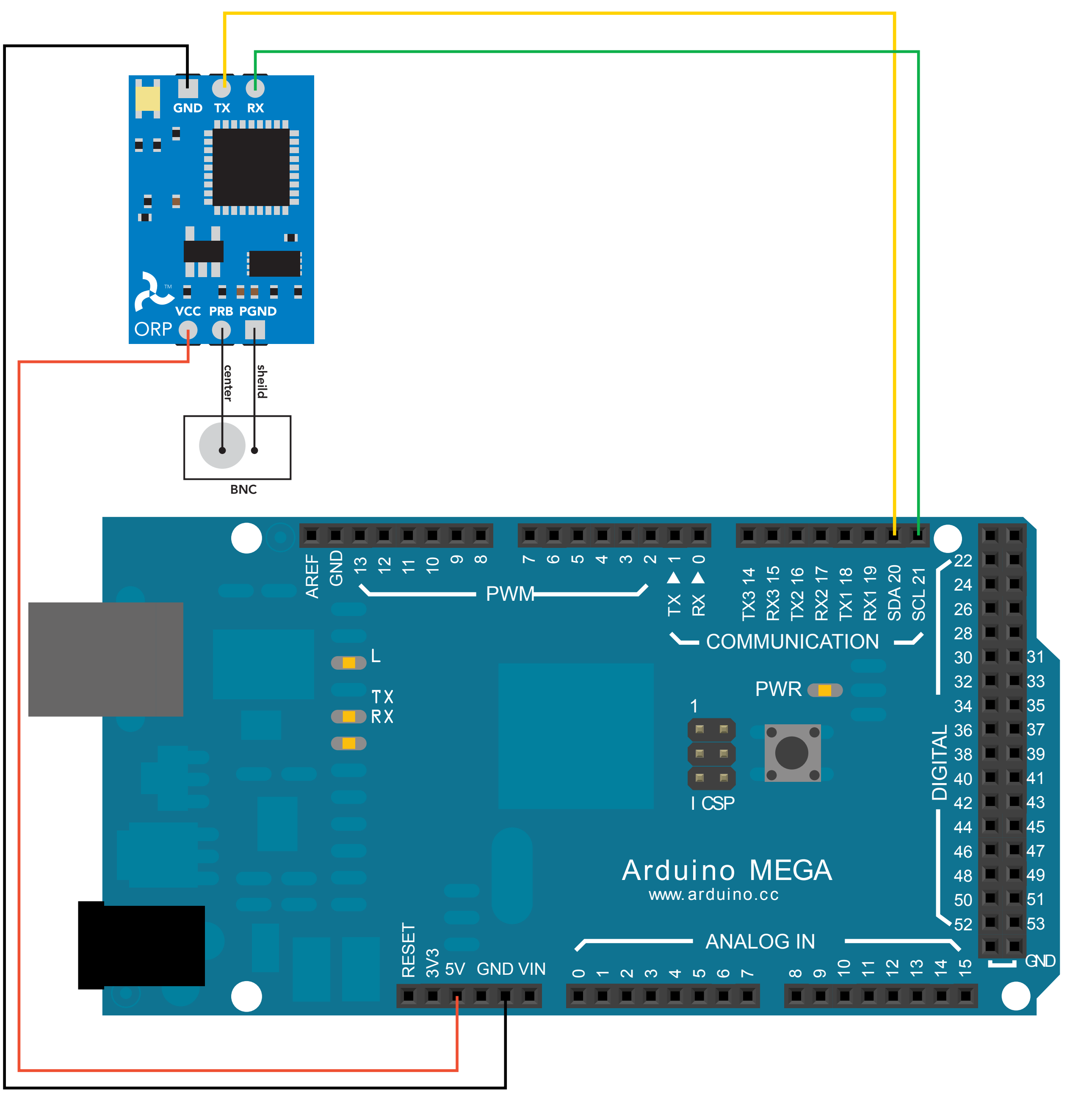
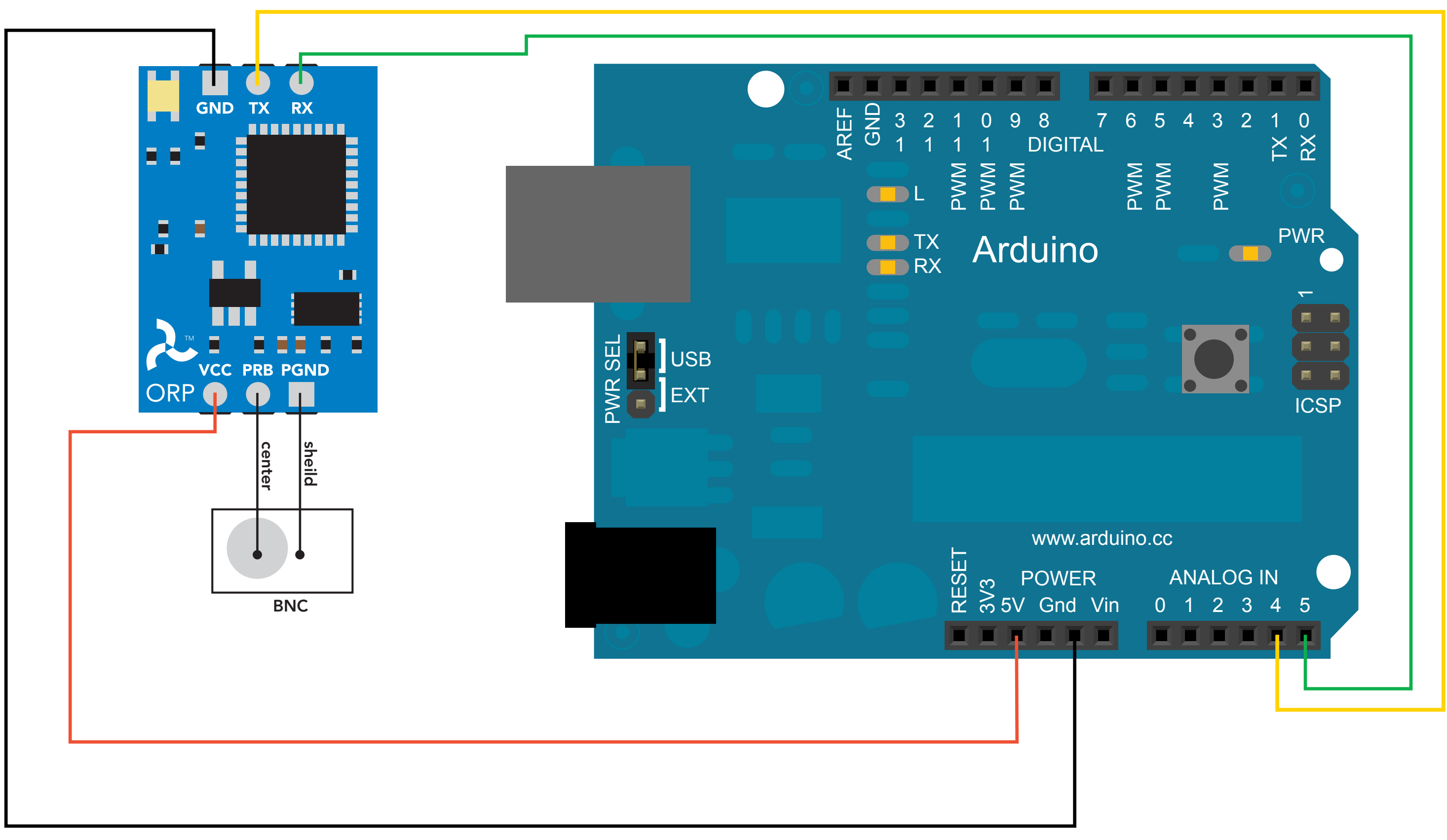
Atlas Scientific
Environmental Robotics

ORP I²C Sample Code

ORP I²C





```
//**THIS CODE WILL WORK ON ANY ARDUINO**
//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 EZO ORP Circuit in I²C mode.


