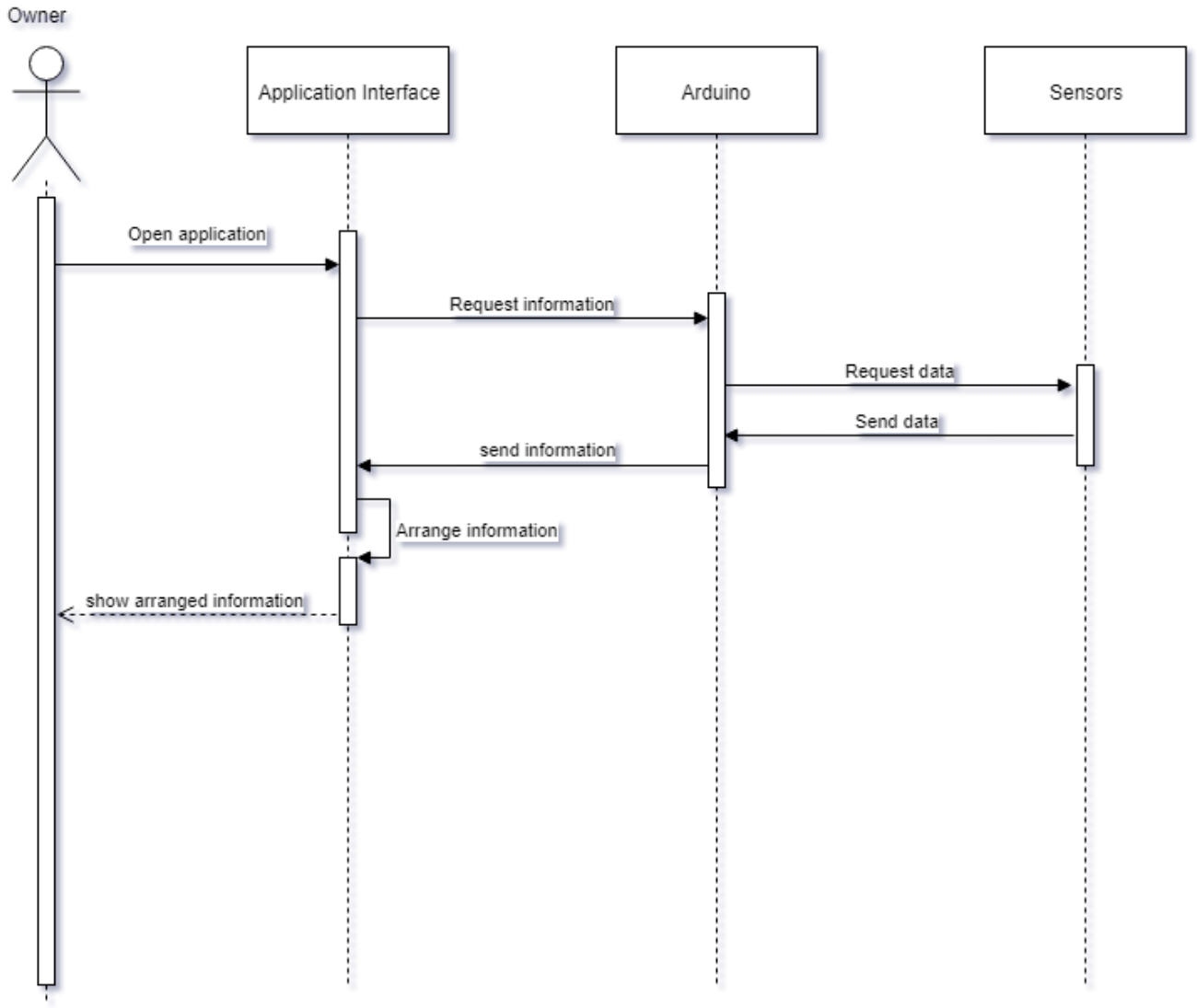

Fig. Sequence Diagram level 0

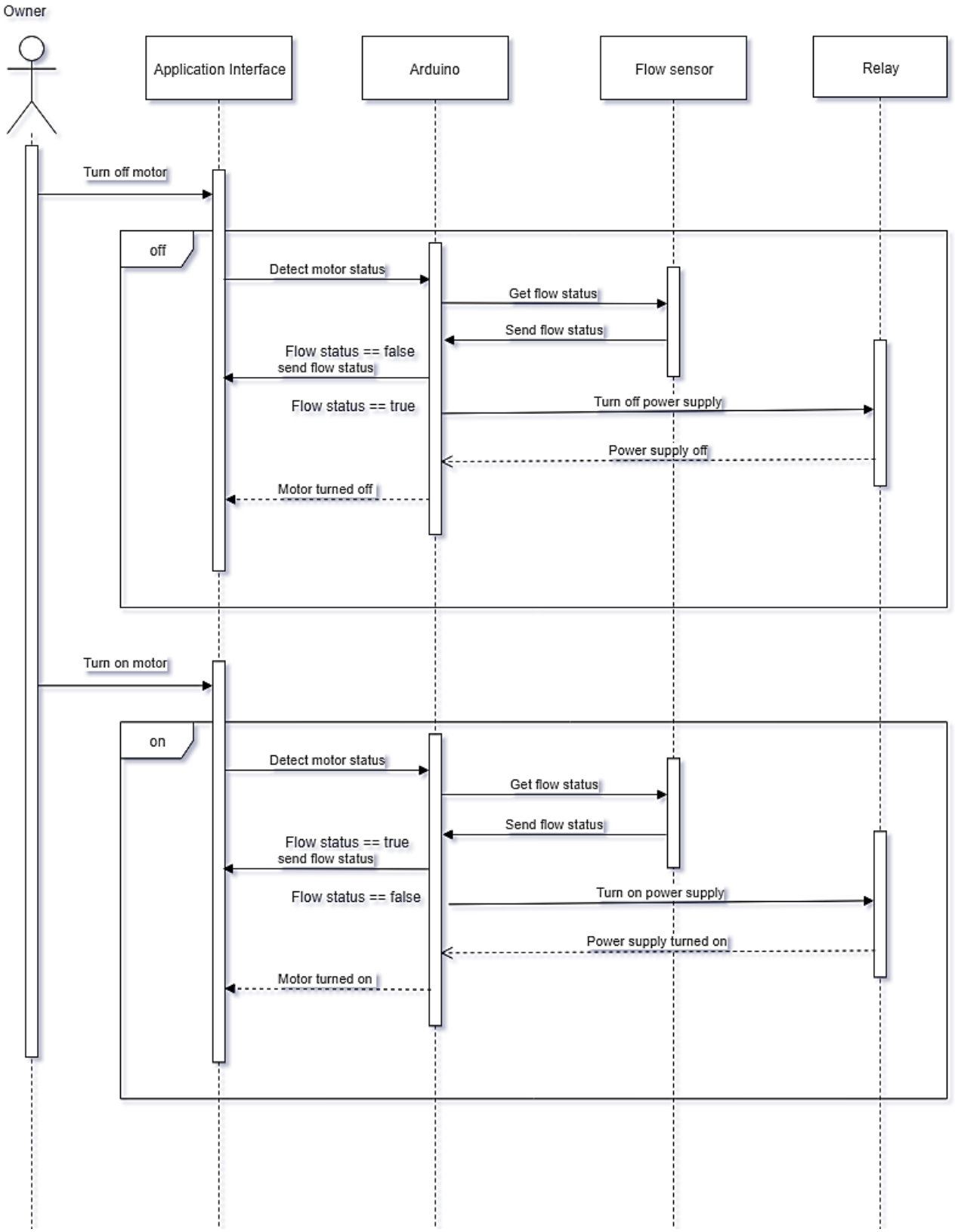

Fig. Sequence Diagram level 1

5.1.5 Use Case Diagram:

Use case: Monitoring Sensors

Primary actor: System owner.

