LAMPIRAN

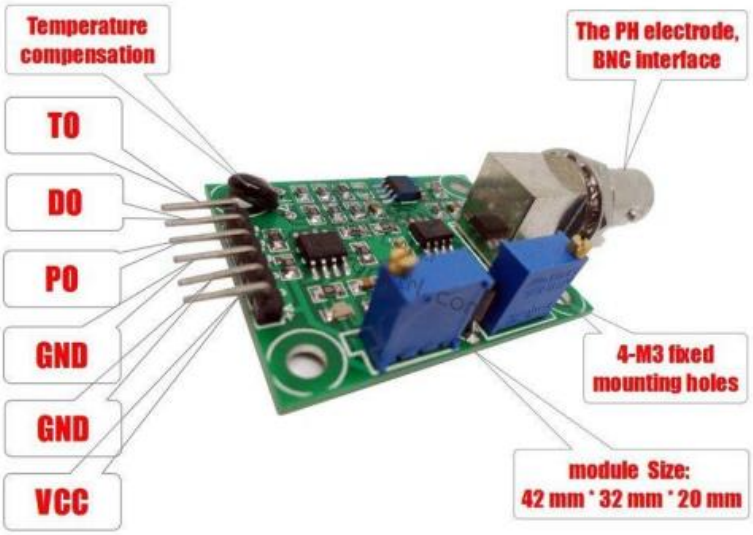
How to use a PH probe and sensor

If you worked with PH metering before you will know that PH values range from 0-14. Where PH 0 Will be very acidic, PH 7 will be neutral and PH 14 very alkaline. Water is near a PH 7 and this is usually around here that we will need to monitor PH of many things. A swimming pool, for example, should be slightly alkaline at 7.2, hydroponics systems around 6 (for optimum plant nutrition takeup) and aquaponics around 6.8.

I wrote this **PH probe and sensor** "how to" because it is not as straightforward as one would think (but quite easy when you understand the ins and outs) mostly because there is not a lot of information on this on the Internet, surely not detailed information.

We will first look at the ph probe module board and then the PH probe because both the PH probe and sensor have to be set correctly:

- offset setting
- limit setting
- sketch to test the board analogue range
- sketch for PH reading and calibration.
- calibration of PH probe
- PH probe usage
  The ph probe module in this tutorial is available on our site here: PH probe module BNC conector



PH Probe Sensor Pinout

TO – Temperature output
DO – 3.3V Output (from ph limit pot)
PO – PH analog output ==> **Arduino A0**
Gnd – Gnd for PH probe (can come from Arduino GND pin) **==> Arduino GND**
Gnd – Gnd for board (can also come from Arduino GND pin) **==> Arduino GND**
VCC – 5V DC (can come from Arduino 5V pin) **==> Arduino 5V pin**
POT 1 – Analog reading offset (Nearest to BNC connector)
POT 2 – PH limit setting

PH probe module Offset and how to use it.

This board have the ability to supply a voltage output to the analogue board that will represent a PH value just like any other sensor that will connect to an analog pin. Ideally, we want a PH 0 represent 0v and a PH of 14 to represent 5V.

BUT there is a catch……, this board by default have PH 7 set to 0V (or near it, it differs from one PH probe to another, that is why we have to calibrate the probe as you will see later on), This means that the voltage will go into the minuses when reading acidic PH values and that cannot be read by the analog Arduino port. The offset pot is used to change this so that a PH 7 will read the expected 2.5V to the Arduino analog pin, the analog pin can read voltages between 0V and 5V hence the 2.5V that is halfway between 0V and 5V as a PH 7 is halfway between PH 0 and PH 14,

You will need to turn the offset potentiometer to get the right offset, The offset pot is the blue pot nearest to the BNC connector.

To set the offset is easy. First, you need to disconnect the probe from the circuit and short-circuit the inside of the BNC connector with the outside to simulate a neutral PH (PH7). I took a piece of wire, strip both sides, wrap the one side around the outside of the BNC connector and push the other side into the BNC hole. This short-circuit represents about a neutral PH reading of 7.

There are two ways you can do the adjustment.