#include <Wire.h>                   //enable I²C.
#define address 98                  //default I²C ID number for EZO ORP Circuit.


char computerdata[20];              //we make a 20 byte character array to hold incoming data from a pc/mac/other.
byte received_from_computer=0;      //we need to know how many characters have been received.
byte serial_event=0;                //a flag to signal when data has been recived from the pc/mac/other.
byte code=0;                        //used to hold the I²C response code.
char ORP_data[20];                  //we make a 48 byte character array to hold incoming data from the ORP circuit.
byte in_char=0;                     //used as a 1 byte buffer to store in bound bytes from the ORP Circuit.
byte i=0;                           //counter used for ORP_data array.
int time=1400;                      //used to change the delay needed depending on the command sent to the EZO Class ORP Circuit.
float ORP_float;                    //float var used to hold the float value of the ORP.


void setup()                        //hardware initialization.
{
 Serial.begin(9600);                //enable serial port.
 Wire.begin();                      //enable I²C port.
}


void serialEvent(){                                                       //this interrupt will trigger when the data coming from
      received_from_computer=Serial.readBytesUntil(13,computerdata,20);   //the serial monitor(pc/mac/other) is received.
      computerdata[received_from_computer]=0;                             //we read the data sent from the serial monitor
      serial_event=1;                                                     //(pc/mac/other) until we see a <CR>. We also count
     }                                                                    //how many characters have been received.
                                                                          //stop the buffer from transmitting leftovers or garbage.


void loop(){                                                    //the main loop.

 if(serial_event){                                              //if the serial_event=1.
    if(computerdata[0]=='c'||computerdata[0]=='r')time=1400;    //if a command has been sent to calibrate or take a reading we
   else time=300;                                               //wait 1400ms so that the circuit has time to take the reading.
                                                                //if any other command has been sent we wait only 300ms.


  Wire.beginTransmission(address);      //call the circuit by its ID number.
  Wire.write(computerdata);             //transmit the command that was sent through the serial port.
  Wire.endTransmission();               //end the I²C data transmission.


  delay(time);                          //wait the correct amount of time for the circuit to complete its instruction.

  Wire.requestFrom(address,20,1);       //call the circuit and request 20 bytes (this is more then we need).
  code=Wire.read();                     //the first byte is the response code, we read this separately.


  switch (code){                        //switch case based on what the response code is.
   case 1:                              //decimal 1.
    Serial.println("Success");          //means the command was successful.
   break;                               //exits the switch case.

   case 2:                              //decimal 2.
    Serial.println("Failed");           //means the command has failed.
   break;                               //exits the switch case.

   case 254:                            //decimal 254
    Serial.println("Pending");          //means the command has not yet been finished calculating.
   break;                               //exits the switch case.

   case 255:                            //decimal 255.
    Serial.println("No Data");          //means there is no further data to send.
   break;                               //exits the switch case.
  }


 while(Wire.available()){               //are there bytes to receive.
  in_char = Wire.read();                //receive a byte.
  ORP_data[i]= in_char;                 //load this byte into our array.
  i+=1;                                 //incur the counter for the array element.
   if(in_char==0){                      //if we see that we have been sent a null command.
      Wire.endTransmission();           //reset the counter i to 0.
      break;                            //end the I²C data transmission.
      }                                 //exit the while loop.
  }


 Serial.println(ORP_data);              //print the data.
 serial_event=0;                        //reset the serial event flag.
  }
}


//Uncomment this section if you want to take the ORP value and convert it into floating point number.
   /*
   ORP_float=atof(ORP_data);
   */
  }
```

Click here to download the *.ino file

Atlas-Scientific.com