SMART WATER QUALITY MONITORING SYSYTEM

PROJECT REPORT

TEAM MEMBERS

SAI SRI VARDHAN REDDY LINGALA	23EC10038
VEERA VENKATA SAI SRINIVAS PAYYAVULA	23EC10058
NITHISH KUMAR BANOTH	23EC10018
SOUNAK SAHA	23EC10078

SECTION 8

GROUP ID 9

MOTIVATION

The motivation for smart water quality monitoring is to keep our water safe and clean. By using technology to check and manage water quality, we can detect and prevent pollution and ensure that the water we use is healthy for people and the environment. This helps us live healthier lives and protects our planet.

METHODOLOGY

Creating a smart water quality monitoring system with a DS18B20 temperature sensor, Grove TDS sensor, Arduino, LCD 16x2 with I2C, and a breadboard involves several steps

COMPONENTS NEEDED:

- Arduino board (e.g., Arduino Uno)
- DS18B20 temperature sensor
- Grove TDS sensor
- LCD 16x2 display with I2C backpack
- Breadboard and jumper wires
- Power source for Arduino (USB cable or battery)

STEP 1: HARDWARE CONNECTION

- **Connect the DS18B20 temperature sensor:**
  1. **Connect the VCC pin of DS18B20 to the 5V pin on the Arduino.**
  2. **Connect the GND pin of DS18B20 to the GND pin on the Arduino.**
  3. **Connect the DATA pin of DS18B20 to a digital pin on the Arduino (e.g., D2).**

- **Connect the Grove TDS sensor:**
  1. **Connect the VCC pin of the Grove TDS sensor to the 5V pin on the Arduino.**
  2. **Connect the GND pin of the Grove TDS sensor to the GND pin on the Arduino.**
  3. **Connect the analog output pin of the Grove TDS sensor to an analog pin on the Arduino (e.g., A0).**
- **Connect the LCD 16x2 with I2C:**
  1. **Connect the SDA(SIGNAL) pin of the I2C backpack to the A4 (SDA) pin on the Arduino.**
  2. **Connect the SCL(SIGNAL) pin of the I2C backpack to the A5 (SCL) pin on the Arduino.**
  3. **Connect the VCC and GND pins of the I2C backpack to the 5V and GND on the Arduino.**
- **Connect the LCD display to the I2C backpack.**
- **Power the Arduino using a USB cable or a battery or to laptop.**

1. **NOTE :**

DS18B20 Temperature Sensor (Digital):

- The DS18B20 is a digital temperature sensor that communicates using the OneWire protocol.
- It uses a single digital pin (e.g., D2) for communication. Each DS18B20 sensor has a unique address, allowing multiple sensors to be connected to the same digital pin.
- The sensor sends digital signals to the Arduino, and the Arduino uses the OneWire library to communicate with and read data from the sensor.

Grove TDS Sensor (Analog):

- The Grove TDS sensor provides an analog output voltage that corresponds to the Total Dissolved Solids (TDS) in a solution.
- Analog pins on the Arduino are used to read continuous analog signals. The analog pin (e.g., A0) reads the voltage level and converts it to a digital value using the Arduino's analog-to-digital converter (ADC).
- The specific voltage-to-TDS conversion formula may vary between different TDS sensors, and it's typically provided in the sensor's datasheet or documentation.

LCD Pins:

- VCC: Power supply for the LCD (usually connected to +5V).
- GND: Ground connection.
- V0: Contrast control voltage. Adjusting this voltage controls the contrast of the display.
- RS (Register Select): Selects whether data or an instruction is sent to the LCD. High for data, low for instruction.
- RW (Read/Write): Selects whether data is read from or written to the LCD. Normally set to write mode (grounded) since we often only write to the display.
- E (Enable): Strobe signal. Falling edge triggered to latch data.
- D0-D7: Data lines for sending 8-bit ASCII or custom character data to the display.

I2C Pins:

- SDA (Serial Data Line): Bi-directional data line for sending and receiving serial data.
- SCL (Serial Clock Line): Clock line for synchronizing data transfer between devices.

ARDUINO PINS

1. **Digital Pins (D2-D13):** Versatile I/O warriors, capable of digital input or output.
2. **Analog Pins (A0-A5):** Sensitive souls that read analog voltages, perfect for sensors.
3. **Power Pins (5V, 3.3V, GND):** The energy brigade—5V and 3.3V for power, GND for grounding.
4. **Communication Pins (TX, RX, SDA, SCL):** Talkative types—TX/RX for serial chat, SDA/SCL for I2C conversations.
5. **PWM Pins (D3, D5, D6, D9, D10, D11):** Masters of Pulse Width Modulation, turning digital into analog-like magic.
6. **Interrupt Pins (D2, D3):** Swift responders, alerting the Arduino to immediate action.

