INTERNET OF THINGS

SMART WATER MANAGEMENT:

PROJECT REPORT

PHASE-5

SUBMITTED BY

TEAM LEADER: MOAMMED JAFFER K

TEAM MEMBERS: HEMA KUMAR S

TEAM MEMBER: NAVEEN KUMAR V

TEAM MEMBER: THUTHESH KUMAR R

TABLE OF CONTENTS

Page(no)

1.Introduction:

Public swimming pools provide a refreshing escape for countless individuals seeking respite from the sweltering heat or an avenue for fitness and recreation. However, beneath the crystal-clear surface of these aquatic sanctuaries lies a complex world of chemical reactions, microbial life, and potential health risks. Ensuring the safety and well-being of swimmers in public pools demands the vigilant oversight of water quality.

Water quality monitoring in public pools is a paramount responsibility of pool operators, management, and regulatory authorities. It encompasses a multifaceted approach aimed at preserving the health, hygiene, and enjoyment of all who venture into the pool's waters. The essence of this monitoring lies in the systematic evaluation and maintenance of various water parameters that collectively define the pool's aquatic environment.

1.1 Aim:

- ➢ To improve water quality by monitoring real-time parameters such as pH value and turbidity.
- ➢ To enhance public health and safety by continuously monitoring the quality of drinking water.
- ➢ To provide a reliable and cost-effective system for real-time water quality monitoring.
- ➢ To enable global accessibility through the use of Internet of Things (IoT) technology.
- ➢ To apply and implement the knowledge and skills acquired during undergraduate education to address the issue of water pollution.

1.2 Objectives:

Develop a real-time IoT-based water quality monitoring system with the following key features:

Cost Reduction:

Create a system that reduces the import cost by using indigenous technology and components, thus promoting self-reliance.

Capacity Building:

Improve local capacity building by enabling skill development and knowledge transfer in the field of water quality monitoring.

Mobile Application Integration:

Provide a feature that allows users to monitor water quality through an Android app for convenient access and real-time data.

Email Alerts:

Implement an alert system that sends email notifications in response to water quality anomalies or issues, ensuring timely response and action.

2. Ideation and proposed solution

2.1 Problem Statement:

- The traditional system to determine water quality is not reliable
- It rely on collecting water samples, testing and analyses in water laboratories that is costly and time consuming
- Human errors can occur and may not provide accurate result
- It lack capability for real-time data capture, analyses and stakeholders can’t do fast dissemination on information for making timely and appropriate decisions.

2.2 Proposed Solution:

The proposed system consists of 4 major stages.

- **Sensing stage**
  - At the sensing stage the four sensors are used that sense the parameters pH, Turbidity, TDS and Temperature of water
- **Data acquisition stage**
  - In this stage the ESP32 microcontroller acquired the sensors data and compute according the program burn in it
- **Wireless data transmission to IoT cloud stage**
  - The computed data is send to the IoT cloud ThingSpeak wirelessly using the Wi-Fi technology
- **Visualization and Monitoring stage**
  - In the last stage we can monitor the water quality via the ThingSpeak dashboard, an Android App, and 16x2 LCD

2.3 Ideation & Brainstorming:

Step-1:Team Gathering and problem Statement

MOHAMMED JAFFER

- what sensors and hardware components will be required to collect and transmit weather data?
- How can the system be designed to handle and process large amounts of data in real-time?
- How can the system be designed to provide accessibility and inclusivity for users with disabilities?
- What measures can be taken to reduce maintenance costs and increase system longevity?
- What type of partnerships and collaborations will be needed to ensure successful integration with other smart city technologies?

HEMA KUMAR S

- How can the system be designed to minimize power usage and maximize efficiency?
- How can machine learning and artificial intelligence be incorporated to improve accuracy and efficiency?
- How can machine learning and artificial intelligence be incorporated to improve accuracy and efficiency?
- How can the system be designed to minimize waste and maximize sustainability?
- What type of compatibility will be required to ensure the system can be easily integrated with existing infrastructure and technology?

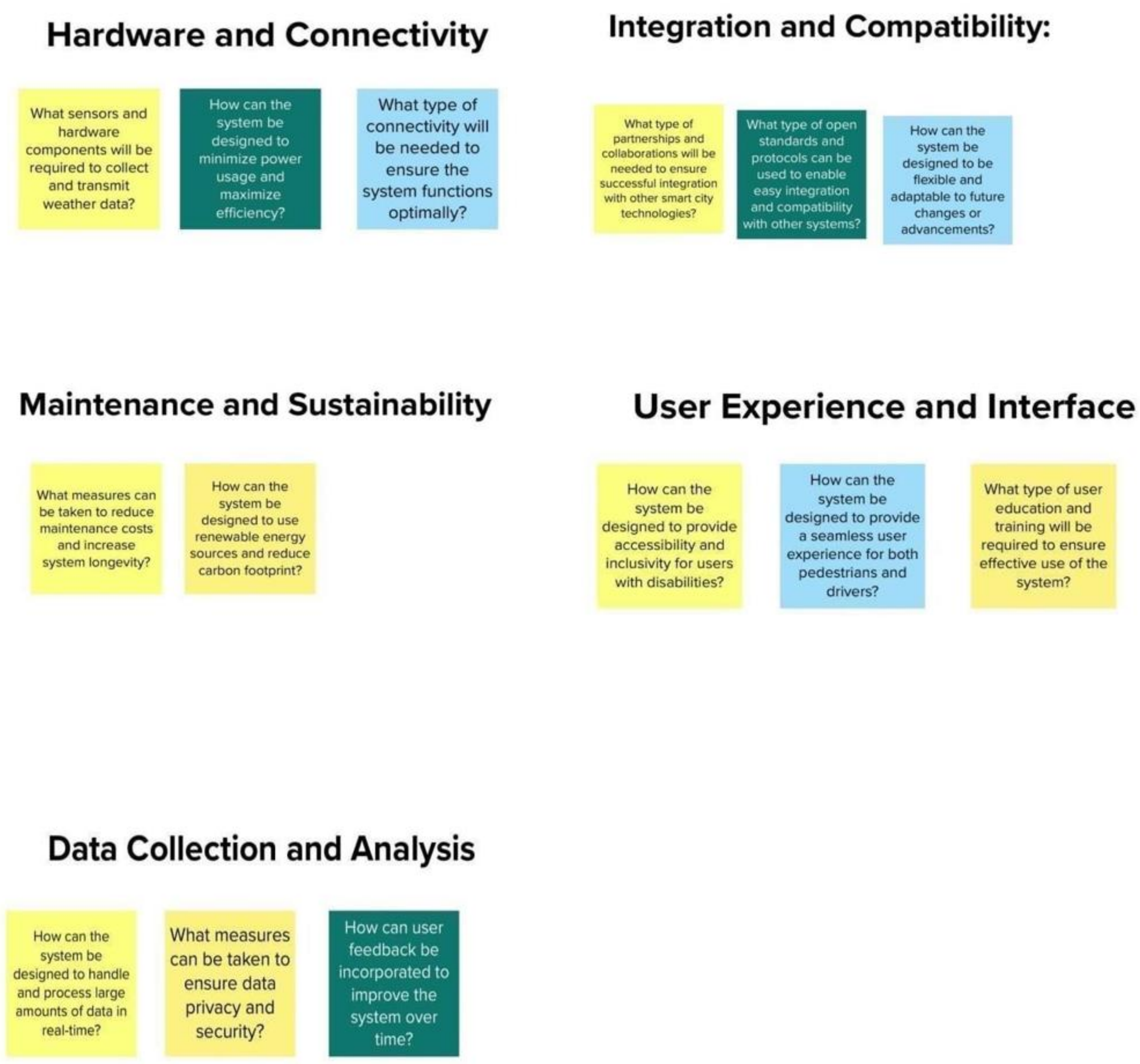
NAVEEN KUMAR

- What type of connectivity will be needed to ensure the system functions optimally?
- How can the data collected from sensors be analyzed to determine the optimal levels?
- How can the system be designed to provide a seamless user experience for both pedestrians and drivers?
- What type of business model can be used to ensure the system remains profitable and scalable?
- How can the system be designed to be flexible and adaptable to future changes or advancements?

