A

# PROJECT REPORT

ON

## "IOT BASED AQUACULTURE MONITORING SYSTEM"

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

## BACHELOR OF TECHNOLOGY

## IN

## ELECTRONICS AND COMMUNICATION ENGINEERING

Under the guidance of

**Dr. N. K. SHARMA** (Head of Department)

(Electronics &Communication Engineering Department)


**SUBMITTED BY:**

| | |
|---|---|
| Kishan Dixit | CT-2928/19 |
| Shrasti Bhadauria | CT-2929/19 |
| Areeba Irfan | CT-2930/19 |
| Ashish Yadav | CT-2931/19 |
| Nandani Chaudhary | CT-2933/19 |

**Baba Saheb Dr. B. R. Ambedkar College of Agricultural Engineering & Technology,**

**( A Faculty of Chandra Shekhar Azad University of Agriculture & Technology, Kanpur)**

**Campus-Etawah**


**Session 2022-2023**

# CERTIFICATE

This is to certify that the project entitled **"IOT BASED AQUACULTURE MONITORING SYSTEM"** submitted to **Baba Saheb Dr. Bhim Rao Ambedkar College of Agricultural Engineering and Technology, Etawah** in the partial fulfillment of the requirement of award of the degree of Bachelor of Technology in Electronics and Communication Engineering is Bonafide record of project report carried out by **Kishan Dixit (CT-2928/19), Shrasti Bhadauria (CT-2929/19), Areeba Irfan (CT-2930/19), Ashish Yadav (CT-2931/19), Nandani Chaudhary (CT-2933/19).** The project is recommended for the acceptance. The embodies result of original work carried out by the candidates under my supervision and guidance, and that no part has submitted for any degree or diploma. The technical guidance is provided by the faculty available in the department in the course of study.

I hereby approve its submission for above mentioned degree.

**Dr. N. K. Sharma**
**In charge of**
**E. & C.E. Department**

# ACKNOWLEDGEMENT

In the accomplishment of completion of our project on **"IOT BASED AQUACULTURE MONITORING SYSTEM"**. We would like to convey our special gratitude to **Dr. N. K. Sharma** who guided us with their valuable help in channeling our efforts into the right direction during the completion of the project. We are thankful to our **ECE faculty** for his technical guidance to us. Your valuable guidance and suggestion helped us in various phases of the completion of this project. We will always be thankful to you in this regard. We are highly grateful to the Dean **Dr. N. K. Sharma** and Hon'ble Vice Chancellor **Dr. A. Singh** for providing encouragement. In Addition, we thankfully acknowledge the help we received from all faculty members of Electronics & Communication Engineering, CAET, Etawah.

DATE:

PLACE:

**Kishan Dixit (CT-2928/19)**

**Shrasti Bhadauria (CT-2929/19)**

**Areeba Irfan (CT-2930/19)**

**Ashish Yadav (CT-2931/19)**

**Nandani Chaudhary (CT-2933/19)**

# ABSTRACT

A comprehensive solution created especially for B2B applications in the aquaculture sector is the IoT-based Aquaculture Monitoring System. It makes use of the Internet of Things (IoT) to manage, control, and monitor aquaculture operations in real-time, enhancing sustainability, production, and efficiency. The system integrates a network of sensors and smart devices deployed throughout aquaculture facilities, including fish farms, hatcheries, and processing plants. These IoT-enabled devices collect and transmit valuable data on various environmental parameters such as water quality, temperature, pH levels, and feed management. The collected data is securely transmitted to a centralized cloud-based platform for storage, analysis, and visualization. Through the cloud-based platform, aquaculture operators and stakeholders gain access to a user-friendly dashboard that provides real-time insights into the crucial metrics affecting the health and performance of aquatic organisms. This allows for proactive decision-making, timely interventions, and optimized resource allocation. Furthermore, the system incorporates advanced analytics and machine learning algorithms to generate predictive models and actionable recommendations. By leveraging historical data, the system can identify trends, patterns, and potential risks, enabling proactive measures to prevent disease outbreaks, optimize feeding schedules, and improve overall operational efficiency. The IoT-based Aquaculture Monitoring System offers several benefits to B2B stakeholders in the aquaculture industry. It enhances production yields by enabling precise and optimal control of water quality parameters, minimizing stress on aquatic organisms, and promoting their growth. The system also aids in resource optimization by reducing feed wastage, energy consumption, and operational costs.

**Keywords**—Internet of Things (IoT), Aquaculture, Water Quality Monitoring, Arduino, pH, Temperature, Sensors, Wi-Fi, Internet, Smartphone.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF ABBREVIATION

| ABBREVIATION | DETAILS |
|---|---|
| ***IoT*** | Internet of Things |
| ***AI*** | Artificial Intelligence |
| ***ML*** | Machine Learning |
| ***LPWAN*** | Low-Power Wide-Area Network |
| ***MQTT*** | Message Queuing Telemetry Transport |
| ***LoRa*** | Long Range |
| ***NB-IoT*** | Narrowband Internet of Things |
| ***CO2*** | Carbon Dioxide |
| ***O2*** | Oxygen |
| ***pH*** | Potential of Hydrogen |
| ***EC*** | Electrical Conductivity |
| ***TDS*** | Total Dissolved Solids |
| ***DO*** | Dissolved Oxygen |
| ***LED*** | Light Emitting Diode |
| ***PIR*** | Passive Infrared Sensor |
| ***ADC*** | Analog-to-Digital Converter |
| ***SOC*** | System on a Chip |
| ***PCB*** | Printed Circuit Board |
| ***GPS*** | Global Positioning System |
| ***WLAN*** | Wireless Local Area Network. |

# 1.1 INTRODUCTION:

Aquaculture is one of the thriving areas in many countries in the world since demand for fish and the fish prepared food is expanding day by day. According to The United Nations Food and Agriculture Organization(UNFAO) "2012 State of World Fisheries and Aquaculture "Worldwide yearly production of fishery items adds up to around 128 million tons. The animal protein intake per individual is about 15% and increase the human reliance on fishery resources. The average consumption of fish products is 19 to 20 kg per person per year today and will be 16.7 kg per year in 2030 according to UNFAO. Production of fisheries, advancement and future food needs are firmly related.

Aquaculture comprises of the set of exercises, information and techniques for the rearing of aquatic plants and a few animal groups. This activity has an awesome significance in financial improvement and food production. Commercial aquaculture is confronting numerous issues because of sudden climatic vacillation leading to changes in water quality parameters. Aqua farmers are relying upon manual testing for knowing the condition of the various parameters of the water. But this manual testing is time consuming and also give inappropriate results as parameters for measuring water quality changes continuously. It will be better if automatic monitoring can be done somehow. So modern technology should be brought to aquaculture to overcome this problem. For rural development, technologies have to support several key application areas, forexample, living quality, wellbeing, environmental change etc. So we have to be more selective in choosing the appropriate technologies for this kind of advancement.

An integrated on chip computer Arduino is used in our system as data processing and storing device which has an inbuilt Wi-Fi module. Using the Dataplicity service we can also access the Arduino through internet. So, no additional Wi-Fi or Internet module is required. Smartphones are very obtainable and most of the smartphones have Media Transfer Protocol (MTP) today. Using these and performing some analysis on the water quality parameters make our work unique.determine the overall quality of the water effectively. This is the point which is the base of our research.

After a lots of study, we have realized that all parameters need not to be monitored. Because there are some parameters whose imbalances cause the imbalances of other parameters and from the quantity of some parameters we can assume the condition of others.

## 1.2 OBJECTIVES:

The objectives of our projects to design Aquaculture Monitoring System by using Iot as given below:

- To monitor the pH.
- To monitor temperature and turbidity.

Now we will mention the reasons behind this.

Temperature pronouncedly affects biological and chemical procedures. Rates of biological and chemical responses double for each 10˚C increment in temperature in general. Temperature significantly influences chemical treatments. Fish have poor resistance to sudden changes in temperature. Often, a quick change in temperature of as low as 5˚C will stress or even slaughter fish. pH, DO, conductivity, salinity etc. are directly dependent on temperature. So, temperature should be in the expected range first before checking other parameters. General threshold range for temperature is 21˚C-33˚C which can be maintained easily. For these reasons, we consider temperature as our first working parameter.

The pH is a ration of the hydrogen ion concentration and designates whether the water is acidic or basic inreaction. Phytoplankton and other marine plant life eliminate carbon dioxide from the water during photosynthesis, so the pH water body increases during the day and drops during the night. Waters with low aggregate alkalinity regularly have pH estimations of 6 to 7.5 preceding sunrise, however when phytoplankton development is substantial, at evening pH esteems may ascend to 10 or significantly higher. The pH of natural waters is significantly impacted by the convergence of carbon dioxide which is an acidic gas. pH changes in pond water are for the most part affected via carbon dioxide and ions in harmony with it. Control of pH is necessary for diminishing ammonia and $H_2S$ poisonousness. We see that pH is directly or indirectly related to many other parameters and controlling it is comparatively easier. That's why pH is our second consideration.

## 2. REVIEWS OF LITERATURE:

***Aleksandar Petkovski (September 2021)*** Aquaculture is one of the areas that is rapidly growing in the world in the last years. The demand for food is increasing in recent years and aquaculture can significantly contribute to secure food in the future. The usage of IoT technology in aquaculture can improve the production process and quality of the water. The IoT technology and cloud computing increase the productivity and reduce the costs for fish farming. The systems that use these technologies in aquaculture are monitoring systems, systems for controlling water quality and intelligent fish feeding systems. The development of these systems will lead to more efficient aquaculture. The identification of hardware, protocols, platforms, and benefits of smart fish farming is necessary for the fish farmers and the researchers. The idea of this research is to present a systematic literature review of the IoT -based solutions in aquaculture. This work aims to identify the sensors, hardware, network protocols, software applications and benefits of using IoT in aquaculture. This paper is a systematic literature review about the IoT-based applications in aquaculture, IoT -based hardware used for fish farming, network protocols and cloud-based platforms used in IoT solutions, as well as the benefits of using these technologies.

***M. H. Islam et al. (2020)*** titled "Internet of Things (IoT) in Healthcare: A Comprehensive Review" provides a detailed overview of the use of IoT in healthcare. The authors discuss the various applications of IoT in healthcare, including remote patient monitoring, real-time tracking of vital signs, and smart healthcare systems. The paper also provides an overview of the different IoT technologies used in healthcare, such as sensors, wearables, and mobile devices, and discusses their potential benefits and challenges. Additionally, the authors discuss the challenges associated with implementing IoT in healthcare, such as privacy and security concerns, interoperability issues, and the need for data analytics and management. Overall, the paper provides valuable insights into the potential of IoT in healthcare and the challenges that need to be addressed for successful implementation. It serves as a useful resource for researchers and practitioners interested in developing and implementing IoT-based healthcare systems.

