****

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A PROJECT REPORT

ON

**INTELLIGENT SUBMERSIBLE PUMP FOR IRRIGATION PURPOSES**

Submitted in the partial fulfillment of the requirement for the award of the degree of

**MASTER OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**SANJANA KULKARNI**

**SRN: R21TD011**

Under the Guidance of

**Dr. Laxmi Rananavare**

Associate professor

School of Computer Science and Engineering

2023-2024

**REVA UNIVERSITY**

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064

[www.reva.edu.in](http://www.reva.edu.in)



**SCHOOL OF** **COMPUTER SCIENCE AND ENGINEERING**

A Project Report

on

Intelligent Submersible Pump for Irrigational Purposes

Submitted in fulfillment of the requirements for the award of the Degree of

Master of Technology

Submitted by

Sanjana Kulkarni

R21TD011

Under the guidance of

Dr. Laxmi Rananavare

Associate professor

2023-2024

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064

www.reva.edu.in **DECLARATION**

I, ***SANJANA KULKARNI***, student of M.Tech (Computer Science and Engineering), at School of Computer Science and Engineering, REVA University, declare that this Project Report/Dissertation entitled “**Intelligent Submersible Pump for Irrigation Purposes**” is based on the results of the research work carried out by me and reported by me under the guidance and supervision of **Dr. Laxmi B Rananavare**, Associate Professor at School of Computer Science and Engineering.

I am submitting this Project Report / Dissertation in partial fulfilment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering by REVA University, Bengaluru during the academic year **2023-24**.

I declare that this project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 20% and it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

I further declare that this project / dissertation report or any part of it has not been submitted for award of any other Degree / Diploma of this University or any other University/ Institution.

*(Signature of the candidates)*

*Certified that this project work submitted by* ***SANJANA KULKARNI*** *has been carried out under my / our guidance and the declaration made by the candidate is true to the best of my/our knowledge.*

----------------------------- --------------------------------------

*Signature of Guide Signature of Director of School*

*Date: \_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_*

*Official Seal of the School*

****

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING**

**CERTIFICATE**

Certified that the project work entitled “**Intelligent Submersible Pump for Irrigation Purposes**” carried out under my/our guidance by **SANJANA KULKARNI (R21TD011)**, bonafide student of REVA University during the academic year 2023-2024, is submitting the project report in partial fulfilment for the award of **Master of Technology** in **Computer Science and Engineering** during the academic year **2023–2024**. The project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 20%. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

---------------------------- ----------------------------------

(GUIDE NAME,signature) (Director name,signature)

**Dr. Laxmi Rananavare Dr. Ashwinkumar U M**

Associate Professor & Supervisor Director, School of CSE

External Examiners

**Name of the Examiner with Affiliation Signature with Date**

**1.**

**2.**

**ACKNOWLEDGEMENT**

I would like to acknowledge our esteemed institute REVA UNIVERSITY for providing its excellent support and environment for the entire duration of the course. I am glad to take this opportunity to express my thanks and appreciation to the people who have been instrumental in the successful completion of this research project.

I would like to express my sincere gratitude to ***Dr. P. Shyama Raju,*** Chancellor, REVA University for providing us congenial environment and surroundings to work on.

I’d also like to express my gratitude to ***Mr. Umesh Raju,*** Pro Chancellor, REVA University wholeheartedly.

I wish to place on record my grateful thanks to ***Dr. N. Ramesh***, Vice-Chancellor I/C, REVA University, for providing excellent facilities in college, which helped us in every way during the proceedings of the project.

I would like to extend my heartfelt thanks to **Dr. R C Biradar**, Pro Vice – Chancellor, REVA University.

I would like to express my sincere thanks to ***Dr. Ashwinkumar U M,*** Director and HoD, School of Computer Science and Engineering, for his valuable guidance and encouragement.

I would like to express my sincere thanks to ***Dr. Ranjitha*** ***U. N*** HoD- M.Tech, School of Computer Science and Engineering, for her valuable guidance and encouragement.

I am highly indebted to my guide **Dr. Laxmi Rananavare**, Associate Professor, School of Computer Science & Engineering, on her periodic inspection, time to time evaluation of the project and kind support to bring the project work to the completion stage through proper guidelines.

I extend my sincere thanks to the entire faculty of School of Computer Science & Engineering, REVA University who have encouraged me throughout the course of our Masters. I would like to thank my parents, family and friends and the one above all of us, the omnipresent god for the kind support they provided.

**(SANJANA KULKARNI, R21TD011)**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **PRELIMINARY PAGES** | | |
| **S.no** | **CONTENTS** | **PAGE NUMBER** |
| **1** | **TITLE** | **1** |
| **2** | **DECLARATION** | **2** |
| **3** | **CERTIFICATE** | **3** |
| **4** | **ACKNOWLEDGMENT** | **4** |
| **5** | **TABLE OF CONTENTS** | **5** |
| **6** | **LIST OF FIGURES** | **7** |
| **7** | **ABSTRACT** | **8** |

|  |  |  |
| --- | --- | --- |
| **MAIN CONTENT** | | |
| **CHAPTER** | **TITLE** | **PAGE NUMBER** |
| **1** | **INTRODUCTION** | **9** |
| **2** | **RELATED WORK** | **11** |
| **3** | **METHODOLOGY** | **15** |
| **4** | **RESULTS & DISCUSSION** | **37** |
| **5** | **CONCLUSION AND FUTURE WORK** | **42** |
|  | **REFERENCES** | **43** |

|  |  |  |
| --- | --- | --- |
| **APPENDIX** | | |
| **S.no** | **TITLE** | **PAGE NUMBER** |
| **1** | **ACCEPTANCE LETTER(Paper)[** | **35** |
| **2** | **PARTICIPATION CERTIFICATE** | **36** |
| **3** | **THESIS PLAGIARISM CHECK REPORT** | **37** |
| **4** | **MY PAPER** | **38** |
| **5** | **PAPER PLAGIARISM CHECK REPORT** | **44** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Fig.no** | **Label** | **Page.no** |
| **1** | **1** | ESP8266 Pin Configuration | **14** |
| **2** | **2** | Pin Configuration of 5V Relay | **16** |
| **3** | **3** | Water Level Sensor | **17** |
| **4** | **4** | Temperature Sensor | **18** |
| **5** | **5** | Water Level Sensor | **19** |
| **6** | **6** | Arduino Software Window | **20** |
| **7** | **7** | Thing Speak (IoT Platform) | **23** |
| **8** | **8** | Circuit Diagram of Intelligent Submersible Pump | **24** |
| **9** | **9** | Flow Chart of the Intelligent Submersible Pump | **25** |
| **10** | **10** | Prototype of Intelligent Submersible Pump | **26** |
| **11** | **11** | Voltage Fluctuating Output on Ardino Software tool | **26** |
| **12** | **12** | E-mail Alert send to mail ID indicating Low Voltage fault | **27** |
| **13** | **13** | Voltage Fluctuating Output on Ardino Software tool | **28** |
| **14** | **14** | E-mail Alert send to mail ID indicating High Voltage fault | **29** |
| **15** | **15** | High Winding Temperature Output on Arduino Software tool | **30** |
| **16** | **16** | E-mail Alert send to mail ID indicating High Temperature Alert | 40 |
| **17** | **17** | Water Level Alert on Ardino Software tool | 41 |
| **18** | **18** | Tank Water Level Alert sent to user mail ID | **41** |

**LIST OF FIGURES**

**19 19** Pink : sufficient water;red : low level on ThingSpeak 41

**ABSTRACT**

Abstract—India, with its villages and thriving agriculture sector, heavily relies on climate conditions for successful cultivation. Unfortunately, the insufficient availability of water sources during the monsoon season poses a challenge. To combat this issue, farmers have resorted to utilizing irrigation methods in their agricultural fields. Notably, bore well electric pumps and lift irrigation systems have become the go-to choices for irrigation in India. These electric pumps, ranging in capacity from 3HP to 15HP, are responsible for efficiently extracting and pumping water. However, many farmers in rural areas encounter multiple problems, such as unscheduled load shedding, under-voltage, over-voltage, dry run scenarios, and consequent motor failures. The high costs associated with motor replacements or repairs have raised concerns among farming communities. Consequently, ongoing research is focused on developing robust motor protection mechanisms to prevent these faults. An intelligent protection and control system is developed for safeguarding electric motors in agricultural environments, leveraging the Internet of Things (IoT) and cutting-edge sensors.

**CHAPTER 1: INTRODUCTION**

     India, being primarily an agricultural country, heavily relies on farming as the main occupation for a majority of its population. With approximately 70% of the people depending on agriculture, it contributes significantly to the nation's economy. However, despite the sector's crucial role, its share in the country's GDP has declined due to the growth of other sectors. To tackle various challenges faced by farmers, such as inadequate water supply and unreliable power sources, innovative solutions incorporating the Internet of Things (IoT) and intelligent pump control systems are being introduced.

In the latest interconnected environment, utilization of Embedded Internet of Things (IoT) devices has become increasingly prevalent. These devices are equipped with Wi-Fi communication modules, enabling seamless connection to the Internet. Within the realm of IoT, we encounter various embedded systems such as STM32L4, which assists in discovering IoT nodes, and Node MCU esp8266, a reliable IoT component. Gathering environmental data, particularly temperature and humidity, is made possible through the application of the advanced DHT11 sensor. Employing C++ and similar low-level programming languages, developers can effectively craft IoT devices and implement IoT Cloud software. In this context, the IoT Cloud assumes significance, as it has been specifically modified to handle data processing emanating from IoT devices. With the evolution of IoT, the Cloud has transcended geographical boundaries and can be leveraged from any part of the world. For seamless integration and exchange of information between IoT devices or with the IoT Cloud, access to the Internet is imperative[[1].](#one) In today's interconnected world, the utilization of Embedded Internet of Things devices has become increasingly prevalent [[1].](#two)

In the revolutionary era of the IoT, sensors play a crucial part in the process of facilitating advanced environmental sensing, remote surveillance, and health monitoring [[3]](#three). We regularly interact with IoT technology due to the monitoring, measurement, and management of various objects through technologies like RFID, wireless, and WAN. In the field of electronic engineering, a significant number of commercial operations focus on embedded systems, and many embedded devices run code in C/C++. Common tasks in traditional C/C++ programming include providing text and graphics support for Windows and Linux. C/C++ API modules are used for sensor integration, particularly for monitoring liquid levels in tanks used for storage purposes and containers. Sensors measuring water level are important in tanks used in distribution of water supplied for drinking purposes [[4]](#four).

.

**CHAPTER 2: RELATED WORK**

There is research that has been carried out in the field of Agricultural Submersible Pump,its related Issues, IoT, and Sensors. Below are Information and ideas collected from the paper and cited respectively.

The Electric Submersible Pump (ESP) plays a vital role in achieving a high recovery percentage in the oil and gas industry. These pumps are commonly either low-speed progressive cavity pumps or multi-stage high-speed centrifugal pumps. Traditionally, an enclosed submersible induction motor (IM) is used to power the ESP. Typically, these pumps are utilized for process of removing gas and oil from underground geological formations, situated within wells underground of 100-1000 ft. Powering the ESP is achieved through long cables connected to variable frequency drive or adjustable frequency drive or VSD on surface installation or platform. Unfortunately, conventional IPM drives with position sensor-based speed controllers cannot be utilized by ESPs [[4].](#four)

A range of straightforward drivers for file systems, networks, and easy-to-use APIs have been developed to enhance the input-output capabilities of the NXP1768 chip [[5]](#five).