THUTHESH KUMAR

- What is the optimal placement of sensors and hardware components to ensure accurate data collection?
- What measures can be taken to ensure data privacy and security?
- What type of user education and training will be required to ensure effective use of the system?
- How can the system be designed to use renewable energy sources and reduce carbon footprint?
- How can the system be designed to ensure interoperability with other systems and devices?

Step-2: Brainstorm,Idea Listing and Grouping



Hardware and Connectivity

- What sensors and hardware components will be required to collect and transmit weather data?
- How can the system be designed to minimize power usage and maximize efficiency?
- What type of connectivity will be needed to ensure the system functions optimally?

Integration and Compatibility:

- What type of partnerships and collaborations will be needed to ensure successful integration with other smart city technologies?
- What type of open standards and protocols can be used to enable easy integration and compatibility with other systems?
- How can the system be designed to be flexible and adaptable to future changes or advancements?

Maintenance and Sustainability

- What measures can be taken to reduce maintenance costs and increase system longevity?
- How can the system be designed to use renewable energy sources and reduce carbon footprint?

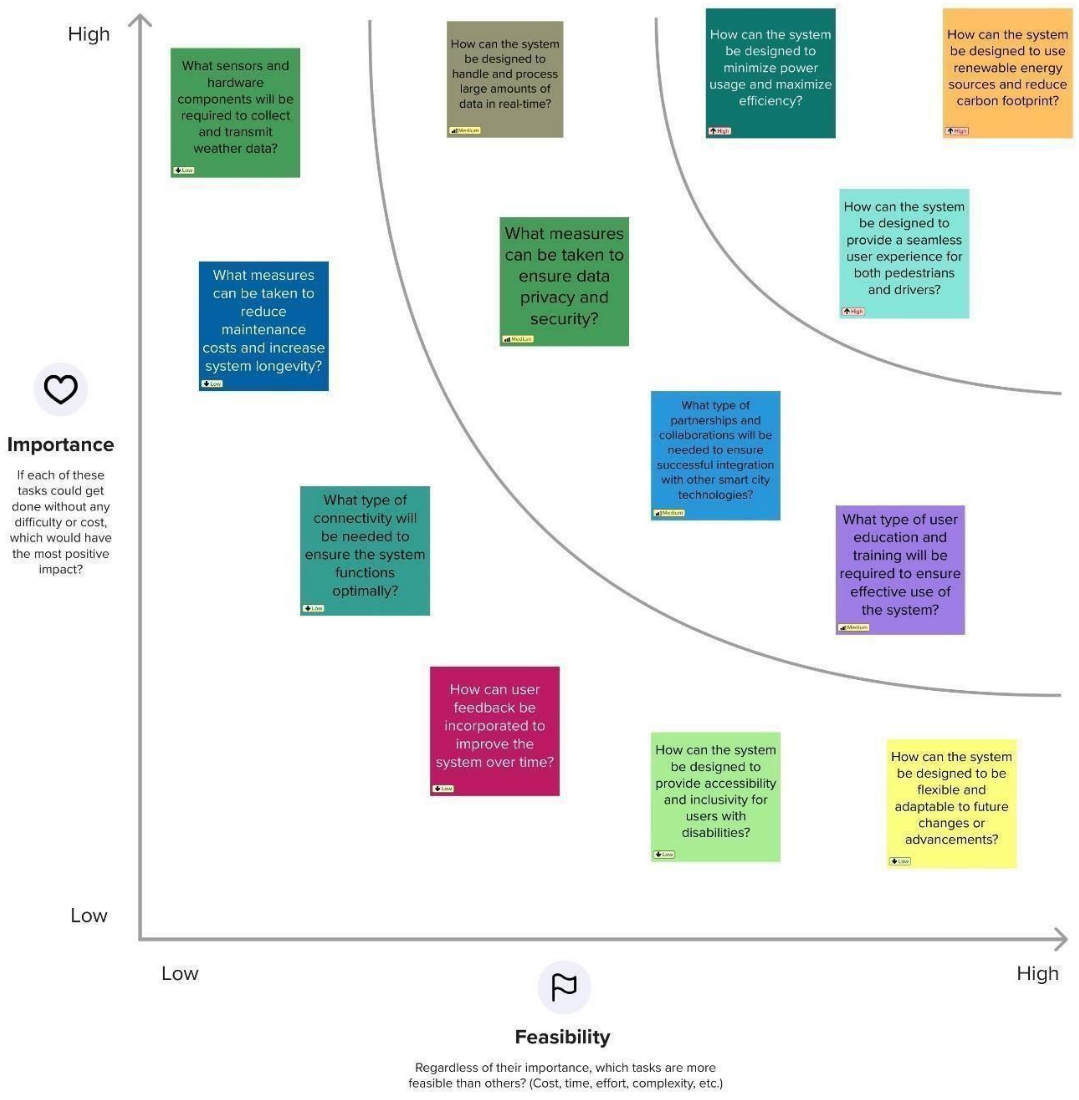
User Experience and Interface

- How can the system be designed to provide accessibility and inclusivity for users with disabilities?
- How can the system be designed to provide a seamless user experience for both pedestrians and drivers?
- What type of user education and training will be required to ensure effective use of the system?