***M. N. Uddin et al. (2020)*** titled "IoT-Based Smart Home Automation: A Systematic Review" provides a comprehensive review of the state of the art in IoT-based smart home automation

systems. The paper discusses the various IoT technologies used in smart homes, such as smart sensors, actuators, and controllers. The authors provide a detailed overview of the benefits of IoT-based smart homes, including increased energy efficiency, improved security and safety, and enhanced convenience and comfort. They also discuss the challenges associated with implementing IoT-based smart homes, such as interoperability issues, privacy and security concerns, and the need for effective data management and analysis.

***R. M. Yaqub et al. (2020)*** titled "An Overview of Internet of Things (IoT) Applications in Transportation Systems" provides a comprehensive overview of IoT-based transportation systems. The paper discusses the various IoT technologies used in transportation systems, such as smart traffic management, intelligent transportation systems, and connected vehicles. The authors highlight the benefits of IoT-based transportation systems, including improved traffic flow, reduced congestion, enhanced safety, and reduced environmental impact. They also discuss the challenges associated with implementing IoT-based transportation systems, such as the need for reliable and secure communication networks, interoperability between different IoT devices and platforms, and data privacy and security concerns. Overall, the paper provides valuable insights into the potential of IoT in transportation systems and the challenges that need to be addressed for successful implementation. It serves as a useful resource for researchers and practitioners interested in developing and implementing IoT-based transportation systems.

***S. A. Khan et al. (2020)*** titled "Internet of Things (IoT) Based Smart Grid: A Comprehensive Review" provides a comprehensive review of the use of IoT in smart grid systems. The paper discusses the various IoT technologies used in smart grids, such as smart sensors, advanced metering infrastructure, and smart grid communication networks. The authors highlight the benefits of using IoT in smart grids, including improved energy efficiency, increased reliability and security, and enhanced grid management and control. They also discuss the challenges associated with implementing IoT-based smart grids, such as the need for interoperability and standardization, data privacy and security concerns, and the need for effective data analytics and management. Overall, the paper provides valuable insights into the potential of IoT in smart grid systems and the challenges that need to be addressed for successful implementation. It serves as

a useful resource for researchers and practitioners interested in developing and implementing IoT-based smart grid systems.

***Adnan Sarwar (August 2022)*** Aquaculture fastest growing business worldwide especially in developing countries. Fisheries are marine species and required an oceanic environment where fisheries could grow and live naturally. Off-shore aquaculture businesses need a real-time water quality monitoring system. So, aquafarmers could maintain the required environment for a sustainable and profitable business. This work represents an IoT-based realtime health management system designed for aquaculture and considered the most required health metrics for aquaculture. The proposed system used four primary sensors: water level, temperature, pH, and dissolved oxygen. Sensors connected with microcontroller Arduino Uno R3 and ESP 8266 wi-fi module are used for data transmission to the IoT source ThingSpeak. The designed system could access online through the web interface and phone App for aquafarmers. The sensor data was accurate, and the system worked as designed.

***S.S. Sahoo and A. K. Panda (2020)*** titled "IoT-based smart farming: A review" provides a comprehensive review of IoT-based smart farming. The paper discusses the various IoT technologies used in agriculture, including sensor networks, cloud computing, and the Internet of Things.The authors highlight the benefits of using IoT in agriculture, such as efficient resource management and increased crop yields. They also discuss the challenges associated with implementing IoT-based smart farming systems, such as data management, privacy and security concerns, and interoperability between different IoT devices and platforms. Overall, the paper provides valuable insights into the potential of IoT-based smart farming and the challenges that need to be addressed for successful implementation. It serves as a useful resource for researchers and practitioners interested in developing and implementing IoT-based smart farming systems.

## 3.1 HARDWARE:

The hardware required to achieve the object are as follows:

- LCD 16*2
- Aurdino Uno
- ESP8266 module
- LM35 temperature sensor
- Jumper wire
- Relay
- Pump
- Buzzer
- pH sensor
- Turbidity sensor
- Powering Arduino with a battery
- LED indicator

# HARDWARE COMPONENTS DESCRIPTION

An IoT-based aquaculture monitoring system typically consists of various hardware components that work together to collect and transmit data from the aquaculture environment. Here are some essential hardware components commonly used in such systems:

## 3.1 LCD 16*2

An LCD (Liquid Crystal Display) screen is an electronic display module and has a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data. The Command register stores various commands given to the display. Data register stores data to be displayed. The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register.

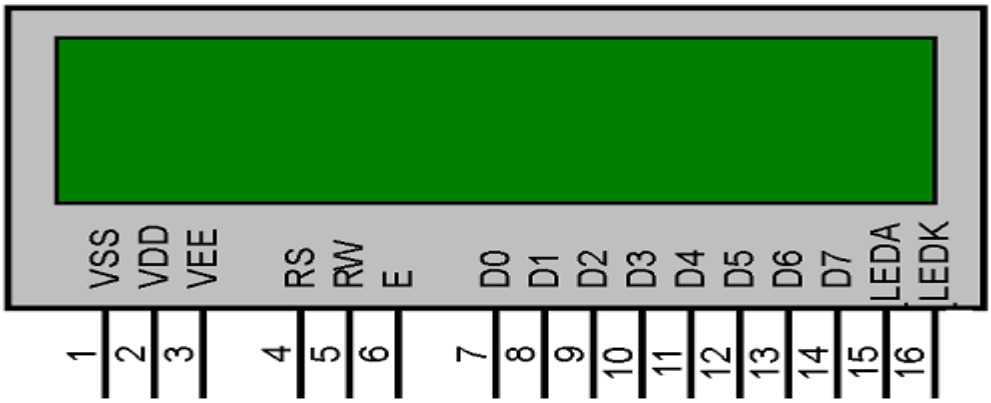



**Fig-3.1: LCD 16x2**

The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

**Table-3.1. Internal Structure of LCD**

| Sr. No | Pin No. | Pin Name | Pin Type | Pin Description | Pin Connection |
|---|---|---|---|---|---|
| 1 | Pin 1 | Ground | Source Pin | This is a ground pin of LCD | Connected to the ground of the MCU/ Power source |
| 2 | Pin 2 | VCC | Source Pin | This is the supply voltage pin of LCD | Connected to the supply pin of Power source |
| 3 | Pin 3 | V0/VEE | Control Pin | Adjusts the contrast of the LCD. | Connected to a variable POT that can source 0-5V |
| 4 | Pin 4 | Register Select | Control Pin | Toggles between Command/Data Register | Connected to a MCU pin and gets either 0 or 1.<br>0 -> Command Mode<br>1-> Data Mode |
| 5 | Pin 5 | Read/Write | Control Pin | Toggles the LCD between Read/Write Operation | Connected to a MCU pin and gets either 0 or 1.<br>0 -> Write Operation<br>1-> Read Operation |
| 6 | Pin 6 | Enable | Control Pin | Must be held high to perform Read/Write Operation | Connected to MCU and always held high. |

| | | | | | |
|---|---|---|---|---|---|
| 7 | Pin 7-14 | Data Bits (0-7) | Data/Command Pin | Pins used to send Command or data to the LCD. | In 4-Wire Mode<br>Only 4 pins (0-3) is connected to MCU<br>In 8-Wire Mode<br>All 8 pins(0-7) are connected to MCU |
| 8 | Pin 15 | LED Positive | LED Pin | Normal LED like operation to illuminate the LCD | Connected to +5V |
| 9 | Pin 16 | LED Negative | LED Pin | Normal LED like operation to illuminate the LCD connected with GND. | Connected to ground |

It is okay if you do not understand the function of all the pins, I will be explaining in details below. Now, let us turn back our LCD.

4-bit and 8-bit mode of LCD: The LCD can work in two different modes, namely the 4-bit mode and the 8-bit mode. In **4 bit mode** we send the data nibble by nibble, first upper nibble and then lower nibble. For those of you who don't know what a nibble is: a nibble is a group of four bits, so the lower four bits (D0-D3) of a byte form the lower nibble while the upper four bits (D4-D7) of a byte form the higher nibble. This enables us to send 8 bit data. Whereas **in 8 bit mode** we can send the 8-bit data directly in one stroke since we use all the 8 data lines. Now you must have guessed it, Yes 8-bit mode is faster and flawless than 4-bit mode. But the major drawback is that it needs 8 data lines connected to the microcontroller. This will make us run out of I/O pins on our MCU, so 4-bit mode is widely used. No control pins are used to set these modes. It's just the way of programming that change.

**Read and Write Mode of LCD:**

As said, the LCD itself consists of an Interface IC. The MCU can either read or write to this interface IC. Most of the times we will be just writing to the IC, since reading will make it more complex and such scenarios are very rare. Information like position of cursor, status completion interrupts etc. can be read if required, but it is out of the scope of this tutorial.

**a) Interfacing 16x2 LCDE with Arduino**

To establish a good communication between human world and machine world, display units play an important role. And so they are an important part of embedded systems. Display units - big or small, work on the same basic principle. Besides complex display units like graphic displays and 3D dispays, one must know working with simple displays like 16x1 and 16x2 units. The 16x1 display unit will have 16 characters and are in one line. The 16x2 LCD will have 32 characters in total 16in 1st line and another 16 in 2nd line.

**Components Required**

**Hardware:** ARDUINO UNO, power supply(5v), LCD screen, capacitor.

**Software:** Arduino IDE (Arduino nightly).

**Circuit Diagram and Explaination:** In 16x2 LCD there are 16 pins over all if there is a back light, if there is no back light there will be 14 pins. One can power or leave the back light pins. Now in the 14 pins there are 8 data pins (7-14 or D0-D7), 2 power supply pins (1&2 or VSS&VDD or GND&+5v), 3$^{rd}$ pin for contrast control (VEE-controls how thick the characters should be shown), and 3 control pins (RS&RW&E). In the circuit, you can observe I have only took two control pins, this gives the flexibility. The contrast bit and READ/WRITE are not often used so they can be shorted to ground. This puts LCD in highest contrast and read mode. We just need to control ENABLE and RS pins to send characters and data accordingly.