Goal in context: To set the system to monitor sensors when the water tank level decreases to 20% and increases up to 90% and quality of water.

Preconditions: System has been programmed for water levels and water quality to recognize various sensors.

Trigger: The System owner decides to “set” the system, i.e., to turn on the system functions.

Scenario:

1. System owner: observes mobile app
2. System owner: enters login credentials
3. System owner: selects “automatic” or “manual”
4. System owner: observes water tank level and water quality that Water tank management system has been armed

Exceptions:

1. Mobile app is not ready: System owner checks all sensors to determine which are open; restarts system.
2. Login credentials is incorrect (mobile app warns once): System owner re-enters correct login credentials.
3. Manual selected: mobile app notifies level of water; user clicks on start fill or stop fill button.
4. Automatic selected: mobile app notifies level of water; system starts and stops filling water tank automatically.

Priority: Essential, must be implemented

When available: First increment

Frequency of use: Many times per day depending on use of water

Channel to actor: Via control mobile app

Secondary actors: System administrator, sensors

Channels to secondary actors:

System administrator: mobile app

Sensors: hardware and ultrasonic frequency interfaces

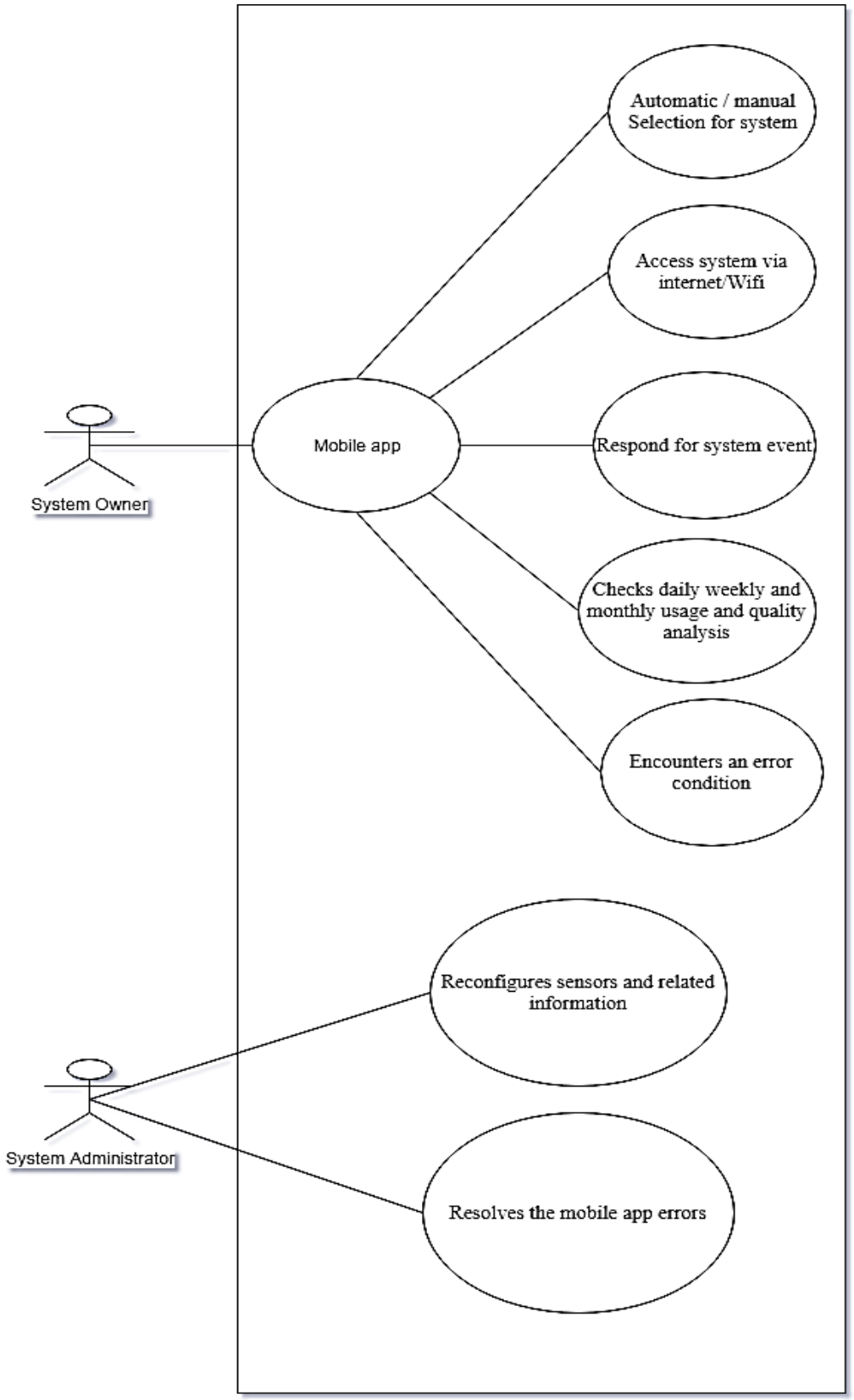

Fig. Use Case Diagram

5.2 Schedule:

Sr. No.	Details of activity	Week no.	Name of responsible Team members
1	Literature Survey	Week 1	Prathamesh Jadhav
2	Problem Definition	Week 2	Omkar Kajarekar
3	Defining Scope and methodology	Week 3	Atharva Joshi
4	Gathering information about arduino and data sharing through arduino	Week 4 - 5	Bhavya Oswal
5	Gathering information about MIT app development environment	Week 6	Omkar Kajarekar
6	Information collection and developing SRS	Week 7 - 8	Bhavya Oswal Prathamesh Jadhav
7	Analysis of consumables and resources required	Week 9 - 10	Atharva Joshi
8	Defining Procedure	Week 11	Omkar Kajarekar
9	Gathering resources and consumables required	Week 12 -13	Bhavya Oswal
10	Drawing Tentative designs and uml diagrams	Week 14 - 15	Prathamesh Jadhav Atharva Joshi Bhavya Oswal Omkar Kajarekar
11	Designing Mobile Application Layout	Week 16	Omkar Kajarekar Prathamesh Jadhav

12	Designing Circuit as per circuit diagram	Week 17	Atharva Joshi Bhavya Oswal
13	Developing Arduino Program	Week 18	Omkar Kajarekar Prathamesh Jadhav
14	Developing Mobile application	Week 19 - 20	Bhavya Oswal Atharva Joshi
15	Performing unit Testing of Mobile application	Week 21	Atharva Joshi
16	Performing Integration Testing of Mobile application	Week 22 -23	Bhavya Oswal
17	Performing test system platforms	Week 24	Omkar Kajarekar
18	Performing test I/O signal conditioning	Week 25	Prathamesh Jadhav
19	Testing Cabling & test fixtures	Week 26	Bhavya Oswal
20	Testing Automated report generation feature	Week 27	Atharva Joshi
21	Resolving the defects found in Testing	Week 28 -29	Omkar Kajarekar Prathamesh Jadhav
22	Testing system at Acceptance level	Week 30	Prathamesh Jadhav
23	Measuring Quality Assurance	Week 31	Atharva Joshi
24	Implementation and Demonstration of the developed System	Week 32	Bhavya Oswal Prathamesh Jadhav Omkar Kajarekar Atharva Joshi

6.0 Results and Applications:

6.1 Implementation and Design:

1. Implementation of application:

Implementation of application starts with the user login screen where the user must enter correct user name, password and phone number to access application. If user checks the checkbox of keep me signed-in and enters the correct credentials and phone number then the application gets directly to home screen when opened next time, here the application stores the login credentials and phone number of user.