The submersible diaphragm pump finds extensive use in oil wells, but this utilization often leads to various issues such as wax deposition, corrosion of oil pipes, cable damage, and obstruction in monitoring pump parameters, results in expensive and easily worn pipe rods etc [[6]](#six).

IMs have lower efficiency and thermal stability due to power losses in the rotor through slippage. The energy efficiency of ESP depends on the reduction of energy losses in the supply channel. Converters for energy-efficient SEM control are available in the country and abroad. The permanent Magnet Synchronous Motor (PMSM)-controlled Electric Submersible Pumps (ESPs) provide enhanced efficiency, reduced dimensions, expanded operating ranges, lower heat production, and superior performance compared to equivalently size Induction Motor (IM)-controlled ESPs. Nonetheless, PMSM-ESPs encounter control challenges attributed to the nonlinearity inherent in Permanent Magnet (PM) motors [[7]](#seven).

Inductive loads such as AC and DC motors are considered common in industrial applications. In industry, three-phase induction machines are popular, for domestic use, single-phase induction motors are preferred for their low maintenance, low operating costs, robust construction, and also for their efficiency and reliability. Motors can face damage such as cooling, temperature and vibration, lubrication, stator, rotor, bearing, and winding failures. Even small problems will result in damages and financial losses. Monitoring of motors and their parameters is necessary to avoid damage. Thanks to IoT technology, engines can monitor and facilitate data communication between machines [[8]](#eight).

It is important to identify errors while they are still being created. According to the literature, the most damaging faults in electrical equipment are stator and bearing abnormalities, which account for more than 80% of induction motor problems. Stator rotation faults are caused by aging and damage to the windings, as well as other insulation techniques used in the construction of new motors. Once thresholds are exceeded, the insulation continues to degrade, resulting in thermal hot spots that affect stator rotation. Similarly, the main causes of bearing-related failures include aging, lubricant and its viscosity diminishes over time. As a result, the rolling balls of the bearings begin to seize [[9]](#nine).

The Internet of Things technology has lately received much study attention and has emerged as a very promising technological paradigm. The number of connected devices in the Internet of Things is expected to see a substantial increase to a large number .Wireless sensor networks (WSNs) perform better when the IoT is used, especially in requirements which requires environmental and healthcare monitoring. IoT has made it easy for consumers to get continuously updated information about the conditions and characteristics of the environment and the physiological state data from anywhere, anytime through web browsers or mobile software application. The Internet of Things enables seamless data collection, wireless networking, cloud storage, and analytics that are critical to the design and deployment of these systems. [[10]](#Ten).Wearable Body Area Network ,WSN type is widely used to monitor physiological signals and improve overall well-being and quality of life. For example, the study used WBANs to monitor safety-related variables such as temperature, humidity, and ultraviolet (UV) radiation levels. The design and deployment of a Wearable Sensor Network by utilizing the technology IoT has several advantages for the wearable technology industry, some of which are specified below: 1. Data collection 2. Wireless connectivity 3. Storage on cloud platforms also enables data processing, analysis, and long-term storage 4. Data analytics: Data analytics technologies that can process and analyse sensor-collected data are included in the IoT. Various machine learning algorithms can detect patterns and similarities in data to offer insightful information about people's health [[10]](#ten).

The effectiveness of IOT and automation is evident for the following reasons:

1. Diverse connectivity: IoT can connect and connect a wide range of devices, from mobile devices and tablets to stand-alone devices, enabling comprehensive monitoring and control

2. Reduced dependence on human labour: Both IoT and automation are essential to decrease the human interference and completely rule out the possibility of human error, leading to more reliable and consistent results.

3. Remote Monitoring: In agriculture, farmers can use a variety of equipment to remotely monitor crop and soil health from any location, enabling quick intervention and decision-making.

4. Time-saving: IOT and automation simplify the report generation and monitoring process, saving farmers valuable time and effort.

5. Advanced analytics: IoT and automation facilitate a wide range of analytics, such as monitoring average rainfall and evaluating soil condition gradients, providing valuable insights for informed agricultural decision-making [[11].](#eleven)

Arduino UNO: An open-source microcontroller board that features sets of analog and digital I/O pins, and devices.

16x2 Liquid Crystal Display: An electronic device generally used as a screen in TVs, PCs, and smartphones, which helps to control the device independently and displays moving images.

DHT11: The DHT11 is a combination of humidity and temperature sensor that provides a simple,easy-to-read interface for reading humidity and temperature. These can be used mainly in the fields to select a suitable environment for growing crops [[11].](#eleven)

GSM module: GSM modules are communication modules that use mobile networks to wirelessly receive and send data, creating a data transfer between the device and the network [[11].](#Eleven)

Wi-Fi Module: A device that allows Wi-Fi signals to be transmitted to smart devices such as laptops and phones, allowing users to gain high-speed network access. A popular example of such a module is the ESP8266 module. It enables us to meet the demands of the IoT industry [[12]](#twelve).

The system for calculating fault distance relies only on measuring original flash events and analysing the reflection pattern of traveling waves caused by the fault. Unlike other approaches that generally uses Rogowski coils for measuring step waves, this particular system explores the operation of Capacitive voltage detectors within a medium voltage substation [[13]](#thirteen).

Water can be saved without being wasted with help of to the Internet of Things and sensors; this system shuts down the motor in situation when water reaches the top level in the tank, reducing water wastage. When the water level reaches full in the tank, continuous usage of the motor wastes water and electricity. So IoT turns off the engine by reducing human interference. The relay that connects the motor consists of an oscillator connected to the tank and an IC connected to the circuit that controls how the circuit turns on and off [[14](#Fourteen)]. Requirements for water level monitoring systems include residential areas etc. Pumps that can be completely submerged in water are referred to as submersible pumps, often known as electric submersible pumps. A 3-6V source is used for power supply [[15](#fifteen)].

**CHAPTER 3:** **METHODOLOGY**

**3.1 Hardware and Software Prerequisites:**

Before working on a project there are requirements required to create this project which are software and hardware requirements that are listed below:

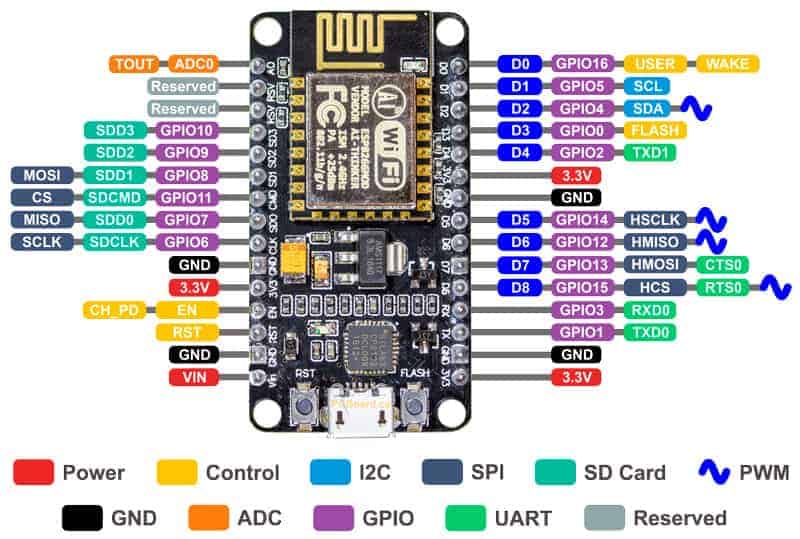
**NodeMCU ESP826**

Fig-1:ESP8266 Pin Configuration

ESP8266 is affordable Microchip, development board that integrates the Wi-Fi model. It enables easy designing of ideas and development of IoT projects. Simple description of its pin configuration is as follows:

Power Pins: These include pins for connecting to power sources such as VCC (typically 3.3V) and ground

GPIO Pins: These are named as GPIO0, GPIO1, and GPIO2 and so on and are designed as digital input / output pins.

Analog Pins: These pins allow analog input to read signals from sensors or other devices. It varies depending on the specific board variant.

Special Pins: There are the Reset pin "RST" which is used to reset the ESP8266 and the Serial pins "TX and RX" used for serial communication.

Totally ESP8266 offers a better platform for building IoT projects.

## **Relay Module Pin DiagramRELAY**

Fig-2:Pin Configuration of 5V Relay

Fig-2:Pin Configuration of 5V Relay

Fig-2:Pin Configuration of 5V Relay

Fig-2:Pin Configuration of 5V Relay

A 5 Volts relay is coated with a blue color material plastic, used for both the AC & DC loads.Works with 5V, so it is known to be a 5 Volts relay. It cost varies based on its design and features.

This relay module includes six pins, where each operating for a different function in its operation.

NO Port maintains open connection until the signal is provided with the COM. The Normally Closed port is in connection with the COM pin, forming a complete circuit. When signaled to relay, the common contact pin makes a connection to the NO pin, thus completing the connection with NO port and detaching from NC port. Then, the Common Contact pin connects load that requires switching via the relay. NO, COM, NC all these ports provides relay functioning for the effective operation of connected gadgets, devices etc.

Signal Pin: Relay activates via this port, operates in two cases. During active low case, the relay activates once if an active low signal is given to the signal pin, whereas in an active high case, the relay would activate once if a high signal is given to the signal pin.

5V VCC: This is used to power the module with 5 volts Direct Current.

Ground: Links to the Ground terminal of the main supply.

Specifications:

* The range of supply is between 3.75V – 6V.
* State if inactivity current is 2mA
* When the relay is active then the current is approximately 70mA
* 250VAC/30VDC is relay’s contact voltage
* 10A is maximum current

Figure 5:Thing Speak (IoT Platform)

## **WATER PUMP**

Fig-3:Water Level Sensor

Mini Pump, sensitive and light in weight made of light plastic, good suite for experiment and is usually used in aquarium, fish tanks and fountain etc. Works with 5V power supply.

Specifications:

* Rated Current: 100mA – 200mA
* Flow rate: 80-100L / H
* Wire Length: ~22 cm
* Mode driving: brushless cc design, magnetic driving

## **Introduction to DHT11 - The Engineering ProjectsDHT11**

Fig-4:Temperature Sensor

The digital sensor DHT11 is inexpensive ones created to measures temperature and humidity of the environment used in , designed combination of thermistor and capacitive humidity sensor. This sensor produces digital signal on the data port. It is designed with a 4.7K or 10K resistor.

