Introduction

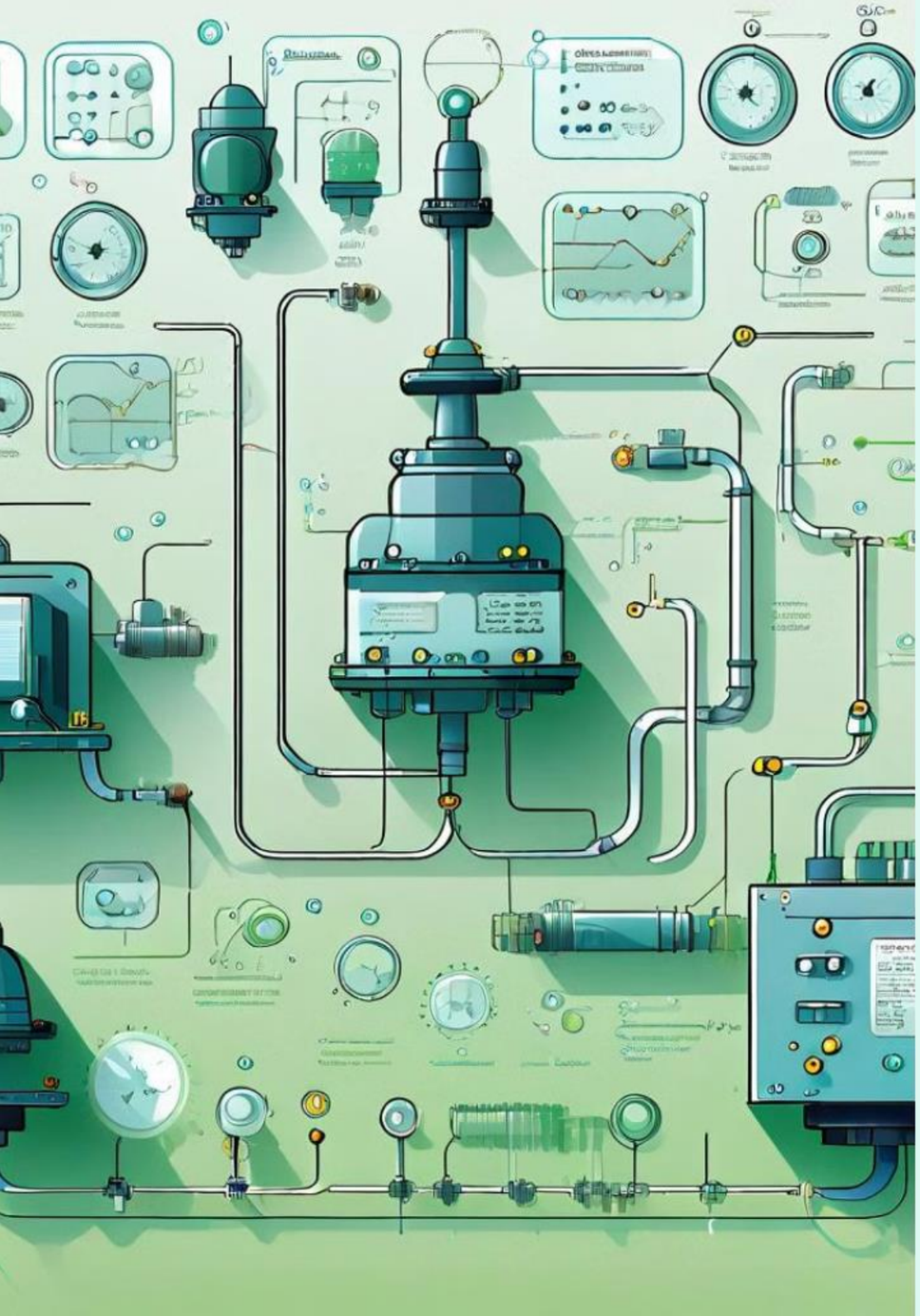
Team-18

Team

1. Virat Garg
2. Varun Gupta
3. Raghav Grover
4. Khwaish Garg

Project Idea

Our IOT-based water monitoring system aims to automate water management using sensors to measure key parameters like TDS, Temperature and pH Readings .It also allows user to see live water analysis along with controlling the supply of water according to thresholds for various sensors.