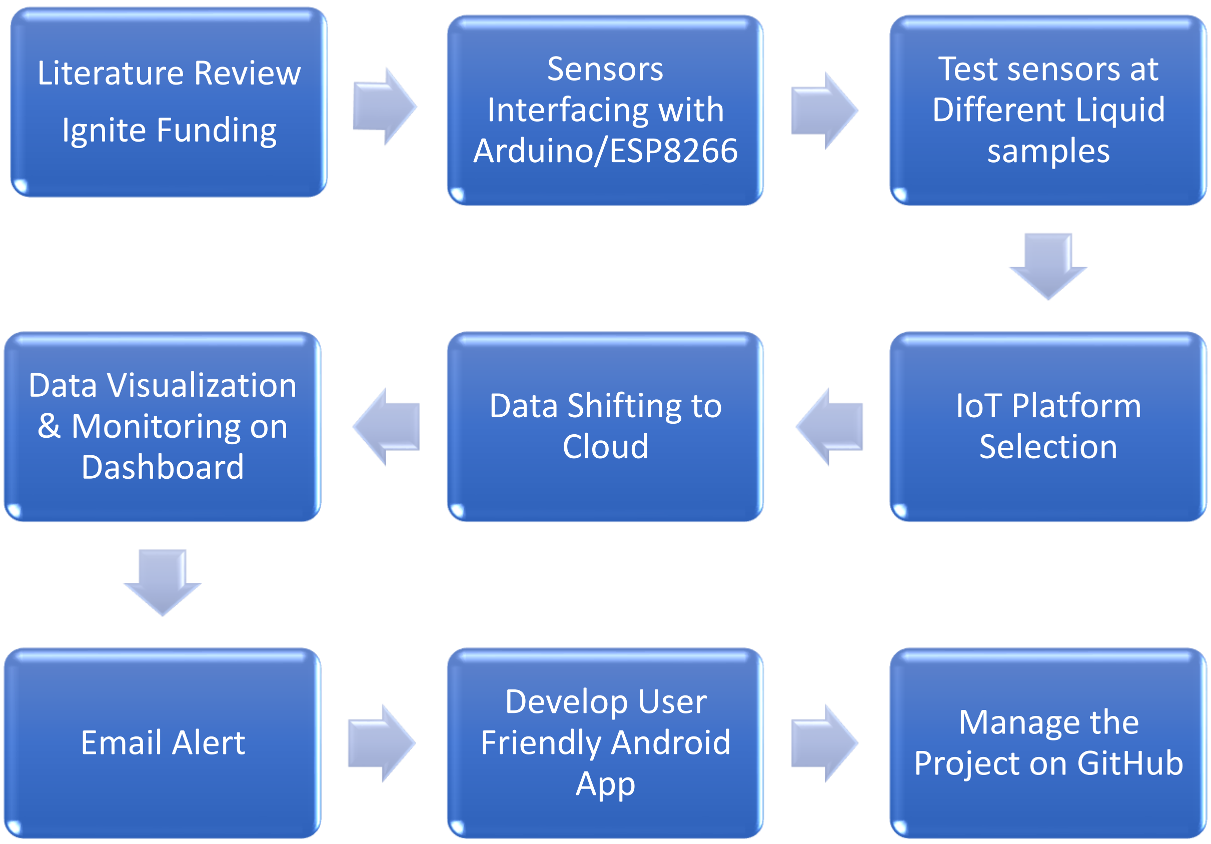
Data Collection and Analysis

- How can the system be designed to handle and process large amounts of data in real-time?
- What measures can be taken to ensure data privacy and security?
- How can user feedback be incorporated to improve the system over time?

Step-3: Idea Prioritization


High
Low
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Low
High
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)
What sensors and hardware components will be required to collect and transmit weather data?
Low
How can the system be designed to handle and process large amounts of data in real-time?
Medium
How can the system be designed to minimize power usage and maximize efficiency?
High
How can the system be designed to use renewable energy sources and reduce carbon footprint?
High
What measures can be taken to reduce maintenance costs and increase system longevity?
Low
What measures can be taken to ensure data privacy and security?
Medium
How can the system be designed to provide a seamless user experience for both pedestrians and drivers?
High
What type of partnerships and collaborations will be needed to ensure successful integration with other smart city technologies?
Medium
What type of connectivity will be needed to ensure the system functions optimally?
Low
What type of user education and training will be required to ensure effective use of the system?
Medium
How can user feedback be incorporated to improve the system over time?
Low
How can the system be designed to provide accessibility and inclusivity for users with disabilities?
Low
How can the system be designed to be flexible and adaptable to future changes or advancements?
Low

What we have done so far?


Literature Review
Ignite Funding
Sensors Interfacing with Arduino/ESP8266
Test sensors at Different Liquid samples
IoT Platform Selection
Data Shifting to Cloud
Data Visualization & Monitoring on Dashboard
Email Alert
Develop User Friendly Android App
Manage the Project on GitHub

2.4 project overview:

The world is moving towards technology rapidly and with the marvellous innovations the life has become more easier than before. But still in the 21st century, some developing countries like Pakistan are facing with the problem of water pollution. The messy and dirty water is still being used for drinking purpose without any monitoring and filtering system. This water pollution is one of the major causes for various types of water-borne diseases such as dengue, cholera and malaria etc. for human beings. 40% of deaths in worldwide are caused by water pollution. Besides the human being the water pollution is also dangerous for animals and agriculture. Therefore, for the socio-economic growth of the country the system is required that monitor the quality of water.

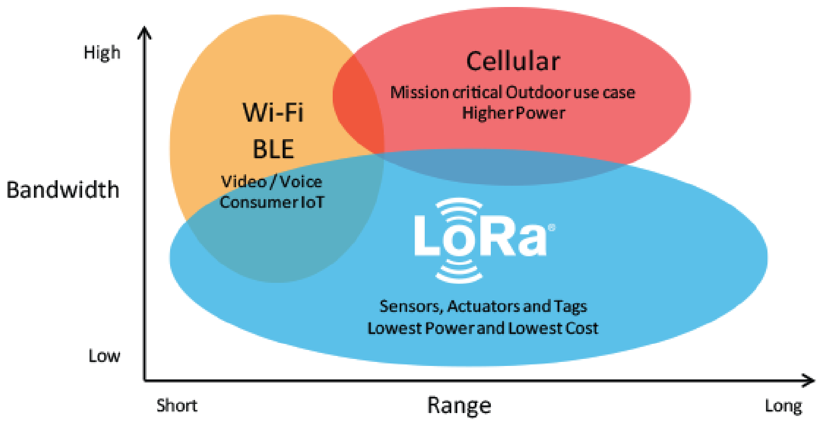
This problem for the developing countries is increasing with the industrialization and growing population. The traditional system to monitor the quality of water is not reliable because it does not provide the information about the parameters of water like pH, turbidity, conductivity, temperature, etc. continuously. Therefore, we need to develop the real time system to monitor the quality of water so that the necessary actions can be taken on the time to remain safe from unsuitable situation.

Hence, there is need of developing better methodologies to monitor the parameters of water in real time. Therefore in this project we present a design and development of a low cost system for real time monitoring of the water quality in IoT (internet of things).The system consist of several sensors is used to measuring physical and chemical parameters of the water. The parameters such as temperature, PH, turbidity, conductivity of the water can be measured.

The **pH sensor (SKU: SEN0161/0169)** measures the concentration of hydrogen ions. It shows the water is acidic or alkaline. Pure water has 7pH value, less than 7pH has acidic, more than 7pH has alkaline. The range of pH is 0-14 ph. For drinking purpose, it should be 6.5-8.5pH. **Turbidity sensor (SKU: SEN0189)** is a measure of the degree to which the water loses its transparency due to the presence of suspended particulates. The more total suspended solids in the water, the murkier it seems and the higher the turbidity. **Conductivity Sensor (SKU: SEN 0244)** is a measure of water's capability to pass electrical flow. This ability is directly related to the concentration of ions in the water. The more ions that are present, the higher the conductivity of water. **Temperature sensor (DS18B20)** is used to measure the temperature of the water.

The microcontroller will obtain and process the data from the sensors and sent the data to the IoT cloud server (**Arduino IoT Cloud** / ThingsSpeak / things network) through LoRa WAN gateway. **Arduino IoT Cloud** / ThingsSpeak is an Internet of Things (IoT) platform that lets you collect and store sensor data in the cloud and develop IoT applications. These IoT platform provides apps that let you visualize, analyze and then act to data.

