IoT- BASED WATER QUALITY MONITORING SYSTEM

AIM:

The aim of the project is to design and Internet of Things enabled water quality monitoring system that utilizes pH and TDS(Total Dissolved Solids) sensors.

The main objectives of the project are:

i) To design and construct a hardware setup consisting of pH sensor, TDS sensor, node MCU.

ii) To collect data and perform analysis and data visualization in real time using Thingspeak.

HARDWARE REQUIREMENTS:

Node MCU, pH sensor, TDS sensor, resistors, breadboard, jumper wires.

SOFTWARE REQUIREMENTS:

Arduino IDE, ThingSpeak cloud

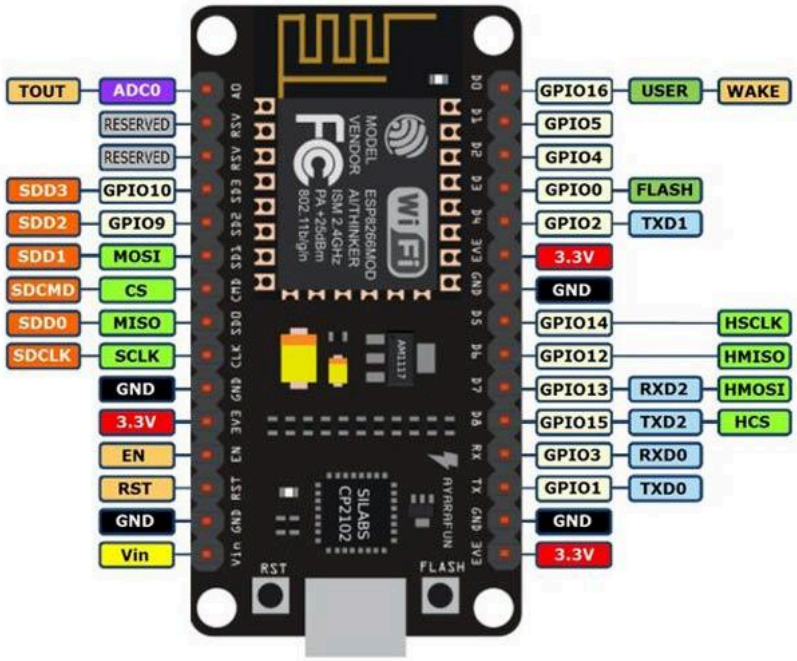
THEORY:

NodeMCU

NodeMCU is an open-source development board that is built around the ESP8266 Wi-Fi module. It is widely used due to its compact size, low cost, and built-in Wi-Fi capabilities.

Features and Specifications

- Microcontroller: NodeMCU is powered by the ESP8266 microcontroller, which integrates a 32-bit RISC processor running at 80MHz. It has 4MB of flash memory for program storage.
- Wifi Connectivity: One of the key features of NodeMCU is its built-in Wi-Fi module, which allows for easy connectivity to wireless networks and the internet. It supports 802.11 b/g/n protocols and can act as a client or an access point.
- Pins: NodeMCU provides a number of General Purpose Input/Output(GPIO) pins that can be used to interface with external devices and sensors. These pins can be configured as digital input/output or as analog inputs.
- USB Interface: NodeMCU has a micro USB port for power supply and programming purposes. It can be easily connected to a computer for programming and debugging.
- Development Environment: NodeMCU can be programmed using the Arduino IDE(Integrated interface for writing, compiling and uploading code to the NodeMCU board.
- Libraries and Community Support: NodeMCU has a vast ecosystem of libraries and resources available online, making it easier to integrate with various sensors, modules and platforms.



pH SENSOR:

The pH sensor plays a crucial role in the IoT-based water quality monitoring system as it provides essential information about the acidity or alkalinity of the water being tested. This sensor utilizes the principles of electrochemistry to measure the pH level accurately. It consists of two electrodes: a glass electrode and a reference electrode.

Glass electrode:

The glass electrode is the sensing component of the pH sensor. It is constructed using a specialized glass membrane that is sensitive to hydrogen ions (H+). When this membrane comes into contact with water, it interacts with the H+ ions present in the solution. This interaction generates an electrical potential based on the concentration of H+ ions. The glass electrode acts as a pH-sensitive probe, converting the ionic activity of H+ ions into a voltage signal.