Specifications:

* 3 – 5 Volts power with I/O
* Maximum current use of 2.5-microampere during conversion
* Suitable temperature readings range from 0 to 50°C with an accuracy of +- 2°C
* 1 Hz of maximum sampling rate per second
* Gadget dimensions {(15.5) x (12) x (5.5)}mm
* 4 pins spacing 0.1 inches
* RoHS compliant

**WATER LEVEL SENSOR**

*Fig-5: Water Level Sensor*

*Fig-5: Water Level Sensor*

*Fig-5: Water Level Sensor*

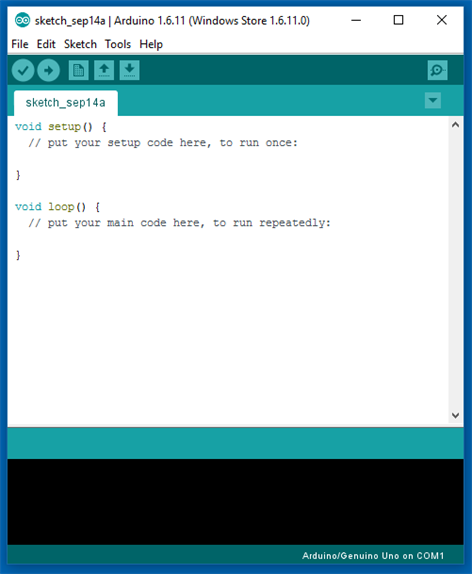
*Fig-5: Water Level Sensor*

A float switch/Water level sensor is a tool that provides the facility to monitor the liquid level in any type of container. It enables various responses like switching On/OFF of a pump, enabling an indicator, alarm sound, etc. Float switches are flexible and are used in hydroponic setups, water storage tanks, agricultural use etc.

Specifications:

* Its Maximum Load is 50W and Voltage is 100V DC with 0.5A of current.
* Range is -20°~ 80° C temperature
* Maximum Load current is 1A
* Maximum Contact Resistance is 0.4 ohms

**ARDUINO SOFTWARE (IDE)**



*Fig- 6:Arduino Software Window*

In the Arduino Software, program developed is called as sketch and is written in text editor. It must have file name followed by “.” and “ino".Terminal Editor enables users to cut, paste, replace, search the text. The message area shows errors. The console shows output with errors and other info. It is possible to verify, upload the developed programs. These Arduino program can also be opened, created from the toolbar and can be saved and serial terminal is also accessible from toolbar section.

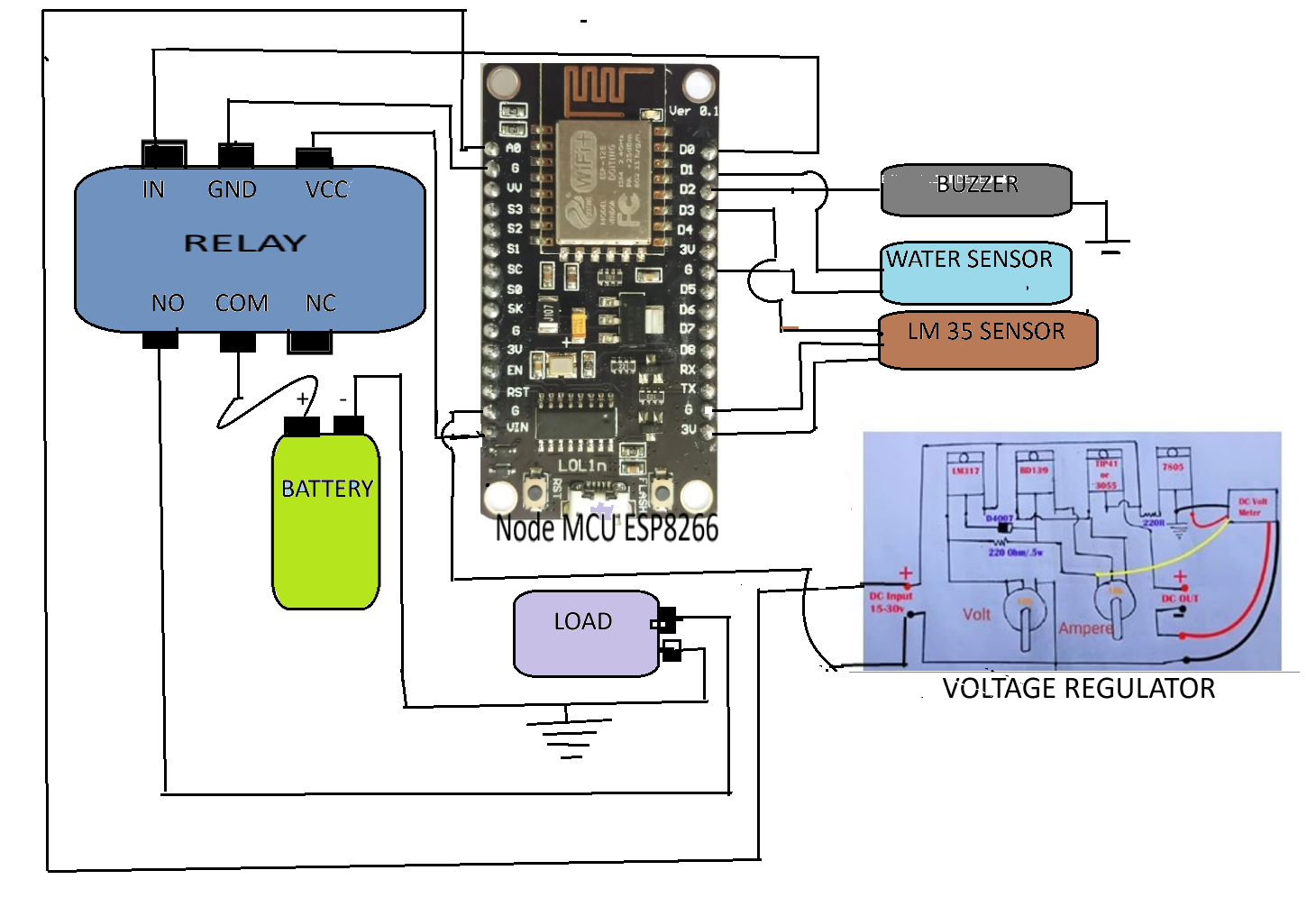
**IoT PLATFORM**

*Fig-7:Thing Speak (IoT Platform)*

In this project plan, Thing Speak is used as IoT platform. It is an IoT-applied, open-sourced API designed to store and retrieve data from connected devices by utilizing the “Hypertext Transfer across the global network of interconnected computers and devices or within a LAN”. ThingSpeak offers for the development of applications for logging sensor data and a network of connected devices with latest data."

**3.2 Circuit Diagram:**

Using the Hardware Components mentioned under section 3.1, the below circuit Diagram is designed and explained as follows:



Pump

*Fig-8 :Circuit Diagram of Intelligent Submersible Pump*

In the above provided Circuit Diagram , Node MCU ESP8266 pin D1 is connected to the “+” terminal of Water Level Sensors and its other terminal is connected to GND as shown .D2 is connected to “+” terminal of Buzzer and “–“ end terminal is grounded by connecting to G pin of Node MCU .

The DTH11 sensor has 3 pins one is VCC, other is data pin for data transfer( which senses factor from external environment ) is connected to D3 pin of Node MCU and GND pin of sensor is connected to G pin of the MCU i.e GND pin. T

he 5V Relay has 6 terminals. Signal pin, VCC pin and GND pin of relay is connected to D0 pin for Relay controlling, VIN pin for 12V supply and G pin of Node MCU respectively.

The Battery’s “+” terminal is connected to the Common Contact of the Relay’s pin and NC pin is in earlier stages connected to the COM pin, forming a closed circuit. However, this connection is down broken once the relay is triggered by a high/low signal sent to the signal pin of the relay from a microcontroller. The load (here aquarium pump) “+” terminal is connected to NO pin of relay. The “-” terminals of Battery and Load are grounded.

The Voltage Regulator “+” pin is connected to A0 pin of MCU for providing voltage read to MCU and the “-” side of the relay is grounded by connecting to GND pin of MCU.

**3.3 Program Design**

The below Software is programmed in Arduino programming language in Ardino Software by selecting Port “COM 8” and Board “Node MCU 1.0 (ESP -12E Module)” ,compiled and flashed into Node MCUESP 8266.

Program:

#include <DHT.h>

#include <Arduino.h>

#if defined(ESP32)

#include <WiFi.h>

#elif defined(ESP8266)

#include <ESP8266WiFi.h>

#endif

#include <ESP\_Mail\_Client.h>

#include <ESP8266HTTPClient.h>

#define DHTPIN 0 //D3

#define DHTTYPE DHT11 //DHT 11

#define BUZZER 4 //D2

#define WATER\_FLOAT 5 //D1

#define SMTP\_HOST "smtp.gmail.com"

#define SMTP\_PORT 465

/\* The sign in credentials \*/

#define AUTHOR\_EMAIL "smartagri2024@gmail.com"

#define AUTHOR\_PASSWORD "vagbpnnthmwwscin"/\* Recipient's email\*/

#define RECIPIENT\_EMAIL "Sanjanakulkarni1995@gmail.com"

/\* The SMTP Session object used for Email sending \*/

SMTPSession smtp;

static int CNT=0;

void smtpCallback(SMTP\_Status status);

const int channelID = 2393731;

String apiKey = "HKDRT1ROAHCYAX4Q"; // Enter your Write API key from ThingSpeak

const char \*ssid = "Narasimha"; // replace with your wifi ssid and wpa2 key

const char \*pass = "ssjnk@106";

const char\* server = "api.thingspeak.com";

unsigned long lastConnectionTime = 0;

const unsigned long postingInterval = 15L \* 1000L; // Post data every 15 seconds.

const int checkTalkBackInterval = 2 \* 1000; // Time interval in milliseconds to check TalkBack (number of seconds \* 1000 = interval)

float t,h;

float voltage;

int float\_stat;

const int voltagePin = A0; // Analog input pin for the voltage sensor

const int relayPin = 16; // Digital pin D0 connected to the relay module

const float low\_threshold\_Voltage = 2.0; // Set your threshold low voltage level here

const float high\_threshold\_Voltage = 4.0; // Set your threshold high voltage level here

WiFiClient client; // Initialize the Wifi client library.