7. **Reset Pin (RESET):** The restart button—applying low voltage here triggers a fresh beginning.
8. **Vin (Voltage In):** The entry point for external power, sustaining your Arduino.
9. **ARef (Analog Reference):** Guides analog pins, setting the reference voltage for precise measurements.

STEP 2: SOFTWARE SETUP

- **Install the Arduino IDE on your computer if you haven't already.**
- **Open the Arduino IDE and install any required libraries for the DS18B20 temperature sensor, Grove TDS sensor, and the LCD 16x2 with I2C.**
- **Write the Arduino code for your water quality monitoring system.**

STEP 3: UPLOAD THE CODE

- **Connect your Arduino to your computer via USB.**
- **Select the correct Arduino board and COM port in the Arduino IDE.**
- **Click the "Upload" button to upload the code to the Arduino.**

STEP 4: TESTING

- **Once the code is uploaded, the LCD should display the temperature and TDS readings. You can dip the Grove TDS sensor into a water sample to measure its TDS level. The DS18B20 sensor will provide the temperature reading. With this setup, you have created a smart water quality monitoring system using an Arduino, DS18B20 temperature sensor, Grove TDS sensor, and an LCD display for real-time monitoring of**

temperature and TDS levels in water. You can further enhance the system by adding data logging, alerts, and wireless connectivity for remote monitoring.

ITEMS REQUIRED

- **Arduino board (e.g., Arduino Uno)**
- **DS18B20 temperature sensor**
- **Grove TDS sensor**
- **LCD 16x2 display with I2C backpack**
- **Breadboard**
- **Jumper wires for connecting components**
- **One Wire library for DS18B20 sensor**
- **Dallas Temperature library for DS18B20 sensor**
- **LiquidCrystal_I2C library for the LCD with I2C**
- **USB cable or battery for powering the Arduino**
- **Computer with the Arduino IDE for programming the Arduino board**
- **Water sample for testing water quality**
- **Resistors**

RESULTS

1. **Temperature Reading (DS18B20 Sensor):**
    - The DS18B20 sensor measures the temperature of the water sample.
    - The temperature reading is displayed on the LCD screen in degrees Celsius (°C).
    - The accuracy of the temperature reading is typically within ±0.5°C in the specified range.
2. **TDS (Total Dissolved Solids) Reading (Grove TDS Sensor):**
    - The Grove TDS sensor measures the TDS level in the water sample, which represents the concentration of dissolved solids in the water.

- The TDS reading is displayed on the LCD screen in parts per million (ppm).
- The accuracy of the TDS reading is typically within ±10% of full scale (F.S.).
- The TDS reading can give you an idea of the water's purity or the concentration of various substances in the water.

3. **LCD Display:**
   - The LCD 16x2 display will show the real-time temperature and TDS readings.
   - The display will be updated at regular intervals, as specified in your Arduino code.

NOTE HOW SENSORS WORK

1. **Temperature Sensor (e.g., DS18B20):**
   - **Physics Principle - Thermal Expansion:** The DS18B20 relies on the principle of thermal expansion. Most materials expand when heated and contract when cooled. The sensor contains a temperature-sensitive resistor made of a material with predictable thermal characteristics.
   - **Resistance and Temperature:** As the temperature changes, the material's resistance changes. In the DS18B20, this change in resistance is used to determine the temperature. The relationship between temperature and resistance is often linear and well-characterized.
2. **TDS Sensor (e.g., Grove TDS Sensor):**
   - **Physics Principle - Electrical Conductivity:** The TDS sensor operates on the basis of electrical conductivity, which is influenced by the concentration of ions in a solution.
   - **Ion Movement:** When the TDS sensor is placed in water, ions in the water (such as salts and minerals) facilitate the flow of electric current between the sensor's electrodes. The higher the concentration of dissolved solids, the better the water conducts electricity.
   - **Calibration:** The sensor's output is calibrated to correlate the electrical conductivity with the actual concentration of dissolved solids. This calibration involves understanding the relationship between ion concentration and conductivity under specific conditions.

CHALLENGES

1. **Sensor Calibration:**

Calibrating the sensors for accurate temperature and TDS readings can be challenging. The sensors may need periodic recalibration to maintain accuracy.

2. **Data Accuracy**

Ensuring the accuracy of the data from the sensors can be a challenge, especially in changing environmental conditions or when dealing with sensor drift over time.

