Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

PH meter(SKU: SEN0161)

Analog pH Meter Kit SKU: SEN0161

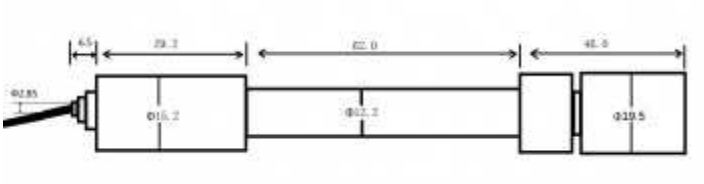
Analog pH Meter Kit SKU: SEN0169

Contents

Introduction

Need to measure water quality and other parameters but haven't got any low cost pH meter? Find it difficult to use with Arduino? Here comes an analog pH meter, specially designed for Arduino controllers and has built-in simple, convenient and practical connection and features. It has an LED which works as the Power Indicator, a BNC connector and PH2.0 sensor interface. You can just connect the pH sensor with BNC connector, and plug the PH2.0 interface into any analog input on Arduino controller to read pH value easily.

Specification



SEN0161 dimension

- Module Power: 5.00V
- Circuit Board Size: 43mm×32mm
- pH Measuring Range: 0-14
- Measuring Temperature: 0-60 ℃
- Accuracy: ± 0.1pH (25 ℃)
- Response Time: ≤ 1min
- pH Sensor with BNC Connector
- PH2.0 Interface (3 foot patch)
- Gain Adjustment Potentiometer
- Power Indicator LED

Precautions

- Before and after use of the pH electrode every time, you need to use (pure)water to clean it.
- The electrode plug should be kept clean and dry in case of short circuit.
- **Preservation**: Electrode reference preservation solution is the **3N KCL** solution.
- Measurement should be avoided staggered pollution between solutions, so as not to affect the accuracy of measurement.
- Electrode blub or sand core is defiled which will make PTS decline, slow response. So, it should be based on the characteristics of the pollutant, adapted to the cleaning solution, the electrode performance recovery.

- Electrode when in use, the ceramic sand core and liquid outlet rubber ring should be removed, in order to make salt bridge solution to maintain a certain velocity.

NOTE: **Differences between the probes, SEN0161 and SEN0169**

Their usages/ specifications are almost the same. The differences locates at

Long-firing Operation: SEN0169 supports, while SEN0161 NOT, i.e. you can not immerse SEN0161 in water for Continuous Testing.

Life Span: In 25 ℃, pure water, do Continuous Testing with them both, SEN0169 can work two years, while SEN0161 can only last for 6 months. And just for reference, if put them in turbid, strongly acid and alkali solution, 25℃, the life span would drop to one year (SEN0169), 1 month(or shorter, SEN0161).
Tempreture, pH, turbidity of the water effect the probe life span a lot.

Waterproof: You can immerse the whole probe SEN0169 into the water, while you can only immerse the front part of the probe SEN0161, the electrode glass bulb, into water, the rear part, from the white shell to the cable, MUST NOT be under water.

Strongly Acid and Alkali: SEN0169 are preferred for strongly acid and alkali test. And if your testing range is usually within pH6~8, then SEN0161 is capable for that.