**The connections which are done for LCD are given below:**

PIN1 or VSS to ground

PIN2 or VDD or VCC to +5v power

PIN3 or VEE to ground (gives maximum contrast best for a beginner)

PIN4 or RS (Register Selection) to PIN0 of ARDUINO UNO

PIN5 or RW (Read/Write) to ground (puts LCD in read mode eases the communication for user)

PIN6 or E (Enable) to PIN1 of ARDUINO UNO

PIN11 or D4 to PIN8 of ARDUINO UNO

PIN12 or D5 to PIN9 of ARDUINO UNO

PIN13 or D6 to PIN10 of ARDUINO UNO

PIN14 or D7 to PIN11 of ARDUINO UNO

The ARDUINO IDE allows the user to use LCD in 4 bit mode. This type of communication enables the user to decrease the pin usage on ARDUINO, unlike other the ARDUINO need not to be programmed separately for using it in 4 it mode because by default the ARDUINO is set up to communicate in 4 bit mode. In the circuit you can see we have used 4bit communication (D4-D7).

## 3.2:- Aurdino Uno

The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board. Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits. The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.

**Arduino Uno** is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an 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 a 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.


**Fig3.2: Aurdino UNO**

## How to use Arduino boards?

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11:** These pins provide an 8-bit PWM output by using analogWrite() function.
- **SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK):** These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

## Table-3.2: Arduino Uno Pinout Configuration

| Pin Category | Pin Name | Details |
|---|---|---|
| Power | Vin, 3.3V, 5V, GND | Vin: Input voltage to Arduino when using an external power source.<br>5V: Regulated power supply used to power microcontroller and other components on the board.<br>3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.<br>GND: ground pins. |
| Reset | Reset | Resets the microcontroller. |
| Analog Pins | A0 – A5 | Used to provide analog input in the range of 0-5V |
| Input/Output Pins | Digital Pins 0 - 13 | Can be used as input or output pins. |
| Serial | 0(Rx), 1(Tx) | Used to receive and transmit TTL serial data. |
| External Interrupts | 2, 3 | To trigger an interrupt. |
| PWM | 3, 5, 6, 9, 11 | Provides 8-bit PWM output. |

| | | |
|---|---|---|
| SPI | 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK) | Used for SPI communication. |
| Inbuilt LED | 13 | To turn on the inbuilt LED. |
| TWI | A4 (SDA), A5 (SCA) | Used for TWI communication. |
| AREF | AREF | To provide reference voltage for input voltage. |

## Communication

Arduino can be used to communicate with a computer, another Arduino board or other microcontrollers. The ATmega328P microcontroller provides UART TTL (5V) serial communication which can be done using digital pin 0 (Rx) and digital pin 1 (Tx). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The ATmega16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. There are two RX and TX LEDs on the arduino board which will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328P also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

## Several function of Arduino:

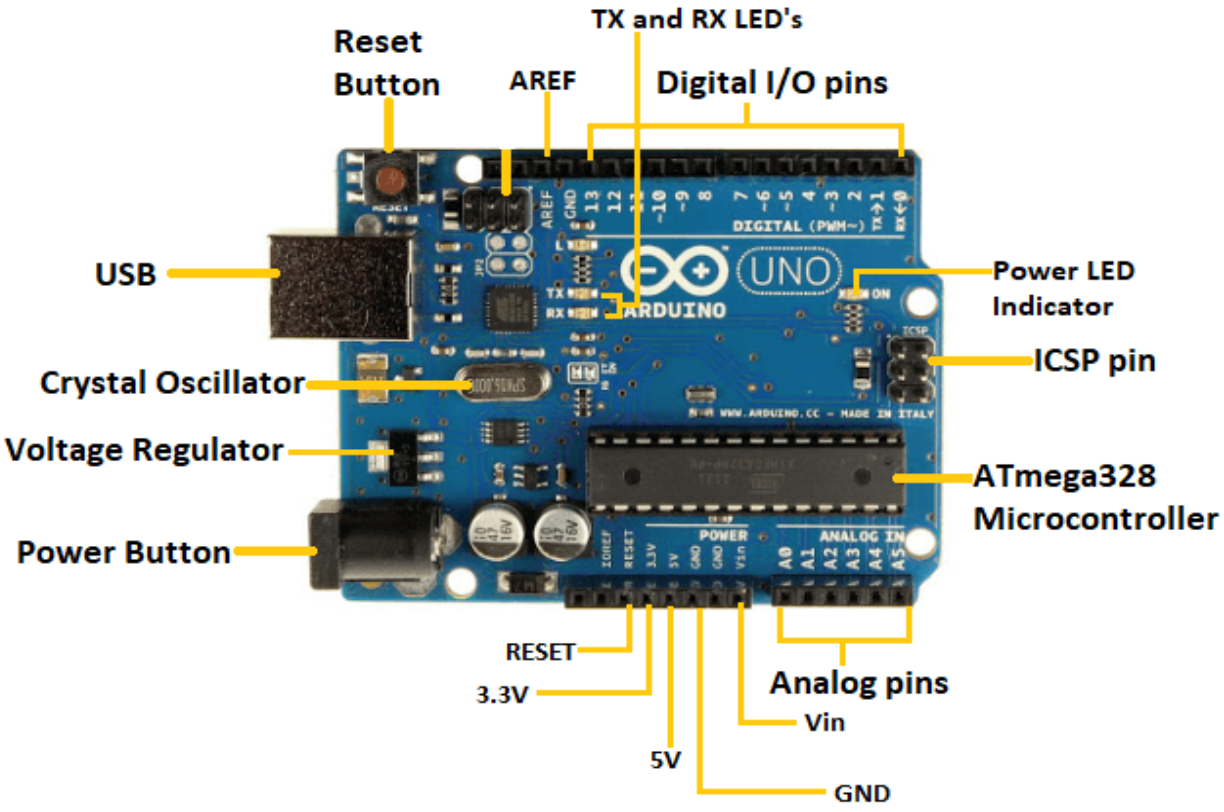



**Fig-3.3: Function of Arduino**

### Board Description of Arduino UNO

- **ATmega328 Microcontroller-** It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.
- **ICSP pin -** The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator-** The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins-** The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.

- **TX and RX LED's-** The successful flow of data is represented by the lighting of these LED's.
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button-** It is used to add a Reset button to the connection.
- **USB-** It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator-** The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator-** The voltage regulator converts the input voltage to 5V.
- **GND-** Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin-** It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

**ATmega328P** is a popular 8-bit microcontroller based on the AVR architecture, manufactured by Microchip Technology (previously Atmel Corporation). It is widely used in various embedded systems and microcontroller-based projects due to its versatility, ease of use, and availability.

**ATmega328 Pinout Configuration**

ATMEGA328P is a 28 pin chip as shown in pin diagram above. Many pins of the chip here have more than one function. We will describe functions of each pin in below table.

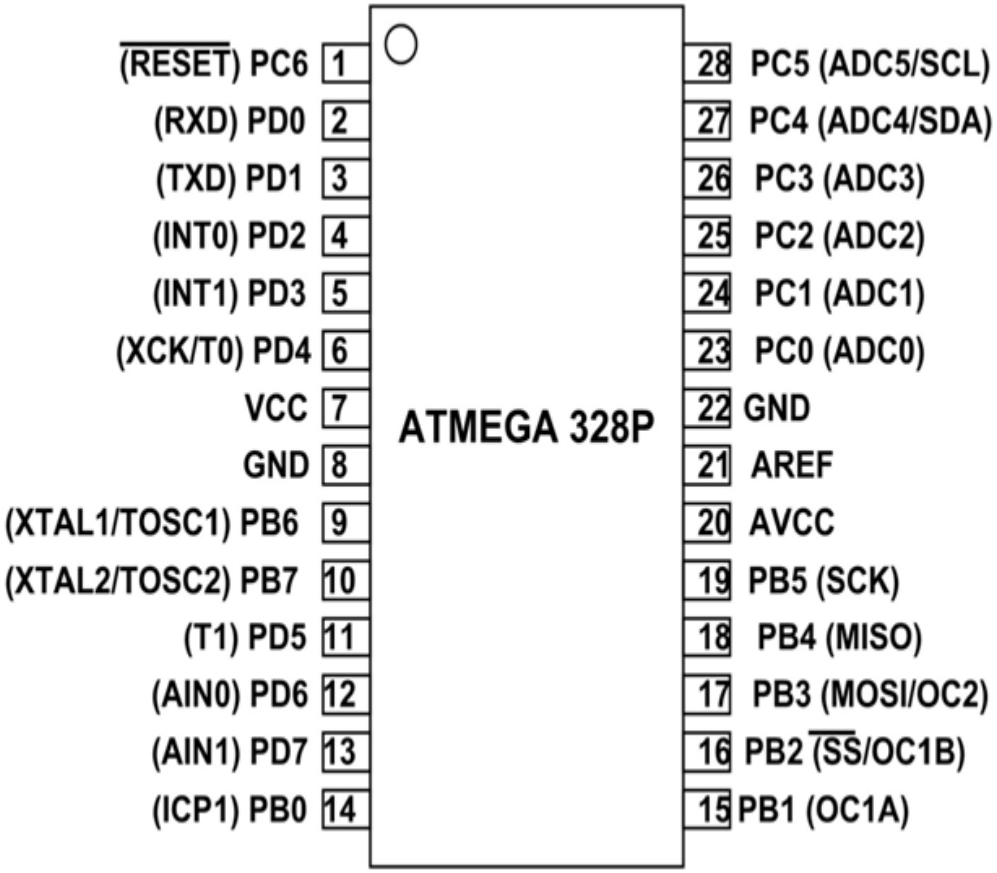



**Fig 3.4: ATMEGA 328 Pinout Configuration**

**Table-3.3: Internal structure of ATMega328P microcontroller**

| Pin No. | Pin name | Description | Secondary Function |
|---|---|---|---|
| 1 | PC6 (RESET) | Pin6 of PORTC | Pin by default is used as RESET pin. PC6 can only be used as I/O pin when RSTDISBL Fuse is programmed. |
| 2 | PD0 (RXD) | Pin0 of PORTD | RXD (Data Input Pin for USART) USART Serial Communication Interface |

| | | | |
|---|---|---|---|
| | | | [Can be used for programming] |
| 3 | PD1 (TXD) | Pin1 of PORTD | TXD (Data Output Pin for USART)<br>USART Serial Communication Interface<br>[Can be used for programming]<br><br>INT2( External Interrupt 2 Input) |
| 4 | PD2 (INT0) | Pin2 of PORTD | External Interrupt source 0 |
| 5 | PD3 (INT1/OC2B) | Pin3 of PORTD | External Interrupt source1<br><br>OC2B(PWM - Timer/Counter2 Output Compare Match B Output) |
| 6 | PD4 (XCK/T0) | Pin4 of PORTD | T0( Timer0 External Counter Input)<br>XCK ( USART External Clock I/O) |
| 7 | VCC | | Connected to positive voltage |
| 8 | GND | | Connected to ground |
| 9 | PB6 (XTAL1/TOSC1) | Pin6 of PORTB | XTAL1 (Chip Clock Oscillator pin 1 or External clock input)<br>TOSC1 (Timer Oscillator pin 1) |
| 10 | PB7 | Pin7 of | XTAL2 (Chip Clock Oscillator pin 2) |

