#include <SPI.h>

#include <WiFiNINA.h>

#include <PubSubClient.h>

const char\* ssid = "psu-iot";

const char\* password = "y2nfu9jih82q";

const char\* mqttServer = "41.193.5.154";

const int mqttPort = 24500;

const char\* mqttClientName = "MKRWiFi1010Client";

const char\* tdsTopic = "tdsSensorData";

WiFiClient wifiClient;

PubSubClient client(wifiClient);

#define TdsSensorPin A1

#define VREF 5.0

#define SCOUNT 30

int analogBuffer[SCOUNT];

int analogBufferIndex = 0;

float averageVoltage = 0;

float tdsValue = 0;

float temperature = 16;

void setup() {

Serial.begin(9600);

setupWiFi();

setupMQTT();

}

void loop() {

handleWiFi();

handleMQTT();

readSensor();

processSensorData();

delay(2000);

}

void setupWiFi() {

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("Connected to WiFi");

}

void setupMQTT() {

client.setServer(mqttServer, mqttPort);

client.setCallback(callback);

while (!client.connected()) {

Serial.println("Connecting to MQTT...");

if (client.connect(mqttClientName)) {

Serial.println("Connected to MQTT");

client.subscribe(tdsTopic);

} else {

Serial.print("Failed to connect to MQTT with state ");

Serial.println(client.state());

delay(2000);

}

}

}

void handleWiFi() {

if (WiFi.status() != WL\_CONNECTED) {

Serial.println("WiFi disconnected. Reconnecting...");

setupWiFi();

}

}

void handleMQTT() {

if (!client.connected()) {

Serial.println("MQTT disconnected. Reconnecting...");

setupMQTT();

}

client.loop();

}

void readSensor() {

static unsigned long analogSampleTimepoint = millis();

if (millis() - analogSampleTimepoint > 40U) {

analogSampleTimepoint = millis();

analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin);

analogBufferIndex = (analogBufferIndex + 1) % SCOUNT;

}

}

void processSensorData() {

static unsigned long printTimepoint = millis();

if (millis() - printTimepoint > 800U) {

printTimepoint = millis();

averageVoltage = calculateAverageVoltage();

float compensationCoefficient = 1.0 + 0.02 \* (temperature - 25.0);

float compensationVoltage = averageVoltage / compensationCoefficient;

tdsValue = calculateTDS(compensationVoltage);

Serial.print("TDS Value:");

Serial.print(tdsValue, 0);

Serial.println("ppm");

publishData(tdsValue); // Publish TDS value to MQTT broker

}

}

float calculateAverageVoltage() {

float sum = 0;

for (int i = 0; i < SCOUNT; i++) {

sum += analogBuffer[i];

}

return sum / SCOUNT \* VREF / 1024.0;

}

float calculateTDS(float voltage) {

return (133.42 \* voltage \* voltage \* voltage - 255.86 \* voltage \* voltage + 857.39 \* voltage) \* 0.5;

}

void publishData(float data) {

char messageBuffer[50];

snprintf(messageBuffer, 50, "%.2f", data);

if (client.publish("TDS Sensor", messageBuffer)) {

Serial.println("Publish succeeded");

} else {

Serial.println("Publish failed");

}

}

void callback(char\* topic, byte\* payload, unsigned int length) {

// Handle MQTT messages if needed

}