pH Electrode Characteristics

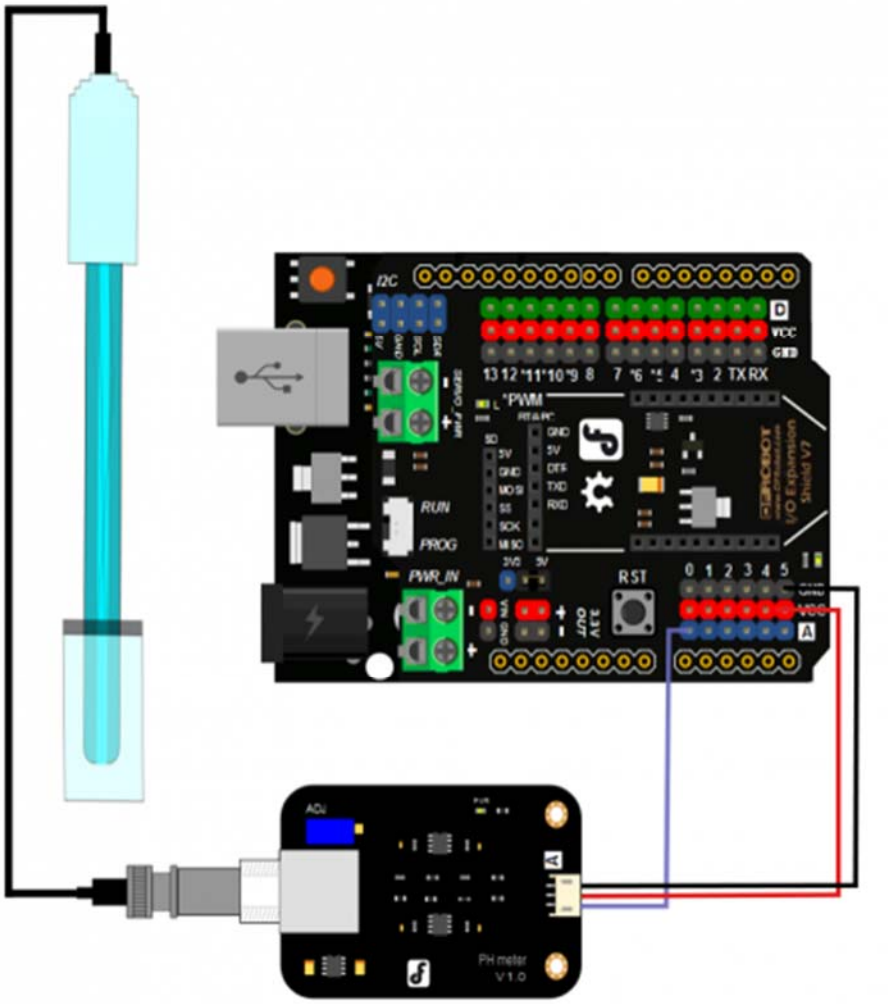
The output of pH electrode is Millivolts, and the pH value of the relationship is shown as follows (25 ℃):

VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

NOTE: It is normal that if your reading is much different with the table since you are not reading from the electrode directly but from the voltage adapter, it has converted the original voltage (-5V ~ +5V) to Arduino compatible voltage, i.e. 0 ~ 5V. See the discussion on Forum.

Usage

Connecting Diagram

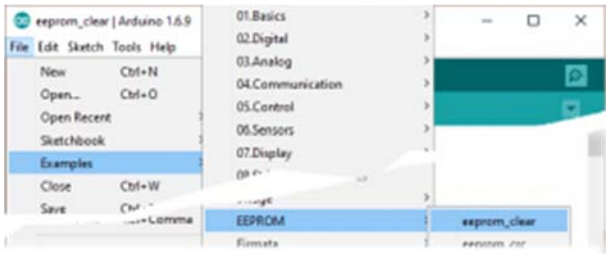


NOTE:

Before you insert the pH probe into one solution from another, or after you finish using the sensor, you must wash the pH electrode with pure water everytime (distilled water is the best)!
The closer **power supply** to +5.00V, the more accurate pH readings you could get.
You have to immerse the pH probe into stationary solution instead of the running one to get relative stable pH readings.
How long should it be under the solution? It depends on the pH value, the closer to neutral solution (pH = 7.00), the longer it will take. As we tested in water pH = 6.0, the blue one costs 6 minutes, and in standard Acid/ Alkali (4.00/ 10.00) solutions, it only needs 10 seconds.