DHT dht(DHTPIN, DHTTYPE);

void setup()

{

Serial.begin(9600);

pinMode(voltagePin, INPUT);

pinMode(relayPin, OUTPUT);

digitalWrite(relayPin, LOW); // Initialize relay to an open state

pinMode(BUZZER,OUTPUT);

pinMode(WATER\_FLOAT,INPUT\_PULLUP);

digitalWrite(BUZZER,LOW);

dht.begin();

Serial.println("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}

void loop()

{

static int talkback\_stat=0;

voltage = analogRead(voltagePin) \* (5.0 / 1023.0); // Read and convert analog input to voltage

//Serial.print(analogRead(voltagePin));

Serial.print("\nVoltage: ");

Serial.println(voltage);

t = dht.readTemperature();

h = dht.readHumidity();

float\_stat=digitalRead(WATER\_FLOAT);

Serial.print("\nTemperature:");

Serial.print(t);

Serial.print("\tHumidity:");

Serial.print(h);

Serial.print("\nWater float:");

Serial.print(float\_stat);

talkback\_stat=check\_water\_pump\_Control();

if(talkback\_stat==1)

{

digitalWrite(relayPin, HIGH);

}

else if(talkback\_stat==0)

{

digitalWrite(relayPin, LOW);

}

if (voltage < low\_threshold\_Voltage || voltage > high\_threshold\_Voltage)

{

digitalWrite(relayPin, LOW); // Turn off the circuit by activating the relay

digitalWrite(BUZZER, HIGH);

delay(1000);

digitalWrite(BUZZER, LOW);

send\_Vtg\_Alert\_mail();

}

if(float\_stat==1)

{

Serial.print("\nWater Level is Low in tank");

digitalWrite(relayPin, LOW);

digitalWrite(BUZZER, HIGH);

delay(1000);

digitalWrite(BUZZER, LOW);

send\_Water\_level\_Alert\_mail();

}

if(t>32 && t<45)

{

Serial.print("\nWinding Temperature is high");

digitalWrite(relayPin, LOW);

digitalWrite(BUZZER, HIGH);

delay(1000);

digitalWrite(BUZZER, LOW);

send\_Temp\_Alert\_mail();

}

send\_data();

delay(500);

}

void send\_data()

{

if (client.connect(server,80)) // "184.106.153.149" or api.thingspeak.com

{

String postStr = apiKey;

postStr +="&field1=";

postStr += String(t);

postStr +="&field2=";

postStr += String(h);

postStr +="&field3=";

postStr += String(float\_stat);

postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(postStr.length());

client.print("\n\n");

client.print(postStr);

delay(postingInterval);

}

}

/\* Callback function to get the Email sending status \*/

void smtpCallback(SMTP\_Status status)

{

/\* Print the current status \*/

Serial.println(status.info());

/\* Print the sending result \*/

if (status.success())

{

//Serial.println("----------------");

ESP\_MAIL\_PRINTF("Message sent success: %d\n", status.completedCount());

ESP\_MAIL\_PRINTF("Message sent failled: %d\n", status.failedCount());

Serial.println("----------------\n");

struct tm dt;

for (size\_t i = 0; i < smtp.sendingResult.size(); i++)

{

/\* Get the result item \*/

SMTP\_Result result = smtp.sendingResult.getItem(i);

time\_t ts = (time\_t)result.timestamp;

localtime\_r(&ts, &dt);

ESP\_MAIL\_PRINTF("Message No: %d\n", i + 1);

ESP\_MAIL\_PRINTF("Status: %s\n", result.completed ? "success" : "failed");

// ESP\_MAIL\_PRINTF("Date/Time: %d/%d/%d %d:%d:%d\n", dt.tm\_year + 1953, dt.tm\_mon + 4, dt.tm\_mday+3, dt.tm\_hour, dt.tm\_min, dt.tm\_sec);

ESP\_MAIL\_PRINTF("Recipient: %s\n", result.recipients.c\_str());

ESP\_MAIL\_PRINTF("Subject: %s\n", result.subject.c\_str());

}

Serial.println("----------------\n");

}

}

void send\_Water\_level\_Alert\_mail()

{

/\*\* Enable the debug via Serial port

\* none debug or 0

\* basic debug or 1

\*/

smtp.debug(1);

/\* Set the callback function to get the sending results \*/

smtp.callback(smtpCallback);

/\* Declare the session config data \*/

ESP\_Mail\_Session session;

/\* Set the session config \*/

session.server.host\_name = SMTP\_HOST;

session.server.port = SMTP\_PORT;

session.login.email = AUTHOR\_EMAIL;

session.login.password = AUTHOR\_PASSWORD;

session.login.user\_domain = "";

/\* Declare the message class \*/

SMTP\_Message message;

/\* Set the message headers \*/

message.sender.name = "Alert Mail";

message.sender.email = AUTHOR\_EMAIL;

message.subject = "Tank Water Level Alert";

message.addRecipient("Client", RECIPIENT\_EMAIL);

/\*Send HTML message\*/

//String htmlMsg = "<div style=\"color:#2f4468;\"><h1>Temperature crossed threshold</h1></div>"+String("Temperature:")+String(t)+String("°C")+String("<p>Humidity:")+String(h)+String("%")+String("<p>- Sent from ESP8266 board</p>");

String htmlMsg = "<div style=\"color:pink;\"><h1>Water level is Low</h1></div>"+String("<p>- Sent from ESP8266 board</p>");

message.html.content = htmlMsg.c\_str();

message.html.content = htmlMsg.c\_str();

message.text.charSet = "us-ascii";

message.html.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

/\*

//Send raw text message

String textMsg = "Hello World! - Sent from ESP8266 board";

message.text.content = textMsg.c\_str();

message.text.charSet = "us-ascii";

message.text.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

message.priority = esp\_mail\_smtp\_priority::esp\_mail\_smtp\_priority\_low;

message.response.notify = esp\_mail\_smtp\_notify\_success | esp\_mail\_smtp\_notify\_failure | esp\_mail\_smtp\_notify\_delay;\*/

/\* Set the custom message header \*/

//message.addHeader("Message-ID: <abcde.fghij@gmail.com>");

/\* Connect to server with the session config \*/

if (!smtp.connect(&session))

return;

/\* Start sending Email and close the session \*/

if (!MailClient.sendMail(&smtp, &message))

Serial.println("Error sending Email, " + smtp.errorReason());

}

void send\_Temp\_Alert\_mail()

{

/\*\* Enable the debug via Serial port

\* none debug or 0

\* basic debug or 1

\*/

smtp.debug(1);

/\* Set the callback function to get the sending results \*/

smtp.callback(smtpCallback);

/\* Declare the session config data \*/

ESP\_Mail\_Session session;

/\* Set the session config \*/

session.server.host\_name = SMTP\_HOST;

session.server.port = SMTP\_PORT;

session.login.email = AUTHOR\_EMAIL;

session.login.password = AUTHOR\_PASSWORD;

session.login.user\_domain = "";

/\* Declare the message class \*/

SMTP\_Message message;

/\* Set the message headers \*/

message.sender.name = "Temperature Alert Mail";

message.sender.email = AUTHOR\_EMAIL;

message.subject = "Winding Temperature is high";

message.addRecipient("Client", RECIPIENT\_EMAIL);

/\*Send HTML message\*/

String htmlMsg = "<div style=\"color:pink;\"><h1>Temperature crossed threshold</h1></div>"+String("Temperature:")+String(t)+String("°C")+String("<p>Humidity:")+String(h)+String("%")+String("<p>- Sent from ESP8266 board</p>");

message.html.content = htmlMsg.c\_str();

message.html.content = htmlMsg.c\_str();

message.text.charSet = "us-ascii";

message.html.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

/\*

//Send raw text message

String textMsg = "Hello World! - Sent from ESP8266 board";

message.text.content = textMsg.c\_str();

message.text.charSet = "us-ascii";

message.text.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

message.priority = esp\_mail\_smtp\_priority::esp\_mail\_smtp\_priority\_low;

message.response.notify = esp\_mail\_smtp\_notify\_success | esp\_mail\_smtp\_notify\_failure | esp\_mail\_smtp\_notify\_delay;\*/

/\* Set the custom message header \*/

//message.addHeader("Message-ID: <abcde.fghij@gmail.com>");

/\* Connect to server with the session config \*/

if (!smtp.connect(&session))

return;

/\* Start sending Email and close the session \*/

if (!MailClient.sendMail(&smtp, &message))

Serial.println("Error sending Email, " + smtp.errorReason());

}

void send\_Vtg\_Alert\_mail()

{

/\*\* Enable the debug via Serial port

\* none debug or 0

\* basic debug or 1

\*/

smtp.debug(1);

/\* Set the callback function to get the sending results \*/

smtp.callback(smtpCallback);

/\* Declare the session config data \*/

ESP\_Mail\_Session session;

/\* Set the session config \*/

session.server.host\_name = SMTP\_HOST;

session.server.port = SMTP\_PORT;

session.login.email = AUTHOR\_EMAIL;

session.login.password = AUTHOR\_PASSWORD;

session.login.user\_domain = "";

/\* Declare the message class \*/

SMTP\_Message message;

/\* Set the message headers \*/

message.sender.name = "Voltage Alert Mail";

message.sender.email = AUTHOR\_EMAIL;

message.subject = "Voltage fluctuating";

message.addRecipient("Client", RECIPIENT\_EMAIL);

/\*Send HTML message\*/

String htmlMsg = "<div style=\"color:Black;\"><h1>Voltage is fluctuating</h1></div>"+String("Voltage:")+String(voltage)+String("<p>- Sent from ESP8266 board</p>");

message.html.content = htmlMsg.c\_str();

message.html.content = htmlMsg.c\_str();

message.text.charSet = "us-ascii";

message.html.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

/\*

//Send raw text message

String textMsg = "Hello World! - Sent from ESP8266 board";

message.text.content = textMsg.c\_str();

message.text.charSet = "us-ascii";

message.text.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

message.priority = esp\_mail\_smtp\_priority::esp\_mail\_smtp\_priority\_low;

message.response.notify = esp\_mail\_smtp\_notify\_success | esp\_mail\_smtp\_notify\_failure | esp\_mail\_smtp\_notify\_delay;\*/

/\* Set the custom message header \*/

//message.addHeader("Message-ID: <abcde.fghij@gmail.com>");

/\* Connect to server with the session config \*/

if (!smtp.connect(&session))

return;

/\* Start sending Email and close the session \*/

if (!MailClient.sendMail(&smtp, &message))

Serial.println("Error sending Email, " + smtp.errorReason());

}

int check\_water\_pump\_Control()

{

int stat;

String talkBackAPIKey = "KMGEHQOEKWP3XZG4"; // Buzzer Control TalkBack API key

String talkBackID = "50997"; // Buzzer control TalkBack ID

HTTPClient http;

String thingSpeakAPI = "api.thingspeak.com";

char charIn;

String talkBackURL = "http://" + thingSpeakAPI + "/talkbacks/" + talkBackID + "/commands/execute?api\_key=" + talkBackAPIKey;

// Make a HTTP GET request to the TalkBack API:

http.begin(client,talkBackURL);

int httpCode = http.GET();

if (httpCode > 0)

{

String talkBackCommand = http.getString(); //Get the request response payload

//Serial.print("\nTalkBackCommand=");

Serial.print(talkBackCommand.c\_str()); //Print the response payload

http.end(); //Close connection

if (!strcmp(talkBackCommand.c\_str(), "TURN\_ON"))

{

//digitalWrite(LED\_BUILTIN, LOW);

stat= 1;

}

else if (!strcmp(talkBackCommand.c\_str(), "TURN\_OFF"))

{

//digitalWrite(LED\_BUILTIN, HIGH);

stat= 0;

}

else

{

Serial.print("");

}

Serial.flush();

delay(1000);

}

return stat;

}

The coding is done by keeping the following constraints noted :

When Voltage falls below the limit or if it exceeds the limit, the Pump/Load is switched off. Thus saving the pump during voltage issues.