If you have a multimeter handy you can measure the value of the PO pin and adjust the offset potentiometer until PO measures 2.5V.

I prefer to just use the sketch below. Just download it to your Arduino as you will with any other sketch, open serial monitor and view the reading there. All this sketch does is to print the volts it receives from the analog pin and print it to the serial monitor. It of course first changes the digital value to volts to make it easier. Now simply turn the offset pot until it is exactly 2.5V. You can learn more about reading voltages and digital representation of volts here: https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

Offset sketch

```
void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
}

// the loop routine runs over and over showing the voltage on A0
void loop() {
 // read the input on analog pin 0:
 int sensorValue = analogRead(A0);
 // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
 float voltage = sensorValue * (5.0 / 1023.0);
 // print out the value you read:
 Serial.println(voltage);
 delay(300);
}
```

PH limit setting

There is another pot that acts like a limit switch. Basically, the D0 pin on the sensor board will supply 3.3V to the pin until a preset PH value (that you set with the limit pot) is reached, at this point a red LED will light up and the pin will go down to about 0V.

I did not play with this much but suppose it can be handy if you want to activate a buzzer or something if a certain PH is reached, it will work great on an Arduino digital port – that will go high from about 2V up.

This will work if the PH value goes higher than the set value. If you want it to trigger something when the PH goes lower, you need to monitor the digital pin to trigger when the digital pin goes low.

You will unfortunately not be able to set this limit between two values, either if the pH goes up to high or if the PH drop to low. Programmatically you of cause can do an upper and lower limit.

Connecting and calibrating the PH probe.

The hard part is over and this offset does not have to be set again, even if you change PH probes. We have PH probes available here: PH probe Electrode BNC connector
Here is a couple of things to know about PH probes:

1. The probes readings change over time and need to be calibrated every now and again to make sure the value is still the same and be adjusted if it did change.
2. You need at least one PH buffer solution to calibration your PH probe. They are available at many different PH values, A buffer solution of 6.86 and 4.01 is most common as it covers the range of most applications. If you are only going to use one buffer solution make sure its value is near the value range you will use in your normal tests – if it is pool water a buffer solution of 6.86 is usually near enough.
3. Buffers come in pre-made solutions or as a powder. I prefer the powder because it is cheaper and does not have an expiration date. The powder is easy to make up as well, I suppose it depends on the power you will use, the one I use you add the powder to 250ml distilled water and stir until all powder is dissolved. It will last about a month once you added water to it.

A PH probe **takes some time** to get to the right value, allow it to be in the liquid you want to measure for at least two minutes or longer, it does not mean it will be stable at one ph value,

it will jump around a bit between 3 or 4 values but on the last digit, for example, between 6.84 – 6.88

4. PH values differ in different temperatures, although that might sound cumbersome, in the temperature range between 10 – 30 degrees Celcius the PH does not differ and from 30 degrees Celcius it goes up with about a pH of 0.01 to 50 degrees Celcius that is about 0.06. In most uses, it will be below 30 degrees Celcius and temperature do not have to be calculated in.

Hook up your PH probe after you removed the wire you used to short-circuit the BNC connector and download the sketch below.

PH measurement sketch

```
float calibration = 0.00; //change this value to calibrate
const int analogInPin = A0;
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;
void setup() {
 Serial.begin(9600);
}

void loop() {
 for(int i=0;i<10;i++)
 {
 buf[i]=analogRead(analogInPin);
 delay(30);
 }
 for(int i=0;i<9;i++)
 {
 for(int j=i+1;j<10;j++)
 {
 if(buf[i]>buf[j])
 {
 temp=buf[i];
 buf[i]=buf[j];
 buf[j]=temp;
 }
 }
```

```
}
avgValue=0;
for(int i=2;i<8;i++)
avgValue+=buf[i];
float pHVol=(float)avgValue*5.0/1024/6;
float phValue = -5.70 * pHVol + calibration;
Serial.print("sensor = ");
Serial.println(phValue);


delay(500);
}
```

A note on buffer solutions: do not **CROSS CONTAMINATE!** What I mean by this is to not take the probe from one buffer solution to another or from a liquid sample you tested to a buffer solution without rinsing it thoroughly with distilled water first. You will change your buffer solutions ph and your calibration will be off.