| | | | |
|---|---|---|---|
| | (XTAL2/TOSC2) | PORTB | TOSC2 (Timer Oscillator pin 2) |
| 11 | PD5<br>(T1/OC0B) | Pin5 of PORTD | T1(Timer1 External Counter Input)<br><br>OC0B(PWM - Timer/Counter0 Output Compare Match B Output) |
| 12 | PD6<br>(AIN0/OC0A) | Pin6 of PORTD | AIN0(Analog Comparator Positive I/P)<br><br>OC0A(PWM - Timer/Counter0 Output Compare Match A Output) |
| 13 | PD7 (AIN1) | Pin7 of PORTD | AIN1(Analog Comparator Negative I/P) |
| 14 | PB0<br>(ICP1/CLKO) | Pin0 of PORTB | ICP1(Timer/Counter1 Input Capture Pin)<br><br>CLKO (Divided System Clock. The divided system clock can be output on the PB0 pin) |
| 15 | PB1 (OC1A) | Pin1 of PORTB | OC1A (Timer/Counter1 Output Compare Match A Output) |
| 16 | PB2 (SS/OC1B) | Pin2 of PORTB | SS (SPI Slave Select Input). This pin is low when controller acts as slave.<br>[Serial Peripheral Interface (SPI) for programming] |

| | | | |
|---|---|---|---|
| | | | OC1B (Timer/Counter1 Output Compare Match B Output) |
| 17 | PB3 (MOSI/OC2A) | Pin3 of PORTB | MOSI (Master Output Slave Input). When controller acts as slave, the data is received by this pin. [Serial Peripheral Interface (SPI) for programming]<br>OC2 (Timer/Counter2 Output Compare Match Output) |
| 18 | PB4 (MISO) | Pin4 of PORTB | MISO (Master Input Slave Output). When controller acts as slave, the data is sent to master by this controller through this pin.<br>[Serial Peripheral Interface (SPI) for programming] |
| 19 | PB5 (SCK) | Pin5 of PORTB | SCK (SPI Bus Serial Clock). This is the clock shared between this controller and other system for accurate data transfer.<br>[Serial Peripheral Interface (SPI) for programming] |
| 20 | AVCC | | Power for Internal ADC Converter |
| 21 | AREF | | Analog Reference Pin for ADC |
| 22 | GND | | GROUND |

| | | | |
|---|---|---|---|
| 23 | PC0 (ADC0) | Pin0 of PORTC | ADC0 (ADC Input Channel 0) |
| 24 | PC1 (ADC1) | Pin1 of PORTC | ADC1 (ADC Input Channel 1) |
| 25 | PC2 (ADC2) | Pin2 of PORTC | ADC2 (ADC Input Channel 2) |
| 26 | PC3 (ADC3) | Pin3 of PORTC | ADC3 (ADC Input Channel 3) |
| 27 | PC4 (ADC4/SDA) | Pin4 of PORTC | ADC4 (ADC Input Channel 4)<br>SDA (Two-wire Serial Bus Data Input/output Line) |
| 28 | PC5 (ADC5/SCL) | Pin5 of PORTC | ADC5 (ADC Input Channel 5)<br>SCL (Two-wire Serial Bus Clock Line) |

**Table-3.4: Features of ATMEGA328P**

| **ATMEGA328P – Simplified Features** | |
|---|---|
| CPU | 8-bit AVR |
| Number of Pins | 28 |
| Operating Voltage (V) | +1.8 V TO +5.5V |

| | |
|---|---|
| Number of programmable I/O lines | 23 |
| Communication Interface | Master/Slave SPI Serial Interface(17,18,19 PINS) [Can be used for programming this controller]<br>Programmable Serial USART(2,3 PINS) [Can be used for programming this controller]<br>Two-wire Serial Interface(27,28 PINS)[Can be used to connect peripheral devices like Servos, sensors and memory devices] |
| JTAG Interface | Not available |
| ADC Module | 6channels, 10-bit resolution ADC |
| Timer Module | Two 8-bit counters with Separate Prescaler and compare mode, One 16-bit counter with Separate Prescaler,compare mode and capture mode. |
| Analog Comparators | 1(12,13 PINS) |
| DAC Module | Nil |
| PWM channels | 6 |
| External Oscillator | 0-4MHz @ 1.8V to 5.5V<br>0-10MHz @ 2.7V to 5.5V<br>0-20MHz @ 4.5V to 5.5V |

| | |
|---|---|
| Internal Oscillator | 8MHz Calibrated Internal Oscillator |
| Program Memory Type | Flash |
| Program Memory or Flash memory | 32Kbytes[10000 write/erase cycles] |
| CPU Speed | 1MIPS for 1MHz |
| RAM | 2Kbytes Internal SRAM |
| EEPROM | 1Kbytes EEPROM |
| Watchdog Timer | Programmable Watchdog Timer with Separate On-chipOscillator |
| Program Lock | Yes |
| Power Save Modes | Six Modes[Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby] |
| Operating Temperature | -40°C to +105°C(+105 being absolute maximum, -40 being absolute minimum) |

**Where to Use ATMEGA328P**?

Although we have many controllers ATMEGA328P is most popular of all because of its features and cost. ARDUINO boards are also developed on this controller because of its features.

- With program memory of 32 Kbytes ATMEGA328P applications are many.

- With various POWER SAVING modes it can work on MOBILE EMBEDDED SYSTEMS.
- With Watchdog timer to reset under error it can be used on systems with minimal human interference.
- With advanced RISC architecture, the controller executes programs quickly.
- Also with in chip temperature sensor the controller can be used at extreme temperatures.

These all features add together promoting ATMEGA328P further.

**How to Use ATMEGA328P**?

ATMEGA328 is used similar to any other controller. All there to do is programming. Controller simply executes the program provided by us at any instant. Without programming controller simply stays put without doing anything.

As said, first we need to program the controller and that is done by writing the appropriate program file in the ATMEGA328P FLASH memory. After dumping this program code, the controller executes this code and provides appropriate response.

Entire process of **using an ATMEGA328P** goes like this:

1. List the functions to be executed by controller.
2. Write the functions in programming language in IDE programs.

**Applications**:

There are hundreds of applications for ATMEGA328P:

- Used in ARDUINO UNO, ARDUINO NANO and ARDUINO MICRO boards.
- Industrial control systems.
- SMPS and Power Regulation systems.
- Digital data processing.
- Analog signal measuring and manipulations.
- Embedded systems like coffee machine, vending machine.
- Motor control systems.

- Display units.
- Peripheral Interface system.

**Related Boards:**

If you are interested in boards with similar functionality, at Arduino you can find:

- Arduino Uno Rev3
- Arduino Uno Wi-Fi Rev2

**Different types of Arduinos:**

- Arduino Uno (R3)
- Arduino Nano
- Arduino Micro
- Arduino Due
- LilyPad Arduino Board
- Arduino Bluetooth
- Arduino Diecimila
- RedBoard Arduino Board
- Arduino Mega (R3) Board
- Arduino Leonardo Board

## 3.3:- ESP8266 module

The 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. 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. Its high degree of on-chip

integration allows for minimal external circuitry. The ESP8266 Module is not capable of 5-3V logic shifting and will require an external Logic Level Converter. Please do not power it directly from your 5V dev board.

The ESP8266 module enables microcontrollers to connect to 2.4 GHz Wi-Fi, using IEEE 802.11 bgn. It can be used with ESP-AT firmware to provide Wi-Fi connectivity to external host MCUs, or it can be used as a self-sufficient MCU by running an RTOS-based SDK.

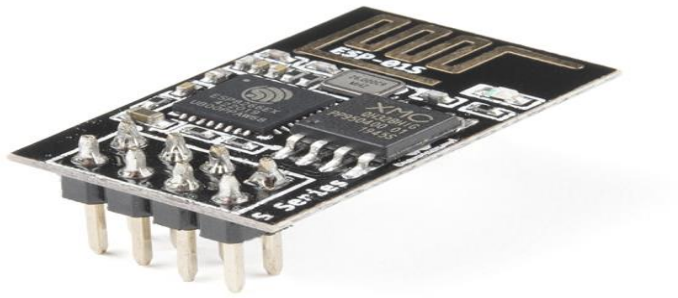



**Fig-3.5: Wi-Fi module ESP8266**

**The range of Wi-Fi module ESP8266:**

The module has a wireless Wi-Fi transceiver operating in an unlicensed frequency range of 2400-2484 MHz in the IEEE 802.11 b/g/n standard, with support for TCP/IP communication protocol stack and Wi-Fi security including WAP3.

**ESP8266 Features:**

- It is also known as a system-on-chip (SoC) and comes with a 32-bit Tensilica microcontroller, antenna switches, RF balun, power amplifier, standard digital peripheral interfaces, low noise receive amplifier, power management module and filter capability.
- The processor is based on Tensilica Xtensa Diamond Standard 106Micro and runs at 80 MHz.
- It incorporates 64 KiB boot ROM, 80 KiB user data RAM and 32 KiB instruction RAM.
- It supports Wi-Fi 802.11 b/g/n around 2.4 GHz and other features including 16 GPIO, Inter-Integrated Circuit ($I^2C$), Serial Peripheral Interface (SPI), 10-bit ADC, and $I^2S$ interfaces with DMA.

- External QSPI flash memory is accessed through SPI and supports up to 16 MiB and 512 KiB to 4 MiB is initially included in the module.
- It is a major development in terms of wireless communication with little circuitry. and contains onboard regulator that helps in providing 3.3V consistent power to the board.
- It supports APSD which makes it an ideal choice for VoIP applications and Bluetooth interfaces.

**ESP8266 Projects and Applications:**

- Wireless Web Server
- Geolocation using ESP8266
- Pressure Sensors on Railway Tracks
- Air Pollution Meter
- Temperature logging system
- World's smallest IoT project
- Wi-Fi controlled robot
- Humidity and temperature monitoring
- M2M using ESP8266

## 3.4:-LM35 temperature sensor

LM35 is an analog, linear temperature sensor whose output voltage varies linearly with change in temperature. LM35 is three terminal linear temperature sensor from National semiconductors. It can measure temperature from-55 degree celsius to +150 degree celsius. The voltage output of the LM35 increases 10mV per degree Celsius rise in temperature. LM35 can be operated from a 5V supply and the stand by current is less than 60uA.

LM35 sensor uses the basic principle of a diode, where as the temperature increases, the voltage across a diode increases at a known rate. By precisely amplifying the voltage change, it is easy to generate an analog signal that is directly proportional to temperature.