On the home screen functionalities such as tank simulation, motor status and motor mode control, turbidity and flow value display are present.

The tank simulation displays the water percentage filled in the tank. Application retrieve's the data from ThingSpeak server and checks for new data after each 1-2 seconds. As the data on the server updates the simulation also displays the changes. Simulation displays the water level from 10% to 100% increasing 10 % at a time.

The application also displays the water remaining to be filled in percentage which is opposite to the tank simulation which displays the water filled in the tank. The turbidity value and flow values are also retrieved by application from ThingSpeak server and checks for new data after each 1-2 seconds.

User can change the motor mode from automatic to manual and vice versa as per requirement. In manual mode user can turn on or turn off the motor using application, only limitation is user can change the motor status after 10 to 12 seconds. In automatic mode system will control the motor status based on water level i.e. if the water level drops below a certain level system will turn on the motor and if the water level rise to a certain level system will turn off the motor.

When changing motor mode or status if the application is unable to send command to server then an error message as "unable to perform action" will be displayed and user has to select the motor mode and status again. If application successfully sends command to server then a message as "action performed successfully" will be displayed.

On the home screen two buttons are present one navigates to the screen which displays charts present on the ThingSpeak server and second navigates to the screen which displays the weekly and daily reports.

If the application is unable to retrieve the data from server due to unavailable or bad internet connection then an error message as “Internet not available” will be displayed until the application is unable to fetch data.

2. Implementation of hardware:

The implementation of hardware side starts with the WIFI module and NodeMCU. These two modules establish connection between the server and other sensors. There are two circuits which performs two major tasks.

First circuit that is Arduino circuit calculates the water level of tank with help of ultrasonic sensor. The ultrasonic sensor uses ultrasonic wave technology to measure the distance front of it. Ultrasonic sensor is placed on the top facing the tank bottom. This position is best to calculate the water level.

The second major component in Arduino circuit is turbidity sensor which calculates the water turbidity value. The water turbidity value can be helpful in order to determine the quality of water. The unit used for turbidity is in voltage.

The third component in Arduino circuit is flow sensor. Flow sensor is used to find rate that is speed of flow of water to the motor inlet. If flow sensor gives value less than threshold value then motor is immediately stopped. When the water rate get back to normal the motor will start automatically depending upon the condition chosen by user.

The second circuit that is NodeMCU circuit which is used to control the motor depending upon the user choice and water percentage in the tank. The component that is relay controls over the motor on/off state depending on the signal transmitted by the NodeMCU.

6.2 Testing:

1. Black Box Testing :

Black box testing involves looking at the specifications and does not require examining the code of a program. It is done from customer's point of view. The test engineer only knows the set of inputs and expected outputs and is unaware of how those inputs are transformed into outputs by the s/w.

- It is convenient to administer because they use the complete finished product and do not require any knowledge of its construction.
- Independent test laboratories can administer black box test to ensure functionality and compatibility.
- It requires a functional knowledge of the product to be tested.
- It is not mandatory to have knowledge of internal logic of the system nor does it require knowledge of programming language used to build the product.

a. Requirement based Testing:

It deals with validating the requirements given in the software requirement specifications of the software system. Explicit requirements are stated and documented as part of the requirements specification. Implied or implicit requirements are those that are not documented but assumed to be incorporated in the system. The precondition for requirement testing is a detailed review of the requirements specification. This process ensures that some implied requirements are converted and documented as explicit requirements, thereby bringing better clarity to requirements and making requirements based testing more effective.The process of testing is as follows:

- All explicit requirements and implicit requirements are collected and documented as "Test requirement specification" (TRS).
- Requirement based testing can be conducted on such as TRS.
- Requirements are tracked by Requirements Traceability Matrix (RTM). An RTM traces all the requirements through design, development and testing. In the sample RTM requirement identifier and description can be taken from the Requirements Specifications.

- Each requirement is assigned a requirement priority, classified as high, medium or low. This not only enables prioritizing the resources for development of a feature but is also used to sequence and run tests.

RTM provides some metrics that can be collected such as:

- Requirement addressed priority wise- that helps to know test coverage based on requirements.
- Number of test cases requirement wise- Total number of test cases created for each requirement.
- Total number of test cases prepared- Total of all test case prepared for all requirements.

Test results can be used to collect metrics such as:

- Total number of test cases (or requirements) passed.
- Total number of test cases (or requirements) failed.
- Total number of defects in requirements.
- Number of requirements completed.
- Number of requirements pending.

b. Boundary Value Analysis (BVA):

Conditions and boundaries are two major sources of defects in a software product. By condition we mean situations wherein, based on the values of various variables, certain actions would have to be taken. By boundaries, we mean "limits" of values of the various variables. BVA is method useful for arriving at tests that are effective in catching defects that happen at boundaries. BVA is useful to generate test cases when the input (or output) data is made up of clearly identifiable boundaries of ranges.

Example- Concept of discount when we buy goods

First ten units (1-10) $5.00

Next ten units (11-20) $4.75

Next ten units (21-30) $4.50

More than 30 units $4.00

The defects that are found in the situation is when buying 9, 10, 11,19,20,21,29,30,31, and similar number of items.

The reason for performing BVA are:

- Programmers tentativeness in using right comparison operator e.g. whether to use <= or <
- Confusion caused by the availability of multiple ways to implement loops and condition checking. E.g. for, while, repeat have different terminating conditions for the loop, this could cause confusion is deciding which operator to use.
- The requirements themselves may not be clearly understood, especially around boundaries, thus causing even correctly coded program to not perform the correct way.

c. Equivalence Partitioning:

It is a software testing technique that involves identifying a small set of representative input values that produce as many different output conditions as possible. It reduces the number of permutation and combinations of input, output values used for testing, thereby increasing the coverage and reducing the effort involved in testing. The set of input values that generate one single expected output is called a partition. When the behavior of the software is the same for a set of values, then the set is termed as equivalence class or partition.