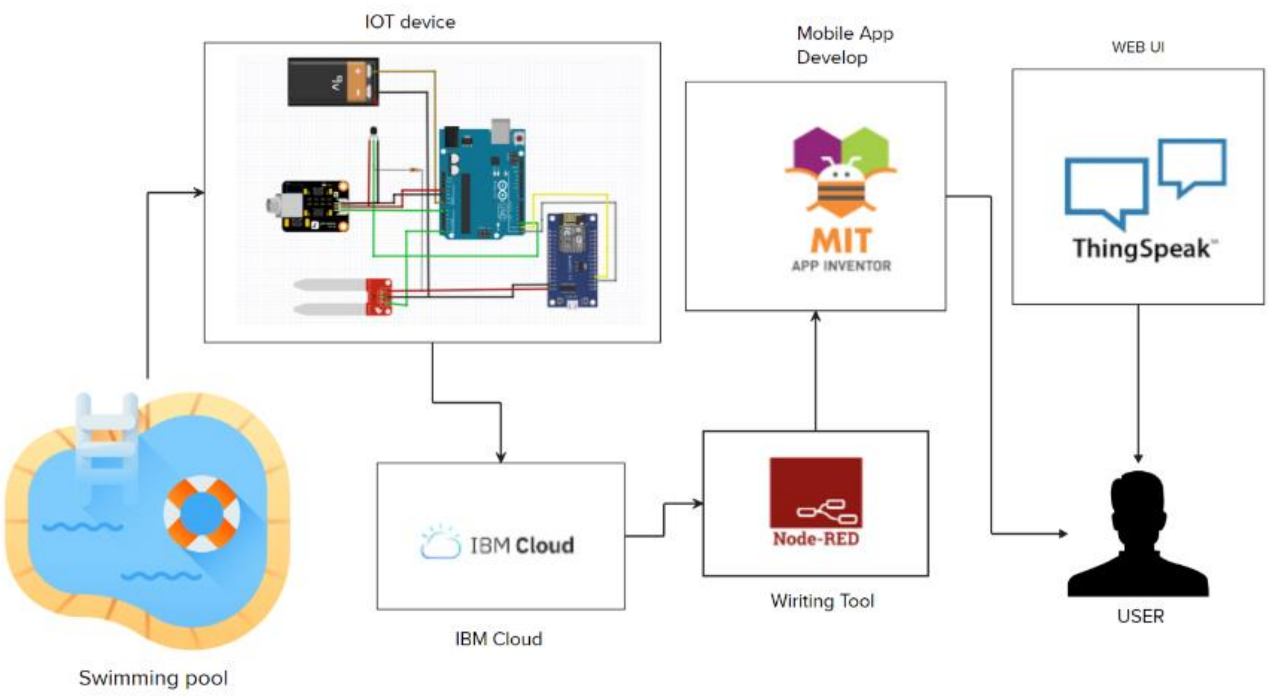
LoRa WAN is a protocol designed for creating large-scale public networks. The technology allows for sensors to talk to the internet without 3G/4G or Wi-Fi. It has the long range as compared to the Wi-Fi or GSM. It can send the data to 10s of kilometre. It is lower power consumption technology. But it can't sent the big data like videos. The maximum data that can be sent is 50Kbps. Therefore it can be used to transfer the sensors data that does not change rapidly.


High
Bandwidth
Low
Short
Range
Long
Wi-Fi
BLE
Video / Voice
Consumer IoT
Cellular
Mission critical Outdoor use case
Higher Power
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost

The Arduino MKR WAN 1310 board provides a practical and cost effective solution to add LoRa connectivity to projects requiring low power. This board can be connected to the Arduino IoT Cloud, your own LoRa network using the Arduino LoRa PRO Gateway.

3. Project Design:

3.1System Architecture:



3.2 Hardware Components:

COMPONENTS REQUIRE:

- **Arduino UNO**
- **NodeMCU**
- **Gravity Analog pH sensor**
- **DS18B20 temperature sensor**
- **Turbidity sensor**
- **Power supply**
- **Jumpers**
- **Breadboard**

COMPONENTS DESCRIPTION:

Arduino:

Arduino is an open-source electronics platform and a family of microcontroller boards designed for building and prototyping a wide range of digital devices and interactive objects. It provides both the hardware and software necessary to create and control various electronic projects, from simple LED blinking experiments to complex robotics and automation applications.

Connect the sensors and provide power to the NodeMCU.

Connect the 5V pin from Arduino UNO to VCC (or power) on the NodeMCU.

Connect GND from Arduino UNO to GND on the NodeMCU.

Connect Arduino UNO's TX (Transmit) to NodeMCU's RX (Receive).

Connect Arduino UNO's RX (Receive) to NodeMCU's TX (Transmit).

Arduino	PH Sensor board
5V	V+
GND	G

A0	Po

NodeMCU:

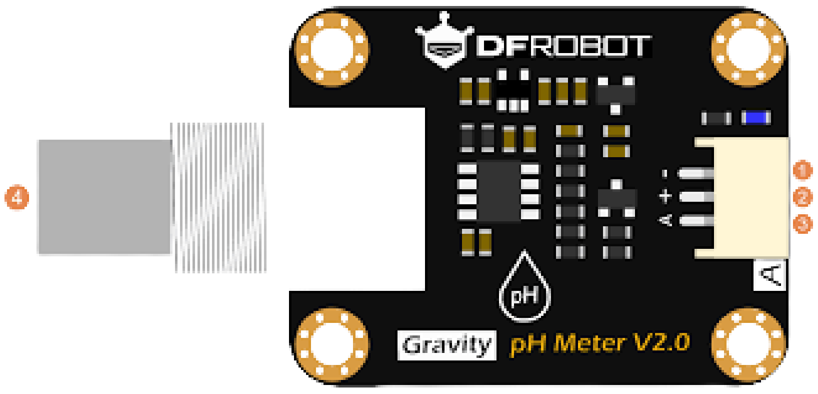
NodeMCU is an open-source development board that integrates the ESP8266 microcontroller and provides a platform for building IoT projects. It features built-in WiFi connectivity, a USB interface for easy programming, and GPIO pins for interfacing with various sensors and components. NodeMCU supports the Arduino IDE, Lua scripting language, and offers an accessible and versatile solution for developing WiFi-enabled applications and devices.

Connect the pH sensor to an analog pin on the NodeMCU

Power the pH sensor using 3.3V from the NodeMCU and connect its GND to NodeMCU's GND.

Gravity Analog pH sensor:

The "Gravity Analog pH Sensor" is a specific pH (acidity or alkalinity) measurement sensor designed for use in electronic projects and applications. It is part of the "Gravity" series, which is a line of electronic sensors and modules developed by DFRobot, a company specializing in open-source hardware and robotics. The Gravity series is known for its ease of use and compatibility with various development platforms, including Arduino and Raspberry Pi.



Pin Description:

V+: 5V DC input

G: Ground pin

Po: pH analog output

Do: 3.3V DC output

To: Temperature output

DS18B20 temperature sensor:

The DS18B20 is a digital temperature sensor known for its precision and versatility. It is part of the Dallas Semiconductor 1-Wire family of sensors and is widely used in various applications for measuring temperature.

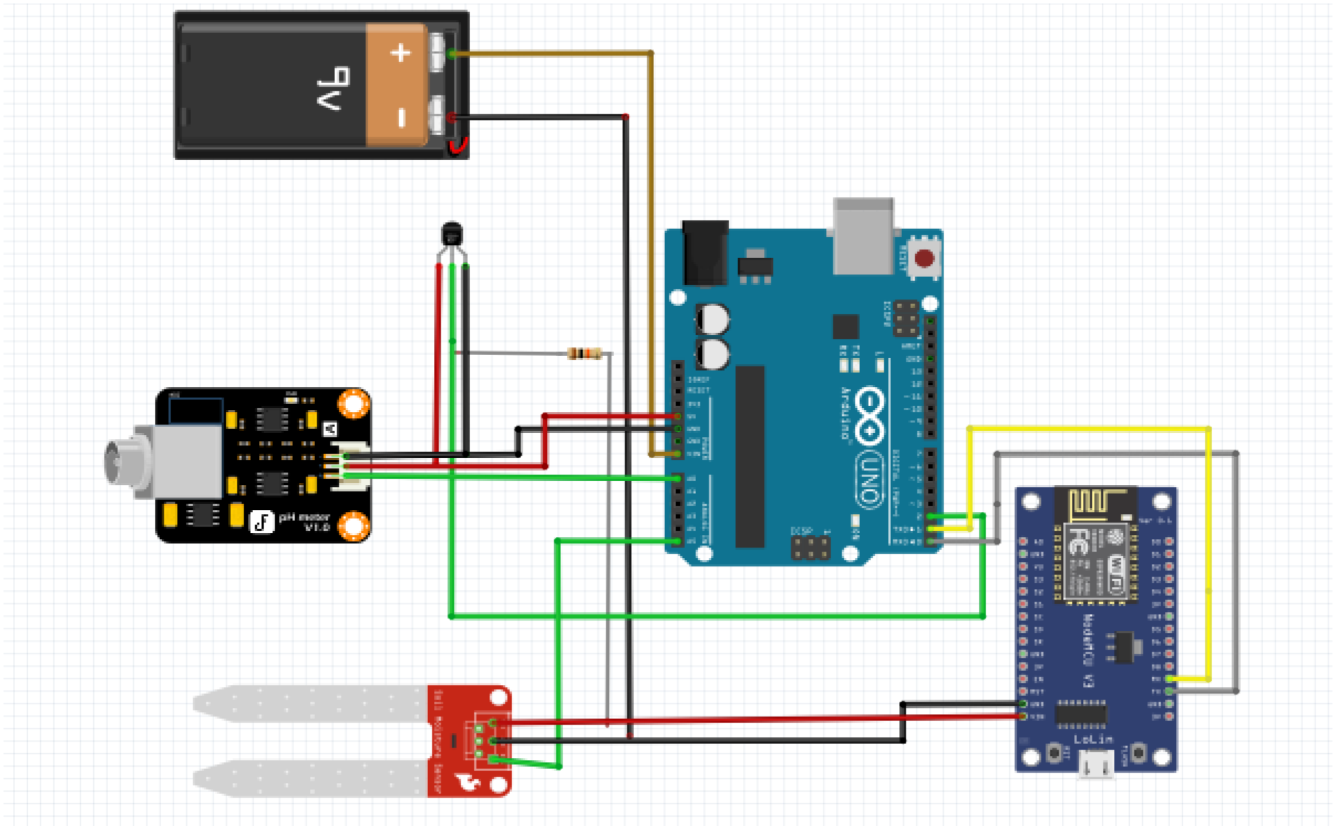
Pinout of DS18B20:

- VCC: Power input: (3.3 – 5) V DC
- Ground: Ground pin of the circuit
- Data: 1 Wire temperature value data output pin

TURBIDITY SENSOR:

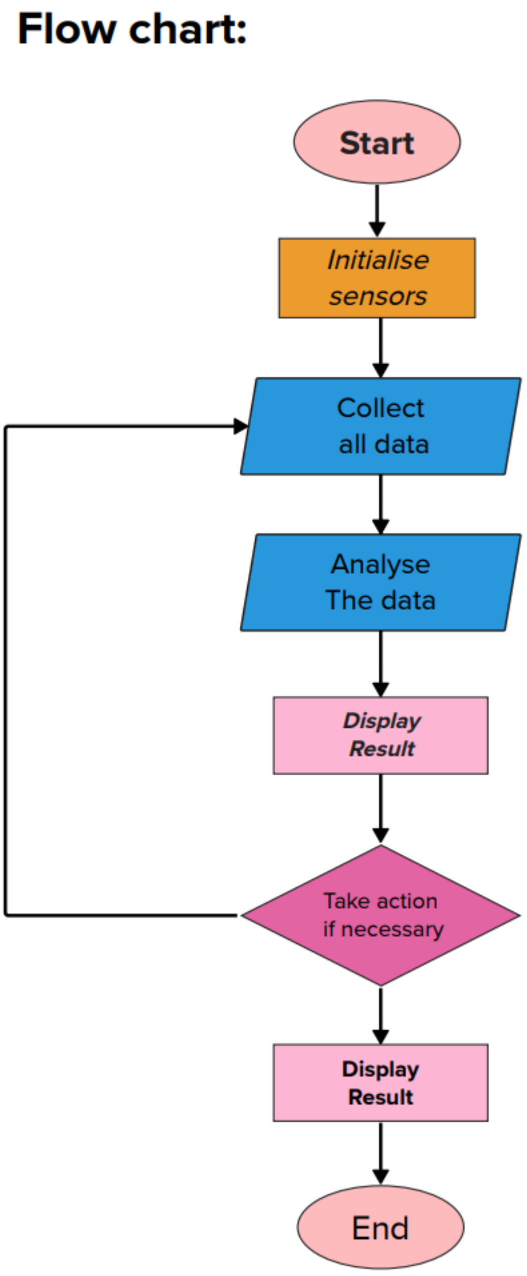
A turbidity sensor is an analytical device that measures the cloudiness or haziness of a fluid. It does this by measuring the amount of light that is scattered by particles in the fluid. The more particles there are in the fluid, the more light will be scattered, and the higher the turbidity reading will be. Turbidity sensors are used in a wide variety of applications, including

3.3 CIRCUIT DESIGN:



WATER QUALITY MONITORING PUBLIC SWIMMING POOL CIRCUIT DESIGN

3.4 Flow Chart:



4. Program:

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

const int TdsSensorPin = 34;
const int oneWireBus = 4;
const int sensorPin = 35;
const int SensorPin = 32;
#define VREF 5.0
#define SCOUNT 30
float volt;
float ntu;
unsigned long int avgValue;
float b;
int buf[10], temp;
int analogBuffer[SCOUNT];
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0, copyIndex = 0;
float averageVoltage = 0, tdsValue = 0, temperature = 25;

OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
WiFiClient client;

String apiWritekey = "7H3WMWDXLOCPPBKF";
const char* ssid = "DXMJ";
const char* password = "dxmdjaffer";
const char* server = "api.thingspeak.com";

void setup() {
  Serial.begin(115200);
  sensors.begin();
  delay(1000);

  // WiFi setup
  WiFi.disconnect();
  delay(10);
  WiFi.begin(ssid, password);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  pinMode(TdsSensorPin, INPUT);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
}

void loop() {
  // TDS Sensor
  static unsigned long analogSampleTimepoint = millis();
  if (millis() - analogSampleTimepoint > 40U) {
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin);
    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 / 4095.0;
    float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
    float compensationVolatge = averageVoltage / compensationCoefficient;

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

    Serial.print("TDS: ");
    Serial.print(tdsValue, 0);
    Serial.println("ppm");
  }

  // Temperature Sensor
  sensors.requestTemperatures();
  float temperatureC = sensors.getTempCByIndex(0);
  float temperatureF = sensors.getTempFByIndex(0);
  Serial.print("Temp: ");
  Serial.print(temperatureC);
  Serial.print("ºC ---> ");
  Serial.print(temperatureF);
  Serial.println("ºF");

  // Turbidity Sensor
  volt = 0;
  for (int i = 0; i < 800; i++) {
    volt += ((float)analogRead(sensorPin) / 4095) * 3.3;
  }
  volt = volt / 800;
  volt = round_to_dp(volt, 1);
  volt = volt + 2.85;

  if (volt < 2.5) {
    ntu = 3000;
  } else {
    ntu = -1120.4 * (volt * volt) + 5742.3 * volt - 4353.8;
    ntu = ntu + 580.56;
  }

  Serial.println("Turbidity = " + String(ntu) + " NTU");

  // pH Sensor
  for (int i = 0; i < 10; i++) {
    buf[i] = analogRead(SensorPin);
    delay(10);
  }
  for (int i = 0; i < 9; i++) {
    for (int j = i + 1; j < 10; j++) {
      if (buf[i] > buf[j]) {
        temp = buf[i];
```

```
        buf[i] = buf[j];
        buf[j] = temp;
      }
    }
  }
  avgValue = 0;
  for (int i = 2; i < 8; i++)
    avgValue += buf[i];
  float phValue = (float)avgValue * 3.3 / 4095 / 6;
  phValue = 5 + phValue;

  Serial.print("pH: ");
  Serial.print(phValue, 2);
  Serial.println(" ");

  // Upload data to ThingSpeak
  uploadToThingSpeak();
}

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;
}

void uploadToThingSpeak() {
  String url = "/update";

  String headers = "X-THINGSPEAKAPIKEY: " + apiWritekey + "\r\n";
  String dataPayload = "field1=" + String(123) + "&field2=" + String(7.2)
                      + "&field4=" + String(25.5) + "&field5=" + String(150);

  int contentLength = dataPayload.length();

