

Sprint 3

Date	14.11.2022
Team ID	PNT2022TMID28239
Project Name	Real-Time Water QualityMonitoring And Control System

```
#include <ESP8266HTTPClient.h>#include <FirebaseArduino.h> #include <DNSServer.h>
#include <ESP8266WiFi.h> #include <ESP8266WebServer.h>#include <WiFiManager.h>
#include <TimeLib.h> //library to get time and date#include <WiFiUdp.h>
#include <OneWire.h>#include <Servo.h>
```

```
// Set these to run example.
```

```
#define FIREBASE_HOST "iot839-a034d.firebaseio.com"#define FIREBASE_AUTH
```

```
#define WIFI_SSID "xxx"
```

```
#define WIFI_PASSWORD "yyy"
```

```
#define StartConvert 0
```

```
#define ReadTemperature 1#define ecSwitch D6 #define tempSwitch D7 #define turbiditySwitch
D8
```

```
/* EC and Temp */
```

```
const byte numReadings = 20; //the number of sample times byte ECSensorPin = A0; //EC
Meter analog output,pin on analog 1byte DS18B20_Pin = D2; //DS18B20 signal, pin on digital 2
unsigned int
```

```
AnalogSampleInterval=25,printInterval=700,tempSampleInterval=850; unsigned int
```

```
readings[numReadings];
```

```
byte index = 0; // the index of the current reading unsigned long AnalogValueTotal = 0;
// the running total
```

```

unsigned int AnalogAverage = 0,averageVoltage=0;           // the averageunsigned long
AnalogSampleTime,printTime,tempSampleTime;
float temperature,ECcurrent;

//Temperature chip I/o
OneWire ds(DS18B20_Pin); // on digital pin 2

#define pHsensorPin A0      //pH meter Analog output to Arduino Analog Input
#define phSwitch D5
unsigned long int avgValueForPH; //Store the average value of the ph sensorfeedback
int pHbuffer[10],tempValueForPH;float pHValue;

#define turbiditysensorPin A0    //turbidity meter Analog output to ArduinoAnalog Input 0
unsigned long int avgValueForTurbidity; //Store the average value of the turbiditysensor
feedback
int turbiditybuffer[10],tempValueForTurbidity;float turbidityValue;

#define MUX_A D3#define MUX_B D4

Servo myservo; #define servoPin D1

// NTP Servers:
static const char ntpServerName[] = "asia.pool.ntp.org";
const int timeZone = +6; // Convert to Bangladesh Standard Time (BST)

WiFiUDP Udp;
unsigned int localPort = 8888; // local port to listen for UDP packets

time_t getNtpTime();
void sendNTPpacket(IPAddress &address);void setup()

```

```

{
  Serial.begin(115200);
  // connect to wifi using WifiManager library.WiFiManager wifiManager;
  //wifiManager.autoConnect("AutoConnectAP");wifiManager.autoConnect("PureraWater");

  Serial.println(); Serial.print("connected: "); Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //Connection toInternet &
  Firebase

  for (byte thisReading = 0; thisReading < numReadings; thisReading++)
    readings[thisReading] = 0;
  TempProcess(StartConvert); //let the DS18B20 start the convert
  AnalogSampleTime=millis();
  printTime=millis(); tempSampleTime=millis();

  pinMode(MUX_A, OUTPUT); pinMode(MUX_B, OUTPUT);
  pinMode(phSwitch,OUTPUT); pinMode(ecSwitch,OUTPUT);
  pinMode(turbiditySwitch,OUTPUT);

  myservo.attach(servoPin);Udp.begin(localPort);
  //Serial.print("Local port: ");
  //Serial.println(Udp.localPort());
  //Serial.println("waiting for sync");setSyncProvider(getNtpTime); setSyncInterval(300);

  Serial.begin(115200);
}

void loop()
{

```

```

    /* READING PH */motorOn(); delay(3000);
digitalWrite(phSwitch, HIGH); //power up ph sensor changeMux(LOW, LOW); // selector
S1=0, S0 = 0; ph setup in Y0delay(3000);
phRead(); //taking reading
digitalWrite(phSwitch, LOW); //power down ph sensor after readingdelay(3000); //reading
complete, now preaparing to take next reading

motorOFF();delay(3000);
for (int i=0;i<3;i++){ digitalWrite(ecSwitch,HIGH);
    changeMux(LOW, HIGH); // selector S1=0, S0 = 0; ph setup in Y1delay(3000);
    EcAndTempReading(); digitalWrite(ecSwitch,LOW);delay(3000);
}

while(ECcurrent<0){ digitalWrite(ecSwitch,HIGH);
    changeMux(LOW, HIGH); // selector S1=0, S0 = 0; ph setup in Y1delay(3000);
    EcAndTempReading(); digitalWrite(ecSwitch,LOW);delay(3000);
}
motorOn(); delay(3000);
/* READING turbidity */
digitalWrite(turbiditySwitch, HIGH); //power up turbidity sensor changeMux(HIGH,
LOW); // selector S1=1, S0 = 0; turbidity setup in Y2delay(3000);
turbidityRead();//taking reading
digitalWrite(turbiditySwitch, LOW);//power down turbidity sensor after reading
delay(3000);//reading complete, now preaparing to take next reading