**Fig-3.6: LM35 Temperature Sensor**

**Pin Configuration of LM35 Temperature Sensor:**

- Pin 1-Vcc-Input voltage is +5V for typical applications
- Pin 2-Analog Out-There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
- Pin 3-Ground-Connected to ground of circuit

**LM35 Sensor Features**:

- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically 5V.
- Can measure temperature ranging from -55°C to 150°C
- Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
- ±0.5°C Accuracy
- Drain current is less than 60uA
- Low cost temperature sensor
- Small and hence suitable for remote applications
- Available in TO-92, TO-220, TO-CAN and SOIC package

## 3.5: Jumper Wire

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires. A **jump wire** (also known as jumper, jumper wire, DuPont wire) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

**Fig-3.7: Jumper wire**

Though jumper wires come in a variety of colors, the colors don't actually mean anything. This means that a red jumper wire is technically the same as a black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power. Jumper wires typically come in three versions: male-to-male, male-to-female and female-to-female. The difference between each is in the end point of the wire. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into. Male-to-male jumper wires are the most common and what you likely will use most often. When connecting two ports on a breadboard, a male-to-male wire is what you'll need.

## 3.6: Relay

ADIY 1 channel 5v relay module is designed for switching only a single high powered device from your Arduino. It has a relay rated up to 10A per channel at 250VAC or 30VDC. There are two LEDs on the relay module indicating the position of the relay. We have three channels of the relay broken out to screw pin terminals. The channels are labeled for their function: common (COM), normally closed (NC), and normally open (NO). On the other side of the module, there are three pins – a Ground pin and a VCC pin to power the module and an input pin IN to control the relay. The input pin is active low, meaning the relay will be activated when you pull the pin LOW and it will become inactive when you pull the pin HIGH.


**Fig-3.8: A Relay**

Relays can be of different types like electromechanical, solid state. Electromechanical relays are frequently used. Let us see the internal parts of this relay before knowing about it working. Although many different types of relay were present, their working is same.Every electromechanical relay consists of an consists of an

1. Electromagnet
2. Mechanically movable contact
3. Switching points and
4. Spring

Electromagnet is constructed by wounding a copper coil on a metal core. The two ends of the coil are connected to two pins of the relay as shown. These two are used as DC supply pins.

## 3.7: Pump

A pump is a device that moves fluids (liquids or gases), or sometimes slurries, by mechanical action, typically converted from electrical energy into hydraulic energy. A small, electrically powered pump. Mechanical pumps serve in a wide range of applications such as pumping water from wells, aquarium filtering, pond filtering and aeration, in the car industry for water-cooling and fuel injection, in the energy industry for pumping oil and natural gas or for operating cooling towers and other components of heating, ventilation and air conditioning systems. In the medical industry, pumps are used for biochemical processes in developing and manufacturing medicine, and as artificial replacements for body parts, in particular the artificial heart and penile prosthesis.

**Fig-3.9: A Pump**

## 3.8: Buzzer

An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.


**Fig-3.10: Buzzer**

The pin configuration of the buzzer is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the '-'symbol or short terminal and it is connected to the GND terminal.

**Applications:**

The applications of the buzzer include the following.

- Communication Devices
- Electronics used in Automobiles
- Alarm Circuits
- Portable Devices
- Security Systems
- Timers
- Household Appliances
- Electronic Metronomes
- Sporting Events
- Game Show

## 3.9: pH Sensor

A pH sensor is one of the most essential tools that's typically used for water measurements. This type of sensor is able to measure the amount of alkalinity and acidity in water and other solutions**.** When used correctly, pH sensors are able to ensure the safety and quality of a product and the processes that occur within a wastewater or manufacturing plant.

In most cases, the standard pH scale is represented by a value that can range from 0-14. When a substance has a pH value of seven, this is considered to be *neutral*. Substances with a pH value above seven represent higher amounts of alkalinity whereas substances with a pH value that's lower than seven are believed to be more acidic. For instance, toothpaste typically comes with a pH value of 8-9. On the other hand, stomach acid has a pH value of two.

**Fig-3.11: pH Sensor**

The difference between an alkaline substance and an acidic substance is very important for any company that uses a cooling tower, boiler, manufacturing processes, swimming pool control, and various types of environmental monitoring. The human body has a standard pH level of 7.4, which is essential for the body to run effectively. If the composition of the body every becomes too acidic or overly alkaline, it will look to return to the neutral state.

A pH meter has three components: the pH meter itself (moving-coil or digital meter), a reference pH electrode, and a pH probe that is inserted into the solution being tested. Most pH probes contain two electrodes inside the body: a measuring electrode (glass electrode) and a reference electrode. The glass electrode contains a reference electrolyte (usually potassium chloride), which has a neutral pH of 7, therefore, it contains a specific amount of hydrogen ions.

The glass electrode works by measuring the difference in pH between the pH electrode and the solution being tested. The electrode does this by measuring the differences in the voltages of the hydrogen ions produced in both the electrode and the solution. This is easy to work out as we already know the pH value of the glass electrode composition.

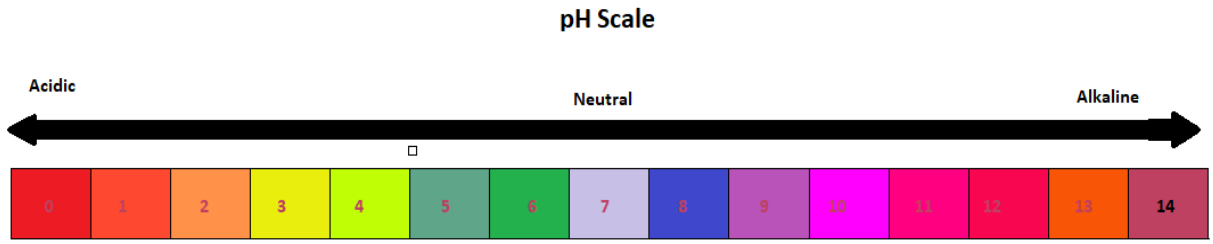



**Fig-3.12: pH Scale**

- pH is a scale that is used to measure the acidity or basicity of an aqueous solution. The pH scale ranges from 0 to 14. 0 being the most acidic and 14 being the most basic. Pure water is neutral and thus has a PH of 7.
- pH is the logarithmic scale and it inversely indicates the concentration of hydrogen ions in an aqueous solution.

$$\mathrm{pH} = -\log_{10}(a_{\mathrm{H}^+}) = \log_{10}\left(\frac{1}{a_{\mathrm{H}^+}}\right)$$

- The above mentioned is the formula to calculate the pH of an aqueous solution.

## 3.10: Turbidity Sensor

AZDM01 has a built-in photoelectric sensor. The light-emitting element generates an infrared light source. The light passes through the transparent plastic casing and the measuring tank. When the liquid with different turbidity levels flows through the measuring tank, the degree of blocking of the light is different, and the intensity of the transmitted light is different, and the photosensitive element will receive it. The change in light intensity is converted into an analogue

voltage signal, and the turbidity of the liquid is calculated by detecting the change in the voltage signal at the output terminal.

**Features:**

1. High reliability
2. Quick response
3. Good stability
4. Waterproof design
5. long life
6. Output mode analog signal

**Fig-3.13: Turbidity Sensor**

The basic principle behind a turbidity sensor involves the measurement of light scattered by the suspended particles in the liquid. Here's a general overview of how it works:

Light Source: The turbidity sensor typically includes a light source, such as an LED (Light Emitting Diode) or a laser diode. The light emitted is usually in the visible spectrum.

Photodetector: The sensor also contains a photodetector, which is capable of detecting the intensity of light.

Optical Path: The light emitted by the source travels through the liquid sample in a specific optical path.

Scattering of Light: As the light passes through the liquid, it encounters the suspended particles. These particles scatter the light in different directions, deviating it from its original path.

Photodetection: The scattered light is then detected by the photodetector. The photodetector measures the intensity of the scattered light, which is proportional to the turbidity of the sample.

Calibration: To convert the measured intensity into a meaningful turbidity value, the sensor is calibrated using standard reference solutions with known turbidity levels. This calibration process establishes a correlation between the detected light intensity and turbidity units, typically expressed in NTU (Nephelometric Turbidity Units).

Output: The turbidity sensor provides an output, often in the form of an electrical signal or a digital reading, that represents the turbidity level of the liquid sample.

## 3.11: Powering Arduino with a battery

We can use the Vin pin to Power your Arduino with an unregulated 7 to 12-volt power source. Like a 9V battery or a wall adapter that is in the range of 7 to 12 volts. Alternatively, you can power your Arduino through the 5V pin with an external regulated 5V power supply. It can be a wall adapter that gives out constant 5V or a DC-DC converter that is connected to a battery or a set of batteries. You can use both the 5V pin and the 3.3V pin to provide power to modules that are connected to the Arduino. But you can't use the 3.3V pin to power your Arduino Uno/Nano.

We can use the 3.3V pin to power sensors and modules that need 3.3V power.

It can supply about 100 to 150mA of current. The 3.3V regulator is connected to the output of the 5V regulator. Drawing current from the 3.3V regulator will dissipate heat in both the 3.3V regulator and the 5V regulator. This means that if you connect a 3.3V device to the 3.3V pin, then it also limits the maximum current you can use for the 5V modules connected to the 5V pin.

## 3.12: LED Indicator

LED indicators are illuminated components used to show the status of a function, a battery, or electronics. APEM's indicator range includes high-quality options from 6 mm all the way up to 30 mm. These premiums, robust and energy-efficient indicators are ideal in a broad range of applications from off-road to agriculture, from military vehicles to commercial electronics.

**Fig-3.14: LED**

A light-emitting diode is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons.

# 4. SOFTWARE USED

## 4.1: Arduino Integrated Development Environment (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. **Arduino IDE** is an open-source software, designed by Arduino.cc and mainly used for writing, compiling & uploading code to almost all Arduino Modules. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. It is available for all operating systems i.e. MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role in debugging, editing and compiling the code. A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more. Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board. The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. This environment supports both C and C++ languages.


**Fig-4.1: Software Arduino**

## 4.2: IOT Online open source development platform (Thingspeak)

ThingSpeak is an open source "Internet of Things" application and API to store and retrieve data from things using HTTP over the Internet or via a Local Area Network. With ThingSpeak, you can create sensor logging applications, location tracking applications, and a social network of things with status updates.