  String request = "POST " + url + " HTTP/1.1\r\n" +
                   "Host: " + server + "\r\n" +
                   "Connection: close\r\n" +
                   "Content-Type: application/x-www-form-urlencoded\r\n" +
                   "Content-Length: " + String(contentLength) + "\r\n" +
                headers +
                "\r\n" + dataPayload + "\r\n\r\n";

  client.print(request);
  // Handle the response from ThingSpeak if needed
  while (client.available()) {
    String line = client.readStringUntil('\r\n');
    Serial.println(line);
  }
```

OUTPUT:

```
TDS: 123 ppm

Temp: 25.5 ºC ---> 77.9 ºF

Turbidity = 150 NTU

pH: 7.2
```

Parameters	Standard value	Measured Value	Error	Accuracy
pH	6.5-8.5	7.79	3.86%	96.14%
Turbidity(NTU)	<5	5	4%	96%
TDS(ppm)	<500(300 for Good Drinking water)	394	23%	77%

5.Integration with Arduino:

5.1 Ardunio:

Arduino is an open-source electronics platform and ecosystem that provides both hardware and software tools for building a wide range of electronic projects. It was created to make it easier for people, particularly hobbyists, students, and makers, to develop and prototype their own electronic devices and interactive projects. Arduino is widely popular due to its accessibility and flexibility.

Here are some key components and features of the Arduino platform:

1.Arduino Boards: Arduino boards are the hardware foundation of the platform. They come in various models and sizes, but they typically consist of a microcontroller (e.g., ATmega series) and input/output pins. The Arduino Uno is one of the most well-known and widely used boards.

2.Arduino Software (IDE): The Arduino Integrated Development Environment (IDE) is the software used for programming Arduino boards. It is a user-friendly environment that supports the Arduino programming language, which is a simplified version of C/C++. Users can write and upload code to their Arduino boards via the IDE.

3.Arduino Shields: Shields are add-on boards that can be stacked on top of Arduino boards to extend their capabilities. There are various shields available for functions like Ethernet connectivity, wireless communication, motor control, and sensor interfacing.

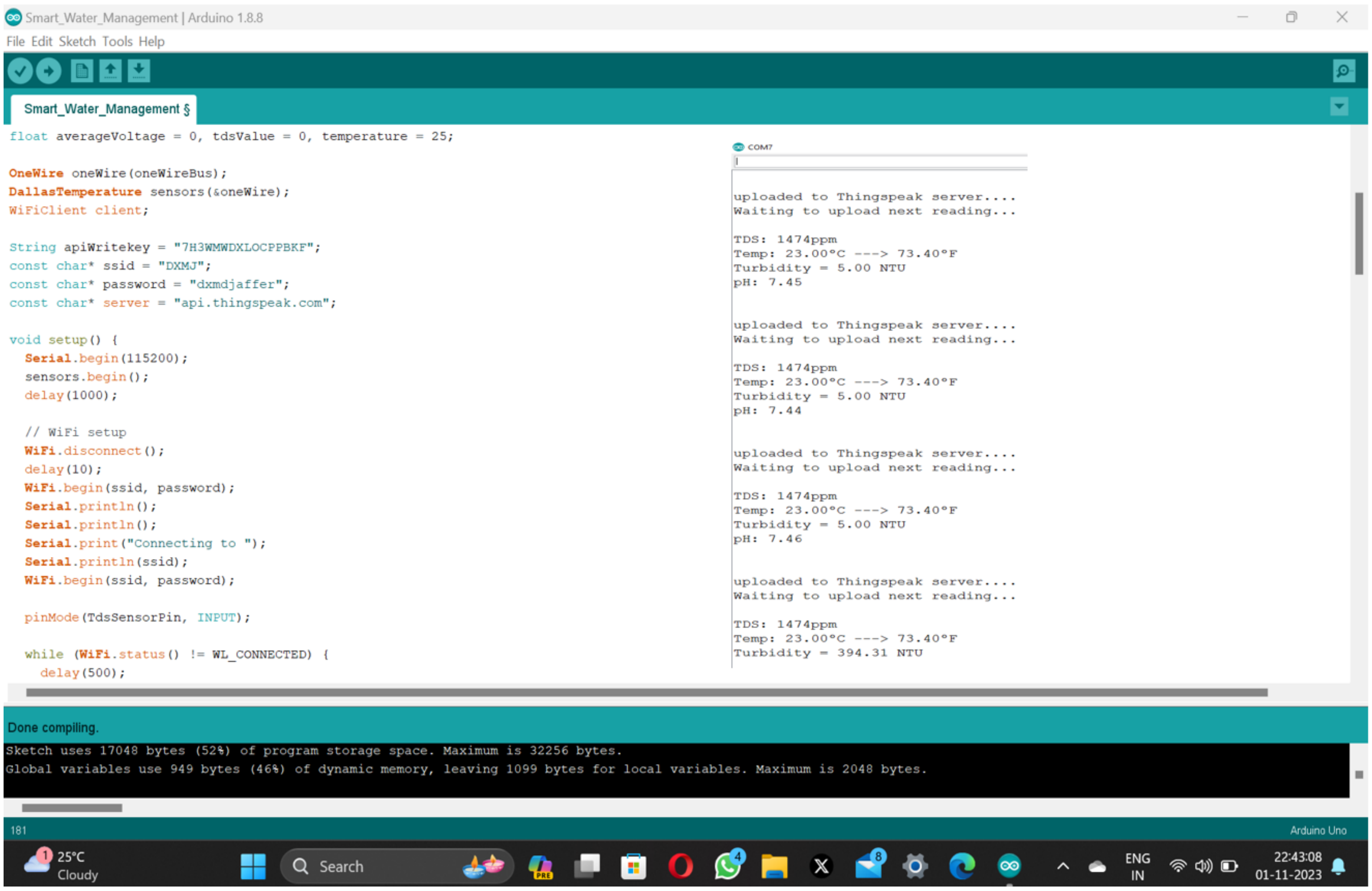
4.Arduino Libraries: Arduino provides a collection of libraries that simplify working with various components and modules. These libraries contain pre-written code that can be easily integrated into your projects.

5.Versatile Inputs and Outputs: Arduino boards have digital and analog input/output pins, making it easy to interface with sensors, actuators, and other electronic components. This flexibility allows for a wide range of applications.

6.Community and Documentation: Arduino has a large and active community of users, makers, and developers. There are numerous online resources, tutorials, and forums where users can seek help, share their projects, and find inspiration.

7.Open Source: Arduino is open source, which means that the hardware and software designs are available to the public. This encourages innovation and allows users to modify and adapt the platform to their specific needs.

5.2 ARDUINO output:



```
float averageVoltage = 0, tdsValue = 0, temperature = 25;

OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
WiFiClient client;

String apiWritekey = "7H3WMWDXLOCPPBKF";
const char* ssid = "DXMJ";
const char* password = "dxmdjaffer";
const char* server = "api.thingspeak.com";

void setup() {
  Serial.begin(115200);
  sensors.begin();
  delay(1000);

  // WiFi setup
  WiFi.disconnect();
  delay(10);
  WiFi.begin(ssid, password);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  pinMode(TdsSensorPin, INPUT);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
```

```
uploaded to Thingspeak server....
Waiting to upload next reading...

TDS: 1474ppm
Temp: 23.00°C ---> 73.40°F
Turbidity = 5.00 NTU
pH: 7.45


uploaded to Thingspeak server....
Waiting to upload next reading...

TDS: 1474ppm
Temp: 23.00°C ---> 73.40°F
Turbidity = 5.00 NTU
pH: 7.44


uploaded to Thingspeak server....
Waiting to upload next reading...

TDS: 1474ppm
Temp: 23.00°C ---> 73.40°F
Turbidity = 5.00 NTU
pH: 7.46


uploaded to Thingspeak server....
Waiting to upload next reading...