Motivation behind the project

Our team found out that there is a need for a smart water quality monitoring system which allows users, to monitor data in real time and move the water to a waste storage whenever water crosses a benchmark water quality measures .We also help user get the notification when water is undrinkable and our prototype right now can store data for about 6 days.

Required Components

1. ESP 32
2. pH sensor
3. Temperature sensor
4. TDS (Total Dissolved Solids) sensor
5. Solenoid valve for water flow control
6. Jumper wires and breadboard for circuit connections
7. Power supply (such as a 9V battery or USB power)
8. Power supply of 12V

Output

Data Visualization

Displays measurements in the form of line graphs.

1

2

Real-Time Monitoring

Provides instant updates on water quality.

Alerts and Notifications

3

Sends alerts when parameters are out of range and give notifications.

How It Works

1

Sensors Detects

pH, temperature, and TDS sensors continuously monitor water conditions

2

Microcontroller Processes

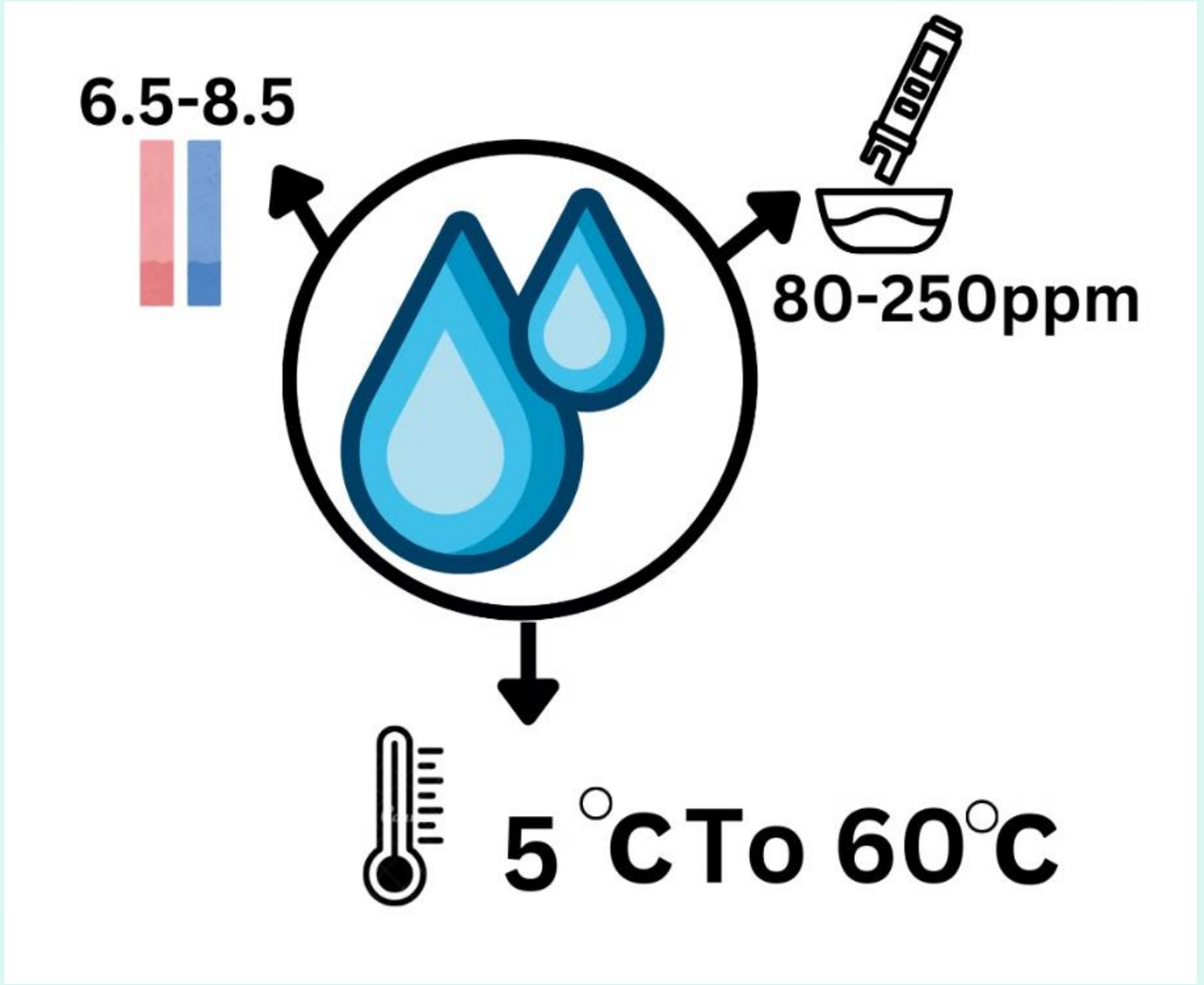
Compares sensor data to target thresholds and triggers actions