In addition to storing and retrieving numeric and alphanumeric data, the ThingSpeak API allows for numeric data processing such as timescaling, averaging, median, summing, and rounding. Each ThingSpeak Channel supports data entries of up to 8 data fields, latitude, longitude, elevation, and status. The channel feeds support JSON, XML, and CSV formats for integration into applications. The ThingSpeak application also features time zone management, read/write API key management and JavaScript-based charts from Highslide Software / Torstein

Hønsi.

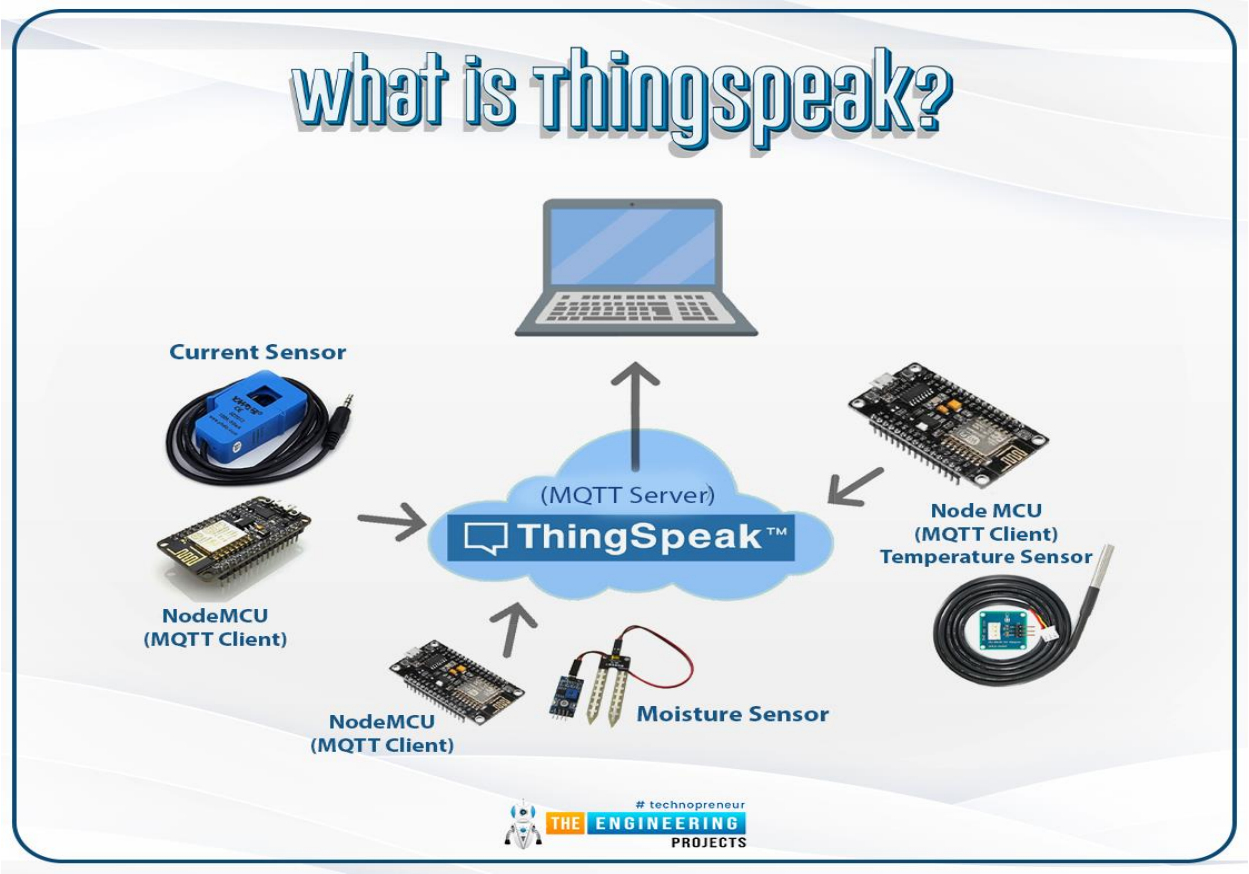



**Fig-4.2: ThingSpeak**

**Features:**

- 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 like Twilio® or Twitter®.

## 4.3: Microcontroller programming language: Embedded C

Embedded C is a programming language commonly used for microcontroller programming. It is a variant of the C programming language that is tailored specifically for embedded systems, which are computer systems designed to perform specific tasks with real-time constraints.

When programming in Embedded C, it is important to have a good understanding of the microcontroller's datasheet, the specific development environment or IDE (integrated development environment) you are using, and any hardware-related specifications or constraints.

There are several popular IDEs and tool chains available for Embedded C development, such as Keil µVision, MPLAB X IDE, Atmel Studio, and Eclipse with CDT, each with their own set of features and support for various microcontroller families.

Arduino boards are commonly programmed using a variant of the C++ programming language. While the Arduino IDE and framework simplify the programming process, it is still based on the C and C++ languages.

**Is Embedded C and C are different?**

Embedded C is an extension of C language and it is used to develop micro-controller based applications. The extensions in the Embedded C language from normal C Programming Language is the I/O Hardware Addressing, fixed-point arithmetic operations, accessing address spaces, etc. Embedded C Program has five layers of Basic Structures**.**

C is a general-purpose programming language, which is widely used to design any type of desktop-based applications. It was developed by **Dennis Ritchie** as a system programming language to develop the operating system. The main features of C language include low-level access to memory, a simple set of keywords, and clean style, these features make C language suitable for system programmings like OS or compiler development. In nature it uses a native platform development scheme, ie the development of the application by it is platform-dependent and can only be used on a single platform.

**Embedded Software Development Tools List**

- PyCharm: A Czech company JetBrains created this IDE specifically for developers working with Python.
- WebStorm: Another IDE from JetBrains is WebStrom, used for creating JavaScript, CSS and HTML solutions.
- Qt Creator.
- MPLAB X.
- Visual Studios.
- Eclipse.
- Net Beans.
- MATLAB.
- Arduino.

## 4.4: MATLAB

MATLAB (an abbreviation of "MATrix LABoratory") is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numeric computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation

- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building.

## 5.1 WORKING:

This system will keep an eye on turbidity, temperature, and pH. The system that is being suggested consists of three sensors that measure pH, temperature, and turbidity. A message is sent to the LCD after a comparison of the measured values from the sensors with the predetermined values. A buzzer and LED will sound an alert if any parameter increases above its limit.

| Parameters | Limiting Range |
|---|---|
| 1. pH | 6-8 |
| 2. Temperature | 25-32deg. Celcius |
| 3. Turbidity | 25-100 NTU |

The data is first shown on the LCD panel before being transferred to the Wi-Fi module. The Wi-Fi module sends the measured data value to the server over the internet. The Wi-Fi module is set up to send measured data to the "ThingSpeak" application running on a distant server. The information delivered to the remote server was updated using a string created from the data gathered from the sensor. Additionally, this location has a relay that activates the aerator pump whenever the water parameters are outside of the predetermined range.