Method 1. Software Calibration

The software calibration is easier than the next part - Hardware Calibration through the Potentiometer. Because it writes the calibration values into Arduino's EEPROM, so you can calibrate once for all if you won't replace your Arduino. It uses mathematical method that to draw a line using two points, i.e. using the Acid standard solution, pH = 4.00 and alkaline pH = 10.00 or 9.18 to draw the linear relation between the voltage and the pH value.



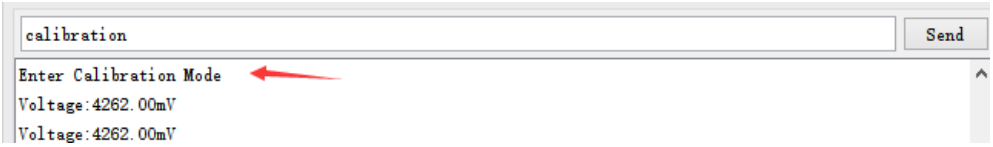
For NOTE 3. Arduino sample sketch "**EEPROM Clear**"

NOTE:

During the calibration (from step 4 to step 7), **power outage** should be avoided, or you will have to start over from step 4.
Software Calibration has nothing to do with the **potentiometer** on the adapter. Especially after you finished the calibration, you should never adjust the potentiometer, or you should start over. Moreover, considering the mechanical vibration might interfere the potentiometer value, you could seal it by Hot Melt Adhesive.
If you want to try Hardware Calibration, you'd better reset the EEPROM setting by uploading the Arduino IDE sample sketch "**EEPROM Clear**" as shown as the right hand picture.

Steps

1. Wiring the pH probe, pH meter adapter (the little PCB board) and Arduino UNO as the Diagram section above.
2. Upload the sample code "Software Calibration" below to UNO.
3. Open Serial Monitor, choose command format as “Both NL & CR”and 115200.
4. Send “**Calibration**” to enter Calibration Mode, and you will see “Enter Calibration Mode” directly.



5. Acid Calibration

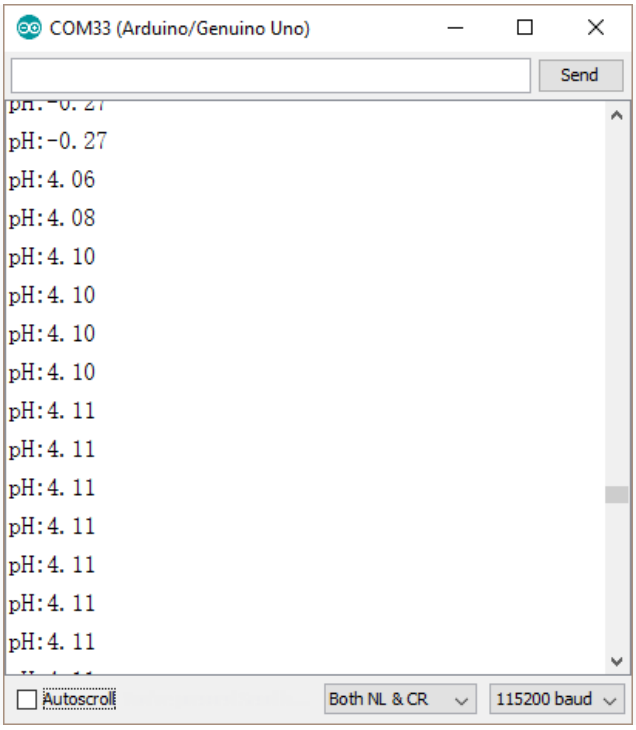
1. Wash your pH probe with pure water (distilled water is best) and dryer it in case of diluting the standard pH solution. Insert it into standard acid solution of pH = 4.0. Wait several seconds till the readings get relative stable.
2. Enter “**acid:4.00**”(no bland space, lower case), and you will get “Acid Calibration Successful” notice. Then go on with Alkali Calibration.