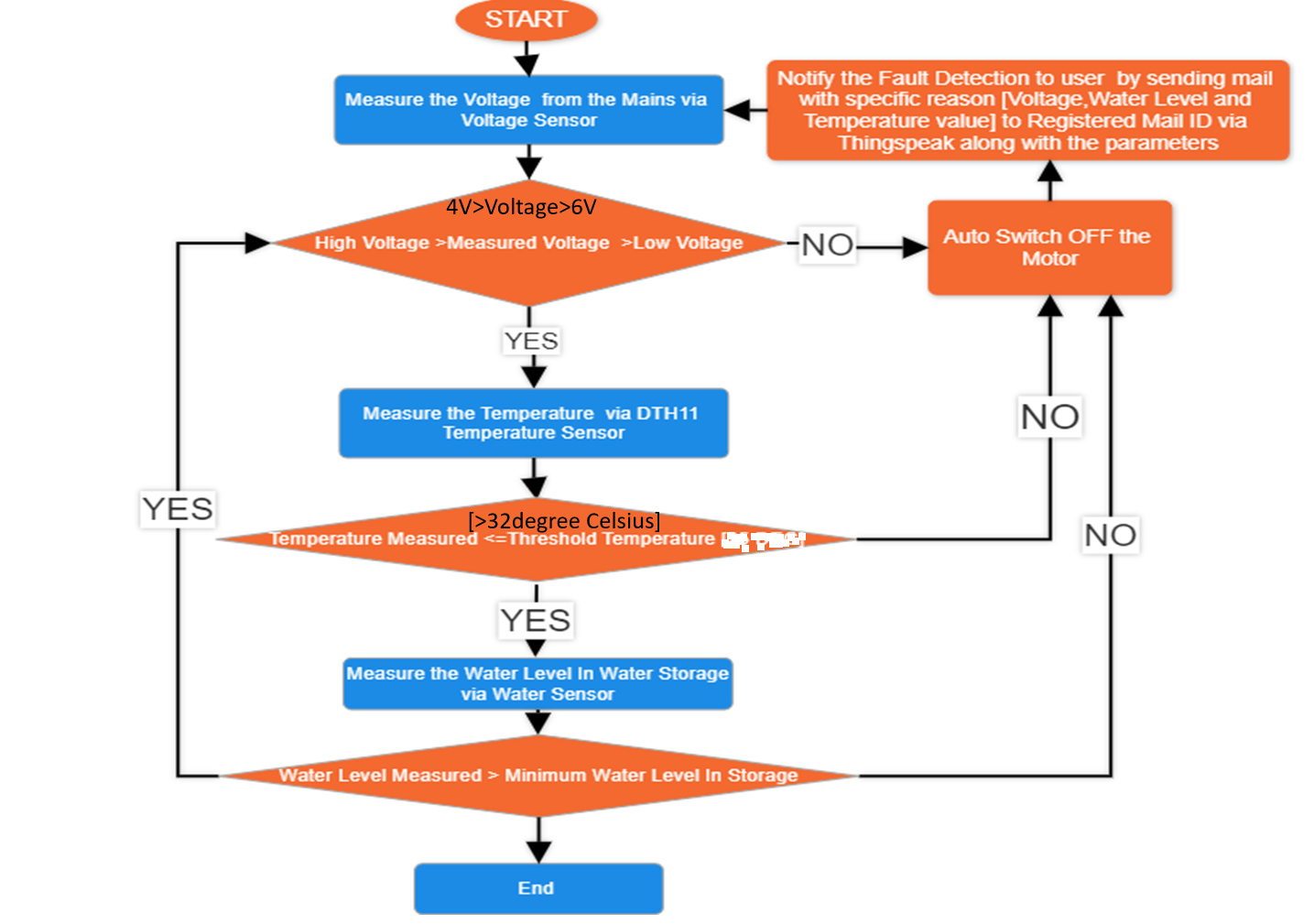
When the DHT11 temperature senses the value more than 32 degree Celsius, the Pump/Load is switched off.

In this experiment, the water level is monitored. When the water level in the tank, well, or in any storage falls below the minimum limit, the water level sensor notifies on this and the pump will be switched off with the help of a relay as the relay will be controlled by MC and program flashed into MCU.

In all the 3 constraints, the user will be notified with the beep alarm i.e with the buzzer. The cause for the fault will be mentioned and will be sent to the registered mail ID with the help of the IoT platform “ThingSpeak”.

With the help of “ThingSpeak,” we can turn the motor automatically ON or OFF.

Below is the flowchart depicting the project concept and its work flow:

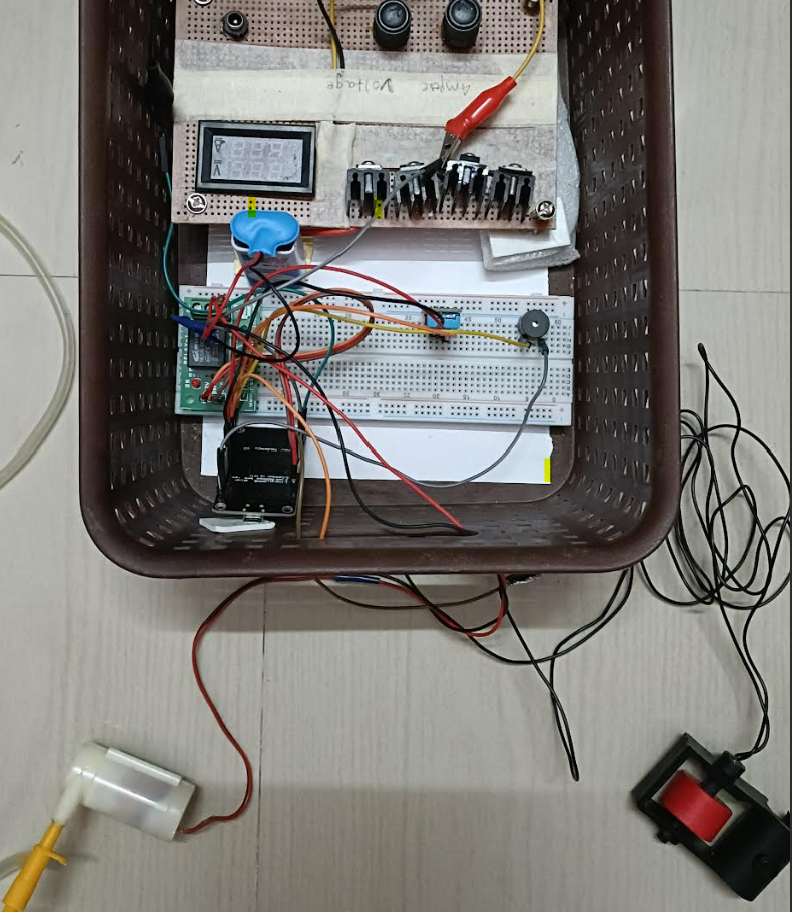


*Fig- 9: Flow Chart of the Intelligent Submersible Pump*

**CHAPTER 4: RESULTS & DISCUSSION**

The proposed project aims to design and develop an intelligent submersible pump for irrigation purposes using IoT for the operation of the motor using Sensors. The system is controlled using Node MCU ESP8266 system. The developed system is programmed to work in both automatic and manual mode to turn on/off motor using a IoT and sensors for easy Operations.

Prototype Designed :



*Fig-10:Prototype of Intelligent Submersible Pump*

**Low Voltage Detection**

When the voltage falls below 2Volts i.e Low Voltage level, the system detects the fault and the pump switches off itself and sends a mail alert to the registered user mail id [sanjanakulkarni1995@gmail.com](mailto:sanjanakulkarni1995@gmail.com) via the ThingSpeak account Platform.

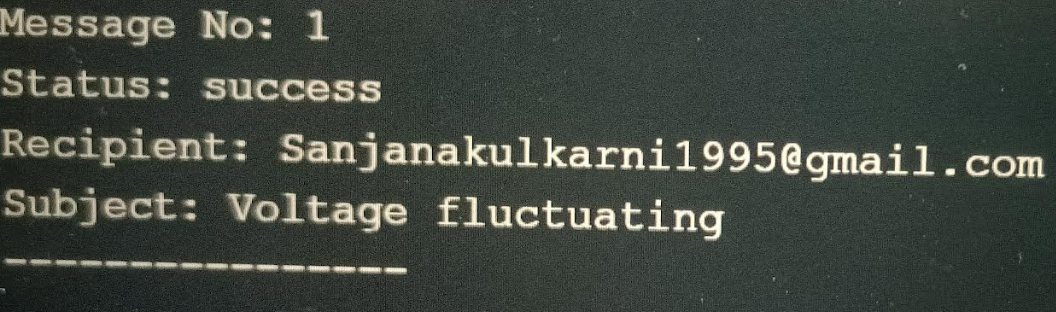


Figure 16:Voltage Fluctuating Output on Ardino Software tool

Figure 16:Voltage Fluctuating Output on Ardino Software tool

Figure 16:Voltage Fluctuating Output on Ardino Software tool

Figure 16:Voltage Fluctuating Output on Ardino Software tool

*Fig-11: Voltage Fluctuating Output on Ardino Software tool*

****

Figure 17:E-mail Alert send to mail ID indicating Low Voltage fault

Figure 17:E-mail Alert send to mail ID indicating Low Voltage fault

Figure 17:E-mail Alert send to mail ID indicating Low Voltage fault

Figure 17:E-mail Alert send to mail ID indicating Low Voltage fault

*Fig-12: E-mail Alert send to mail ID indicating Low Voltage fault*

*Fig-12: E-mail Alert send to mail ID indicating Low Voltage fault*

*Fig-12: E-mail Alert send to mail ID indicating Low Voltage fault*

*Fig-12: E-mail Alert send to mail ID indicating Low Voltage fault*

**High Voltage Detection**

When the voltage crosses above 4 Volts i.e High Voltage level, the system detects the fault and the pump switches off itself and sends a mail alert to the registered user mail id [sanjanakulkarni1995@gmail.com](mailto:sanjanakulkarni1995@gmail.com) via the ThingSpeak account Platform.

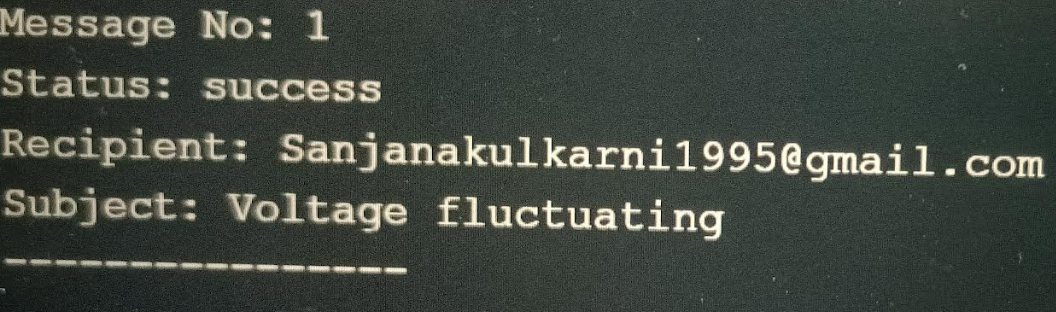


Fig-13:Voltage Fluctuating Output on Ardino Software tool

****

*Fig-14: E-mail Alert send to mail ID indicating High Voltage fault*

*Fig-14: E-mail Alert send to mail ID indicating High Voltage fault*

*Fig-14: E-mail Alert send to mail ID indicating High Voltage fault*

*Fig-14: E-mail Alert send to mail ID indicating High Voltage fault*

**High-Temperature Level Detection**

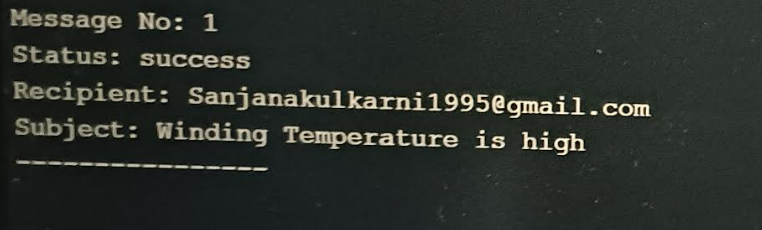
****When the Temperature crosses above 32 degree celsius i.e High Temperature level, the system detects the fault and the pump switches off itself and sends a mail “ alert to the registered user mail id [sanjanakulkarni1995@gmail.com](mailto:sanjanakulkarni1995@gmail.com) via the ThingSpeak account Platform.