When you place the probe in the first solution you might be surprised at how far off it can be, it's normal though. **Remember to leave the probe in the solution for at least two minutes to stabilise.** In the script's top line you will see a variable called ***"calibration".*** Change this value to the difference between what you see in serial monitor and the buffer solutions value, for example, if you read 5.81 and the buffer solution is 6.86 you should change the variable's value to 2.05.

Upload the changed sketch and see how the value looks now.

Some ideas for you.

You can add a potentiometer to your project and program that to change the calibration for you. You always run a risk that the pot might be adjusted my mistake so a button with a 5-second delay can be programmed to put the unit in calibration mode. This becomes great if you add an LCD screen to it and if you add the calibration value in EEPROM so it holds it even if the PH project is powered off.

How about adding a buzzer to the 3.3V limit output to notify you when the PH is out of range. Usually an upper or lower will be enough, in most applications, you will either be worried about high or low PH values but not both. If this is a problem and you need the PH to be in a certain range you can easily do that programmatically and have the buzzer on a digital pin. If it is crucial you can even add a GSM module to this so you can get an SMS or how about a dosing pump to automatically get the PH back in range.

I just might, but do not hold your breath (so much to do but so little time) :-), do something like that in the future with our cheap Skeleton Duino.

PH probe and sensor conclusion

If you go through these steps once it becomes as easy as pie. All the parts, even those I mention in the ideas section is available on our site.

Turbidity sensor SKU: SEN0189

Contents

Introduction

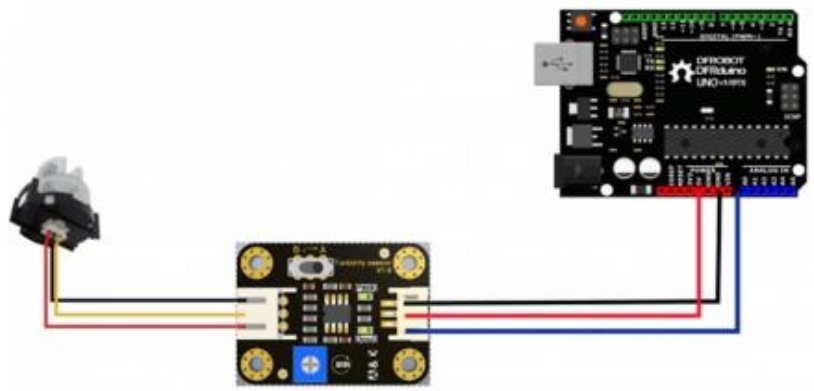
The turbidity sensor detects water quality by measuring the levels of turbidity. It uses light to detect suspended particles in water by measuring the light transmittance and scattering rate, which changes with the amount of total suspended solids (TSS) in water. As the TTS increases, the liquid turbidity level increases.
Turbidity sensors are used to measure water quality in rivers and streams, wastewater and effluent measurements, control instrumentation for settling ponds, sediment transport research and laboratory measurements.
This sensor provides analog and digital signal output modes. The threshold is adjustable when in digital signal mode. You can select the mode according to your MCU.

Note: The top of probe is not waterproof.

Specification

- Operating Voltage: 5V DC
- Operating Current: 40mA (MAX)
- Response Time : <500ms
- Insulation Resistance: 100M (Min)
- Output Method:
  Analog output: 0-4.5V
  Digital Output: High/Low level signal (you can adjust the threshold value by adjusting the potentiometer)
- Operating Temperature: 5℃~90℃
- Storage Temperature: -10℃~90℃
- Weight: 30g
- Adapter Dimensions: 38mm*28mm*10mm/1.5inches *1.1inches*0.4inches

Connection Diagram



Interface Description:

---

1. "D/A" Output Signal Switch
1. "A": Analog Signal Output, the output value will decrease when in liquids with a high turbidity
2. "D": Digital Signal Output, high and low levels, which can be adjusted by the threshold potentiometer
2. Threshold Potentiometer: you can change the trigger condition by adjusting the threshold potentiometer in digital signal mode.