Reference electrode:

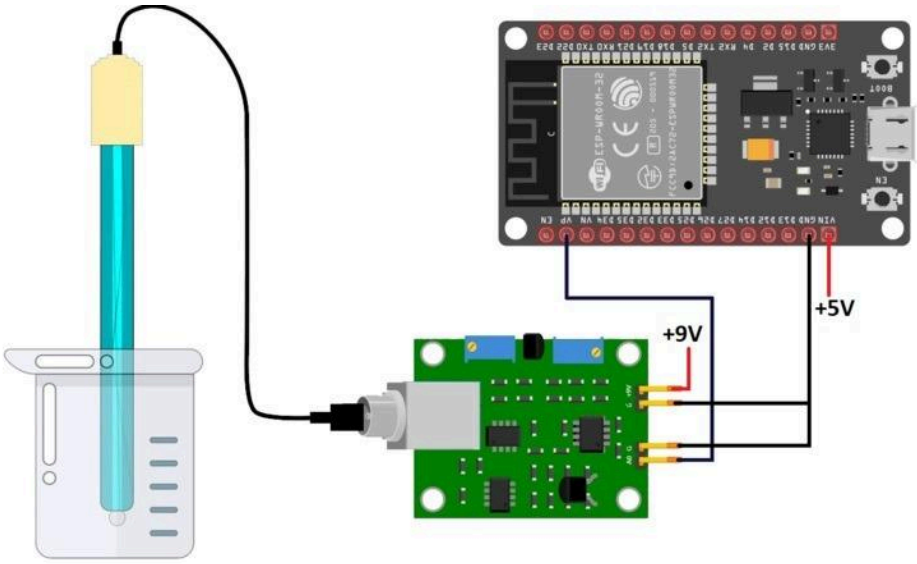
The reference electrode serves as a stable reference point for the pH measurement. It is typically filled with a reference solution, commonly potassium chloride (KCl), which maintains a constant electrical potential. The reference electrode helps establish a baseline against which the potential generated by the glass electrode can be measured accurately.

Working Principle:

When the water sample is tested, the glass electrode and the reference electrode are immersed in the water. The glass electrode's membrane interacts with the H+ ions in the water, while the reference electrode maintains a stable potential. The potential difference between the two electrodes is measured, indicating the ionic activity of H+ ions in the water sample.

Conversion to pH value:

To convert the measured potential difference into a pH value, a calibration process is necessary. This involves comparing the sensor's output voltage at different known pH levels using standard buffer solutions. A calibration curve or equation is then established, relating the measured voltage to the corresponding pH value. Once the calibration is done, the sensor can provide accurate pH measurements.



1. pH SENSOR

TDS SENSOR

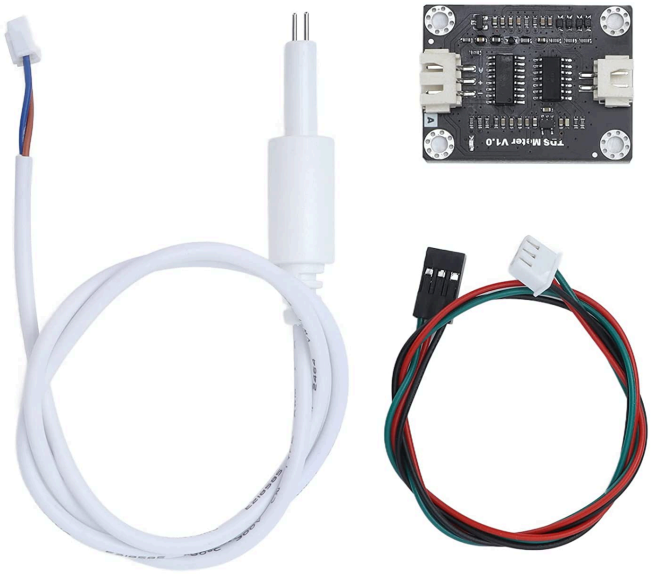
The TDS (Total Dissolved Solids) sensor is a critical component of the IoT-based water quality monitoring system, designed to measure the concentration of dissolved solids in water. The TDS sensor operates based on the principle that the conductivity of water increases with the presence of dissolved solids.