3

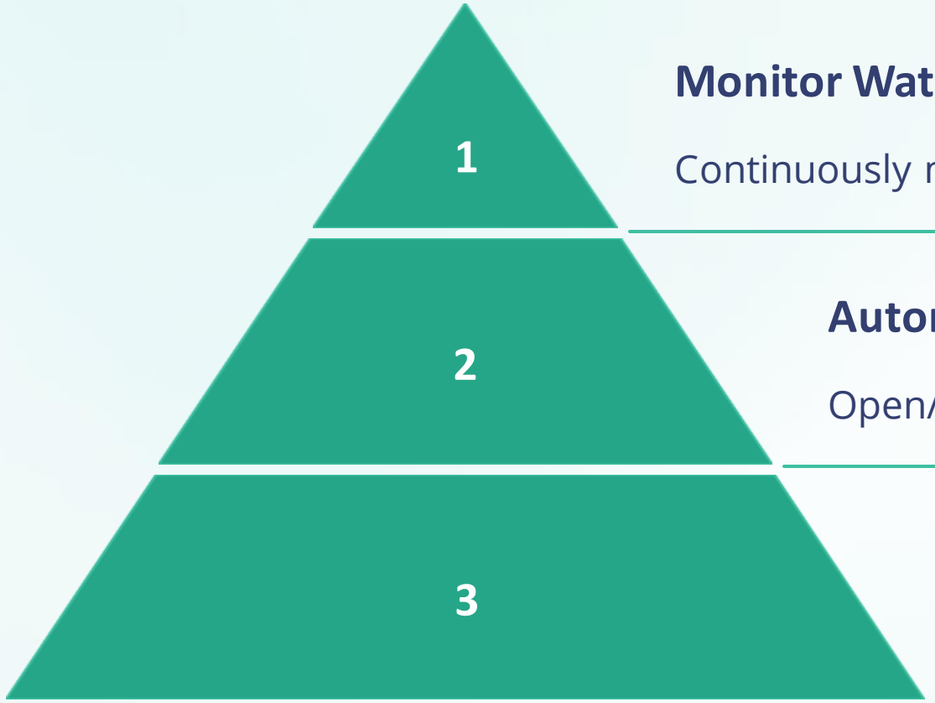
Valve Actuates

Solenoid valve opens or closes to adjust water flow accordingly

Drinking Ranges:



Objective



Monitor Water Quality

Continuously measure pH, temperature, and TDS

Automate Water Treatment

Open/close solenoid valve based on readings

Provide Real-Time Data

Transmit data to mobile app for analysis

The objective of this IoT-based water monitoring system is to continuously measure key water quality parameters - pH, temperature, and total dissolved solids (TDS). Based on these readings, the system will automatically open and close a solenoid valve to adjust the water treatment process. All sensor data will be transmitted to a mobile app, allowing users to monitor water quality in real-time and track historical trends.