TDS: 1474ppm
Temp: 23.00°C ---> 73.40°F
Turbidity = 394.31 NTU
```

Done compiling.

```
Sketch uses 17048 bytes (52%) of program storage space. Maximum is 32256 bytes.
Global variables use 949 bytes (46%) of dynamic memory, leaving 1099 bytes for local variables. Maximum is 2048 bytes.
```

6. Integration with Thingspeak:

6.1 Thingspeak:

ThingSpeak is an Internet of Things (IoT) platform and open-source application for collecting, processing, and visualizing data from various IoT devices. It was created by MathWorks, the company behind MATLAB, and is designed to simplify the process of connecting and working with IoT devices and data. ThingSpeak provides several key features:

1.Data Collection:

ThingSpeak allows users to collect data from a variety of sources, such as sensors, devices, and applications. This data can be sent to ThingSpeak using various communication protocols, including HTTP, MQTT, and the ThingSpeak API.

2.Data Processing:

The platform provides built-in MATLAB analytics that can be applied to the collected data. This enables users to perform real-time data processing, analysis, and visualization directly on the platform.

3.Data Visualization:

ThingSpeak offers customizable charts, graphs, and visualizations to help users understand their IoT data. Users can create real-time dashboards to monitor their devices and sensors.

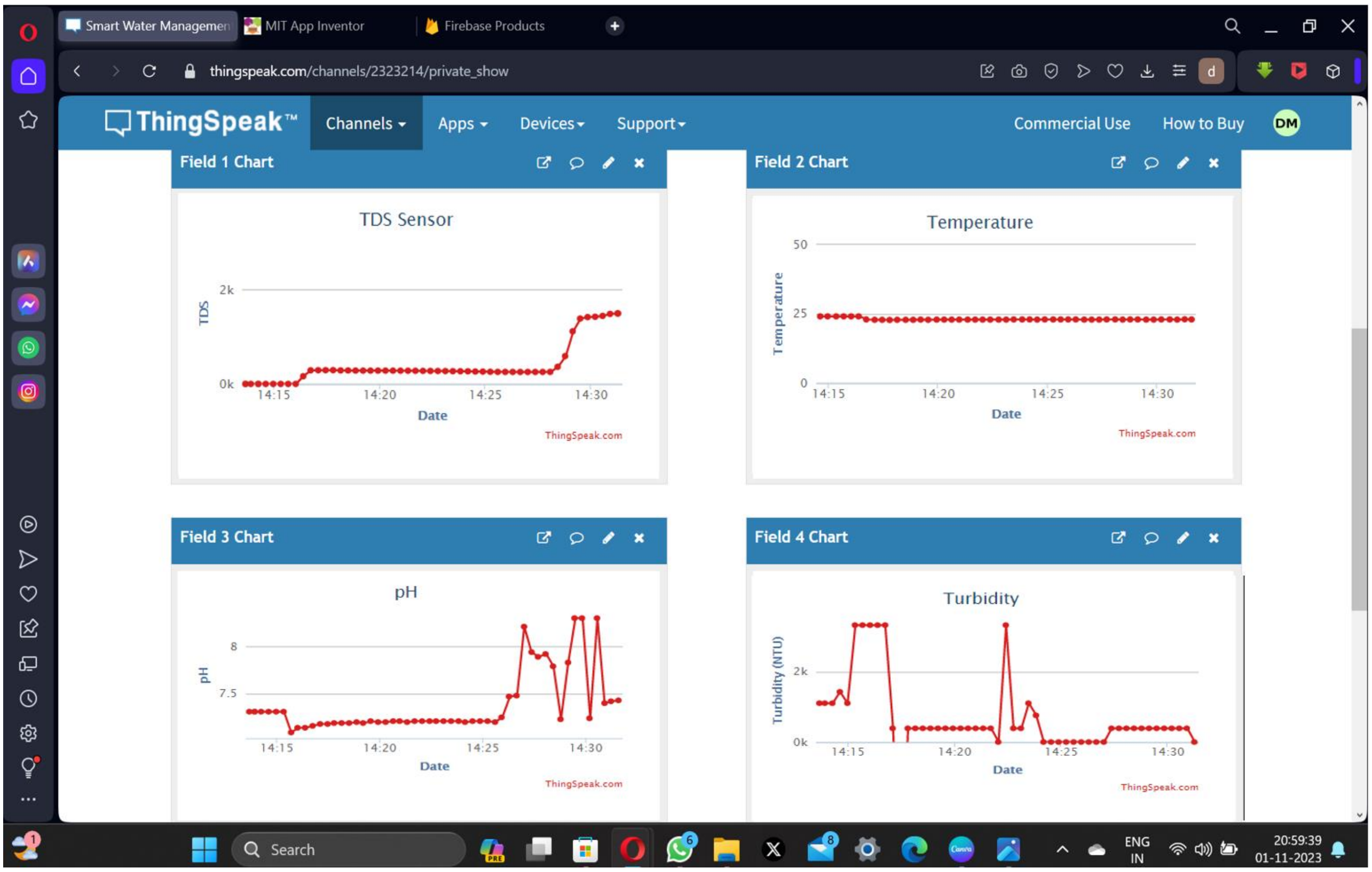
4.Data Sharing:

Users can choose to make their data public or keep it private. This feature is useful for collaborative projects, research, and sharing data with others.

5.Integration:

ThingSpeak can be integrated with other IoT platforms, cloud services, and applications. It provides support for third-party services and devices, making it easier to connect your IoT ecosystem.

6.2 Thingspeak simulation:



7. Integration with Mit app inventor:

7.1 MIT App invertor:

MIT App Inventor is an open-source visual development platform that allows people, including those with little to no programming experience, to create mobile applications for Android devices. It was initially created by Google and then transferred to the Massachusetts Institute of Technology (MIT) for further development and maintenance. MIT App Inventor uses a block-based programming approach that simplifies app development, making it accessible to a wide range of users.

Here are some key features and aspects of MIT App Inventor:

1.Visual Programming: MIT App Inventor uses a visual, drag-and-drop approach to app development. Instead of writing code, users assemble blocks representing different app components and actions. This makes it easy for beginners to create functional apps without having to write traditional code.

2.Real-Time Testing: Users can test their apps on Android devices in real-time by using the MIT App Inventor Companion app. This allows for quick and iterative development, and changes made in the development environment are immediately reflected on the connected Android device.

