**How the code works.**

* The TdsSensorPin variable saves the GPIO where you want to get the readings. We chose GPIO27, but you can use any other ADC pin.

#define TdsSensorPin 27

* Then, insert the analog voltage reference for the ADC. For the ESP32 is 3.3V, for an Arduino, for example, it is 5V.

#define VREF 3.3 // analog reference voltage(Volt) of the ADC

* Before getting a measurement value, we’ll apply a median filtering algorithm to get a more stable value. The SCOUNT variable refers to the number of samples we’ll filter before getting an actual value.

#define SCOUNT 30 // sum of sample point

* Then, we need some arrays to store the readings as well as some index variables that will allow us to go through the arrays.

int analogBuffer[SCOUNT]; // store the analog value in the array, read from ADC

int analogBufferTemp[SCOUNT];

int analogBufferIndex = 0;

int copyIndex = 0;

Initialize the averageVoltage variable and tsdValue as float variables.

float averageVoltage = 0;

float tdsValue = 0;

* The temperature variable saves the current temperature value. The temperature influences the readings, so there is an algorithm that compensates for fluctuations in temperature. In this example, the reference temperature is 25ºC, but you can change it depending on your environment. For more accurate results, you can add a temperature sensor and get the actual temperature at the time of reading the sensor.

float temperature = 25; // current temperature for compensation

* The following function will be used to get a stable TDS value from an array of readings.

// 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;

}

* In the setup(), initialize the Serial Monitor at a baud rate of 115200.

Serial.begin(115200);

* Set the TDS sensor pin as an input.

pinMode(TdsSensorPin,INPUT);

* In the loop(), get new TDS readings every 40 milliseconds and save them in the buffer:

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;

}

}

* Every 800 milliseconds, it gets the latest readings and gets the average voltage by using the filtering algorithm created before:

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 / 4096.0;

* Then, it calculates a temperature compensation coefficient and calculates the TDS value taking that value into account:

//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;

* Finally, it prints the TDS value in ppm:

Serial.print("TDS Value:");

Serial.print(tdsValue,0);

Serial.println("ppm");