6. Alkali Calibration

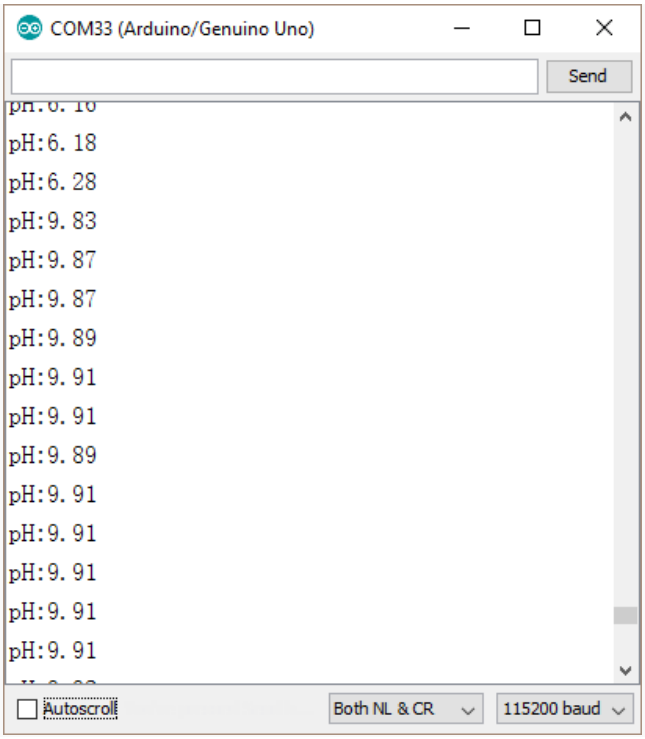
1. Take out the pH probe out of the acid solution, CLEAN it again as you did in last step. After this, insert it into the standard alkali solution with pH = 10 or 9.18. Waiting for the stable readings
2. Enter “**alkali:10.00**”, and you will see “Alkali Calibration Successful”.

7. Enter “exit” to finish calibration. You will see “Calibration Successful,Exit Calibration Mode”.

8. Check if the pH meter was calibrated successfully with the solution pH = 4.00, 9.18, 10.00, if the readings are within the error of 0.1. Congrats!