Testing by this technique involves

1. Identifying all partitions for the complete set of inputs, output values for a product.

2. Picking up one member value from each partition for testing to maximize complete coverage.

d. Performance Testing:

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

Performance Testing Techniques:

- **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.
- **Stress testing -** It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
- **Soak testing -** Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.
- **Spike testing -** Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

6.3 Applications:

1. To save water wastage from overflowing tanks in industrial and local areas.
2. Proposed system gives quantitative as well as qualitative measures about water to the system owner as in some industries water is the main resource and keeping track of it is on priority.
3. Proposed system can be handled in automatic mode for user convenience this feature will helpful where system owner can't check the water level manually each time.

7.0 Future Scope and Conclusion:

7.1 Future Scope:

The developed system has enormous applications. It can be installed in following areas:

1. Private houses or bungalows.
2. Housing societies.
3. Institutions like schools, colleges and hostels.
4. Hospitals
5. Offices.
6. Agriculture.
7. In industry it can be used to check water levels of different tanks consisting of different types of liquids.

The applications stated above are demo applications that are possible with its future development. Initially for limitation of required fund we were just able to develop a Smart Water Tank Management System. The implementation in future can be extended to multiple tanks and data collected can be used for further analysis like consumption forecasting, water leakage detection, etc. Some limitation includes system can be slow for dependency in a single microcontroller, communication of system is depended on high speed internet, if internet becomes slow the communication will be slow. The system will work more efficiently if limitations are solved.

7.2 Conclusion:

This system is designed and implemented using Arduino Uno R3, Wi-Fi module, node MCU and ultrasonic sensor. Various tests are performed to verify if the ultrasonic sensor can identify current water level of water tank and according to the water level tests are conducted to start and stop water motor which is placed in another water tank. Meanwhile ultrasonic sensor detects water level and water motor starts or stops automatically or manually as per the user's choice. The current water level simulation is available in mobile application designed in MIT App Inventor as well as message is displayed when water level reaches a certain level. In the application there are extra features like turbidity value indicator, feature to display flow of water and graphs to display usage of water etc. So system like this is beneficial and effective for homes as well as industrial use to save the water by reducing amount of overflowing water.

8.0 Appendix:

8.1 ThingSpeak:

"ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP and MQTT 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 Key Features:

- ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:
- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.

About ThingSpeak Channel:

At the heart of Thing Speak is a Thing Speak Channel. A channel is where you send your data to be stored. Each channel includes 8 fields for any type of data, 3 location fields, and 1 status field. Once you have a Thing Speak Channel you can publish data to the channel, have Thing Speak process the data, and then have your application retrieve the data.

- 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
- 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.

- 1 status field - A short message to describe the data stored in the channel.

Time Control in ThingSpeak:

ThingSpeak servers automatically execute a TimeControl at the specified time, based on your time zone. Make sure that your time zone is correctly specified in your ThingSpeak profile. Keep in mind that multiple TimeControls that trigger writing to the same channel must still adhere to the message update limits.

Schedule Actions with TimeControl

1. Sign into ThingSpeak.
2. Select Apps > TimeControl.
3. Click New TimeControl.
4. Edit TimeControl settings:
    - Name: Enter a unique name for this TimeControl.
    - Time Zone: The time zone is based on your account settings. To change your profile time zone, click edit.
    - Frequency: Choose whether TimeControl runs once or at recurring intervals.

MATLAB Analysis:

ThingSpeak is an IoT analytics service that allows you to aggregate, visualize, and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak, you can perform online analysis and process data as it comes in. ThingSpeak is often used for prototyping and proof-of-concept IoT systems that require analytics.

With ThingSpeak, you can store and analyze data in the cloud without configuring web servers, and you can create sophisticated event-based email alerts that trigger based on data coming in from your connected devices.

MATLAB Analysis Settings

- **Name**: Enter a name for your analysis. Press enter, or click outside the name box anytime you change the name, and the stored name of your analysis is automatically updated.
- **MATLAB Code**: Enter custom code, or modify the sample code with your data.
- **Save and Run**: Click to save and execute the code.
- **Save**: Click to save your analysis without running the code. An asterisk on this button indicates unsaved changes.
- **Output**: This field displays the output of your code. Use it to debug and modify the code.
- **Clear Output**: Click to clear the code output.
- **Schedule Actions**: Select an action to schedule using this MATLAB code.
    - Receive notification of MATLAB analysis failure by email
    - Schedule your code to run at specified times.
    - Schedule your code to run when channel data meets a predefined condition.
- **Delete**: Click to delete the analysis.
- **My Channels**: Click this tab to see information about your saved channels, including:
    - Channel name
    - Channel ID
    - Write and Read API Keys
    - Channel fields
- **Documentation:** Click this tab for help using the MATLAB Analysis app

- **New Channel**: Click to create a new channel to hold the values of your analyzed data.

ThingSpeak API:

Keywords

Here are some keywords that are used in the API. An understanding of the terms will make the API documentation easier to understand.

- **Channel** – The name for where data can be inserted or retrieved within the ThingSpeak API, identified by a numerical Channel ID.
- **Field** – One of eight specific locations for data inside of a channel, identified by a number between 1 to 8 – A field can store numeric data from sensors or alphanumeric strings from serial devices or RFID readers.
- **Status** – A short status message to augment the data stored in a channel.
- **Location** – The latitude, longitude, and elevation of where data is being sent from.
- **Feed** – The collective name for the data stored inside a channel, which may be any combination of field data, status updates, and location info.
- **Write API Key** – A 16 digit code that allows an application to write data to a channel.
- **Read API Key** – A 16 digit code that allows an application to read the data stored in a channel.

- ## Write API Key:

In order to update a channel, you need to know your Write API Key. If your Write API Key gets compromised you can generate a new key.

Follow these steps to get your Write API Key:

- Select Channels
- Select the Channel to update
- Select Manage API Keys
- **Read API Key:**

  The Read API Key allows your application to read data from the API. You can generate multiple Read API Keys for different applications.

  Follow these steps to get a Read API Key:

  - Select Channels
  - Select the Channel to update
  - Select Manage API Keys
  - Select Generate New Read API Key

8.2 MIT app inventor:

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software (apps) for two operating systems (OS): Android (operating system) |Android, and iOS. MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smartphones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

It uses a graphical user interface (GUI) very similar to the programming languages Scratch (programming language) and the StarLogo, which allows users to drag and drop visual objects to create an application that can run on mobile devices. In creating App Inventor, Google drew upon significant prior research in educational computing, and work done within Google on online development environments.

Blocks as Logic

In MIT App Inventor, users code application behavior using a block-based programming language. There are two types of blocks in App Inventor: built-in blocks and component blocks. The built-in blocks library provides the basic atoms and operations generally available in other programming languages, such as Booleans, strings, numbers, lists, mathematical operators, comparison operators, and control flow operators. Developers use component blocks (properties, methods, and events) to respond to system and user events, interact with device hardware, and adjust the visual and behavioral aspects of components.

Top-Level Blocks

All program logic is built on three top-level block types: global variable definitions, procedure definitions, and component event handlers. Global variables provide named slots for storing program states. Procedures define common behaviors that can be called from multiple places in the code. When an event occurs on the device, it triggers the corresponding application behavior prescribed in the event block. The event handler block may reference global variables or procedures. By limiting the top-level block types, there are fewer entities to reason about.

Fast Iteration and Design Using the Companion

A key feature of MIT App Inventor is its live development environment for mobile applications. App Inventor provides this by means of a companion app installed on the user's mobile device. The App Inventor web interface sends code to the companion app, which interprets the code and displays the app in real time to the developer. This way, the user can change the app's interface and behavior in real time.