LEARNINGS

1. **Sensor Calibration:**

   Learning to calibrate sensors is a valuable skill. Understanding the calibration process and how to apply it can lead to more accurate measurements.

2. **Data Validation:**

   Regularly validating and cross-referencing data from different sensors and sources can help you identify and correct inaccuracies.

CONCLUSIONS

- **Data Accuracy and Reliability:**

  Accurate and reliable data is crucial for monitoring water quality. Regular calibration and validation of sensors are essential to ensure the accuracy of temperature and TDS readings.

- **Real-Time Monitoring:**

  The use of Arduino and the LCD display allows for real-time monitoring and immediate visualization of water quality parameters, making it a useful tool for quick assessments.

- **Environmental Impact:**
  Environmental factors can affect the measurements, and learning how to mitigate these effects is important for obtaining meaningful data.

REFERENCES

- **YOU TUBE (HOW TO ELECTRONICS and MILIOHM)**
- **ARDUINO LEARNING FROM DIY CLASS**
- **WEBISTE (https://arduinogetstarted.com/tutorials/arduino-lcd-i2c)**
- **Git hub for libraries (https://github.com/lucasmaziero/LiquidCrystal_I2C)**
- **One wire library and Dallas temperature library we can download it in Arduino library it self**

CODE FOR ARDUINO IDE TO COMMUNICATE WITH SENSORS AND LCD

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal_I2C.h>

// Data wire is connected to Pin 2 on the Arduino
#define ONE_WIRE_BUS 2
```

```
// Initialize the OneWire and DallasTemperature libraries
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Initialize the LCD library
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Address 0x27, 16 columns, 2 rows

void setup() {
  Serial.begin(9600); //TO START THE SETUP  ,9600BPS
  lcd.backlight();  // Turn on the backlight
  sensors.begin();  // Start the DS18B20  TEMPARATURE sensor
}

void loop() {
  sensors.requestTemperatures();  // Request temperature readings from the
DS18B20 sensor

  // Read the temperature in Celsius
  float temperatureC = sensors.getTempCByIndex(0); //FLOAT MEANS DECIMAL

  // Read the analog value from the Grove TDS sensor
  float sensorValue = analogRead(A0);  // Connect the Grove TDS sensor to the A0
pin on your Arduino
  float tdsValue = sensorValue * 1.25; // ANALOG TO DIGITAL

  // Display the temperature and TDS value on the LCD
  lcd.clear();          // Clear the LCD screen
  lcd.setCursor(0, 0);  // Set cursor to the first line
  lcd.print("Temp (C): ");
  lcd.print(temperatureC);  // Display one decimal place
  lcd.setCursor(0, 1);      // Set cursor to the second line
  lcd.print("TDS (ppm): ");
  lcd.print(tdsValue);

  // Print the temperature and TDS value to the Serial Monitor
  Serial.print("Temperature in Celsius: ");
  Serial.print(temperatureC);  // Display one decimal place
  Serial.println(" °C");

  Serial.print("TDS Value: ");
  Serial.print(tdsValue);  //CALLING THE CODE
  Serial.println(" ppm");

  delay(1000);  // Delay for 1 second before the next reading
}
```

LCD THINGS

LCD ADRESS CODE

```
/*I2C_scanner
  This sketch tests standard 7-bit addresses.
  Devices with higher bit address might not be seen properly.*/
#include <Wire.h
void setup() {
  Wire.begin();
  Serial.begin(9600);
  while (!Serial);
  Serial.println("\nI2C Scanner");
}
void loop() {
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for (address = 1; address < 127; address++ ) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println("  !");
      nDevices++;
    }
    else if (error == 4) {
      Serial.print("Unknown error at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");
  delay(1000);
}
```

RESULTS :

```
Scanning...
I2C device found at address 0x27  !
done
```

LCD GET READY DIY LAB CODE

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2 rows
void setup()
{
   // initialize the lcd
  lcd.backlight();
}
void loop()
{
  lcd.clear();                // clear display
  lcd.setCursor(0, 0);        // move cursor to   (0, 0)
  lcd.print("Arduino");       // print message at (0, 0)
  lcd.setCursor(2, 1);        // move cursor to   (2, 1)
  lcd.print("GetStarted"); // print message at (2, 1)
  delay(2000);                // display the above for two seconds
  lcd.clear();                // clear display
  lcd.setCursor(3, 0);         // move cursor to   (3, 0)
  lcd.print("DIY");       // print message at (3, 0)
  lcd.setCursor(0, 1);         // move cursor to   (0, 1)
  lcd.print("GROUP NO 9"); // print message at (0, 1)
  delay(2000);                 // display the above for two seconds
}
```