Specification:

- Input Voltage: The TDS sensor module requires a DC input voltage ranging from 3.3V to 5.5V. This voltage range ensures compatibility with various power supply sources commonly used in IoT applications.
- Output Voltage: The TDS sensor provides an output voltage ranging from 0V to 2.3V. This voltage signal is proportional to the conductivity of the water sample and can be used to infer the TDS level present in the water.
- Working Current: The TDS sensor module consumes a working current between 3mA to 6mA during operation. This low current requirement ensures efficient power usage and allows for integration into low-power IoT devices.

- TDS Measurement Range: The TDS sensor can measure TDS levels within the range of 0 to 1000 parts per million (ppm). This range covers a wide spectrum of water quality conditions, enabling the detection of both low and high TDS concentrations.

- TDS Measurement Accuracy: The TDS sensor has an accuracy of ±10% Full Scale (F.S.) at a temperature of 25 ℃. This indicates that the measured TDS value may deviate by up to 10% from the true TDS value at maximum range. It is important to consider temperature compensation techniques for accurate measurements in different environmental conditions.
- Module Interface: The TDS sensor module employs an XH2.54-3P interface. This interface provides a standardized connection for seamless integration with other components or microcontrollers within the IoT-based water quality monitoring system.
- Electrode Interface: The TDS sensor utilizes an XH2.54-2P electrode interface. This interface enables the connection of electrodes specifically designed for TDS measurements, allowing for efficient and reliable detection of dissolved solids in the water.



2.TDS Sensor

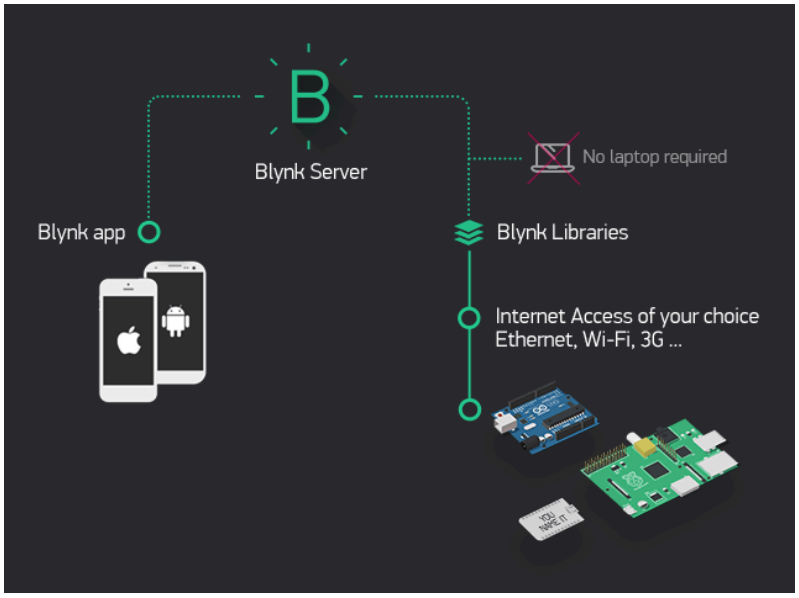
BLYNK:

Blynk can be used in a system as an integral part of an IoT solution. It provides a user-friendly interface for creating custom dashboards and controlling connected devices, making it suitable for various IoT applications and systems.

FEATURES:

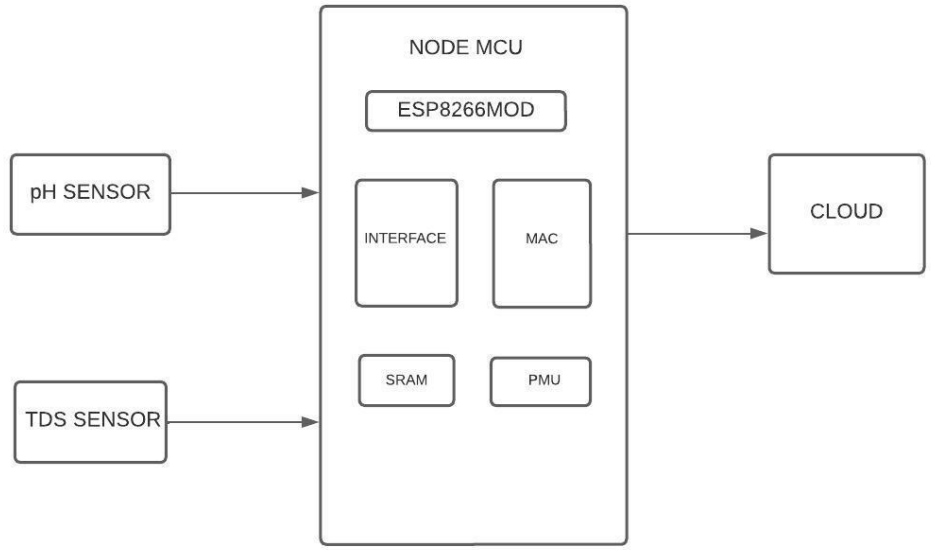
- Sensor Integration: Blynk can be used in a system as an integral part of an IoT solution. It provides a user-friendly interface for creating custom dashboards and controlling connected devices, making it suitable for various IoT applications and systems.
- Data monitoring and control: Blynk allows you to create a customized dashboard within the app, where you can monitor real-time data from your sensors. You can also control connected devices or trigger actions based on sensor readings or user inputs, providing a user-friendly interface for system control and management.
- Widgets library: Blynk offers a wide range of pre-built widgets that users can select from to add functionality to their dashboard. Examples include buttons, sliders, graphs, gauges, and notifications.
- Device connectivity: Blynk supports integration with popular development boards and microcontrollers, such as Arduino, Raspberry Pi, ESP8266, and many others. It provides libraries and code examples to facilitate the connection between the devices and the Blynk app.
- Cloud Integration: Blynk provides cloud storage for data logging and retrieval. Users can store and access historical data from their IoT projects, enabling analysis and tracking of trends over time.

The overview of Blynk is given below:



3.Outlook of Blynk

BLOCK DIAGRAM:

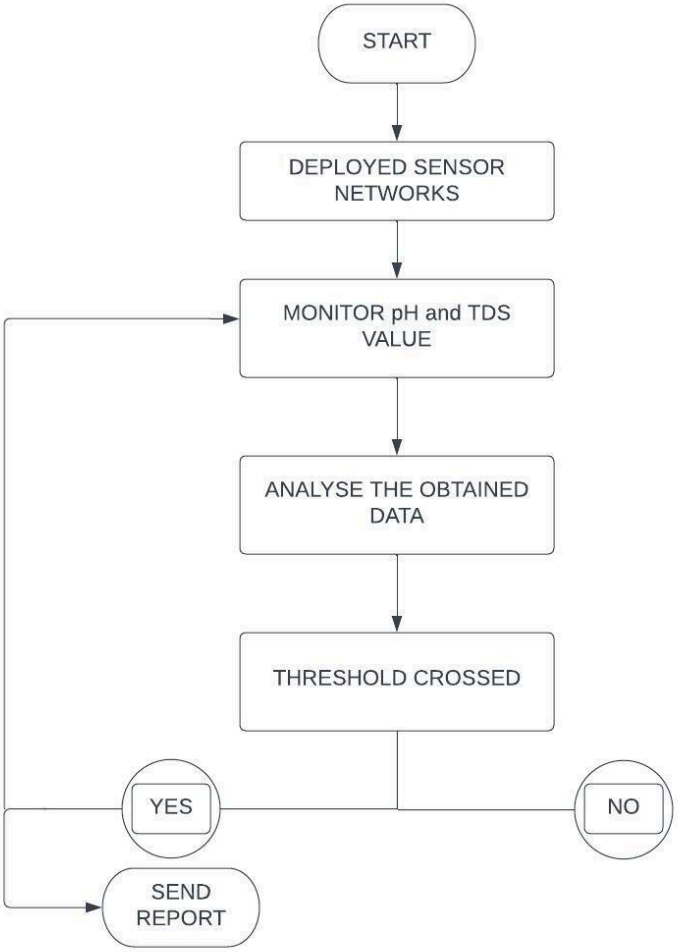


ALGORITHM:

1. Set up hardware:
1.1 Gather the required components, including a NodeMCU development board and sensors for data collection.
1.2 Connect the sensors to the appropriate pins on the NodeMCU board as per the sensor's specifications.
1.3 Ensure that the NodeMCU is connected to a stable power source.
2. Set up Blynk:
2.1 Create an account on Blynk and create a new account.
2.2 Obtain the authentication token.
3. Set up Arduino IDE:
3.1 Install the Arduino IDE on your computer.
3.2 Install the necessary libraries for the NodeMCU and the sensors you are using.
3.3 Configure the Arduino IDE for the NodeMCU board by selecting the appropriate board and port settings.
4. Write and upload the data:
4.1 Open the Arduino IDE and create a new sketch.
4.2 Write the code to read data from the connected sensors and connect it to Blynk through the authentication token.
4.3 Verify the code for any errors or warnings and upload it to the NodeMCU board.