For example, a student making a game involving the ball component may want to bounce the ball off the edge of the play area. However, an initial implementation might have the ball collide with the wall and then stop. After discovering the Ball Edge Reached event, the student can add the event and update the direction of the ball using the Ball Bounce method. By testing the app and adjusting its programming in response to undesired behavior, students can explore more freely.

Components used:

1. Button

Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (Enabled). Its properties can be changed in the Designer or in the Blocks Editor.

Properties

- **BackgroundColor:** Specifies the Button's background color as an alpha-red-green-blue integer. If an Image has been set, the color change will not be visible until the Image is removed.
- **Enabled:** Specifies whether the Button should be active and clickable.
- **Text:** Specifies the text displayed by the Button.

Events

- **Click():** Indicates that the user tapped and released the Button.
- **GotFocus():** Indicates the cursor moved over the Button so it is now possible to click it.
- **LongClick():** Indicates that the user held the Button down.
- **LostFocus():** Indicates the cursor moved away from the Button so it is now no longer possible to click it.
- **TouchDown ():** Indicates that the Button was pressed down.
- **TouchUp():** Indicates that the Button has been released.

2. Checkbox

Checkbox components can detect user taps and can change their Boolean state in response. A Checkbox component raises an event when the user taps it. There are many properties affecting its appearance that can be set in the Designer or Blocks Editor.

Properties

- **BackgroundColor:** Specifies the background color of the Checkbox as an alpha-red-green-blue integer.
- **Checked:** Set to true if the box is checked, false otherwise.
- **Enabled:** Specifies whether the checkbox should be active and clickable.
- **Text:** Specifies the text displayed by the checkbox.
- Visible: Specifies whether the checkbox should be visible on the screen. Value is true if the checkbox is showing and false if hidden.

Events

- **Changed():** User tapped and released the checkbox.
- **GotFocus():** Checkbox became the focused component.
- **LostFocus():** Checkbox stopped being the focused component.

3. Label

Labels are components used to show text. A label displays text which is specified by the Text property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.

Properties:

- **BackgroundColor:** Specifies the label's background color as an alpha-red-green-blue integer.

- **Text:** Specifies the text displayed by the label
- **TextAlignment:** Specifies the alignment of the label's text: center, normal (e.g., left-justified if text is written left to right), or opposite (e.g., right-justified if text is written left to right).
- **Visible:** Specifies whether the Label should be visible on the screen. Value is true if the Label is showing and false if hidden.

4. Notifier

The Notifier component displays alert messages and creates Android log entries through an assortment of methods.

Properties

- **BackgroundColor:** Specifies the background color for alerts (not dialogs).
- **NotifierLength:** Specifies the length of time that the alert is shown – either "short" or "long".
- **TextColor:** Specifies the text color for alerts (not dialogs).

Events

- **AfterChoosing(*choice*)**
- Event after the user has made a selection for ShowChooseDialog.
- **AfterTextInput(*response*)**
- Event raised after the user has responded to ShowTextDialog.
- **ChoosingCanceled()**
- Event raised when the user cancels choosing an option. ShowChooseDialog.
- **TextInputCanceled()**

- Event raised when the user cancels ShowChooseDialog, ShowPasswordDialog, or ShowTextDialog.

5. TextBox

Users enter text in a text box component.

The initial or user-entered text value in a text box component is in the Text property. If Text is blank, you can use the Hint property to provide the user with a suggestion of what to type. The Hint appears as faint text in the box.

The MultiLine property determines if the text can have more than one line. For a single line text box, the keyboard will close automatically when the user presses the done key. To close the keyboard for multiline text boxes, the app should use the Hide Keyboard method or rely on the user to press the Back key.

The NumbersOnly property restricts the keyboard to accept numeric input only.

Other properties affect the appearance of the text box (TextAlignment, BackgroundColor, etc.) and whether it can be used (Enabled).

Text boxes are usually used with the Button component, with the user clicking on the Button when text entry is complete.

If the text entered by the user should not be displayed, use PasswordTextBox instead.

Properties:

- **BackgroundColor:** The background color of the TextBox`. You can choose a color by name in the Designer or in the Blocks Editor. The default background color is ‘default’ (shaded 3-D look).
- **Enabled:** If set, user can enter text into the TextBox.
- **FontBold:** Specifies whether the text of the TextBox should be bold. Some fonts do not support bold.
- **FontItalic:** Specifies whether the text of the TextBox should be italic. Some fonts do not support italic.
- **FontSize:** Specifies the text font size of the TextBox, measured in sp(scale-independent pixels).
- **FontTypeface**

The text font face of the TextBox. Valid values are 0 (default), 1 (serif), 2 (sans serif), or 3 (monospace).

- **Height:** Specifies the Textbox's vertical height, measured in pixels.
- **HeightPercent:** Specifies the Textbox's vertical height as a percentage of the Screen's Height.
- **Hint:** TextBox hint for the user.
- **MultiLine:** If true, then this TextBox accepts multiple lines of input, which are entered using the return key. For single line text boxes there is a done key instead of a return key, and pressing done hides the keyboard. The app should call the HideKeyboard method to hide the keyboard for a mutiline text box.
- **NumbersOnly:** If true, then this TextBox accepts only numbers as keyboard input. Numbers can include a decimal point and an optional leading minus sign. This applies to keyboard input only. Even if NumbersOnly is true, you can set the text to anything at all using the [Text`] (#TextBox.Text) property.
- **ReadOnly:** Whether the TextBox is read-only. By default, this is true.
- **Text:** The text in the TextBox, which can be set by the programmer in the Designer or Blocks Editor, or it can be entered by the user (unless the Enabled property is false).
- **TextAlignment:** Specifies the alignment of the TextBox's text. Valid values are: 0 (normal; e.g., left-justified if text is written left to right), 1 (center), or 2 (opposite; e.g., right-justified if text is written left to right).
- **TextColor:** Specifies the text color of the TextBox as an alpha-red-green-blue integer.
- **Visible:** Specifies whether the TextBox should be visible on the screen. Value is true if the TextBox is showing and false if hidden.
- **Width:** Specifies the horizontal width of the TextBox, measured in pixels.
- **WidthPercent:** Specifies the horizontal width of the TextBox as a percentage of the Screen's Width.

Events

- GotFocus(): Event raised when the TextBox is selected for input, such as by the user touching it.
- LostFocus(): Event raised when the TextBox is no longer selected for input, such as if the user touches a different text box.

6. WebViewer

Component for viewing Web pages.

The HomeUrl can be specified in the Designer or in the Blocks Editor. The view can be set to follow links when they are tapped, and users can fill in Web forms.