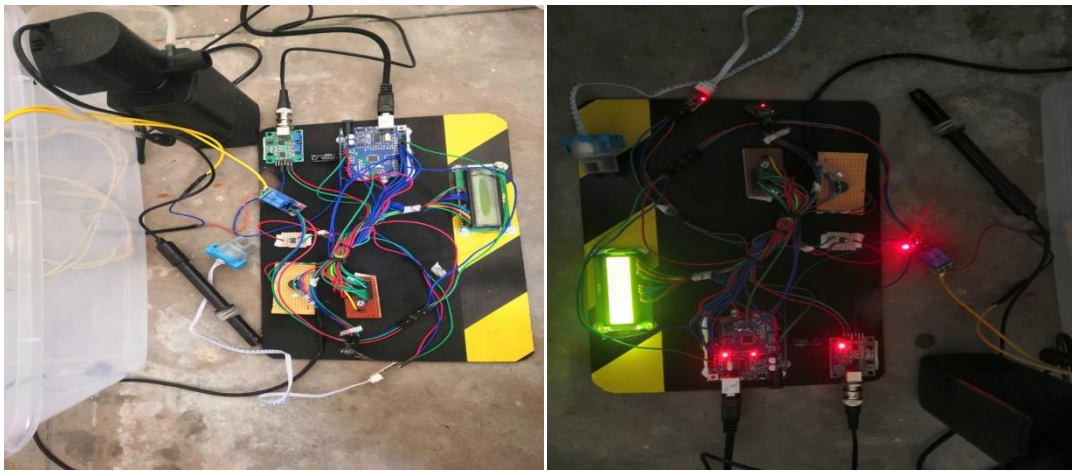
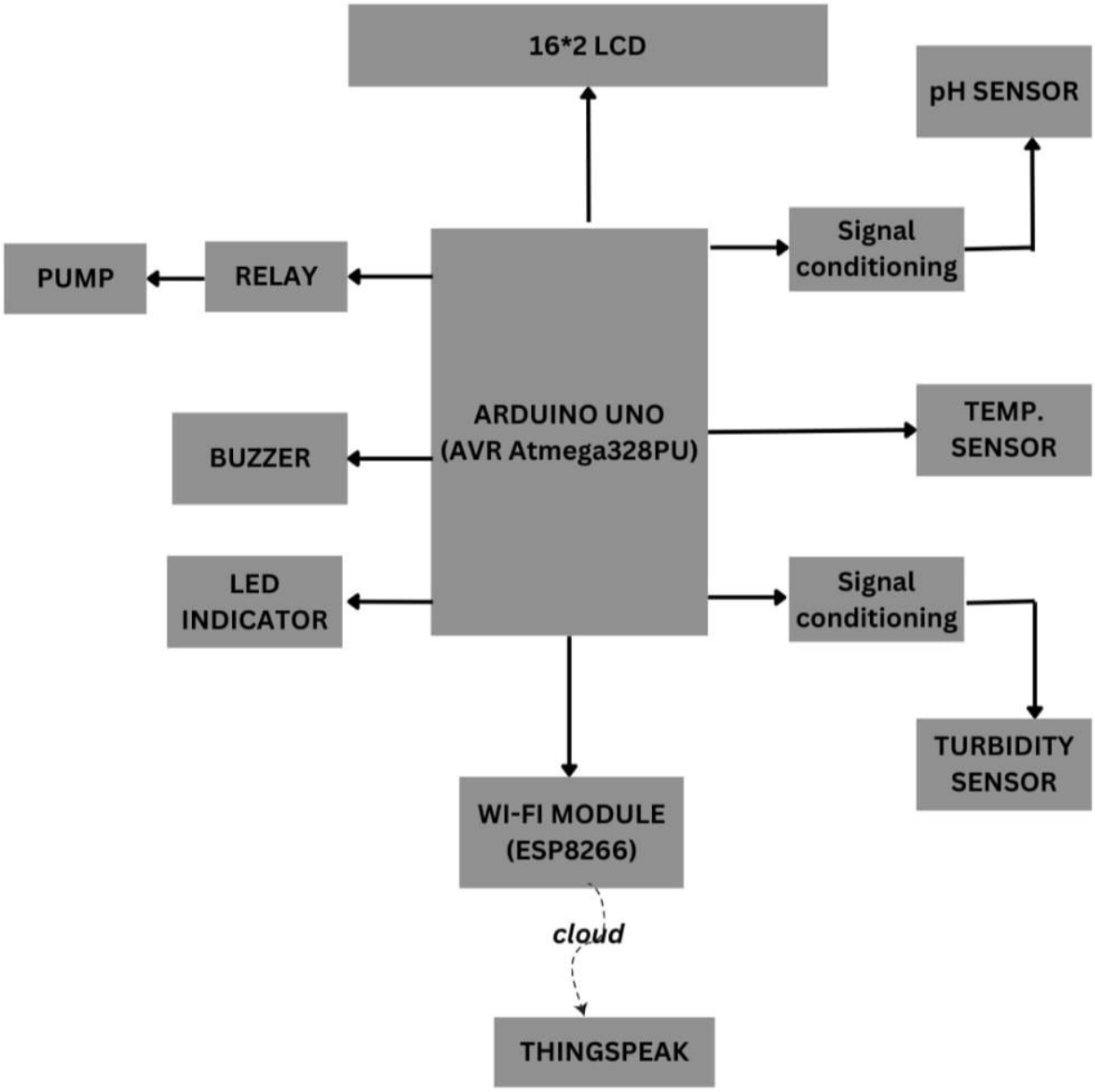
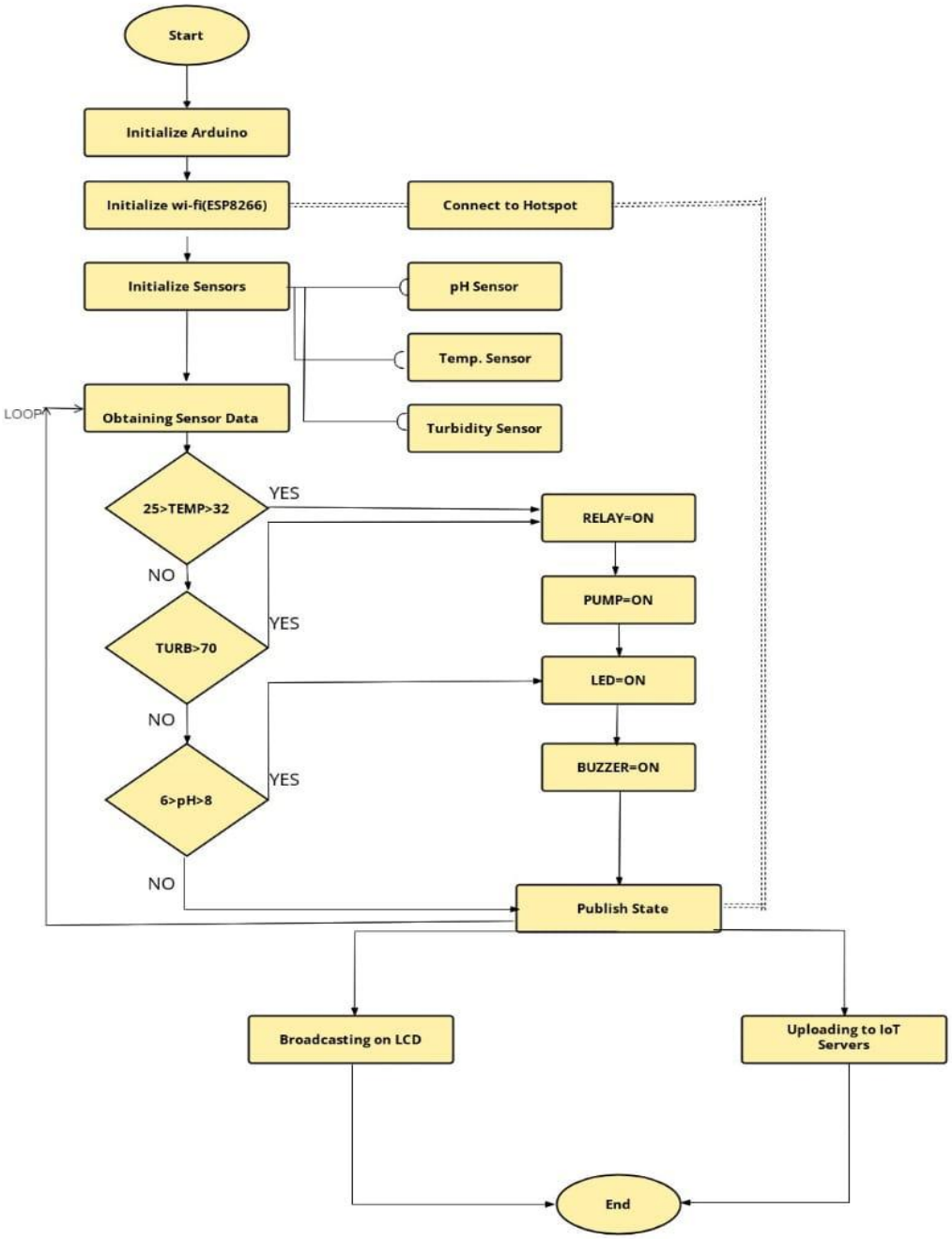


**Fig-5.1: Working Model**

## 5.2 BLOCK DIAGRAM OF SYSTEM



# 5.3 FLOWCHART



## 5.4 ADVANTAGES

- Sensors are easily available.
- Interface any number of sensors to know details content of all parameters present in water.
- Detecting a wide range of temperature, turbidity, pH and etc
- Simple, compact & Easy to handle.
- Sensors have long life time & less cost.
- Simple Drive circuit.
- System is Real time.
- Quality of water can be checked indoor as well as outdoor.
- Visual Output.
- Continuous update of change in percentage of quality.

## 5.5 APPLICATIONS

- Measures the temperature 24/7 for monitoring ,
- It also monitors ph.
- Values and automatically adjusts the ph.
- Values for better shrimp growth.

# 6.1: RESULTS AND DISCUSSION

The software 'ThingSpeak' collects the data from sensors, analyze it and produces the result. Thing-Speak is an open-source internet of things application programming interface used to store and retrieve data from interconnected things using the hypertext protocol over the internet or via a local area network. It also provides access to a broad range of embedded devices and web services. This enables the creation of sensor logging applications that can be updated regularly. The several are shown on the server like temperature, pH and turbidity.

## 6.1.1 Temperature:

In aquaculture, the ideal water temperature varies depending on the species being cultured. However, there is a general temperature range that is considered optimal for most aquatic organisms. The ideal water temperature for aquaculture typically falls between 20°C and 30°C (68°F and 86°F).

This temperature range provides favorable conditions for the growth, reproduction, and overall health of many commonly farmed species. It is important to note that specific temperature requirements may differ for different species, so it's crucial to research and understand the specific needs of the organisms you intend to cultivate.


**Fig-6.1 Result of Temperature on ThingSpeak**

We analyze the project on 09th of June at 4:30pm and observed that the temperature which is most suitable in between 25 to 32. The measured values from the sensors are compared with the

established data and if any of the parameter exceeds its threshold values then an alert will be made using LED indicator and buzzer. It has a relay to trigger the filtering pump which circulates the water in tub.

## 6.1.2 Turbidity

Turbidity refers to the measure of the cloudiness or haziness of a liquid, such as water, caused by suspended particles. It is an important water quality parameter that can impact various aquatic environments, including aquaculture systems. Turbidity is typically measured in Nephelometric Turbidity Units (NTU). Turbidity in water can originate from various sources, including organic matter, clay, silt, algae, and other suspended particles. High turbidity levels can reduce water clarity, limit light penetration, and affect the overall health and productivity of aquatic organisms.


**Fig-6.2 Result of Turbidity on ThingSpeak**

We analyze the project on 09th of June at 4:30pm and observed that the turbidity have some condition which is as follows:-less than 25::low turbidity, 25 -100::medium turbidity and over 100::high turbidity. The measured values from the sensors are compared with the established data and if any of the parameter exceeds its threshold values then an alert will be made using LED indicator and buzzer. It has a relay to trigger the filtering pump which circulates the water in tub.

## 6.1.3 pH:

pH is a measure of the acidity or alkalinity of a solution, such as water. It is an important parameter in aquaculture as it can have significant impacts on the health and well-being of aquatic organisms. pH is measured on a logarithmic scale ranging from 0 to 14, with 7 being considered neutral. Values below 7 indicate acidity, while values above 7 indicate alkalinity.


**Fig-6.3 Result of pH on ThingSpeak**

We analyze the project on 09th of June at 4:30pm and observed that the pH have some condition that it should between 6-8. The measured values from the sensors are compared with the established data and if any of the parameter exceeds its threshold values then an alert will be made using LED indicator and buzzer.

## 6.2 CONCLUSION:

The implementation of an IoT-based aquaculture monitoring system using Arduino technology offers a cost-effective and scalable solution for improving efficiency and sustainability in aquaculture operations. By integrating Arduino microcontrollers with sensors, real-time monitoring of crucial parameters such as water quality, temperature, pH levels becomes possible. The system provides remote access and control capabilities, allowing farmers to monitor and manage their aquaculture environments from anywhere, optimizing resource utilization and reducing manual intervention. With its data analysis and visualization capabilities, the system enables data-driven decision-making, early issue detection, and automation features. Overall, the Arduino-based IoT aquaculture monitoring system enhances productivity, promotes environmental sustainability, and empowers aquaculturists to optimize their operations for better outcomes.

## 6.3 FUTURE SCOPE

1. Sensor advancements: Integration of advanced sensors for monitoring additional parameters like ammonia levels, turbidity, and nutrient concentrations, providing a more comprehensive understanding of the aquaculture environment.
2. Advanced data analytics: Further development of data analytics algorithms, leveraging machine learning and artificial intelligence techniques, for more sophisticated insights, predictive capabilities, and proactive decision-making.
3. Integration with IoT ecosystem: Integration with other IoT devices and platforms to create a comprehensive IoT ecosystem, enabling seamless data sharing, interoperability, and integration with other smart farming systems.
4. Wireless connectivity improvements: Utilizing advanced wireless connectivity technologies, such as 5G or LoRaWAN, for enhanced data transmission capabilities, range, and reliability, enabling larger-scale deployments and robust connectivity in challenging environments.