Code:

```
#define BLYNK_TEMPLATE_ID "TMPL3-Uob0T2A"
#define BLYNK_TEMPLATE_NAME "Water Quality Monitoring"
#define BLYNK_AUTH_TOKEN "Ioqjo68DvtyREcFMAhAQBow3UKQp0SO0"

#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
BlynkTimer timer;

#include <OneWire.h>
#include <DallasTemperature.h>

#define TDS_SENSOR_PIN 27
#define VREF 3.3
#define SCOUNT 30
int analogBuffer[SCOUNT];
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0;
int copyIndex = 0;
float averageVoltage = 0;
float tdsValue = 0;
float temperature = 0;
float phValue = 0;

const char *ssid = "Varun";
const char *password = "varun123";

const int potPin = A0;
#define ONE_WIRE_BUS 4
const int relayPin = 18;

#define TDS_THRESHOLD 800
#define TMP_THRESHOLD 25
#define PH_THRESHOLD 7

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void myTimer()
{
  Blynk.virtualWrite(V4, temperature);
  Blynk.virtualWrite(V0, phValue);
  Blynk.virtualWrite(V27, tdsValue);
}

void setup()
{
  Serial.begin(115200);
  pinMode(relayPin, OUTPUT);
  pinMode(potPin, INPUT);
  pinMode(TDS_SENSOR_PIN, INPUT);
  delay(1000);
  sensors.begin();
  Blynk.begin(auth, ssid, password);
  timer.setInterval(1000L, myTimer);
}
```

Code:

```
void loop()
{

  sensors.requestTemperatures();
  temperature = sensors.getTempCByIndex(0);

  phValue = readPH();

  static unsigned long analogSampleTimepoint = millis();
  if (millis() - analogSampleTimepoint > 40U)
  {
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(TDS_SENSOR_PIN);
    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];

      averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF / 4096.0;

      float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
      float compensationVoltage = averageVoltage / compensationCoefficient;

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

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print("°C | ph: ");
  Serial.print(phValue);
  Serial.print(" | TDS:");
  Serial.print(tdsValue);
  Serial.println(" ppm");

  Blynk.run();
  timer.run();

  if (tdsValue > TDS_THRESHOLD || temperature > TMP_THRESHOLD || phValue > PH_THRESHOLD)
  {
    digitalWrite(relayPin, LOW);
    Serial.println("Opened the Valve : Water quality exceeds threshold emptying the container");
    if(tdsValue > TDS_THRESHOLD){
      Blynk.logEvent("alert_water_quality", String("TDS Crossed the Threshold Value! TDS: ") +
tdsValue);
    }
    else if(temperature > TMP_THRESHOLD){
      Blynk.logEvent("alert_water_quality", String("Temperature Crossed the Threshold Value! Temp: ") +
temperature + String(" °C"));
    }
  }
  else
  {
    digitalWrite(relayPin, HIGH);
  }
}
```

Code:

```
float readPH()
{
  float Value = analogRead(potPin);
  float voltage = Value * (3.3 / 4095.0);
  float ph = (3.3 * voltage);
  delay(500);
  return ph;
}

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;
}
```

Implementation

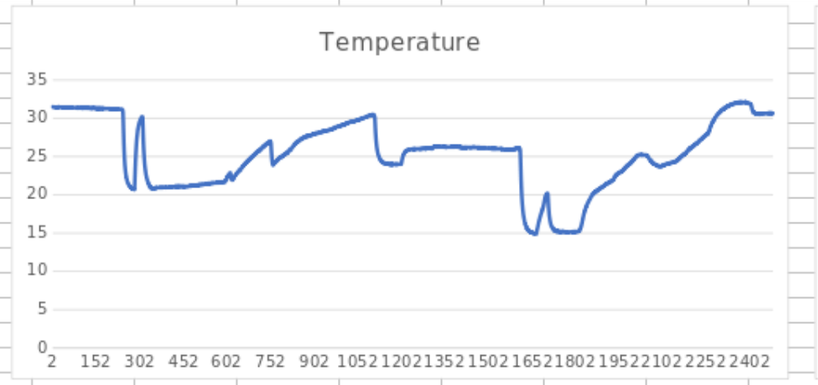
1.All the readings of the pH, turbidity and TDS sensors is used to control the solenoid valve. We have connected the relay module and pH sensor with DC power supplies. The readings of all 3 of them is processed in ESP 32 microcontroller and code is run using arduino IDE with installed libraries.