Fig-15:High Winding Temperature Output on Arduino Software tool

****

Figure 21:E-mail Alert send to mail ID indicating High Temperature Alert

Figure 21:E-mail Alert send to mail ID indicating High Temperature Alert

Figure 21:E-mail Alert send to mail ID indicating High Temperature Alert

Figure 21:E-mail Alert send to mail ID indicating High Temperature Alert

Fig-16: E-mail Alert send to mail ID indicating High Temperature Alert

Fig-16: E-mail Alert send to mail ID indicating High Temperature Alert

Fig-16: E-mail Alert send to mail ID indicating High Temperature Alert

Fig-16: E-mail Alert send to mail ID indicating High Temperature Alert

**Water Level Detection**

When the water level falls below the minimum level in storage tanks, the system detects the minimum water level and the pump switches off thus saving the pump and sending a mail alert to the registered user mail id [sanjanakulkarni1995@gmail.com](mailto:sanjanakulkarni1995@gmail.com) via the ThingSpeak account Platform.

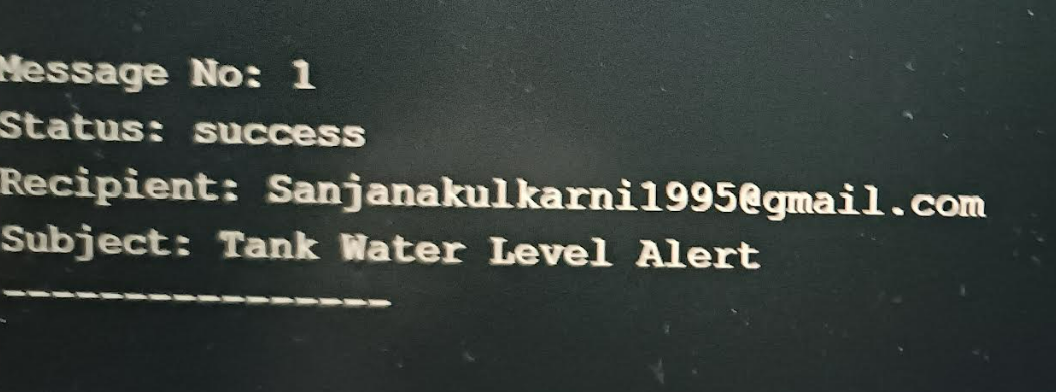


Figure 22:Water Level Alert on Ardino Software tool

Figure 22:Water Level Alert on Ardino Software tool

Figure 22:Water Level Alert on Ardino Software tool

Figure 22:Water Level Alert on Ardino Software tool

*Fig-17:Water Level Alert on Ardino Software tool*



Figure 23:Tank Water Level Alert sent to user mail ID

Figure 23:Tank Water Level Alert sent to user mail ID

Figure 23:Tank Water Level Alert sent to user mail ID

Figure 23:Tank Water Level Alert sent to user mail ID

*Fig-18: Tank Water Level Alert sent to user mail ID*

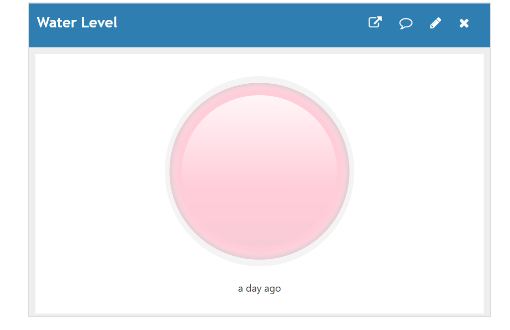
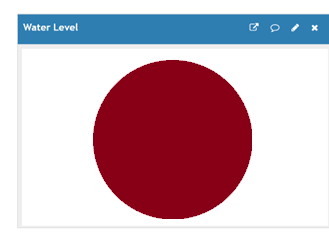
****

Figure 24:Colour Pink in the ThinkSpeak Platform indicates the Water Level is sufficient whereas Red color on the left indicates a low water level on the ThingSpeak Platform.

Figure 24:Colour Pink in the ThinkSpeak Platform indicates the Water Level is sufficient whereas Red color on the left indicates a low water level on the ThingSpeak Platform.

Figure 24:Colour Pink in the ThinkSpeak Platform indicates the Water Level is sufficient whereas Red color on the left indicates a low water level on the ThingSpeak Platform.

Figure 24:Colour Pink in the ThinkSpeak Platform indicates the Water Level is sufficient whereas Red color on the left indicates a low water level on the ThingSpeak Platform.

*Fig-19: Pink : sufficient water;red : low level on ThingSpeak*

*Fig-19:Colour Pink in the ThinkSpeak Platform indicates the Water Level is sufficient whereas Red color on the left indicates a low water level on the ThingSpeak Platform.*

**CHAPTER 5: CONCLUSION AND FUTURE WORK**

**5.1 Conclusions**

In this project, an intelligent protection and control system has been designed and developed for the protection of electric motors used for irrigation purposes using IoT and sensors. The prototype model has been developed which protects the motor against under-voltage faults, over-voltage faults, temperature faults, and dry runs. The project is designed for the detection of water levels, voltage levels, and temperature levels of submersible pumps and to perform safer operation and protection of the system against various faults. The usage of the Node MCU ESP8266 system provides information to the user about the operating condition of the pump and also to control the submersible pump through wireless mode. The project aims to work efficiently by using water level sensors, voltage level monitoring, and LM35 temperature sensors to reduce the interruptions for the working of submersible pumps. In this project, the motor used for irrigation purposes is automatically protected from various faults such as over-voltage, undervoltage, High temperature, and Dry run of motor. The operating conditions of the complete system will be received by the user with the help of Iot and ThingSpeak Platform.

**5.2 Future Work**

* Mobile applications can be developed for easier operations with advanced features.
* Advanced temperature sensors and motor vibration sensors can be used for high-rating pumps to detect mechanical faults in the pump.
* The idea for an autonomous pump system in the future with solar-powered irrigation systems, windmills, and natural energy usage systems that will not require any human intervention can be adopted.

1. **REFERENCES:**

[1] Nikolov, Neven, and Ognyan Nakov. "Research of communication between IoT cloud structure, android application and IoT device using TCP sockets." 2019 X National Conference with International Participation (ELECTRONICA). IEEE, 2019

[2] Thiyagarajan, M., and Chaitanya Raveendra. "Integration in the physical world in IoT using android mobile application." 2015 International Conference on Green Computing and Internet of Things (ICGCIoT). IEEE, 2015.

[3] Ram, Saswat Kumar, et al. "A solar based power module for battery-less IoT sensors towards sustainable smart cities." 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2020.

[4] Brinner, Thomas R., Robert H. McCoy, and Trevor Kopecky. "Induction versus permanent-magnet motors for electric submersible pump field and laboratory comparisons." IEEE Transactions on Industry Applications 50.1 (2013): 174-181.

[5] Hamblen, James O., Zachery C. Smith, and Winne W. Woo. "Introducing embedded systems in the first C/C++ programming class." 2013 IEEE International Conference on Microelectronic Systems Education (MSE). IEEE, 2013.

[6] Ji, Xiaoke, et al. "Application Research of submersible diaphragm pump based on nonmetal coiled tubing." 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP). IEEE, 2021.

[7] Rabbi, S. F., et al. "Modeling and performance evaluation of a hysteresis IPM motor drive for electric submersible pumps." 2015 IEEE Energy Conversion Congress and Exposition (ECCE). IEEE, 2015.

[8] Abid, Ghulam, et al. "IOT based Smart Industrial panel for controlling Three-phase Induction motor." 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). IEEE, 2020.

[9] Surya, Gulamfaruk N., Makarand S. Ballal, and Zafar J. Khan. "Application of observer coil technique for detection of stator turn fault and damaged bearing in cage motors." 2017 National Power Electronics Conference (NPEC). IEEE, 2017.

[10] Wu, Fan, Taiyang Wu, and Mehmet Rasit Yuce. "Design and implementation of a wearable sensor network system for IoT-connected safety and health applications." 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). IEEE, 2019.

[11] Vishwakarma, Satyendra K., et al. "Smart energy efficient home automation system using IoT." 2019 4th international conference on internet of things: Smart innovation and usages (IoT-SIU). Ieee, 2019.

[12] Suresh, M., M. S. Anbarasi, and V. PraveenKumar. "An intelligent smart street light system with predictive model." 2020 International Conference on System, Computation, Automation and Networking (ICSCAN). IEEE, 2020.

[13] Ismail, Shereen, et al. "IoT-Based Water Management Systems: Survey and Future Research Direction." IEEE Access 10 (2022): 35942-35952.

[14] Reddy, Peddamallu Jaya Prakash, Guntimadugu Viswanadh, and Sanjay Kumar Singh. "IOT based Smart Water Pump Switch." 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM). IEEE, 202.

[15] Kulkarni, Sandhya A., et al. "Intelligent water level monitoring system using IoT." 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC). IEEE, 2020.

