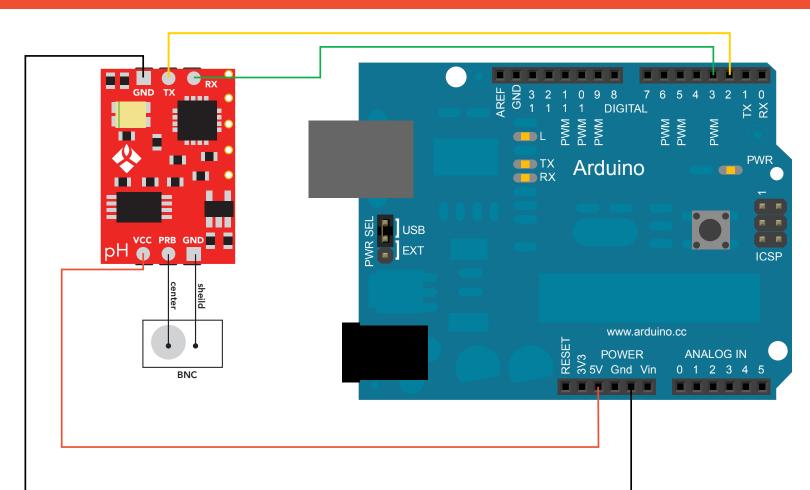


## **Arduino pH Sample Code**





//This code has intentionally has been written to be overly lengthy and includes unnecessary steps. //Many parts of this code can be truncated. This code was written to be easy to understand. //Code efficiency was not considered. Modify this code as you see fit. //This code will output data to the Arduino serial monitor. Type commands into the Arduino serial monitor to control the pH circuit. //set the var Arduino\_only to equal 1 to watch the Arduino take over control of the pH circuit.



#define rx 2 #define tx 3

#include <SoftwareSerial.h>

char ph\_data[20];

//we have to include the SoftwareSerial library, or else we can't use it. //define what pin rx is going to be.

//define what pin Tx is going to be.

SoftwareSerial myserial(rx, tx);

//define how the soft serial port is going to work.

char computerdata[20]; byte received\_from\_computer=0; byte received\_from\_sensor=0; byte arduino\_only=0;

//we need to know how many characters have been received. //we need to know how many characters have been received. //if you would like to operate the pH Circuit with the Arduino only and not use the

//we make a 20 byte character array to hold incoming data from a pc/mac/other.

//serial monitor to send it commands set this to 1. The data will still come out on the //serial monitor, so you can see it working. //used to make sure the Arduino takes over control of the pH Circuit properly.

//we make a 20 byte character array to hold incoming data from the pH.

float ph=0; byte string\_received=0;

byte startup=0;

//used to identify when we have received a string from the pH circuit.

//used to hold a floating point number that is the pH.

void setup(){ Serial.begin(38400); myserial.begin(38400);

//enable the hardware serial port

//enable the hardware serial port

```
void serialEvent(){
```

if(arduino\_only!=1){ received\_from\_computer=Serial.readBytesUntil(13,computerdata,20);

computerdata[received\_from\_computer]=0;

myserial.print('\r');

myserial.print(computerdata);

 $if(myserial.available() > 0){$ 

//the serial monitor(pc/mac/other) is received. //if Arduino\_only does not equal 1 this function will be //bypassed. //we read the data sent from the serial monitor //(pc/mac/other) until we see a <CR>. We also count //how many characters have been received. //we add a 0 to the spot in the array just after the last //character we received.. This will stop us from

//this interrupt will trigger when the data coming from

//transmitting incorrect data that may have been left //in the buffer. //we transmit the data received from the serial monitor //(pc/mac/other) through the soft serial port to the

//all data sent to the pH Circuit must end with a <CR>.

void loop(){

 $received\_from\_sensor=myserial.readBytesUntil(13,ph\_data,20);$ 

string\_received=1;

ph\_data[received\_from\_sensor]=0;

Serial.println(ph\_data);

if(arduino\_only==1){Arduino\_Control();}

}

//if we see that the pH Circuit has sent a character. //we read the data sent from pH Circuit until we see //a <CR>. We also count how many character have //been received. //we add a 0 to the spot in the array just after the last //character we received. This will stop us from

//transmitting incorrect data that may have been left //in the buffer. //a flag used when the Arduino is controlling the //pH Circuit to let us know that a complete string

//has been received. //lets transmit that data received from the pH Circuit //to the serial monitor.

//If the var arduino\_only is set to one we will call this //function. Letting the Arduino take over control of the //pH Circuit

void Arduino\_Control(){ if(startup==0){ myserial.print("e\r");

delay(50); myserial.print("e\r"); delay(50); startup=1; }

delay(800);

}

//on start up sometimes the first command is missed. //so, let's send it twice.

//if the Arduino just booted up, we need to set some things up first.

//a short delay after the pH Circuit was taken out of continues mode is used to make sure we don't //over load it with commands. //startup is completed, let's not do this again during normal operation.

//take the pH Circuit out of continues mode.

//send it the command to take a single reading.

 $myserial.print("R\r");\\$ if(string\_received==1){ ph=atof(ph\_data);  $if(ph>=7.5){Serial.println("high\r");}$ if(ph<7.5){Serial.println("low\r");} string\_received=0;}

//did we get data back from the ph Circuit? //many people ask us "how do I convert a sting into a float?" This is how... //This is the proof that it has been converted into a float. //This is the proof that it has been converted into a float. //reset the string received flag.

//we will take a reading ever 800ms. You can make this much longer or shorter if you like.

//calibrate to a pH of 7

//here are some functions you might find useful

void s\_cal(){ myserial.print("s\r");}

void f\_cal(){

//these functions are not enabled

//calibrate to a pH of 4 //send the "f" command to calibrate to a pH of 4.00

//send the "s" command to calibrate to a pH of 7.00

void t\_cal(){ myserial.print("t\r");}

//calibrate to a pH of 10 //send the "t" command to calibrate to a pH of 10.00

void phFactoryDefault(){

myserial.print("f\r");}

//factory defaults the pH circuit

myserial.print("X\r");}

//send the "X" command to factory reset the device

//get device info

myserial.print("I\r");} void phSetLEDs(byte enabled)

void read\_info(){

if(enabled) myserial.print("L1\r"); myserial.print("L0\r"); //send the "I" command to query the information //turn the LEDs on or off

//if enabled is > 0 //the LED's will turn ON //if enabled is 0 //the LED's will turn OFF