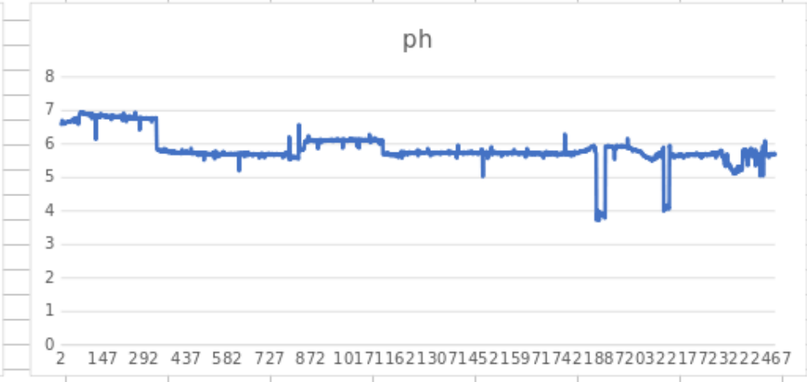
2. These readings are moved on to BLYNK application as well as website. Which gives us real-time data monitoring on our local devices.

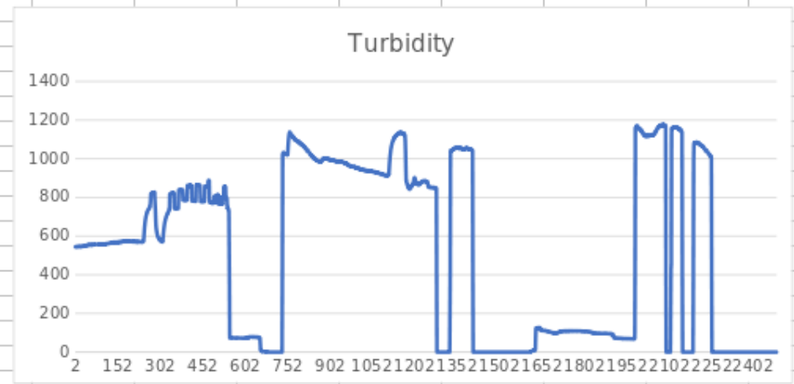
4.When data goes out of bounds(crosses thresholds) solenoid valve is controlled accordingly and a notification reaches the user's device regarding the alarming levels of various sensors.

Graphs:

Temperatu	ph	Turbidity
31.5	6.59	544.26
31.44	6.6	545
31.44	6.61	545
31.44	6.7	545
31.44	6.6	545
31.44	6.6	545.34
31.44	6.6	545.68
31.44	6.61	545.68
31.44	6.6	545.68
31.44	6.62	545.68
31.44	6.64	546.01
31.37	6.6	546.76
31.44	6.64	545.68
31.44	6.63	546.01
31.44	6.64	546.01
31.44	6.64	545.68
31.44	6.64	545.68
31.44	6.64	545.68
31.44	6.64	545.68
31.5	6.64	545.27
31.44	6.64	546.01
31.44	6.66	546.01
31.44	6.65	546.35
31.44	6.65	546.35
31.44	6.65	546.69
31.44	6.66	546.69
31.44	6.67	546.69
31.44	6.67	546.69
31.44	6.66	547.02
31.44	6.67	548.71
31.44	6.67	548.71
31.44	6.68	549.73
31.44	6.68	549.73
31.37	6.68	550.82
31.44	6.68	550.4
31.44	6.68	550.74



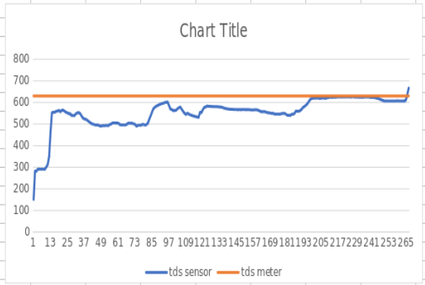


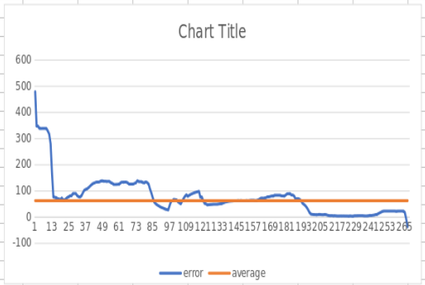


Caliberation:

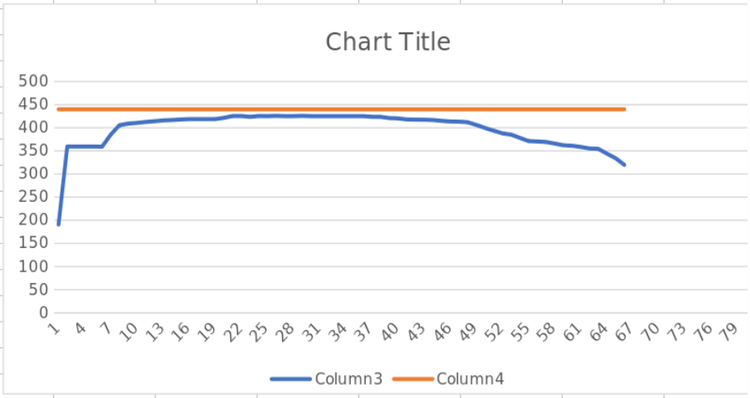
We created many graphs using microsoft excel taking readings of original sample through industrial instruments and reading of our sensors and plotted graphs to observe the average deviation and tried to reduce it as much as possible.

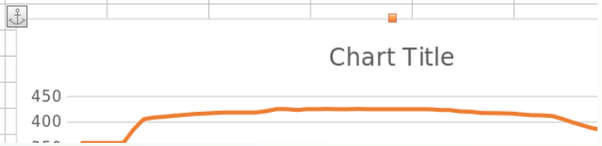
tds sensor	tds meter	error	average
0	150.88	630	479.12	62.9818
0	282.48	630	347.52	62.9818
0	282.13	630	347.87	62.9818
0	291.13	630	338.87	62.9818
0	291.5	630	338.5	62.9818
0	291.13	630	338.87	62.9818
0	291.13	630	338.87	62.9818
0	291.13	630	338.87	62.9818
0	291.13	630	338.87	62.9818
0	300.78	630	329.22	62.9818
0	312.61	630	317.39	62.9818
0	349.66	630	280.34	62.9818
0	461.46	630	168.54	62.9818
0	552.73	630	77.27	62.9818
0	554.77	630	75.23	62.9818
0	554.77	630	75.23	62.9818
0	559.28	630	70.72	62.9818
0	560.22	630	69.78	62.9818
0	563.3	630	66.7	62.9818
0	556.72	630	73.28	62.9818
0	564.24	630	65.76	62.9818
0	564.24	630	65.76	62.9818
0	559.28	630	70.72	62.9818
0	554.36	630	75.64	62.9818
0	551.92	630	78.08	62.9818
0	547.86	630	82.14	62.9818
0	547.86	630	82.14	62.9818
0	539.41	630	90.59	62.9818