Properties

- **CurrentPageTitle:** Returns the title of the page currently being viewed
- **CurrentUrl:** Returns the URL currently being viewed. This could be different from the HomeUrl if new pages were visited by following links.
- **FollowLinks:** Determines whether to follow links when they are tapped in the WebViewer. If you follow links, you can use GoBack and GoForward to navigate the browser history.
- **Height:** Specifies the Web Viewer's vertical height, measured in pixels.
- **HeightPercent:** Specifies the Web Viewer's vertical height as a percentage of the Screen's Height.
- **HomeUrl:** Specifies the URL of the page the WebViewer should initially open to. Setting this will load the page.
- **IgnoreSslErrors:** Determine whether or not to ignore SSL errors. Set to true to ignore errors. Use this to accept self-signed certificates from websites.
- **PromptforPermission:** Determine if the user should be prompted for permission to use the geolocation API while in the WebViewer. If true, prompt the user of

the WebViewer to give permission to access the geolocation API. If false, assume permission is granted.

- **UsesLocation:** Specifies whether or not this WebViewer can access the JavaScript geolocation API.
- **Visible:** Specifies whether the WebViewer should be visible on the screen. Value is true if the WebViewer is showing and false if hidden.
- **WebViewString:** Gets the WebView's String, which is viewable through Javascript in the WebView as the window.AppInventor object.
- **Width:** Specifies the horizontal width of the WebViewer, measured in pixels.
- **WidthPercent:** Specifies the horizontal width of the WebViewer as a percentage of the Screen's Width.

Events

- PageLoaded(url): When a page is finished loading this event is run.
- WebViewStringChange(value): Event that runs when the AppInventor. setWebViewString method is called from JavaScript. The new WebViewString is given by the value parameter.

8.3 Hardware:

1. Arduino Uno R3:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. The Arduino Uno is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features an ATmega16U2 programmed as a USB-to-serial converter. This auxiliary

microcontroller has its own USB bootloader, which allows advanced users to reprogram it.

Why Arduino?

Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino Uno R3?

4. Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
5. Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
6. Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
7. Open source and extensible hardware.

Fig – Arduino R3

Arduino Uno R3 Pin Diagram

The Arduino Uno R3 pin diagram is shown below. It comprises 14-digit I/O pins. From these pins, 6-pins can be utilized like PWM outputs. This board includes 14 digital input/output pins, Analog inputs-6, a USB connection, quartz crystal-16 MHz, a power jack, a USB connection, resonator-16Mhz, a power jack, an ICSP header an RST button.

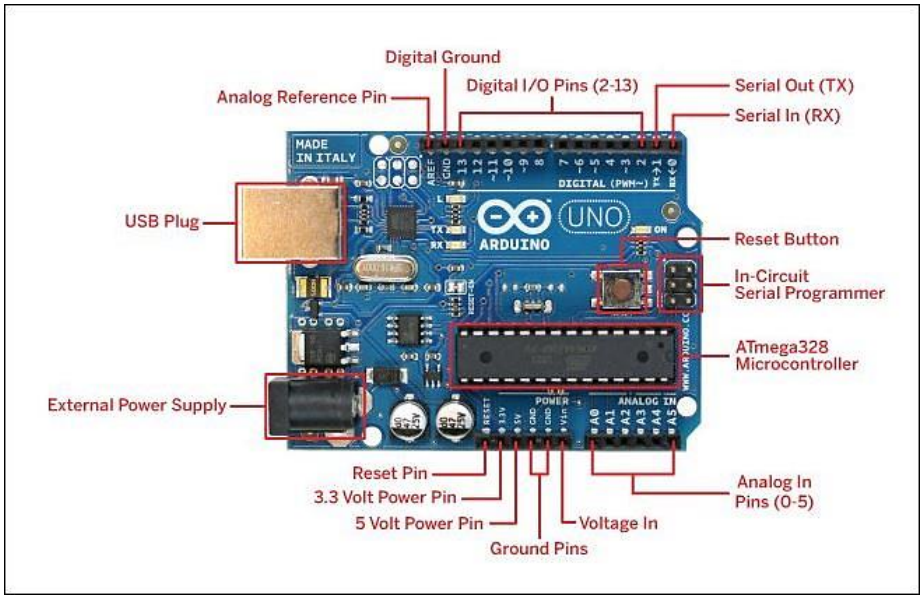

Fig- Arduino Uno Pin Diagram

1. Power Supply- The power supply of the Arduino can be done with the help of an exterior power supply otherwise USB connection. The exterior power supply (6 to 20 volts) mainly includes a battery or an AC to DC adapter. The battery terminals can be placed in the pins of Vin as well as GND. The power pins of an Arduino board include the following.
    1. Vin: The input voltage or Vin to the Arduino while it is using an exterior power supply opposite to volts from the connection of USB or else RPS (regulated power supply). By using this pin, one can supply the voltage.
    2. 5Volts: The RPS can be used to give the power supply to the microcontroller as well as components which are used on the Arduino board.
    3. GND: GND (ground) pins

2. Memory- The memory of an ATmega328 microcontroller includes 32 KB and 0.5 KB memory is utilized for the Boot loader), and also it includes SRAM-2 KB as well as EEPROM-1KB.
3. Input and Output- Uno R3 includes 14-digital pins which can be used as an input otherwise output by using the functions like pin Mode (), digital Read (), and digital Write ().
4. Serial Pins- The serial pins of an Arduino board are TX (1) and RX (0) pins and these pins can be used to transfer the TTL serial data.
5. Reset (RST) Pin- This pin brings a low line for resetting the microcontroller.
6. PWM Pins- the PWM pins of an Arduino are 3, 5, 6, 9, 10, & 11, and used with the function analog Write ().
7. SPI (Serial Peripheral Interface) Pins- SPI is a synchronous serial communication data link that operates in full duplex, which means the data signals carry on both the directions simultaneously. The SPI pins are 10, 11, 12, and 13.

2. Wi-Fi module ESP8266 :

Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much Wi-Fi ability as a Wi-Fi Shield offers. The ESP8266 module is an extremely cost-effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. It has a 64 KB boot ROM, 64 KB instruction RAM and 96 KB data RAM. External flash memory can be accessed through SPI. To communicate with the ESP8266 module, microcontroller needs to use set of AT commands.

Why Wi-Fi module?

The Wi-Fi is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Wi-Fi significantly expand on the geographic range of your phone and will be able to have in controlling the Arduino.

Why Wi-Fi module ESP8266?

1. ESP8266 Wi-Fi module is low cost standalone wireless transceiver that can be used for end-point IoT developments.
2. ESP8266 Wi-Fi module enables internet connectivity to embedded applications. It uses TCP/UDP communication protocol to connect with server/client.
3. This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices.

4. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area.