- 

In Standard acid solution pH = 4.00

- 

In Standard alkali solution pH = 10.00

Sample code: Software Calibration

```
/*************************************************

 This example uses software solution to calibration the ph meter,
not the potentiometer. So it is more easy to use and calibrate.

 This is for SEN0161 and SEN0169.


 Created 2016-8-11

 By youyou from DFrobot <youyou.yu@dfrobot.com>


 GNU Lesser General Public License.

 See <http://www.gnu.org/licenses/> for details.

 All above must be included in any redistribution

 *************************************************/


/***********Notice and Troubleshooting***************

 1.Connection and Diagram can be found here  http://www.dfrobot.co
m/wiki/index.php/PH_meter%28SKU:_SEN0161%29

 2.This code is tested on Arduino Uno.

 *************************************************/


#include <EEPROM.h>

#define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p
);for(; i < sizeof(p); i++) EEPROM.write(address+i, pp[i]);}

#define EEPROM_read(address, p)  {int i = 0; byte *pp = (byte*)&(p
);for(; i < sizeof(p); i++) pp[i]=EEPROM.read(address+i);}


#define ReceivedBufferLength 20

char receivedBuffer[ReceivedBufferLength+1];   // store the serial
command

byte receivedBufferIndex = 0;
```

```
#define SCOUNT  30           // sum of sample point

int analogBuffer[SCOUNT];    //store the sample voltage

int analogBufferIndex = 0;


#define SlopeValueAddress 0     // (slope of the ph probe)store at
the beginning of the EEPROM. The slope is a float number,occupies
4 bytes.

#define InterceptValueAddress (SlopeValueAddress+4)

float slopeValue, interceptValue, averageVoltage;

boolean enterCalibrationFlag = 0;


#define SensorPin A0

#define VREF 5000  //for arduino uno, the ADC reference is the pow
er(AVCC), that is 5000mV


void setup()

{

  Serial.begin(115200);

  readCharacteristicValues(); //read the slope and intercept of th
e ph probe

}


void loop()

{

  if(serialDataAvailable() > 0)

  {

      byte modeIndex = uartParse();

      phCalibration(modeIndex);    // If the correct calibration c
ommand is received, the calibration function should be called.

      EEPROM_read(SlopeValueAddress, slopeValue);     // After cal
ibration, the new slope and intercept should be read ,to update cu
rrent value.

      EEPROM_read(InterceptValueAddress, interceptValue);

  }
```

```
   static unsigned long sampleTimepoint = millis();
   if(millis()-sampleTimepoint>40U)
   {
     sampleTimepoint = millis();
     analogBuffer[analogBufferIndex] = analogRead(SensorPin)/1024.
0*VREF;    //read the voltage and store into the buffer,every 40ms
     analogBufferIndex++;
     if(analogBufferIndex == SCOUNT)
         analogBufferIndex = 0;
     averageVoltage = getMedianNum(analogBuffer,SCOUNT);   // read
the stable value by the median filtering algorithm
   }

   static unsigned long printTimepoint = millis();
   if(millis()-printTimepoint>1000U)
   {
     printTimepoint = millis();
     if(enterCalibrationFlag)             // in calibration mode,
print the voltage to user, to watch the stability of voltage
     {
       Serial.print("Voltage:");
       Serial.print(averageVoltage);
       Serial.println("mV");
     }else{
     Serial.print("pH:");              // in normal mode, print th
e ph value to user
     Serial.println(averageVoltage/1000.0*slopeValue+interceptValu
e);
     }
   }
}

boolean serialDataAvailable(void)
{
```

```
  char receivedChar;
  static unsigned long receivedTimeOut = millis();
  while (Serial.available()>0)
  {
    if (millis() - receivedTimeOut > 1000U)
    {
      receivedBufferIndex = 0;
      memset(receivedBuffer,0,(ReceivedBufferLength+1));
    }
    receivedTimeOut = millis();
    receivedChar = Serial.read();
    if (receivedChar == '\n' || receivedBufferIndex==ReceivedBuffe
rLength){
                receivedBufferIndex = 0;
                strupr(receivedBuffer);
                return true;
    }
    else{
      receivedBuffer[receivedBufferIndex] = receivedChar;
      receivedBufferIndex++;
    }
  }
  return false;
}


byte uartParse()
{
  byte modeIndex = 0;