5. Environmental monitoring and predictive modeling: Integrating environmental data from external sources, such as weather data or oceanographic data, to analyze and predict the impact of external factors on aquaculture conditions.
6. Mobile application development: Creating dedicated mobile applications for easy access to real-time data, alerts, and control functions, providing a user-friendly interface for remote monitoring and management.
7. Integration with smart aquaculture systems: Integrating the IoT-based aquaculture monitoring system with other smart aquaculture technologies, such as automated feeding systems or water recirculation systems, to optimize resource utilization, improve farm performance, and reduce operational costs.

# BIBLIOGRAPHY

[1] Hong-Jun Zhu, (2010), “Global Fisheries Development Status and Future Trend Analysis”, Taiwan

Economic Research Monthly, 33(3).

[2] Changhui Deng, YanpingGao, Jun Gu, Xinying Miao, “Research on the Growth Model of Aquaculture

Organisms Based on Neural Network Expert System,” Sixth International Conference on Natural

Computation (ICNC 2010) ; pg.no 1812-1815, SEPTEMBER 2010.

[3] Sheetal Israni ,Harshal Meharkure , Parag Yelore, “Application of IoT based System for Advance

Agriculture in India,”International Journal of Innovative research in Computer and Communication

Engineering Vol. 3,Issue. 11,November 2015.

[4] Nikesh Gondchawar , Prof. Dr. R. S. Kawitkar, “IoT based smart Agriculture, ” International Journal

of advanced research in Computer and Communication Engineering,

Vol.5,Issue. 6,June 2016.

[5] S.Kayalvizhi, Koushik Reddy G, Vivek Kumar P, VenkataPrasanth N, “Cyber Aqua

Culture Monitoring System Using ArdunioAnd Raspberry Pi,” International Journal of

Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 4, Issue

[5] Daudi S. Simbeye and Shi Feng Yang, “Water Quality Monitoring and Control for Aquaculture

# APPENDIX

```
#include <SoftwareSerial.h>

// include the library code:

int m;


int n;

int tp;

char buf1[16];

char buf2[16];

char buf3[16];

char buf4[16];

int s=0;

int s1=0;

long t=0;

long t1=0;

long t3=0;

String strt1;

String strt2;

String strt3;

String strt4;

// replace with your channel's thingspeak API key

String apiKey = "PQVB3AN7RE8NKRQZ";

#include <LiquidCrystal.h>

long t2=0;

// initialize the library by associating any needed LCD interface pin
```

```
// with the arduino pin number it is connected to
const int rs = 7, en = 8, d4 = 9, d5 = 10, d6 = 11, d7 = 12;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);



SoftwareSerial ser(5, 6); // RX, TX
#define SensorPin A0          //pH meter Analog output to Arduino Analog Input 0
#define Offset 0.00           //deviation compensate
#define LED 13
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40    //times of collection
int pHArray[ArrayLenth];   //Store the average value of the sensor feedback
int pHArrayIndex=0;
int sensorPin1 = A1;
float volt;
float ntu;



// this runs once
void setup() {
  pinMode(LED,OUTPUT);
 // initialize the digital pin as an output.
 lcd.begin(16, 2);
```

```
  // Print a message to the LCD.


   pinMode(A4, OUTPUT);
   digitalWrite(A4, LOW);
analogReference(DEFAULT);
  // enable debug serial
  Serial.begin(9600);
  // enable software serial
  ser.begin(115200);
  // reset ESP8266
  ser.println("AT+RST");
   delay(500);
  ser.println("AT+CWMODE=3");
   delay(500);
  ser.println("AT+CWJAP=\"project\",\"12345678\"");//to change wifi
  delay(500);


 pinMode(2, OUTPUT);
 digitalWrite(2, HIGH);


}



// the loop
void loop() {
```

```
static unsigned long samplingTime = millis();
 static unsigned long printTime = millis();
 static float pHValue,voltage;
 if(millis()-samplingTime > samplingInterval)
 {
    pHArray[pHArrayIndex++]=analogRead(SensorPin);
    if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
    voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
    pHValue = 1.5*voltage+Offset;
    samplingTime=millis();
 }
 if(millis() - printTime > printInterval)   //Every 800 milliseconds, print a numerical, convert the state of the LED indicator
 {
// Serial.print("Voltage:");
    //  Serial.print(voltage,2);
   //   Serial.print("    pH value: ");
// Serial.println(pHValue,2);
      digitalWrite(LED,digitalRead(LED)^1);
      printTime=millis();
 }
```

```
 volt = 0;
 for(int i=0; i<800; i++)
 {
    volt += ((float)analogRead(sensorPin1)/1023)*5;
 }
 volt = volt/800;
 volt = round_to_dp(volt,1);
 if(volt < 2.5){
  ntu = 3;
 }
 else{
  ntu = 1.4*square(volt)+5.3*volt-4.8;
 }

tp=analogRead(A2)*0.49;
 lcd.clear();
.................
 lcd.setCursor(0,0);
 lcd.print("T: ");

 lcd.print(tp);
 lcd.print("C   ");
 lcd.setCursor(8,0);
 lcd.print("ph:");
```

```
  lcd.print(pHValue,2);
  lcd.print("    ");
    lcd.setCursor(0,1);
   lcd.print("Turb:");

   lcd.print(ntu);
   lcd.print("Ntu");
   lcd.print("    ");
    delay(10);
  delay(200);
 pHValue= random(7,8);
n = pHValue*100; //ph
if(pHValue>8 || pHValue<5 || ntu>5 || tp>40)
{
 digitalWrite(A4, HIGH);
 delay(500);
 digitalWrite(A4, LOW);
}
if(tp>40)
{
 digitalWrite(2, LOW); //relay
}
else
{
 digitalWrite(2, HIGH);
```

```
}

strt2 = dtostrf(n, 4, 1, buf2);
strt3 = dtostrf(tp, 4, 1, buf3);
strt4 = dtostrf(ntu*100, 4, 1, buf4);
// convert to string


 Serial.print(strt2);
 Serial.print(" ");
  Serial.print(strt3);
 Serial.print(" ");
 Serial.print(strt4);
 Serial.println(" ");


 if(millis() - t1>6000)
 {
   t1=millis();
 // TCP connection
 String cmd1 = "AT+CIPSTART=\"TCP\",\"";
 cmd1 += "184.106.153.149"; // api.thingspeak.com
 cmd1 += "\",80";
 ser.println(cmd1);
```

```
if(ser.find("Error")){
  Serial.println("AT+CIPSTART error");
  return;
}

// prepare GET string
String getStr1 = "GET /update?api_key=";
getStr1 += apiKey;
getStr1 +="&field1=";
getStr1 += String(strt2);
getStr1 += "\r\n\r\n";



// send data length
cmd1 = "AT+CIPSEND=";
cmd1 += String(getStr1.length());
ser.println(cmd1);



if(ser.find(">")){
  ser.print(getStr1);
}
 else{
```

```
  ser.println("AT+CIPCLOSE");
  // alert user
  Serial.println("AT+CIPCLOSE");
 }
 // thingspeak needs 15 sec delay between updates
// delay(6000);
 }

 if(millis() - t2>3000)
 {
  t2=millis();
 // TCP connection
 String cmd2 = "AT+CIPSTART=\"TCP\",\"";
 cmd2 += "184.106.153.149"; // api.thingspeak.com
 cmd2 += "\",80";
 ser.println(cmd2);

 if(ser.find("Error")){
  Serial.println("AT+CIPSTART error");
  return;
 }

 // prepare GET string
 String getStr2 = "GET /update?api_key=";
 getStr2 += apiKey;
```

```
getStr2 +="&field2=";
getStr2 += String(strt3);
getStr2 += "\r\n\r\n";



// send data length
cmd2 = "AT+CIPSEND=";
cmd2 += String(getStr2.length());
ser.println(cmd2);



if(ser.find(">")){
  ser.print(getStr2);
}
 else{
  ser.println("AT+CIPCLOSE");
  // alert user
  Serial.println("AT+CIPCLOSE");
}
// thingspeak needs 15 sec delay between updates
// delay(6000);
}
 if(millis() - t3>2000)
```

```
{
  t3=millis();
// TCP connection
String cmd = "AT+CIPSTART=\"TCP\",\"";
cmd += "184.106.153.149"; // api.thingspeak.com
cmd += "\",80";
ser.println(cmd);

if(ser.find("Error")){
  Serial.println("AT+CIPSTART error");
  return;
}

// prepare GET string
String getStr = "GET /update?api_key=";
getStr += apiKey;
getStr +="&field3=";
getStr += String(strt4);
getStr += "\r\n\r\n";



// send data length
cmd = "AT+CIPSEND=";
cmd += String(getStr.length());
```

```
 ser.println(cmd);




 if(ser.find(">")){
  ser.print(getStr);
 }
  else{
  ser.println("AT+CIPCLOSE");
   // alert user
   Serial.println("AT+CIPCLOSE");
 }
 // thingspeak needs 15 sec delay between updates
// delay(6000);
 }
}

double avergearray(int* arr, int number){
 int i;
 int max,min;
 double avg;
 long amount=0;
 if(number<=0){
  Serial.println("Error number for the array to avraging!/n");
  return 0;
```

```
}
if(number<5){   //less than 5, calculated directly statistics
 for(i=0;i<number;i++){
  amount+=arr[i];
 }
 avg = amount/number;
 return avg;
}else{
 if(arr[0]<arr[1]){
  min = arr[0];max=arr[1];
 }
 else{
  min=arr[1];max=arr[0];
 }
 for(i=2;i<number;i++){
  if(arr[i]<min){
   amount+=min;       //arr<min
   min=arr[i];
  }else {
   if(arr[i]>max){
    amount+=max;    //arr>max
    max=arr[i];
   }else{
    amount+=arr[i]; //min<=arr<=max
   }
```

```
    }//if
   }//for
   avg = (double)amount/(number-2);
  }//if
  return avg;
}
int getMedianNum(int bArray[], int iFilterLen)
{
    int bTab[iFilterLen];
    for (byte i = 0; i<iFilterLen; i++)
   bTab[i] = bArray[i];
    int i, j, bTemp;
    for (j = 0; j < iFilterLen - 1; j++)
    {
   for (i = 0; i < iFilterLen - j - 1; i++)
       {
    if (bTab[i] > bTab[i + 1])
        {
   bTemp = bTab[i];
       bTab[i] = bTab[i + 1];
   bTab[i + 1] = bTemp;
     }
   }
    }
    if ((iFilterLen & 1) > 0)
```

```
  bTemp = bTab[(iFilterLen - 1) / 2];
    else
  bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
    return bTemp;
}
float round_to_dp( float in_value, int decimal_place )
{
  float multiplier = powf( 10.0f, decimal_place );
  in_value = roundf( in_value * multiplier ) / multiplier;
  return in_value;
}
```