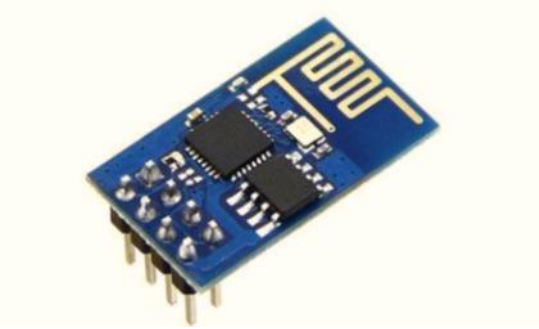

Fig- Wi-Fi module 8266

ESP8266 Module Pin Description-

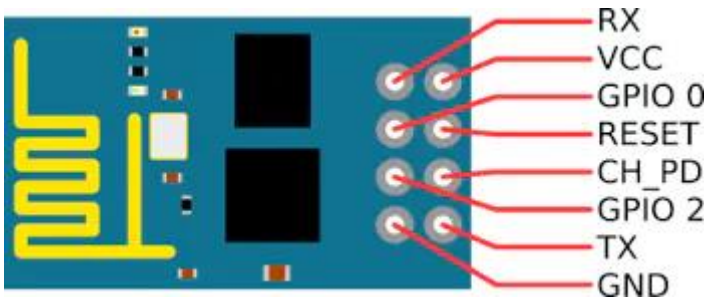

Fig- Wi-Fi module 8266 Module Pins

1. 3V3: - 3.3 V Power Pin.
2. GND: - Ground Pin.
3. RST: - Active Low Reset Pin.
4. CH_PD: - Chip Enable. High: On, chip works properly; Low: Off, small current
5. TX: - Serial Transmit Pin of UART.
6. RX: - Serial Receive Pin of UART.
7. GPIO0 & GPIO2: - General Purpose I/O Pins. These pins decide what mode (boot or normal) the module starts up in. It also decides whether the TX/RX pins are used for Programming the module or for serial I/O purpose's.

ESP8266 AT Command Set:

Sr No.	Function	AT Commands
1	Working	AT
2	Restart	AT+RST
3	Join Access Point	AT+CWJAP=”SSID”,”Password”
4	Set Wi-Fi Mode	AT+CWMODE=<mode> Mode: - 1 = STA (station) 2 = AP (Access Point) 3 = BOTH i.e. STA & AP
5	Set TCP/UDP Connection	AT+CIPMUX=<mode> Mode: - 0 = Single Connection 1 = Multiple Connection
6	Send Data	(CIPMUX=0) AT+CIPSEND=<data length> (CIPMUX=1) AT+CIPSEND=<id>,<data length>
7	Close TCP/UDP Connection	AT+CIPCLOSE

3. NodeMCU-

The NodeMCU (Node MicroController Unit) is an open source software and hardware development environment that is built around a very inexpensive System-on-a-Chip (SoC) called the ESP8266. Simply, NodeMCU is an open source LUA based firmware developed for ESP8266 Wi-Fi chip. Here what is open source means NodeMCU hardware design is open for edit/modify/build. It's like anyone can edit/modify/produce it and market their modified NodeMCU Development boards.

Why NodeMCU?

1. Low cost.
2. Integrated support for Wi-Fi network.
3. Reduced size of the board.
4. Low energy consumption.

Fig- NodeMCU

ESP8266 NodeMCU Pin Description-

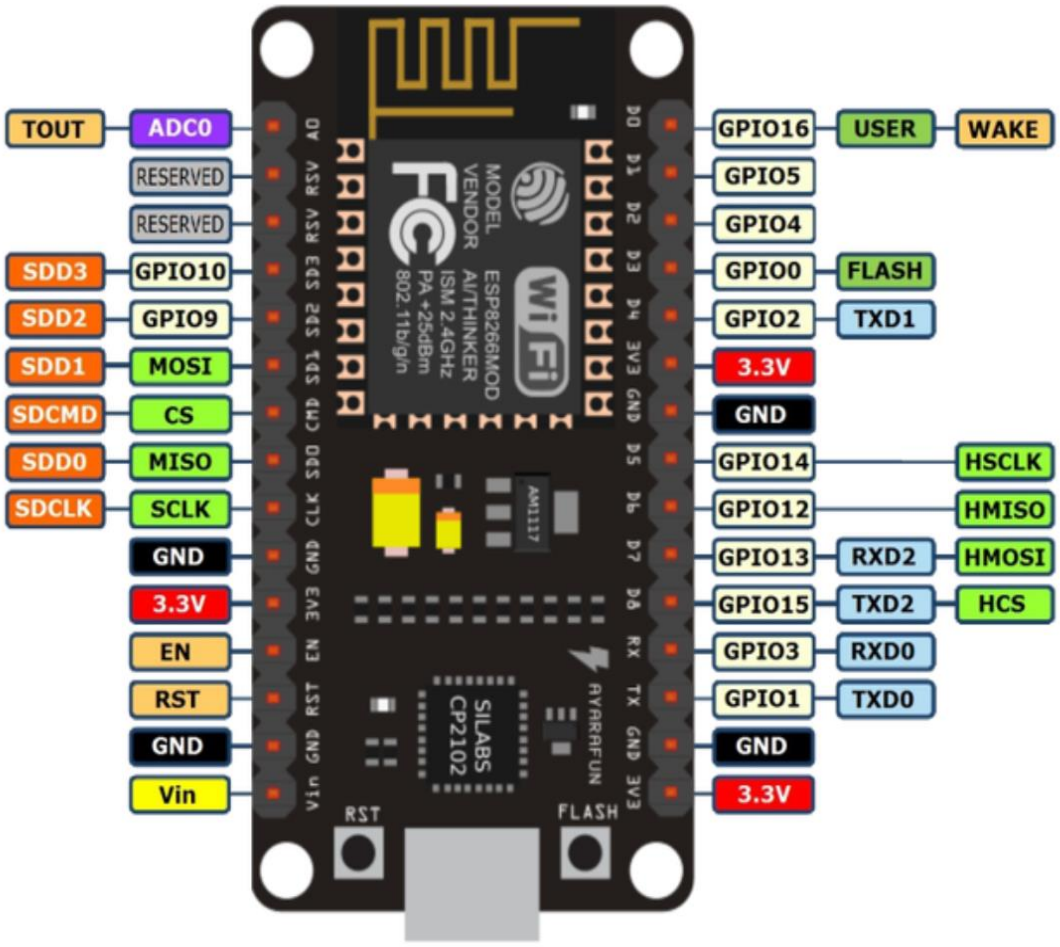

Fig- NodeMCU Pins

1. Power Pins- There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.
2. GND- is a ground pin of ESP8266 NodeMCU development board.
3. I2C Pins- are used to hook up all sorts of I2C sensors and peripherals. Master and I2C Slave are supported.
4. GPIO Pins- NodeMCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically.

5. ADC Channel- The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.
6. UART Pins- NodeMCU has 2 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.
7. SPI Pins- NodeMCU features two SPIs (SPI and HSPI) in slave and master modes.
8. SDIO Pins- NodeMCU features Secure Digital Input/ Output Interface (SDIO) which is used to directly interface SD cards.
9. PWM Pins- The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs.
10. Control Pins- are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
    1. EN pin – The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
    2. RST pin – RST pin is used to reset the ESP8266 chip.
    3. WAKE pin – Wake pin is used to wake the chip from deep-sleep.

5. Ultrasonic sensor-

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

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is D = ½ T x C (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second).

Why Ultrasonic sensor HS-SR04?

1. Excellent non-contact range detection.
2. High accuracy and stable readings in an easy-to-use package.
3. It uses SONAR to determine distance of objects which makes it more reliable.