  if(strstr(receivedBuffer, "CALIBRATION") != NULL)
      modeIndex = 1;
  else if(strstr(receivedBuffer, "EXIT") != NULL)
      modeIndex = 4;
  else if(strstr(receivedBuffer, "ACID:") != NULL)
```

```
      modeIndex = 2;
  else if(strstr(receivedBuffer, "ALKALI:") != NULL)
      modeIndex = 3;
  return modeIndex;
}


void phCalibration(byte mode)
{
    char *receivedBufferPtr;
    static byte acidCalibrationFinish = 0, alkaliCalibrationFinish
= 0;
    static float acidValue,alkaliValue;
    static float acidVoltage,alkaliVoltage;
    float acidValueTemp,alkaliValueTemp,newSlopeValue,newIntercept
Value;
    switch(mode)
    {
      case 0:
      if(enterCalibrationFlag)
         Serial.println(F("Command Error"));
      break;

      case 1:
      receivedBufferPtr=strstr(receivedBuffer, "CALIBRATION");
      enterCalibrationFlag = 1;
      acidCalibrationFinish = 0;
      alkaliCalibrationFinish = 0;
      Serial.println(F("Enter Calibration Mode"));
      break;

      case 2:
      if(enterCalibrationFlag)
      {
```

```
          receivedBufferPtr=strstr(receivedBuffer, "ACID:");
          receivedBufferPtr+=strlen("ACID:");
          acidValueTemp = strtod(receivedBufferPtr,NULL);
          if((acidValueTemp>3)&&(acidValueTemp<5))        //typica
l ph value of acid standand buffer solution should be 4.00
          {
             acidValue = acidValueTemp;
             acidVoltage = averageVoltage/1000.0;        // mV ->
V
             acidCalibrationFinish = 1;
             Serial.println(F("Acid Calibration Successful"));
           }else {
             acidCalibrationFinish = 0;
             Serial.println(F("Acid Value Error"));
           }
      }
      break;

       case 3:
       if(enterCalibrationFlag)
       {
           receivedBufferPtr=strstr(receivedBuffer, "ALKALI:");
           receivedBufferPtr+=strlen("ALKALI:");
           alkaliValueTemp = strtod(receivedBufferPtr,NULL);
           if((alkaliValueTemp>8)&&(alkaliValueTemp<11))        //
typical ph value of alkali standand buffer solution should be 9.18
or 10.01
           {
                 alkaliValue = alkaliValueTemp;
                 alkaliVoltage = averageVoltage/1000.0;
                 alkaliCalibrationFinish = 1;
                 Serial.println(F("Alkali Calibration Successful")
);
            }else{
```

```
                alkaliCalibrationFinish = 0;
                Serial.println(F("Alkali Value Error"));
            }
        }
        break;

         case 4:
         if(enterCalibrationFlag)
         {
             if(acidCalibrationFinish && alkaliCalibrationFinish)
             {
               newSlopeValue = (acidValue-alkaliValue)/(acidVoltage
- alkaliVoltage);
               EEPROM_write(SlopeValueAddress, newSlopeValue);
               newInterceptValue = acidValue - (slopeValue*acidVolt
age);
               EEPROM_write(InterceptValueAddress, newInterceptValu
e);
               Serial.print(F("Calibration Successful"));
             }
             else Serial.print(F("Calibration Failed"));
             Serial.println(F(",Exit Calibration Mode"));
             acidCalibrationFinish = 0;
             alkaliCalibrationFinish = 0;
             enterCalibrationFlag = 0;
         }
         break;
    }
}

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;
}


void readCharacteristicValues()
{
    EEPROM_read(SlopeValueAddress, slopeValue);
    EEPROM_read(InterceptValueAddress, interceptValue);
    if(EEPROM.read(SlopeValueAddress)==0xFF && EEPROM.read(SlopeVa
lueAddress+1)==0xFF && EEPROM.read(SlopeValueAddress+2)==0xFF && E
EPROM.read(SlopeValueAddress+3)==0xFF)
    {
      slopeValue = 3.5;   // If the EEPROM is new, the recommendat
ory slope is 3.5.
```

```
      EEPROM_write(SlopeValueAddress, slopeValue);
    }
    if(EEPROM.read(InterceptValueAddress)==0xFF && EEPROM.read(Int
erceptValueAddress+1)==0xFF && EEPROM.read(InterceptValueAddress+2
)==0xFF && EEPROM.read(InterceptValueAddress+3)==0xFF)
    {
      interceptValue = 0;  // If the EEPROM is new, the recommenda
tory intercept is 0.
      EEPROM_write(InterceptValueAddress, interceptValue);
    }
}
```

