Total Dissolved Solids Measurements Using Arduino

Sunday November 17, 2024

Total dissolved solids (TDS) refer to the combined content of all inorganic and organic substances contained in a liquid, typically water, that are present in a molecular, ionized, or micro-granular form. TDS is usually measured in milligrams per liter (mg/L) or parts per million (ppm). (I'll be using ppm in this article)

The components of TDS can include a variety of substances, such as:

- Salts (e.g., sodium, calcium, magnesium, potassium)
- Minerals (e.g., bicarbonates, sulfates, chlorides)
- Organic matter (e.g., humic substances)

TDS is an important parameter in water quality assessment, as it can affect the physical and chemical properties of water, including its taste, clarity, and suitability for drinking, irrigation, and industrial processes. High levels of TDS can indicate water pollution or the presence of excessive minerals, which may be harmful to health or the environment.

Hardware

- Arduino Uno board
- CQRobot TDS Sensor
- male-to-male jumper wires

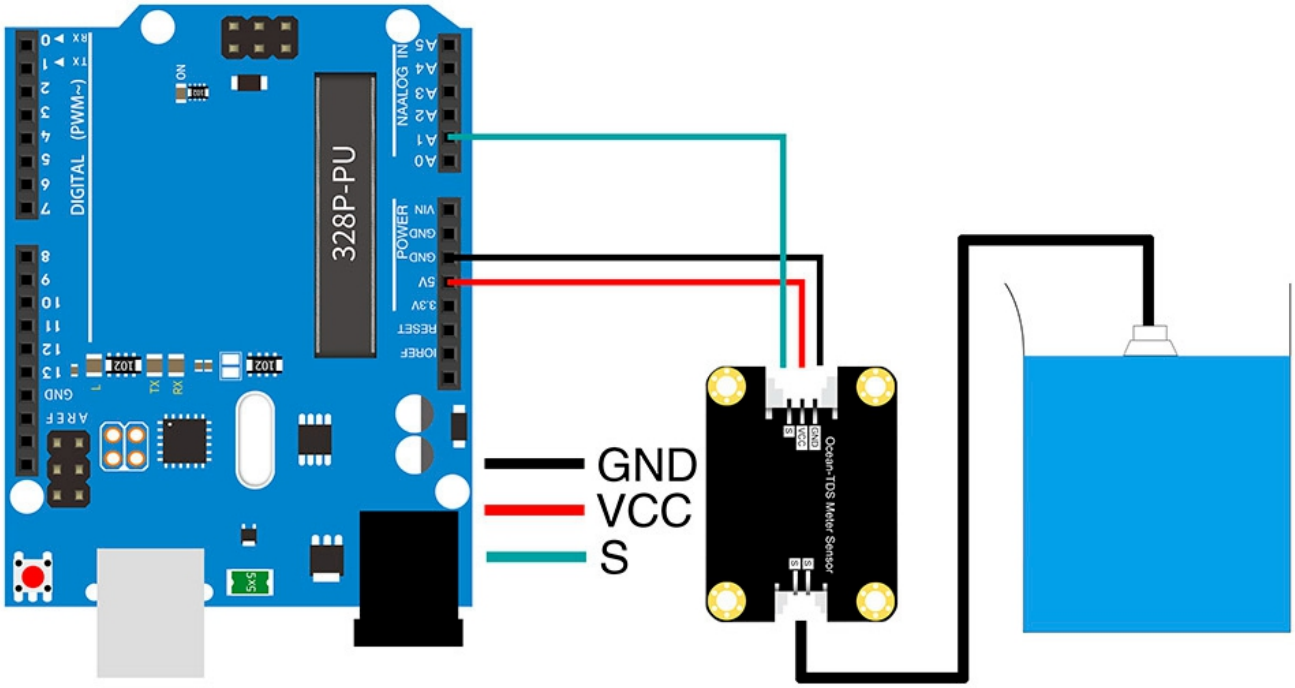
Software

You are welcome to use whichever software combination you prefer. However, I will be using the following tools, and the instructions in this article will be based on that setup:

- Ubuntu Linux *(I'm using Ubuntu MATE, specifically)*
- Zed editor
- Arduino CLI

Hardware Setup

Connect the TDS Sensor controller to the Arduino's 5V, Ground, and A1 pins, and suspend the sensor probe in a container of water:



Code and Test

Use the Arduino CLI to create a sketch:

```
arduino-cli sketch new tds_arduino

cd tds_arduino
```

Create a new Makefile. This will simplify compiling, uploading, and monitoring our Arduino sketch:

```
CORE = arduino:avr
FQBN = $(CORE):uno
CLI_EXE = arduino-cli
PORT = /dev/ttyACM0
BAUD = 115200

default:
    @echo 'Targets:'
    @echo '  list'
    @echo '  compile'
    @echo '  upload'
    @echo '  monitor'

list:
    $(CLI_EXE) board list
```

```
compile:
    $(CLI_EXE) compile --fqbn $(FQBN) .

upload: compile
    $(CLI_EXE) upload -p $(PORT) --fqbn $(FQBN) .

monitor:
    $(CLI_EXE) monitor -p $(PORT) --config $(BAUD)
```