5.Monitor the pH and TDS values:
5.1Open the Blynk app on your smartphone and ensure it is connected to the internet.
5.2The pH and TDS values should be displayed in the Blynk app in real-time.

FLOWCHART:



ARDUINO IDE CODE FOR Ph SENSOR:

```
const int potPin=A0;
float ph;
float Value=0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(potPin,INPUT);
  delay(1000);
}
 void loop(){

        Value= analogRead(potPin);
        Serial.print(Value);
        Serial.print(" | ");
        float voltage=Value*(3.3/4095.0);
        ph=(3.3*voltage);
        Serial.println(ph);
    delay(500);

 }
```

ARDUINO IDE CODE FOR TDS SENSOR:

```
#define TdsSensorPin A0
#define VREF 3.3              // analog reference voltage(Volt) of the ADC
#define SCOUNT  30            // sum of sample point

int analogBuffer[SCOUNT];         // store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0;
int copyIndex = 0;

float averageVoltage = 0;
float tdsValue = 0;
float temperature = 23;       // current temperature for compensation

// median filtering algorithm
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;
}

void setup() {
          Serial.begin(115200);
  pinMode(TdsSensorPin,INPUT);
}


void loop()
{
  // put your main code here, to run repeatedly:
  static unsigned long analogSampleTimepoint = millis();
  if(millis()-analogSampleTimepoint > 40U)
  {       //every 40 milliseconds,read the analog value from the ADC
          analogSampleTimepoint = millis();
          analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin);       //read the
analog value and store into the buffer
          analogBufferIndex++;
          if(analogBufferIndex == SCOUNT)
          {
          analogBufferIndex = 0;
          }
  }
```

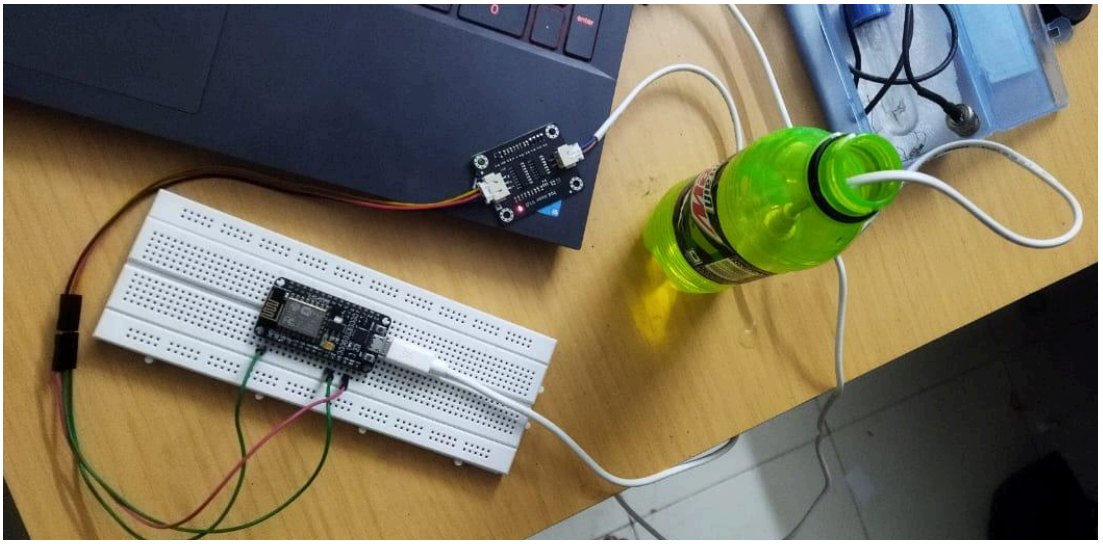
```
static unsigned long printTimepoint = millis();
  if(millis()-printTimepoint > 800U)
  {
        printTimepoint = millis();
        for(copyIndex=0; copyIndex<SCOUNT; copyIndex++)
        {
        analogBufferTemp[copyIndex] = analogBuffer[copyIndex];

        // read the analog value more stable by the median filtering algorithm, and
convert to voltage value
        averageVoltage = getMedianNum(analogBufferTemp,SCOUNT) * (float)VREF /
1024.0;

        //temperature compensation formula: fFinalResult(25^C) =
fFinalResult(current)/(1.0+0.02*(fTP-25.0));
        float compensationCoefficient = 1.0+0.02*(temperature-25.0);
        //temperature compensation
        float compensationVoltage=averageVoltage/compensationCoefficient;

        //convert voltage value to tds value

tdsValue=(133.42*compensationVoltage*compensationVoltage*compensationVoltage -
255.86*compensationVoltage*compensationVoltage +
857.39*compensationVoltage)*0.5;

        //Serial.print("voltage:");
        //Serial.print(averageVoltage,2);
        //Serial.print("V   ");
Serial.print("TDS Value:");
        Serial.print(tdsValue,0);
        Serial.println("ppm");
        }
  }
}
```