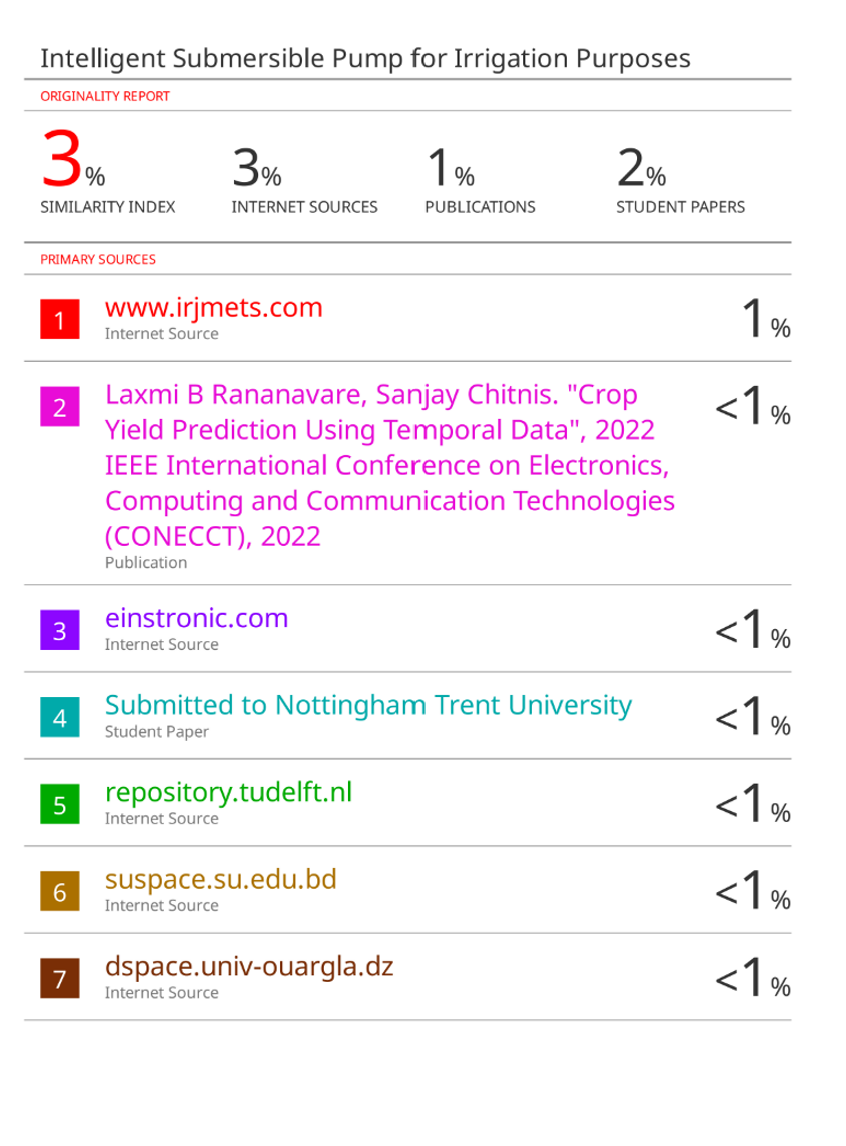
**APPENDIX**

**GMAIL ACCEPTANCE LETTER**

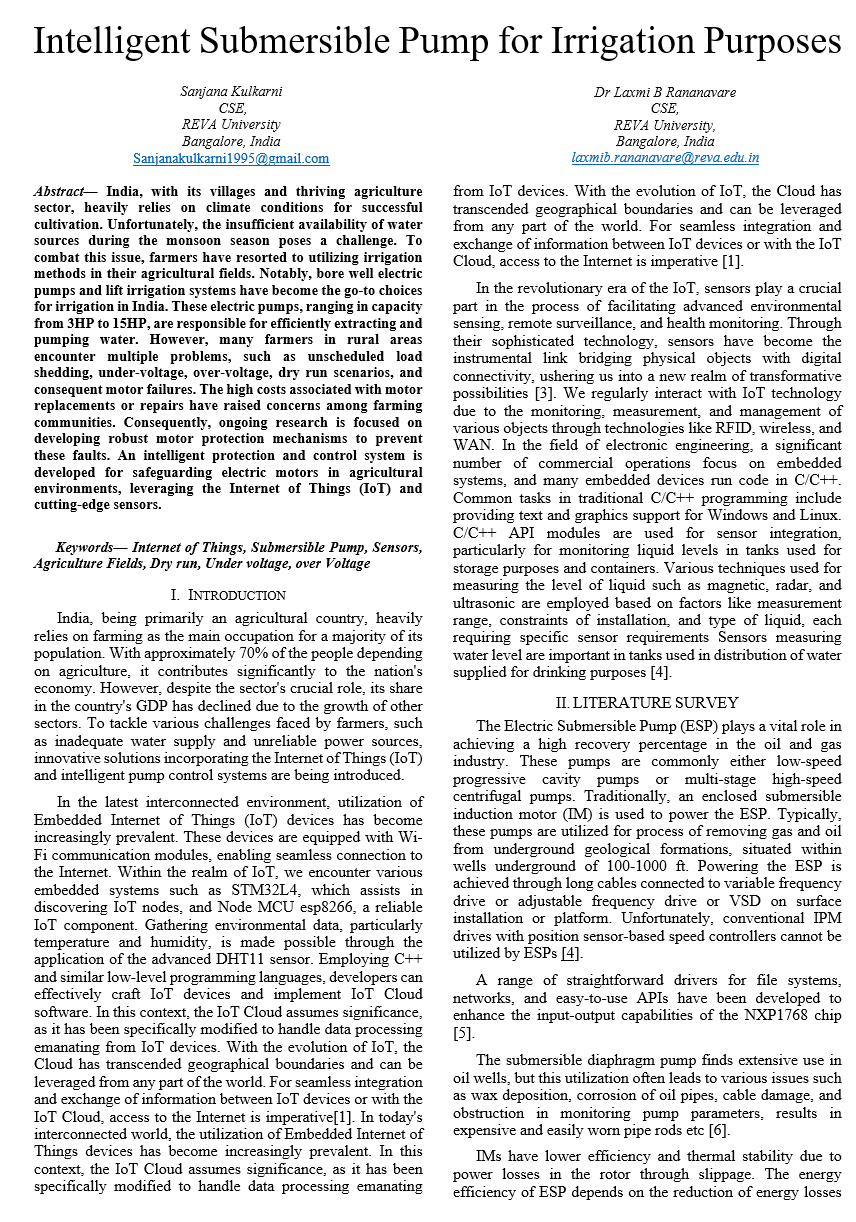




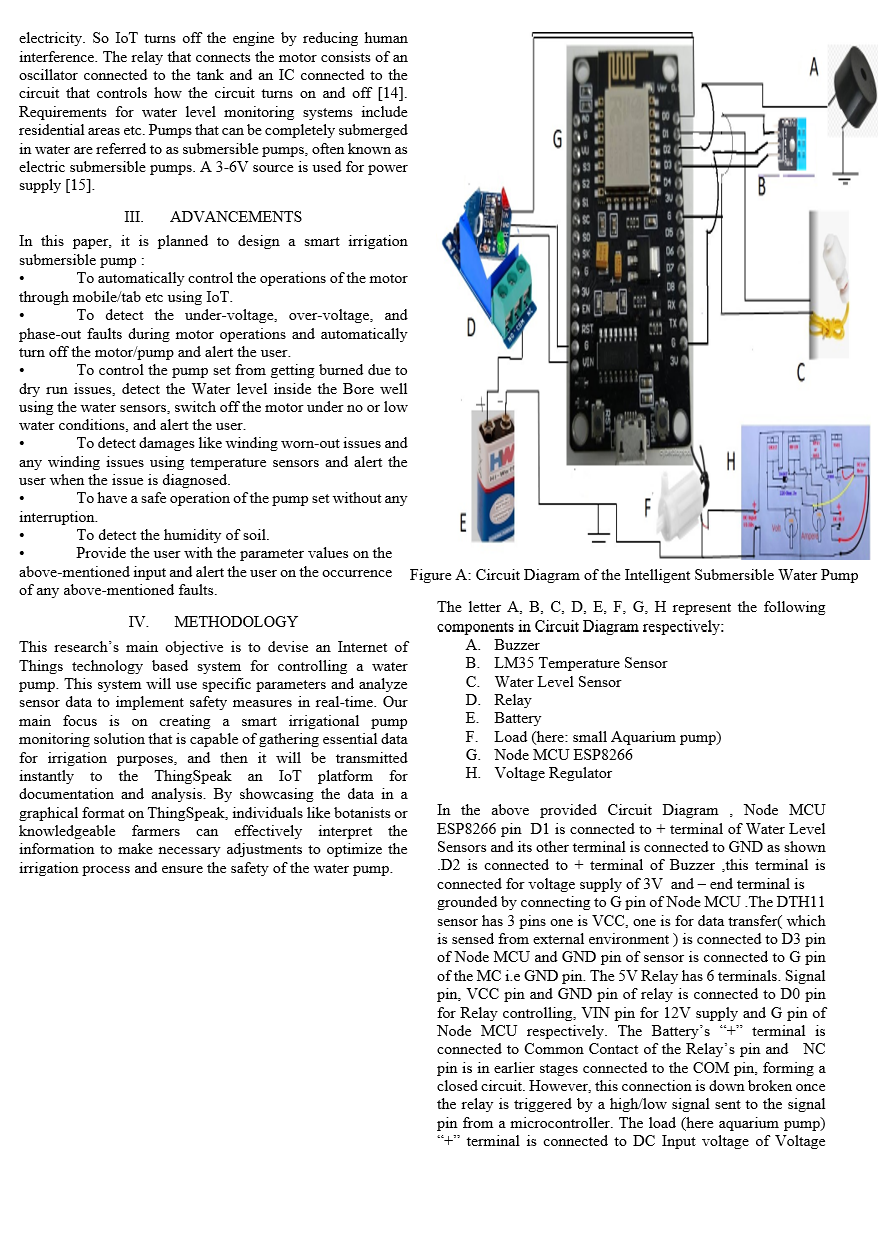
**THESIS PLAGIARISM CHECK REPORT**

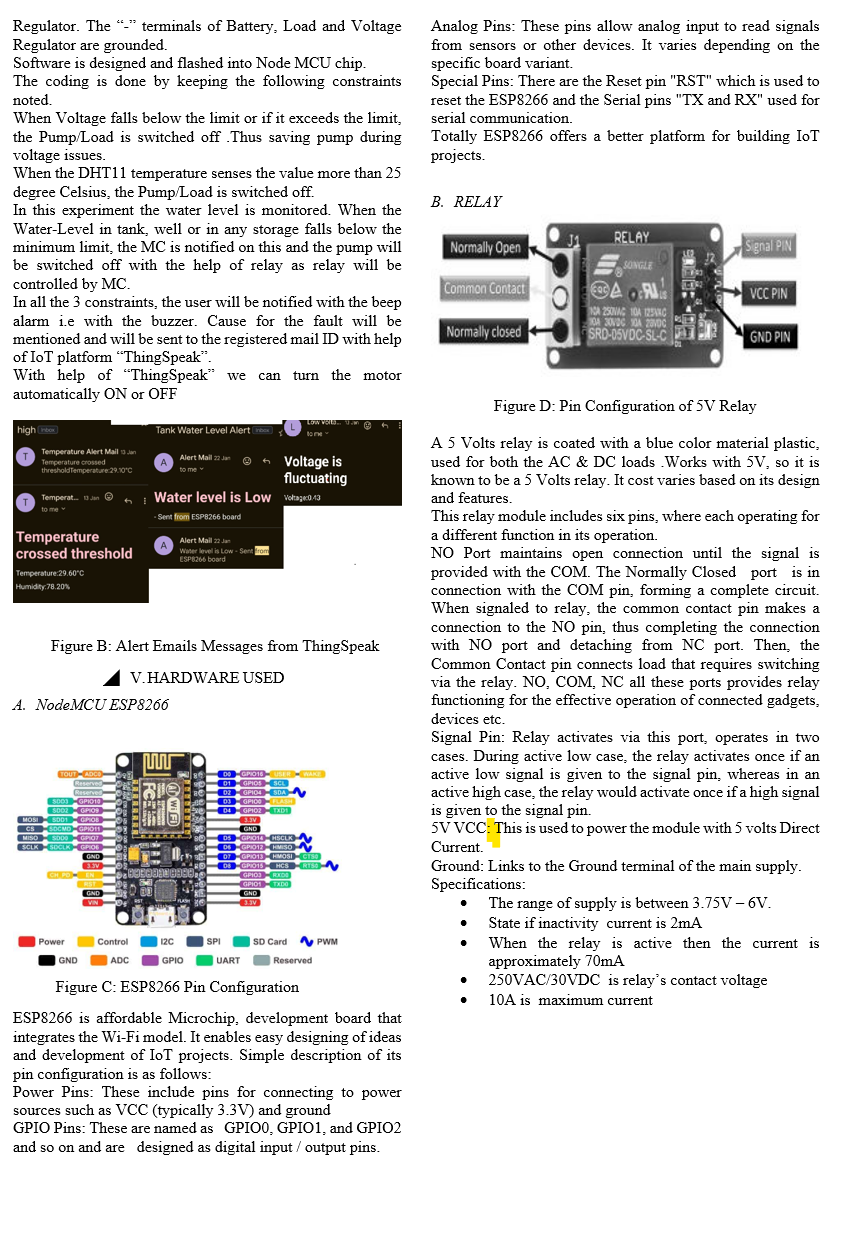