Update the contents of **tds_arduino.ino** as follows:

```
#define TdsSensorPin A1

void setup() {
  // Initialize serial communication at a baud rate of 115200 bits per
second:
  Serial.begin(115200);

  // Set the mode of the pin to INPUT: This pin will be used to read data
from
  // the TDS sensor:
  pinMode(TdsSensorPin, INPUT);
}

void loop() {
  static unsigned long analogSampleTimepoint = millis();

  // take a reading and display it every 1/2 second:
  if (millis() - analogSampleTimepoint > 500U) {
    analogSampleTimepoint = millis();

    // get sensor pin value
    //
https://docs.arduino.cc/language-reference/en/functions/analog-io/analogRead
/
    int sensorValue = analogRead(TdsSensorPin);

    Serial.print("Sensor value: ");
    Serial.println(sensorValue);
  }
}
```

Compile and upload the sketch, then run the monitor:

```
make upload

make monitor
```

Results:

```
Sensor value: 228
```

```
Sensor value: 228
Sensor value: 227
Sensor value: 228
Sensor value: 229
Sensor value: 229
Sensor value: 229
Sensor value: 228
```

The values we're capturing at this point are *not* ppm of total dissolved solids: They're voltage levels being reported from the sensor pin. Moreover, they aren't *actual* voltage levels, but an abstraction of them. (The documentation for `analogRead` explains it well: https://docs.arduino.cc/language-reference/en/functions/analog-io/analogRead/)

In order to be useful, we need to convert them. First we'll add some global values:

```
float referenceVoltage =
    5.0; // analog reference voltage of the analog-to-digital convertor
(ADC)
float tdsValue = 0;
```

Next, our conversion function:

```
float getTdsValue(int sensorValue, float temperatureAtCollection) {
  // Convert the sensor pin reading to actual voltage:
  float sensorVoltage = (float)sensorValue * referenceVoltage / 1024.0;

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

  // temperature compensation
  float compensationVoltage = sensorVoltage / compensationCoefficient;

  // convert voltage value to tds value
  float calculatedTdsValue =
      (133.42 * compensationVoltage * compensationVoltage *
           compensationVoltage -
       255.86 * compensationVoltage * compensationVoltage +
       857.39 * compensationVoltage) *
      0.5;

  return calculatedTdsValue;
}
```

The function takes in two values: the sensor reading and the temperature at the time of collection. (Since we aren't working with a temperature sensor in this exercise, we'll use a default value for that) The logic for voltage, temperature compensation, and voltage-to-tds come from the CQRobot wiki.

Also, the collection temperature must be in Celsius. Since we're likely to be working with Fahrenheit, we'll add a function for that conversion as well:

```
float fahrenheitToCelsius(float temperature) {
  float celsiusValue = ((temperature - 32) * 5) / 9;

  return celsiusValue;
}
```

Finally, we'll update our processing loop to use the new functions:

```
void loop() {
  static unsigned long analogSampleTimepoint = millis();

  // take a reading and display it every 1/2 second:
  if (millis() - analogSampleTimepoint > 500U) {
    analogSampleTimepoint = millis();

    // get sensor pin value
    //
https://docs.arduino.cc/language-reference/en/functions/analog-io/analogRead
/
    int sensorValue = analogRead(TdsSensorPin);

    // Convert temperature to celsius, using a default temperature of 77
degrees
    // Fahrenheit:
    float celsiusValue = fahrenheitToCelsius(77);

    // Calculate the actual TDS value:
    tdsValue = getTdsValue(sensorValue, celsiusValue);

    Serial.print("TDS -- PPM: ");
    Serial.println(tdsValue);
  }
}
```

Now, our complete updated code looks like this:

```
#define TdsSensorPin A1

float referenceVoltage =
    5.0; // analog reference voltage of the analog-to-digital convertor
(ADC)
float tdsValue = 0;

float fahrenheitToCelsius(float temperature) {
  float celsiusValue = ((temperature - 32) * 5) / 9;

  return celsiusValue;
}
```

```
float getTdsValue(int sensorValue, float temperatureAtCollection) {
  // Convert the sensor pin reading to actual voltage:
  float sensorVoltage = (float)sensorValue * referenceVoltage / 1024.0;

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

  // temperature compensation
  float compensationVoltage = sensorVoltage / compensationCoefficient;

  // convert voltage value to tds value
  float calculatedTdsValue =
      (133.42 * compensationVoltage * compensationVoltage *
          compensationVoltage -
       255.86 * compensationVoltage * compensationVoltage +
       857.39 * compensationVoltage) *
      0.5;

  return calculatedTdsValue;
}

void setup() {
  // Initialize serial communication at a baud rate of  115200 bits per
second:
  Serial.begin(115200);

  // Set the mode of the pin to INPUT: This pin will be used to read data
from
  // the TDS sensor:
  pinMode(TdsSensorPin, INPUT);
}

void loop() {
  static unsigned long analogSampleTimepoint = millis();

  // take a reading and display it every 1/2 second:
  if (millis() - analogSampleTimepoint > 500U) {
    analogSampleTimepoint = millis();

    // get sensor pin value
    //
https://docs.arduino.cc/language-reference/en/functions/analog-io/analogRead
/
    int sensorValue = analogRead(TdsSensorPin);

    // Convert temperature to celsius, using a default temperature of 77
degrees
```

```
    // Fahrenheit:
    float celsiusValue = fahrenheitToCelsius(77);

    // Calculate the actual TDS value:
    tdsValue = getTdsValue(sensorValue, celsiusValue);

    Serial.print("TDS -- PPM: ");
    Serial.println(tdsValue);
  }
}
```

When we compile, upload, and run our serial monitor, we now get the PPM values we want:

```
make upload

make monitor
```

```
TDS -- PPM: 403.13
TDS -- PPM: 405.03
TDS -- PPM: 403.13
TDS -- PPM: 403.13
TDS -- PPM: 403.13
TDS -- PPM: 405.03
TDS -- PPM: 403.13
```