```

```
StaticJsonBuffer<200> jsonBuffer; JsonObject& root = jsonBuffer.createObject();
String date = (String) day()+ '/' + month()+ '/' + year(); String timee = (String)
hour()+ ':' + minute()+ ':' + second(); root["date"] = date;
root["time"] = timee; root["turbidity"] = turbidityValue; root["ph"] = pHValue; root["temp"]
= temperature; root["ec"] = ECcurrent; Serial.println("Firebase data: "); Serial.print("Date:
"); Serial.println(date); Serial.print("Time: "); Serial.println(timee); Serial.print("Turbidity:
"); Serial.println(turbidityValue); Serial.print("PH: "); Serial.println(pHValue);
Serial.print("Temperature: "); Serial.println(temperature); Serial.print("EC: ");
Serial.println(ECcurrent);
```

```
// append a new value to /logDHT
```

```
String name = Firebase.push("/sensor_data", root); delay(500000); //END TAKING ALL
READING
```

```
//delay(3000); //END TAKING ALL READING
```

```
}
```

```
void changeMux(int b, int a) { digitalWrite(MUX_A, a); digitalWrite(MUX_B, b);
}
```

```
void motorOn() { myservo.write(180);
```

```

}

void motorOFF(){ myservo.write(0);
}

void pHRead(){
  for(int i=0;i<10;i++)    //Get 10 sample value from the sensor for smooth thevalue
  {
    pHbuffer[i]=analogRead(pHsensorPin);
    // Serial.println(pHbuffer[i]);delay(10);
  }
  avgValueForPH=0;
  for(int i=2;i<8;i++)      //take the average value of 6 center sample
    avgValueForPH+=pHbuffer[i];
  pHValue=(float)avgValueForPH*3.33/1024/6; //convert the analog into millivolt
  pHValue=4.7*pHValue;      //convert the millivolt into pH value Serial.print("
    pH:");
  Serial.print(pHValue,2);Serial.println(" ");
}

void EcAndTempReading(){
  /*
   Every once in a while,sample the analog value and calculate the average.
  */
  if(millis()-AnalogSampleTime>=AnalogSampleInterval)
  {
    AnalogSampleTime=millis();
    // subtract the last reading:
    AnalogValueTotal = AnalogValueTotal - readings[indx];
    // read from the sensor:
    readings[indx] = analogRead(ECsensorPin);
    // add the reading to the total:
    AnalogValueTotal = AnalogValueTotal + readings[indx];
    // advance to the next position in the array:indx = indx + 1;
  }
}

```

```

// if we're at the end of the array...if (indx >= numReadings)
// ...wrap around to the beginning:
indx = 0;
// calculate the average:
AnalogAverage = AnalogValueTotal / numReadings;
}
/*
Every once in a while,MCU read the temperature from the DS18B20 and then let the
DS18B20 start the convert.
Attention:The interval between start the convert and read the temperature should be greater
than 750 millisecond,or the temperature is not accurate!
*/
if(millis()-tempSampleTime>=tempSampleInterval)
{
tempSampleTime=millis();
temperature = TempProcess(ReadTemperature); // read the current temperature from the
DS18B20
TempProcess(StartConvert); //after the reading,start the convert for next reading
}
/*
Every once in a while,print the information on the serial monitor.
*/
if(millis()-printTime>=printInterval)
{
printTime=millis(); averageVoltage=AnalogAverage*(float)5000/1024;
//averageVoltage = averageVoltage *12; //to adjust
Serial.print("Analog value:");
Serial.print(AnalogAverage); //analog average,from 0 to 1023
Serial.print(" Voltage:");
Serial.print(averageVoltage); //millivolt average,from 0mv to 4995mV
Serial.print("mV
");
Serial.print("temp:");
Serial.print(temperature); //current temperature
Serial.print("^C EC:");

```

```

float TempCoefficient=1.0+0.0185*(temperature-25.0); //temperaturecompensation
formula: fFinalResult(25^C) = fFinalResult(current)/(1.0+0.0185*(fTP-25.0));
float CoefficientVolatge=(float)averageVoltage/TempCoefficient; if(CoefficientVolatge<1)
    Serial.println("No solution!"); //25^C 1413us/cm<-->about 216mv if the
voltage(compensate)<150,that is <1ms/cm,out of the range
else if(CoefficientVolatge>3300)
    Serial.println("Out of the range!"); //>20ms/cm,out of the rangeelse
{
    if(CoefficientVolatge<=448)
        ECcurrent=(6.84*CoefficientVolatge)-62.32; //1ms/cm<EC<=3ms/cm else
    if(CoefficientVolatge<=1457) ECcurrent=(6.98*CoefficientVolatge)-125;
    //3ms/cm<EC<=10ms/cm else
    ECcurrent=(5.3*CoefficientVolatge)+2280;
//10ms/cm<EC<20ms/cm
    ECcurrent/=1000; //convert us/cm to ms/cmSerial.print(ECcurrent,2); //two decimal
    Serial.println("ms/cm");
}
}
}
/*
ch=0,let the DS18B20 start the convert;ch=1,MCU read the current temperaturefrom the
DS18B20.
*/
float TempProcess(bool ch)
{
    //returns the temperature from one DS18B20 in DEG Celsiusstatic byte data[12];
    static byte addr[8];
    static float TemperatureSum;if(!ch){
        if ( !ds.search(addr)) {
            Serial.println("no more sensors on chain, reset search!");ds.reset_search();
            return 0;
        }
    }