****

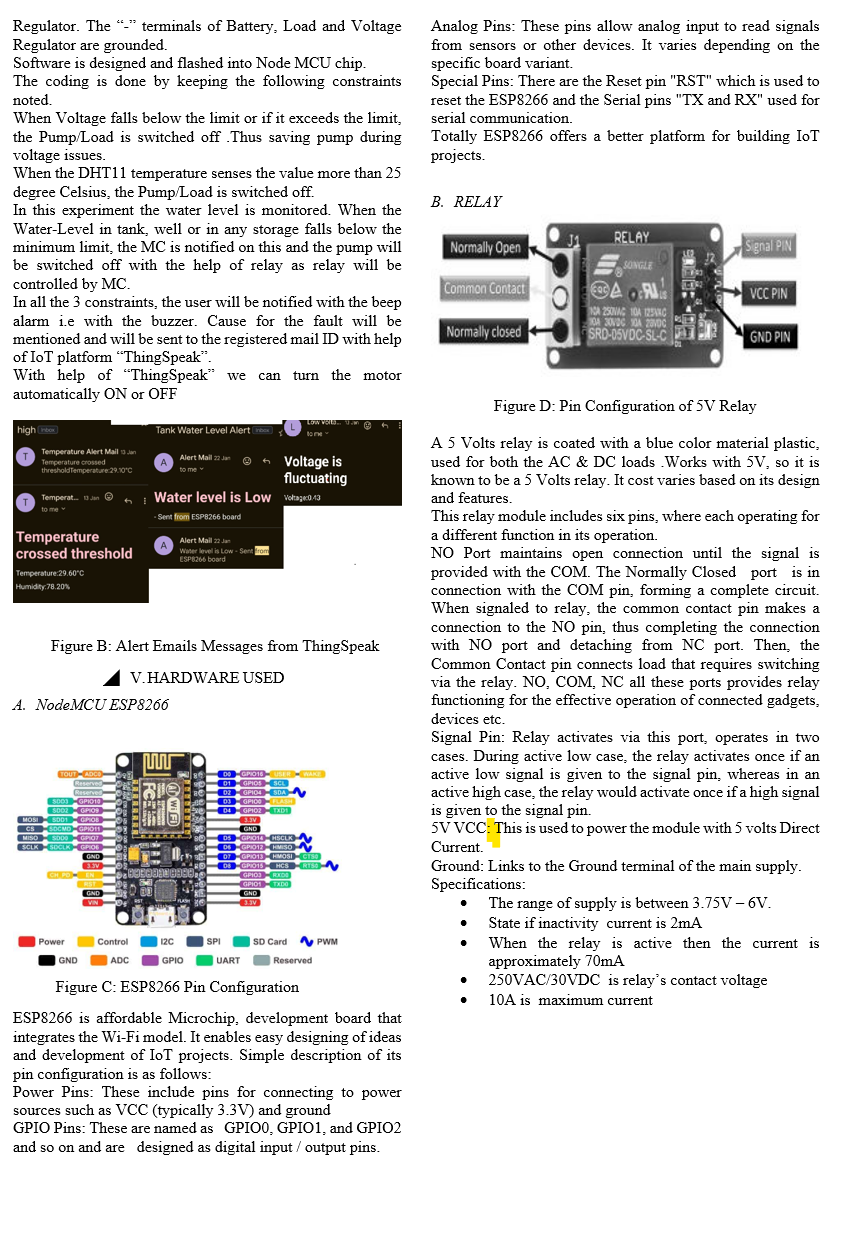
**MY PAPER**

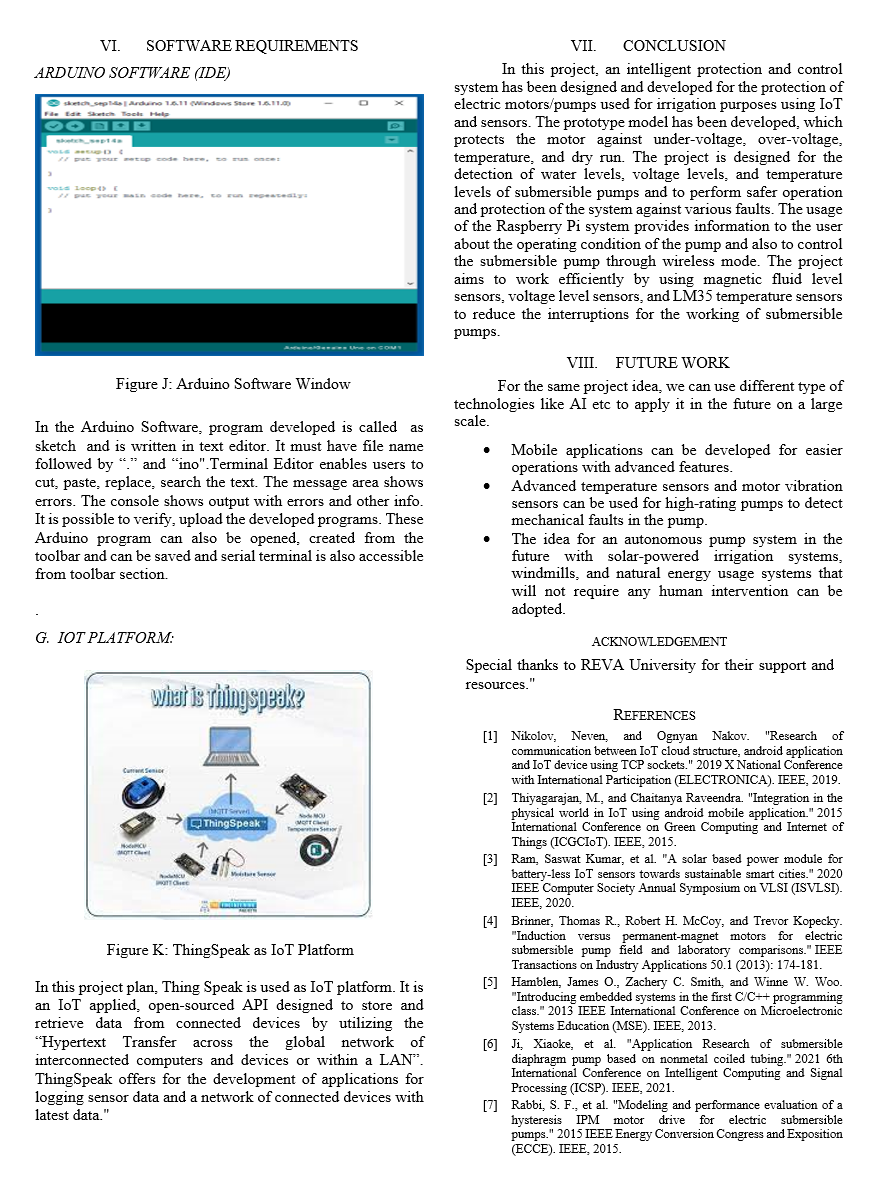
****

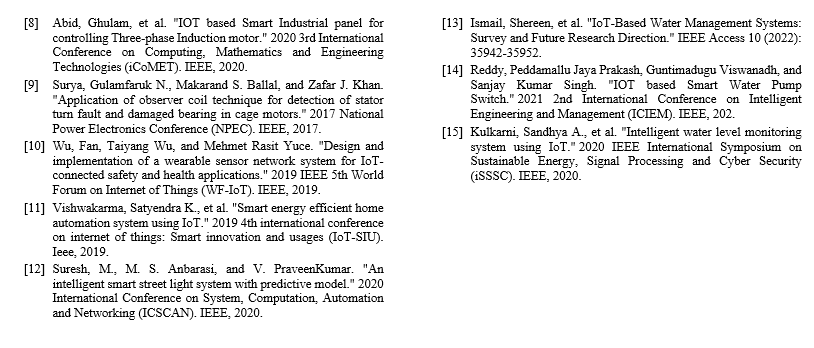
****

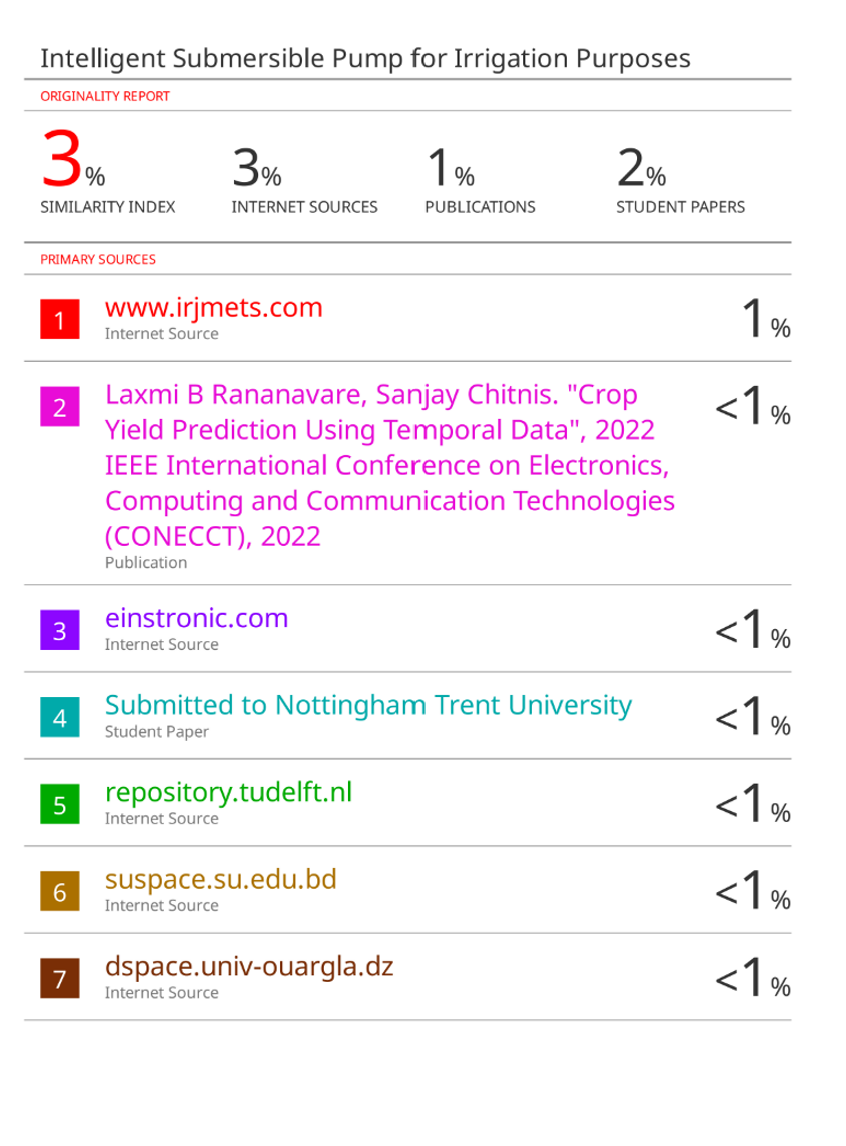
****

****

****

****

****

**PAPER PLAGIARISM CHECK REPORT**