Examples

Here are two examples:
Example 1 uses Analog output mode
Example 2 uses Digital output mode

Example 1

```
void setup() {
  Serial.begin(9600); //Baud rate: 9600
}
void loop() {
  int sensorValue = analogRead(A0);// read the input on analog pin 0:
  float voltage = sensorValue * (5.0 / 1024.0); // Convert the analog read
ing (which goes from 0 - 1023) to a voltage (0 - 5V):
  Serial.println(voltage); // print out the value you read:
  delay(500);
}
```

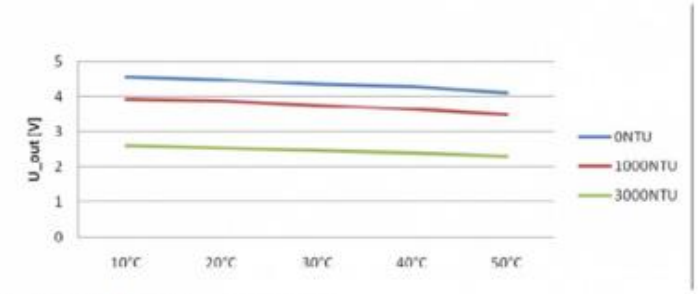
Example 2

```
int ledPin = 13;                // Connect an LED on pin 13, or use the on
board one
int sensor_in = 2;                 // Connect turbidity sensor to Digital
Pin 2
```

```
void setup(){
  pinMode(ledPin, OUTPUT);      // Set ledPin to output mode
  pinMode(sensor_in, INPUT);       //Set the turbidity sensor pin to input
mode
}

void loop(){
   if(digitalRead(sensor_in)==LOW){       //read sensor signal
        digitalWrite(ledPin, HIGH);   // if sensor is LOW, then turn on
     }else{
        digitalWrite(ledPin, LOW);    // if sensor is HIGH, then turn off
the led
     }
}
```

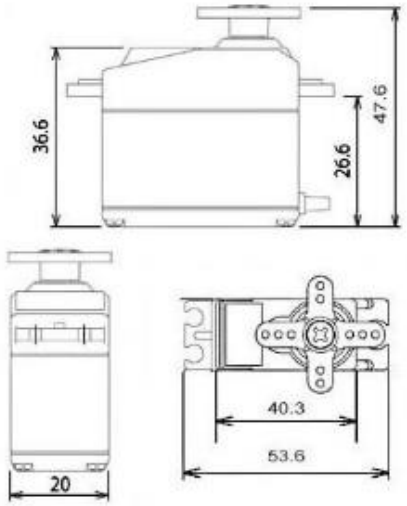
This is a reference chart for the mapping from the output voltage to the NTU according to different temperature. e.g. If you leave the sensor in the pure water, that is NTU < 0.5, it should output "4.1±0.3V" when temperature is 10~50℃.



characteristic curve "Voltage ----Temperature

Note: In the diagram, the unit measuring turbidity is shown as NTU, also it is known as JTU (Jackson Turbidity Unit), 1JTU = 1NTU = 1 mg/L. Refer to Turbidity wikipedia

MG996R High Torque
Metal Gear Dual Ball Bearing Servo



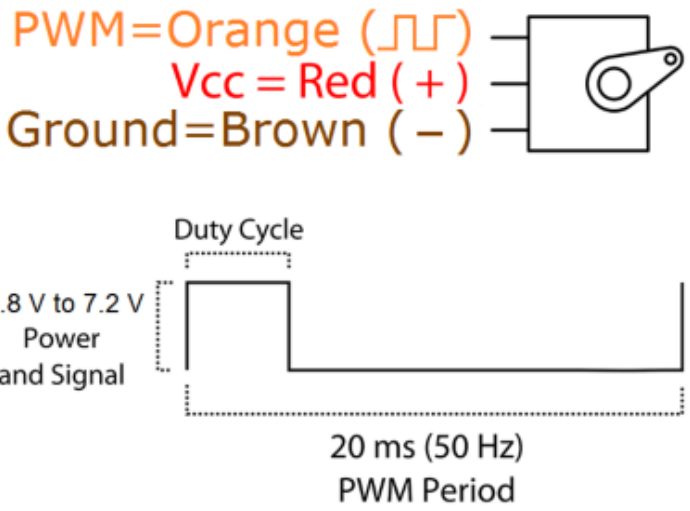
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 °C – 55 °C