Fig- Ultrasonic sensor HS-SR04

Ultrasonic sensor HS-SR04 Pin Description-

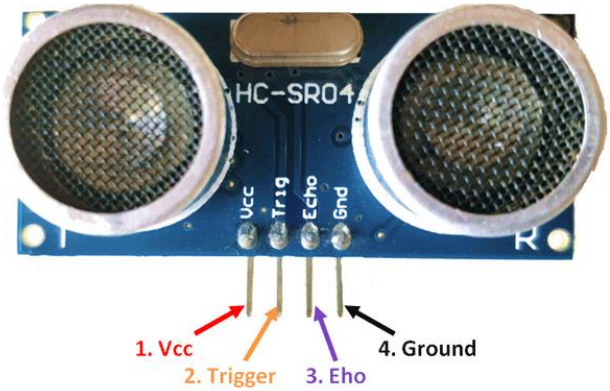

Fig- Ultrasonic sensor HS-SR04 Pins

1. Vcc- The Vcc pin powers the sensor, typically with +5V
2. Trigger- Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3. Echo- Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4. Ground- This pin is connected to the Ground of the system.

6. Relay-

A relay is an electromagnetic switch that is used to turn on and turn off a circuit by a low power signal, or where several circuits must be controlled by one signal. Relays control one electrical circuit by opening and closing contacts in another circuit. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized. When a relay contact is Normally Closed (NC), there is a closed contact when the relay is not energized. In either case, applying electrical current to the contacts will change their state.

Relays are generally used to switch smaller currents in a control circuit and do not usually control power consuming devices except for small motors and Solenoids that draw low amps. Nonetheless, relays can "control" larger voltages and amperes by having an amplifying effect because a small voltage applied to a relays coil can result in a large voltage being switched by the contacts.

Why Relay?

1. Circuits that operate at high voltages or at high currents cannot be controlled directly by an Arduino. Instead, you use a low-voltage control signal from the Arduino to control a relay, which is capable of handling and switching high-voltage or high-power circuits.
2. A relay consists of an electromagnet that, when energized, causes a switch to close or open.
3. Relays provide complete electrical isolation between the control circuit and the circuit being controlled.

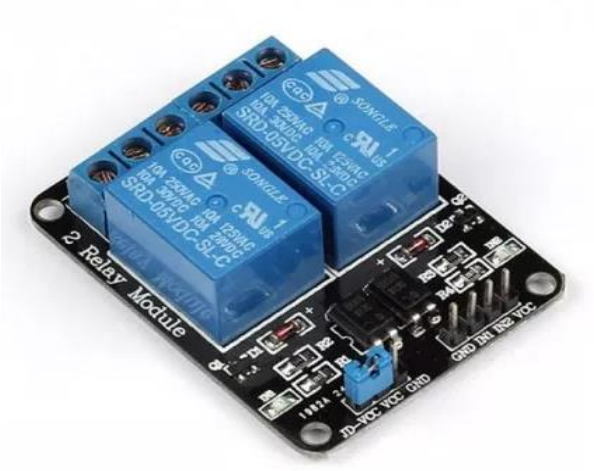

Fig- Relay

Relay Pin Description-

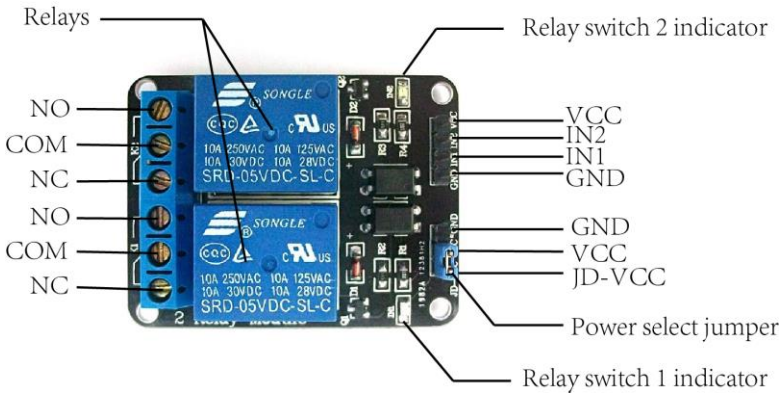

Fig- 2 Channel Relay Pinout

COM: common pin.

NC (Normally Closed): the normally closed configuration is used when you want the relay to be closed by default, meaning the current is flowing unless you send a signal from the Arduino to the relay module to open the circuit and stop the current.

NO (Normally Open): the normally open configuration works the other way around: the relay is always open, so the circuit is broken unless you send a signal from the Arduino to close the circuit.

GND: goes to ground.

IN1: controls the first relay (it will be connected to an Arduino digital pin).

IN2: controls the second relay (it should be connected to an Arduino digital pin if you are using this second relay. Otherwise, you don't need to connect it).

VCC: goes to 5V

7. Turbidity Sensor-

Turbidity sensors measure the amount of light that is scattered by the suspended solids in water. It uses light to detect suspended particles in water by measuring the light transmittance and scattering rate, which changes with the amount of total suspended solids (TSS) in water. As the TTS increases, the liquid turbidity level increases. Turbidity sensors are used to measure water quality in rivers and streams, wastewater and effluent measurements, control instrumentation for settling ponds, sediment transport research and laboratory measurements. This liquid sensor provides analog and digital signal output modes. The threshold is adjustable when in digital signal mode.

Why Turbidity Sensor?

1. It uses light to detect suspended particles in water by measuring the light transmittance and scattering rate, which changes with the amount of total suspended solids (TSS) in water. As the TTS increases, the liquid turbidity level increases.
2. Turbidity sensors are used to measure the water quality.

Fig- Turbidity Sensor

8. Flow Sensor-

Flow sensors are devices which are used to measure a flow rate of Fluid. These sensors are generally part of a flow meter that would help to measure the flow rate. The water flow sensor works on the principle of Hall Effect. Hall Effect is the production of the potential difference across an electric conductor when a magnetic field is applied in the direction perpendicular to that of the flow of current. The water flow sensor is integrated with a magnetic Hall Effect sensor, which generates an electric pulse with every revolution. Its design is in such a way that the hall effect sensor is sealed off from the water, and allows the sensor to stay safe and dry.

Why to use Flow Sensor?

1. To measure the rate of water flowing through the pipe.
2. The water flow sensor consists of a plastic valve body, a water rotor and a hall-effect sensor.
3. When the water flows through the rotor, rotor rolls and the speed of it changes with a different rate of flow. The hall-effect sensor outputs the corresponding pulse signal.

Fig- Flow Sensor

The sensor has 3 wires RED, YELLOW, and BLACK as shown in the figure below. The red wire is used for supply voltage which ranges from 5V to 18V and the black wire is connected to GND. The yellow wire is used for output (pulses), which can be read by an MCU. The water flow sensor consists of a pinwheel sensor that measures the quantity of liquid that has passed through it.

9.0 References and Bibliography:

Sr No.	Reference Title	Author
1	Smart Water Monitoring System using IoT(10 Oct.,2018)	Chinta Rohith Reddy, Saransh Shrivastava
2	Autonomous Water tank Filling System using IoT (9 Sept., 2018)	M. Ramakrishna, G. Dayanandam.
3	Smart water quality monitoring system for real time applications(2018)	K. Ramyadevi, M. Ramya
4	https://www.arduino.cc/en/Guide/ArduinoUno	-
5	https://roboindia.com/tutorials/mit-app-inventor-intro/	-
6	https://www.thegeekpub.com/wiki/list-of-arduino-sensors-and-modules/	-
7	https://electronics-project-hub.com/how-to-read-data-from-thingspeak-arduino-esp8266-nodemcu/	-