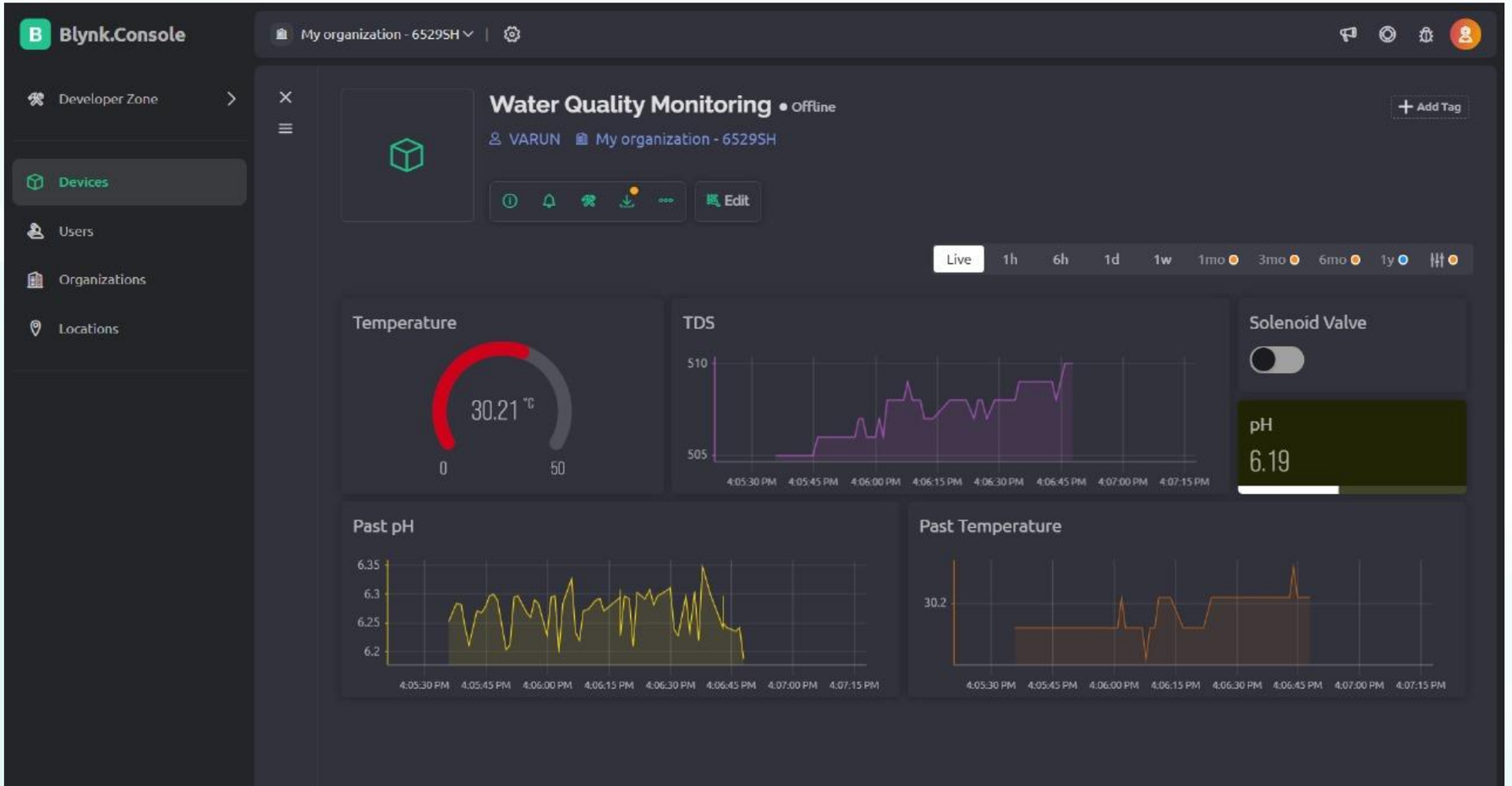
190.9	440	249.1
359.48	440	80.52
359.48	440	80.52
359.48	440	80.52
359.48	440	80.52
359.06	440	80.94
385.31	440	54.69
405.59	440	34.41
409.03	440	30.97
410.51	440	29.49
412.58	440	27.42
414.14	440	25.86
415.84	440	24.16
416.73	440	23.27
417.92	440	22.08
418.82	440	21.18
418.82	440	21.18
418.82	440	21.18
418.82	440	21.18
421.51	440	18.49
425.41	440	14.59