Source Code :

```
/* Mendefinisikan template ID, Nama, dan Token otentikasi pada
Blynk */
#define BLYNK_TEMPLATE_ID "TMPL6pd_Fqf8J"
#define BLYNK_TEMPLATE_NAME "Smart Water Treatment"
#define BLYNK_AUTH_TOKEN "ggEAtbAMfgyHo-akqnnECkyi9wf0iWxm"

// Mengimport Library yang dibutuhkan
#include <ESP8266WiFi.h>                    // Wifi ESP8266
#include <BlynkSimpleEsp8266.h>             // Blynk
#include <Adafruit_ADS1X15.h>               // Modul ADS1115
#include <Servo.h>                          // Motor Servo
#include <LiquidCrystal_I2C.h>              // LCD I2C 16x2
```

```
/* Mendefinisikan jaringan wifi dan token otentikasi pada Blynk
*/
char auth[] = "ggEAtbAMfgyHo-akqnnECkyi9wf0iWxm";
char ssid[] = "Tara";
char pass[] = "00000000";

#define SERVO_PIN 0                          // Pin data D3 atau
GPIO 0 untuk motor servo
#define LCD_ADDRESS 0x27                     // alamat LCD I2C
16x2
Servo servoMotor;                            // Mendefinisikan
Motor Servo sebagai servoMotor
LiquidCrystal_I2C lcd(LCD_ADDRESS, 16, 2);  // Mendefinisikan
LCD I2C sebagai lcd
Adafruit_ADS1015 ads;                        // Mendefinisikan
ADS1115 sebagai ads
BlynkTimer timer;                            // Mendefinisikan
objek BlynkTimer

// Deklarasi variabel global
int16_t adc0, adc1, adc2, adc3;              // Pin pada modul
ADS1115
float volts0, volts1, volts2, volts3;        // variabel untuk
menampung tegangan pada pin ADS1115

double Vclear = 2.02;
double tds1 = 0;                             // Turbidity 1
double tds2 = 0;                             // Turbidity 2

float phValue;                               // pH output
float phStep;                                // pH kalibrasi

String tds1Terbilang;                        // Turbidity 1
terbilang
String tds2Terbilang;                        // Turbidity 2
terbilang
String servoTerbilang;                       // Motor Servo
terbilang
```

```
// Kalibrasi
float ph4 = 3.054;
float ph7 = 2.682;

// Memulai program
void setup(void)
{
  Serial.begin(9600);

  // Menghubungkan ke WiFi
  Blynk.begin(auth, ssid, pass);

  // Inisialisasi LCD
  lcd.init();
  lcd.clear();
  lcd.backlight();

  // Menampilkan Hello dan Loading pada LCD
  lcd.setCursor(0, 0);
  lcd.print("HELLO!");
  delay(1000);
  lcd.setCursor(0, 1);
  lcd.print("LOADING...");
  delay(1000);

  /*  melakukan pengecekan wiring modul ADS1115
      Jika modul ADS1115 tidak terhubung, maka tampilkan pesan
      "Failed to initialize ADS" pada serial monitor
  */
  if (!ads.begin()) {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }

  /*  Jika modul ADS1115 terhubung, maka tampilkan pesan
      "ADS1115 OK" pada LCD
  */
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("ADS1115 OK");
  delay(1000);

  // Setup 3V pada channel 0 & 1 modul ADS1115
  ads.startComparator_SingleEnded(0, 1000);
  ads.startComparator_SingleEnded(1, 1000);
  ads.startComparator_SingleEnded(3, 1000);

  // Inisialisasi motor servo dan mengatur posisi pada 0 derajat
  servoMotor.attach(SERVO_PIN, 500, 2400);
  servoMotor.write(0);
}

