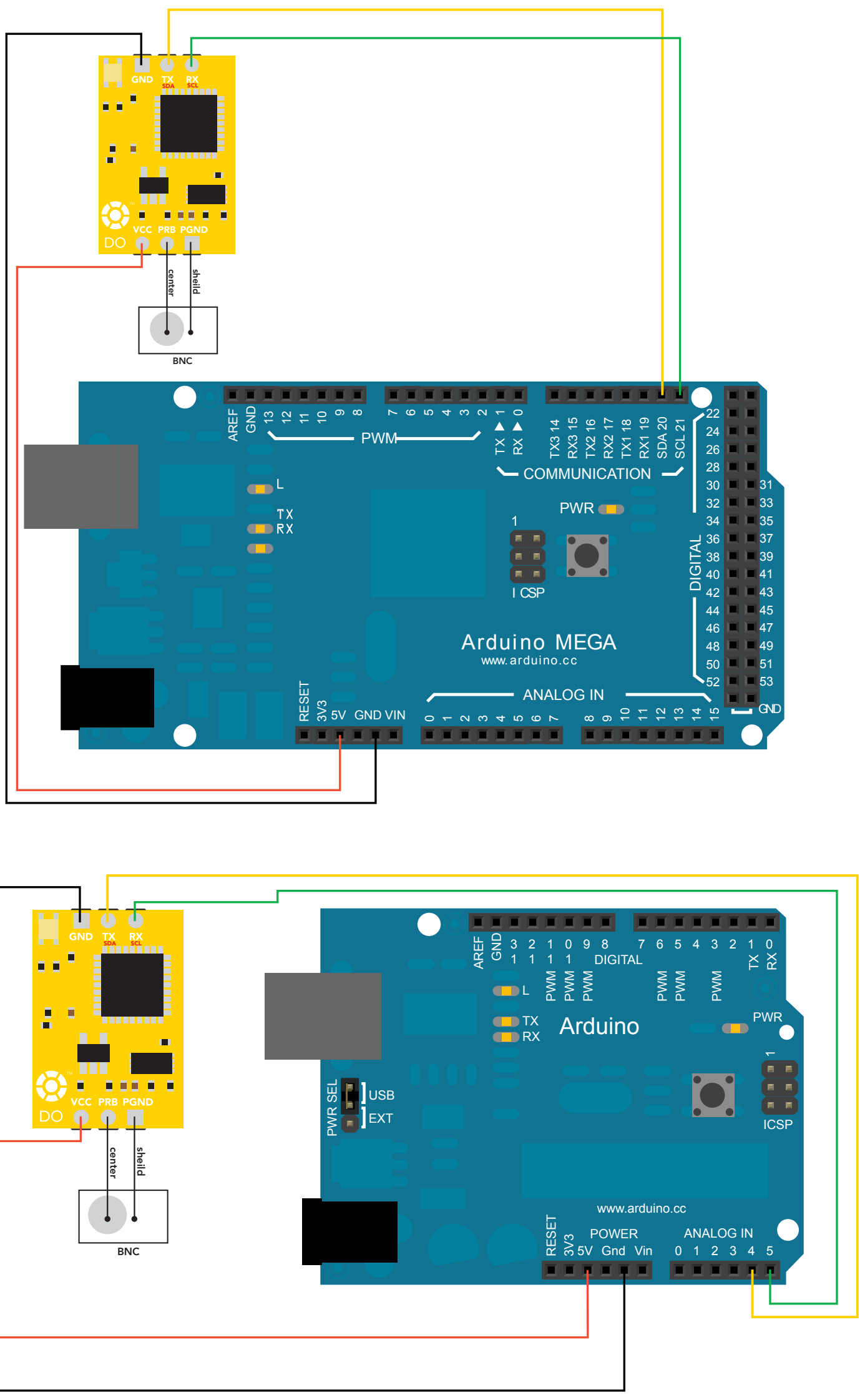
Atlas Scientific™
Environmental Robotics

Dissolved Oxygen I²C Sample Code

Dissolved Oxygen 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 D.O. Circuit in I2C mode.

```
#include <Wire.h>                       //enable I²C.
#define address 97                      //default I²C ID number for EZO D.O. 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 DO_data[20];                       //we make a 20 byte character array to hold incoming data from the D.O. circuit.
byte in_char=0;                         //used as a 1 byte buffer to store in bound bytes from the D.O. Circuit.
byte i=0;                               //counter used for DO_data array.
int time=1400;                          //used to change the delay needed depending on the command sent to the EZO Class D.O. Circuit.
float DO_float;                         //float var used to hold the float value of the DO.
char *DO;                               //char pointer used in string parsing.
char *sat;                              //char pointer used in string parsing.
float do_float;                         //float var used to hold the float value of the dissolved oxygen.
float sat_float;                        //float var used to hold the float value of the saturation percentage.


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.
                                                                    //we read the data sent from the serial monitor
        computerdata[received_from_computer]=0;                     //(pc/mac/other) until we see a <CR>. We also count
        serial_event=1;                                             //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
     else time=300;                                                 //we 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 may be 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.
  DO_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.
      i=0;                              //reset the counter i to 0.
      Wire.endTransmission();           //end the I2C data transmission.
      break;                            //exit the while loop.
      }
  }


 Serial.println(DO_data);               //print the data.
 serial_event=0;                        //reset the serial event flag.
 if(computerdata[0]=='r') string_pars();//Uncomment this function if you would like to break up the comma separated string.
  }
}


 void string_pars(){                    //this function will break up the CSV string into its 2 individual parts, DO and %sat.
                                        //this is done using the C command “strtok”.


 sat=strtok(DO_data, ",");              //let's pars the string at each comma.
 DO=strtok(NULL, ",");                  //let's pars the string at each comma.


 Serial.print("DO:");                   //We now print each value we parsed separately.
 Serial.println(DO);                    //this is the D.O. value.


 Serial.print("Sat:");                  //We now print each value we parsed separately.
 Serial.println(sat);                   //this is the % saturation value.


//Uncomment this section if you want to take the values and convert them into floating point number.
   /*
   DO_float=atof(DO);
   sat_float=atof(sat);
   */
 }
```

Click here to download the *.ino file