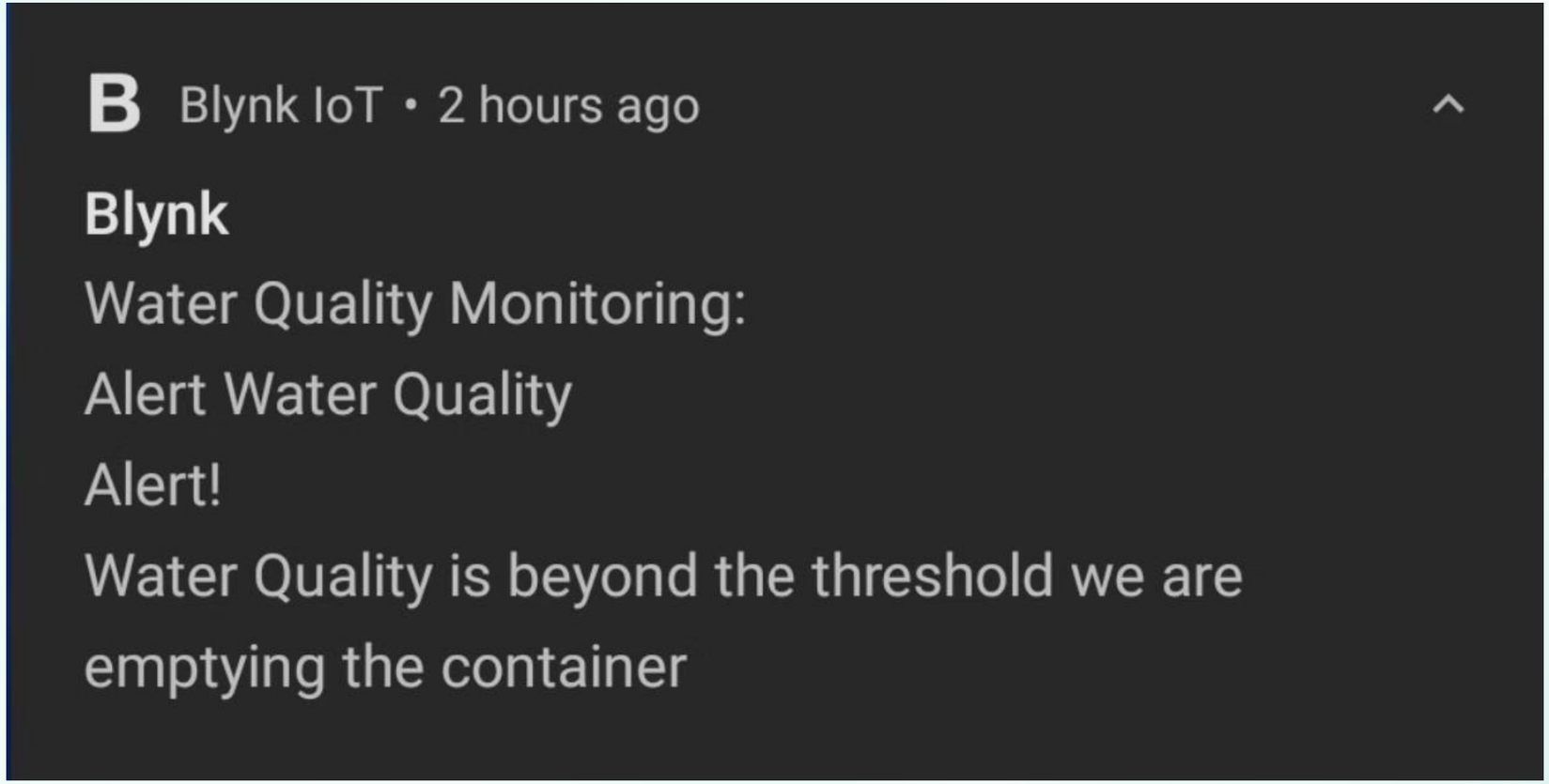


App Interface :

We used 3rd party application(Blynk) to publish our data on website as well as on our application.
We connected it to our different nodes of sensor and provide analysis of data as well as give control of solenoid valve to the user.



Notifications:-



We send notifications like this whenever reading of any sensor goes beyond threshold and also opens the solenoid valve.

Data Analysis:

1. The first graph shows data of pH sensor ,which is uploaded on our website as well as application with some delay.
2. Second graph shows Turbidity and the third one shows the TDS.
3. The solenoid valve is set according to some predefined thresholds to control it according to readings of various sensor.
4. We can store data of up to 6 days but going on large scale we can enable it to store data of up to 30 days.

Failure Analysis

- The issue with our project is that it is not suitable for working in harsh climatic conditions ,currently we set up all the sensors on a cardboard but it will lead to their failure due to their inability to resist water
- Another thing is that our setup is completely dependent on DC power Supply we can ensure a power storing unit to ensure its functioning during power failures.
- Also, our pH sensor was not giving reading up to the mark,its reading is mostly correct but it doesn't show proper variations with change in pH of sample.We also tried its calliberation but weren't much successful.

Further Improvements

- We are planning to add water level sensor to control time for which the valve is open to reduce wastage.
- We also plan to use buzzer with solenoid valve to help user know that levels are crossed.
- We can also use turbidity sensor, adding one more reading to our project and leading to more accuracy.Also currently we are opening the solenoid valve for a fixed delay but later it will check the reading of water level sensor and control accordingly.
- Going at big scale we plan to increase the data storage capacity to at least 6 month from 6 days.
- Currently we are using a 3rd party app to display our data, we can develop our own app and customize it according to our specific needs.
- We can also move towards water treatment depending on the readings ,example using acidic solution to make water drinkable if it's too much basic depending on reading of pH .

Thank You

We, wanted to express our sincere gratitude to our profs as well as our TAs for enabling us to pursue the project and complete it according to expectations. Your support and encouragement means a lot to us, and we are truly grateful for this opportunity.