void loop(void)
{
  Blynk.run();  // Menjalankan Blynk
  timer.run();  // Menjalankan BlynkTimer

  // Membaca data ADC channel 0 & 1 pada modul ADS1115
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  adc3 = ads.readADC_SingleEnded(3);

  // Mengukur tegangan channel 0 & 1 dan 3
  // volts0 = adc0*(3.3/1024);
  // volts1 = adc1*(3.3/1024);
  volts0 = ads.computeVolts(adc0);
  volts1 = ads.computeVolts(adc1);
  volts3 = ads.computeVolts(adc3);

  // Turbidity 1 dan 2
  tds1 = 100.00 - (volts0 / Vclear)*100.00 ;
  tds2 = 100.00 - (volts1 / Vclear)*100.00 ;

  // Ph
  phStep = (ph4 - ph7) / 3;
  phValue = 7.00 + ((ph7 - volts3) / phStep);

  tdsTerbilang();
```

```
  servoControl();

  Serial.println("-----------------------------------------------
-------------");
  Serial.print("AIN0: "); Serial.print(adc0); Serial.print("
Volt: "); Serial.print(volts0);
  Serial.print(" TDS1: "); Serial.print(tds1); Serial.print("
NTU "); Serial.println(tds1Terbilang);
  Serial.print("AIN1: "); Serial.print(adc1); Serial.print("
Volt: "); Serial.print(volts1);
  Serial.print(" TDS2: "); Serial.print(tds2); Serial.print("
NTU "); Serial.println(tds2Terbilang);
  Serial.print("phValue: "); Serial.print(phValue);
Serial.print(" Voltage: "); Serial.println(volts3, 3);
  Serial.print("SERVO: "); Serial.println(servoTerbilang);

  // Display turbidity sensor values on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("TDS1: "); lcd.print(tds1); lcd.print(" NTU");
  lcd.setCursor(0, 1);
  lcd.print("TDS2: "); lcd.print(tds2); lcd.print(" NTU");
  delay(1000);

  // Display motor servo values on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("PH: "); lcd.print(phValue); lcd.print(" pH");
  lcd.setCursor(0, 1);
  lcd.print("SERVO: "); lcd.print(servoTerbilang);

  // Mengirim data ke Blynk
  Blynk.virtualWrite(V0, tds1);
  Blynk.virtualWrite(V1, tds1Terbilang);
  Blynk.virtualWrite(V2, tds2);
  Blynk.virtualWrite(V3, tds2Terbilang);
  Blynk.virtualWrite(V4, servoTerbilang);

  delay(1000);
```

```
}

// Fungsi untuk memeriksa koneksi ke server Blynk
void checkConnection()
{
  if (!Blynk.connected()) {
    Serial.println("Reconnecting...");
    if (Blynk.connect()) {
      Serial.println("Reconnected!");
    } else {
      Serial.println("Reconnect failed!");
    }
  }
}

void tdsTerbilang(){
  if (tds1 < 24.70) {
    tds1Terbilang = "Jernih";
  } else if (24.70 < tds1 > 71.48) {
    tds1Terbilang = "Sedang";
  } else if (tds1 > 71.48) {
    tds1Terbilang = "Keruh";
    {
   Blynk.logEvent("notifikasi_keruh", "notifikasi ketika kondisi
keruh");
    }
  }

  if (tds2 < 24.70) {
    tds2Terbilang = "Jernih";
  } else if (24.70 < tds2 > 71.48) {
    tds2Terbilang = "Sedang";
  } else if (tds2 > 71.48) {
    tds2Terbilang = "Keruh";
  }
}

void servoControl(){
  if (tds1 < 24.70) {
```

```
    servoMotor.write(182);
    servoTerbilang = "OFF";
  } else if (tds1 >= 24.70) {
    servoMotor.write(180);
    servoTerbilang = "ON";
  } else if (tds1 >= 71.48) {
    servoMotor.write(0);
    servoTerbilang = "ON";
  }
// FULL SOURCECODE wa 088801441364
}
```