3.Extensive Component Library: MIT App Inventor includes a wide range of components that can be used to build apps. These components cover UI elements, sensors (such as GPS and accelerometer), multimedia, data storage, and communication features.

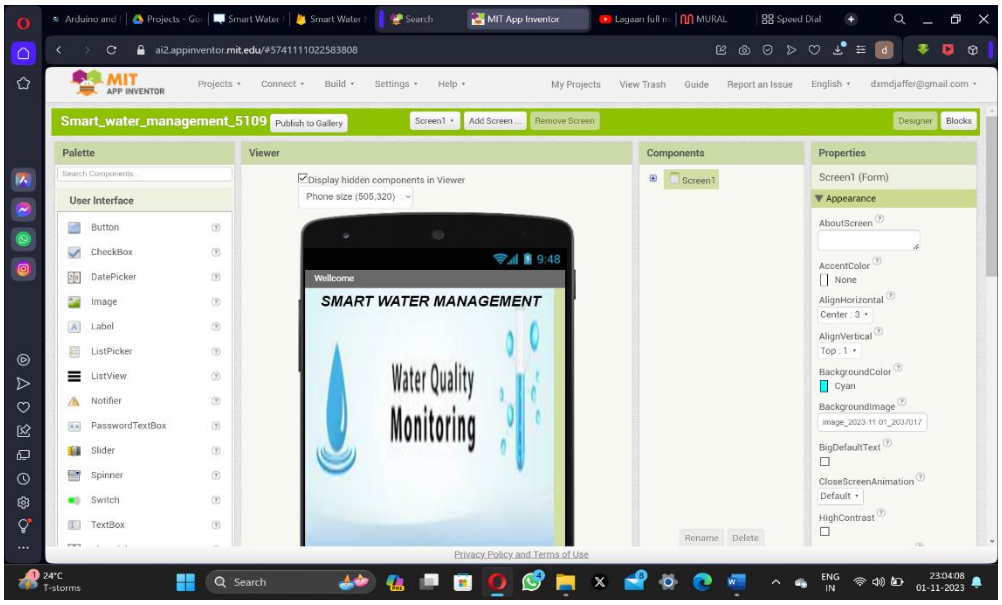
4.Cloud-Based Development: MIT App Inventor is a cloud-based platform, which means that users can access their projects from anywhere with an internet connection. This also simplifies collaboration and sharing of projects.

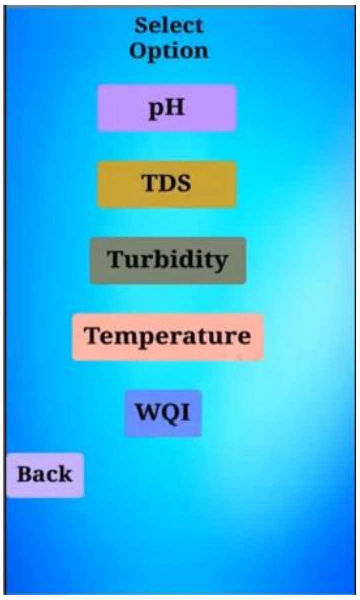
5.Custom Blocks: While the platform offers a wide range of pre-built blocks, users can also create custom blocks, allowing for the encapsulation of complex logic into reusable components.

6.Educational Focus: MIT App Inventor is often used in educational settings to teach programming and app development. It's designed to be user-friendly and intuitive, making it a great tool for introducing students to the world of app creation.

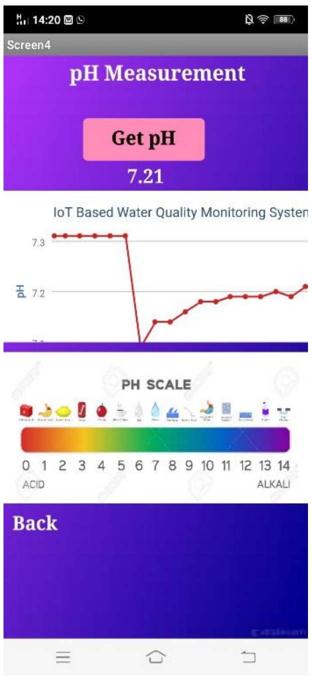
7.Open Source: MIT App Inventor is an open-source project, which means that the source code is available to the public. This openness allows for the platform to be extended and modified by the community.

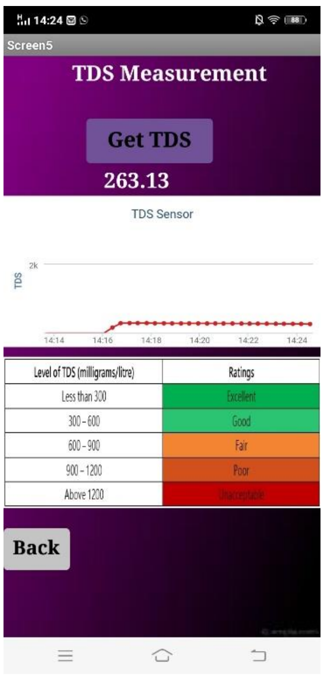
7.2 Mit app inventor simulation:



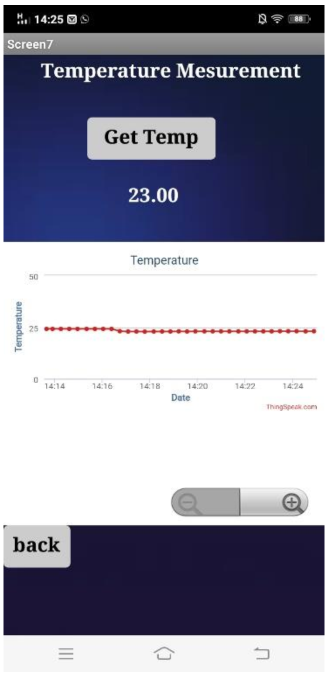


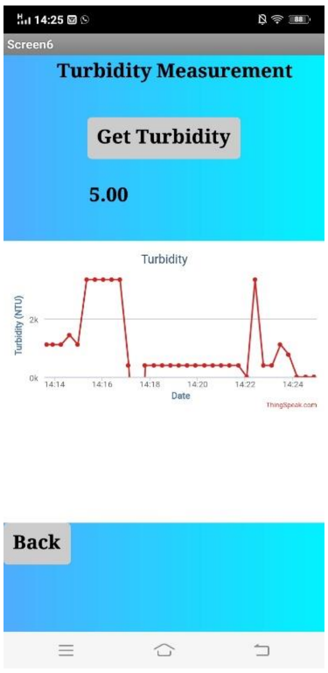
Mobile app design in mit app inventor **Select option**

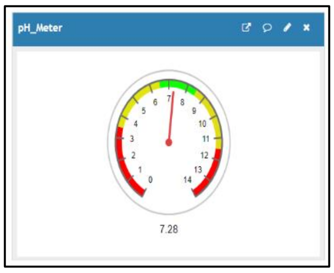


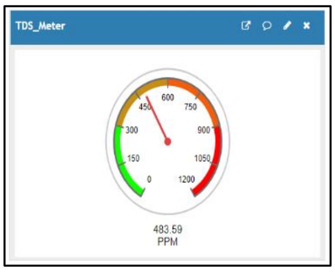


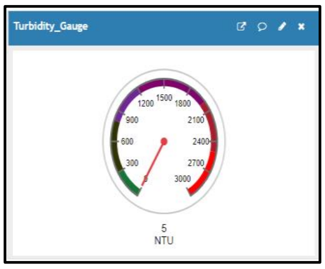
Level of TDS (milligrams/litre)	Ratings
Less than 300	Excellent
300 – 600	Good
600 – 900	Fair
900 – 1200	Poor
Above 1200	Unacceptable

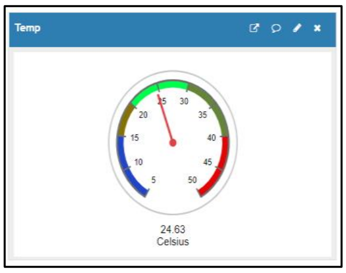












8. Application:

- Monitoring and maintaining chlorine levels: Ensure adequate chlorine levels to disinfect pool water and prevent the growth of harmful bacteria and microorganisms.

- Controlling pH levels: Maintain optimal pH levels for effective chlorine disinfection and to prevent skin and eye irritation.

- Monitoring turbidity: Assess the clarity of pool water, indicating the presence of suspended particles or contaminants.

- Tracking temperature fluctuations: Ensure comfortable swimming temperatures and prevent potential health issues related to extreme water temperatures.

- Optimizing chemical dosing: Adjust chemical dosages based on real-time water quality data, minimizing unnecessary chemical usage and reducing environmental impact.

- Predictive maintenance: Identify potential equipment malfunctions or water quality issues before they lead to pool closures or safety hazards.

9. Conclusion:

IoT-based water quality monitoring systems offer a comprehensive and effective solution for maintaining safe and healthy public swimming pools. By providing real-time insights into critical water quality parameters, these systems enable proactive measures to prevent contamination, optimize pool operations, and ensure a positive swimming experience for all.

10. GitHub Link: https://github.com/DXMJ/Smart-Water-Management-5109