Method 2. Hardware Calibration through potentiometer

If you've taken the Method 1. Software Calibration, you can ignore this part.

1. Connect according to the graphic, that is, the pH electrode is connected to the BNC connector on the pH meter board, and then use the connection lines, the pH meter board is connected to the analog port 0 of the Arduino controller. When the Arduino controller gets power, you will see the blue LED on board is on.
2. Upload the sample code to the Arduino controller.
3. Put the pH electrode into the standard solution whose pH value is 7.00, or directly short circuit the input of the BNC connector. Open the serial monitor of the Arduino IDE, you can see the pH value printed to it, and the error does not exceed 0.3. Record the pH value printed, then compared with 7.00, and the difference should be changed into the "Offset" in the sample code. For example, the pH value printed is 6.88, so the difference is 0.12. You should change the **# define Offset 0.00** into **# define Offset 0.12** in the sample code.
4. **Fine adjustment**

- **For Acid solution:** Put the pH electrode into the pH standard solution whose value is 4.00. Then wait about a minute, adjust the Gain Potential device, let the value stabilise at around 4.00. At this time, the acidic calibration has been completed and you can measure the pH value of an acidic solution.
- **For Alkaline solution:** According to the linear characteristics of pH electrode itself, after the above calibration, you can

directly measure the pH value of the **alkaline** solution, but if you want to get a better accuracy, you can recalibrate it with the standard solution, pH = 9.18. Also adjust the gain potential device, let the value stabilise at around 9.18. After this calibration, you can measure the pH value of the alkaline solution.

Sample Code for Hardware Calibration

```
/*
 # This sample code is used to test the pH meter V1.0.
 # Editor : YouYou
 # Ver    : 1.0
 # Product: analog pH meter
 # SKU    : SEN0161
*/
#define SensorPin A0            //pH meter Analog output to Arduin
o Analog Input 0
#define Offset 0.00            //deviation compensate
#define LED 13
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40    //times of collection
int pHArray[ArrayLenth];   //Store the average value of the sensor
feedback
int pHArrayIndex=0;
void setup(void)
{
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
  Serial.println("pH meter experiment!");    //Test the serial mon
itor
}
```

```
void loop(void)
{
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue,voltage;
  if(millis()-samplingTime > samplingInterval)
  {
      pHArray[pHArrayIndex++]=analogRead(SensorPin);
      if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
      voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
      pHValue = 3.5*voltage+Offset;
      samplingTime=millis();
  }
  if(millis() - printTime > printInterval)   //Every 800 milliseco
nds, print a numerical, convert the state of the LED indicator
  {
        Serial.print("Voltage:");
        Serial.print(voltage,2);
        Serial.print("    pH value: ");
        Serial.println(pHValue,2);
        digitalWrite(LED,digitalRead(LED)^1);
        printTime=millis();
  }
}
double avergearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;
  if(number<=0){
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }
```

```
  if(number<5){   //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else{
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min;        //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max;    //arr>max
          max=arr[i];
        }else{
          amount+=arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
  return avg;
}
```

FAQ

Q1. My PH sensor readings are not correct, what did I miss?

Or the module is defective?

A. 1. Check if the pH sensor circuit board is good? Read on the Forum. or on wiki for the steps. During the transport, there might be crash causing the probe head cracked, please check if the probe is good or not.

2. If you don't use Arduino as the controller, then please check your ADC module that whether it converts the 5V analog input to 1024, if it is 4096(or other byte), please re-determine the equation in the code.

Q2. Big fluctuations in ph meter readings. When I make measurements in a glass, I have correct, stable reading. But when I put it inside the aquarium with the pumping system working, the easurement varies even more than a degree, and it's not stable, if I swicth off the pump the given value doesn´t oscilate anymore.

A. There should be NO working electrical device in the container. Any tiny leakage of electricity will cause the probe working error. Especially, many people bought the EC meter and put it into the same tank for the test, but then the pH meter cannot work well anymore. Please seperate them into different containers, or turning off the EC meter when using the pH meter.

Q3. May I know the Maximum range different if we do not calibrate the pH meter.

A. The maximum range differs from probe, you have to calibrate it before use if the pH probe was kept long.

Q4. I would just like to ask if your pH sensor can be connect to any micro controller aside from arduino. Would it be compatible with a raspberry pi? Thank You!

A. Yes, it can be used on any device as long as it could give 5V power supply and accept 5V analog signal, but as the Rasp pi is only compatible with 3.3V sensor, so an expansion shield is suggested to use with (please make sure which kind of Pi you use)

For any questions and more cool ideas to share, please visit **DFRobot Forum**