ARDUINO IDE CODE FOR TDS SENSOR WITH BLYNK:

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

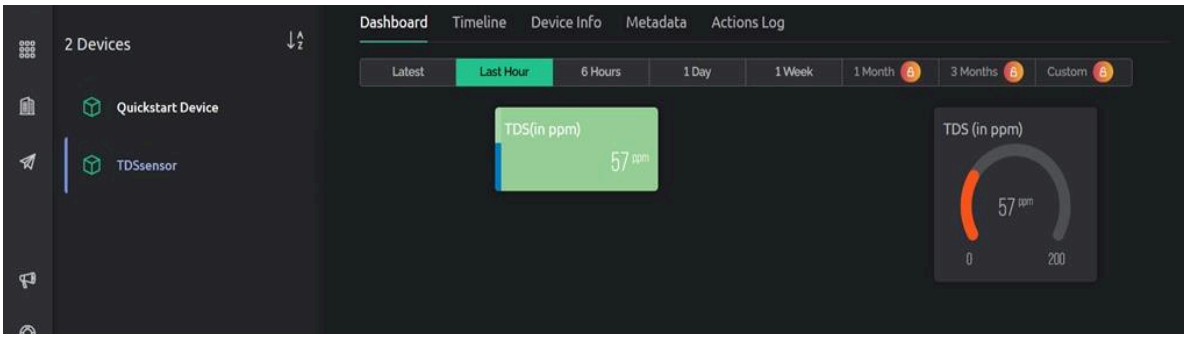
```
#define BLYNK_TEMPLATE_ID "TMPL3SfZ8YPqm"
#define BLYNK_TEMPLATE_NAME "TDSsensor"
#define BLYNK_AUTH_TOKEN "jqI0EfvV9KwIVbI3k1n2vxB06w1y-6jd"

char auth[] = "jqI0EfvV9KwIVbI3k1n2vxB06w1y-6jd";
char ssid[] = "Galaxy M31s234D";
char pass[] = "sanjay43203";

#define TDS_SENSOR_PIN A0

void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
}

void loop()
{
  Blynk.run();

  int tdsValue = analogRead(TDS_SENSOR_PIN);
  Blynk.virtualWrite(V0, tdsValue);
  delay(1000);
}
```

CONNECTION SETUP (TDS SESNOR) :



4.Connection setup for TDS sensor

BLYNK OUTPUT:



5. Blynk setup

INFERENCE:

The development of an IoT-based water quality monitoring system for pH and TDS measurement using pH and TDS sensors has been built. The project aimed to design and implement an efficient solution for real-time monitoring of water quality targeting pH value and TDS value measurements. Through the utilization of NodeMCU, an open-source development board, the system integrated the pH and TDS sensors, enabling data acquisition and transmission to Blynk for analysis and visualization along with alert message in the Blynk console.

RESULT:

The implemented IoT-based water quality monitoring system effectively performs real-time measurement and monitoring of pH and TDS values in water. The system incorporates a pH sensor and a TDS sensor, enabling measurement of these parameters. The acquired data is transmitted to the Blynk application for analysis, visualization and alert message.

IoT MINI PROJECT
IoT BASED WATER QUALITY MONITORING SYSTEM

SRI SUDARSHAN T R (203002109)
SRUTI K A (203002110)
SUBISHRAM S (203002111)
SANTHOSH R (203002305)