```



```

    if ( OneWire::crc8( addr, 7) != addr[7]) { Serial.println("CRC is not valid!"); return 0;
    }
    if ( addr[0] != 0x10 && addr[0] != 0x28) { Serial.print("Device is not recognized!");
        return 0;
    }
    ds.reset(); ds.select(addr);
    ds.write(0x44,1); // start conversion, with parasite power on at the end
}
else{
    byte present = ds.reset();ds.select(addr);
    ds.write(0xBE); // Read Scratchpad
    for (int i = 0; i < 9; i++) { // we need 9 bytes
        data[i] = ds.read();
    }
    ds.reset_search(); byte MSB = data[1];byte LSB = data[0];
    float tempRead = ((MSB << 8) | LSB); //using two's compliment
    TemperatureSum =
    tempRead / 16;
}
    return TemperatureSum;
}

void turbidityRead(){
    Serial.println("Taking Readings from turbidity Sensor");turbidityValue = 0;
    for (int i=0; i<10; i++){
        turbidityValue += analogRead(turbiditysensorPin);delay(10);
    }
    turbidityValue /= 10;
    float turbidityV = turbidityValue/100; turbidityV = round_to_dp(turbidityV, 1);
    turbidityValue = turbidityV;
    //Serial.print("Turbidity level: ");

```

```

//Serial.println(turbidityV);if( turbidityV > 9){
    Serial.print("Turbidity Level: ");Serial.println(turbidityV);
    // Serial.println("NTU"); Serial.println("Very Clean");
    // delay(3000);
}

if( turbidityV >= 8.5 && turbidityV <= 9 ){Serial.print("Turbidity Level: ");
    Serial.println(turbidityV);
    // Serial.println("NTU");Serial.println("Clean");
}

if(turbidityV >= 6 && turbidityV < 8.5){Serial.print("Turbidity Level: ");
    Serial.println(turbidityV);
    // Serial.println("NTU");Serial.println("Dirty");
    // delay(3000);
}
if( turbidityV < 6){ Serial.print("Turbidity Level: ");Serial.println(turbidityV);
    // Serial.println("NTU"); Serial.println("Very Dirty");
    // delay(3000);
}
}

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;
}

/*----- NTP code Don't change Anything----- */

```

```

const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets

time_t getNtpTime()
{
    IPAddress ntpServerIP; // NTP server's ip address

    while (Udp.parsePacket() > 0) ; // discard any previously received packets
    Serial.println("Transmit NTP Request");
    // get a random server from the pool WiFi.hostByName(ntpServerName, ntpServerIP);
    Serial.print(ntpServerName);
    Serial.print(": "); Serial.println(ntpServerIP); sendNTPpacket(ntpServerIP); uint32_t
    beginWait = millis();
    while (millis() - beginWait < 1500) { int size = Udp.parsePacket();
        if (size >= NTP_PACKET_SIZE) { Serial.println("Receive NTP Response");
            Udp.read(packetBuffer, NTP_PACKET_SIZE); // read packet into the buffer
            unsigned long secsSince1900;
            // convert four bytes starting at location 40 to a long integer
            secsSince1900 = (unsigned long)packetBuffer[40] << 24;
            secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
            secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
            secsSince1900 |= (unsigned long)packetBuffer[43];
            return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
        }
    }
    Serial.println("No NTP Response :-("); return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{

```

```
memset(packetBuffer, 0, NTP_PACKET_SIZE);

// Initialize values needed to form NTP request

// (see URL above for details on the packets) packetBuffer[0] = 0b11100011;    // LI,
Version, ModepacketBuffer[1] = 0; // Stratum, or type of clock packetBuffer[2] = 6;    //
Polling Interval packetBuffer[3] = 0xEC; // Peer Clock Precision

// 8 bytes of zero for Root Delay & Root DispersionpacketBuffer[12] = 49;

packetBuffer[13] = 0x4E;packetBuffer[14] = 49;

packetBuffer[15] = 52;

// all NTP fields have been given values, now

// you can send a packet requesting a timestamp: Udp.beginPacket(address, 123); //NTP
requests are to port 123Udp.write(packetBuffer, NTP_PACKET_SIZE); Udp.endPacket();

}
```