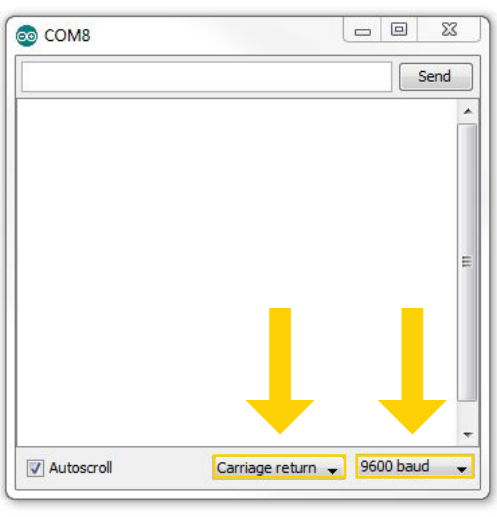
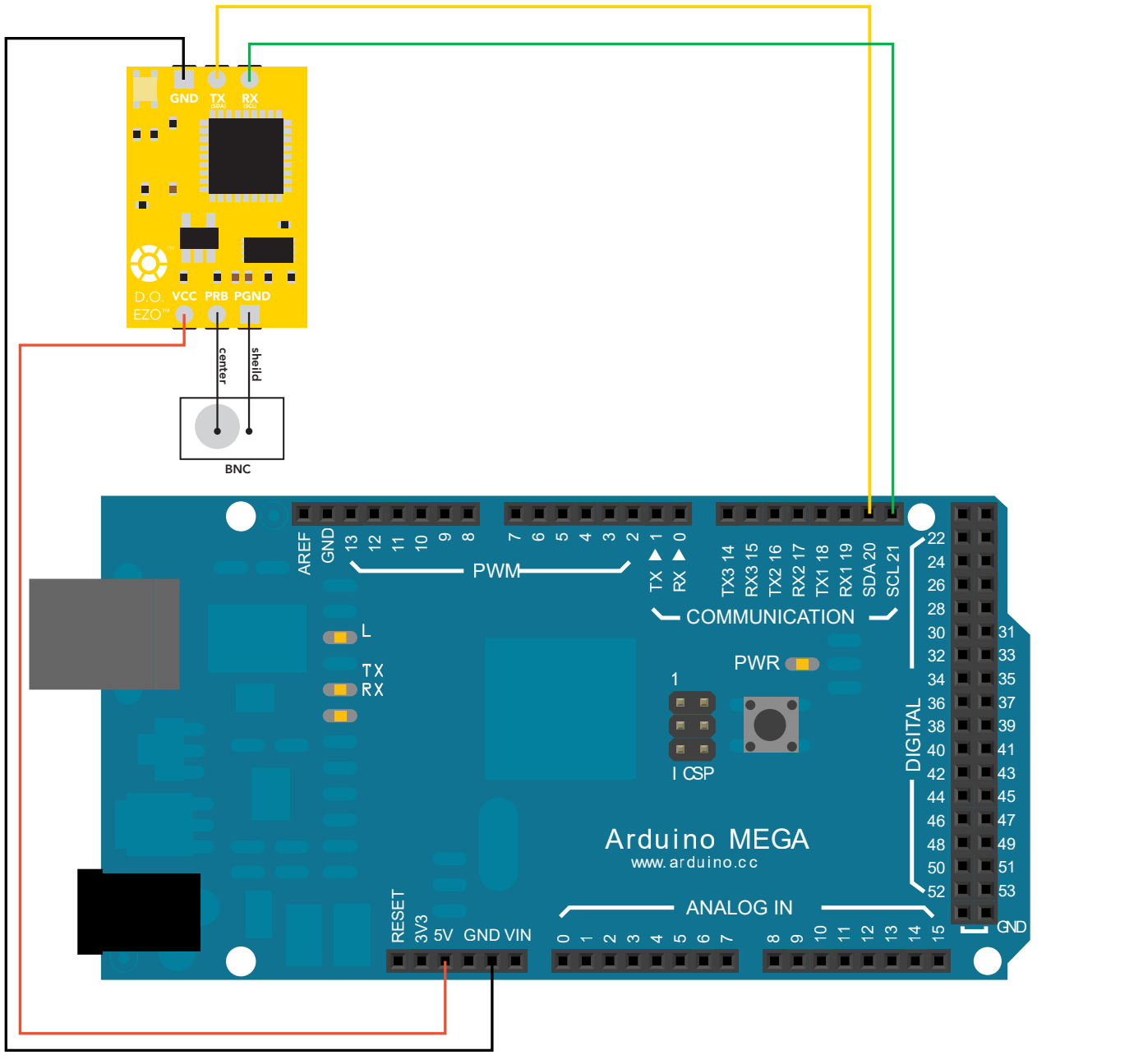
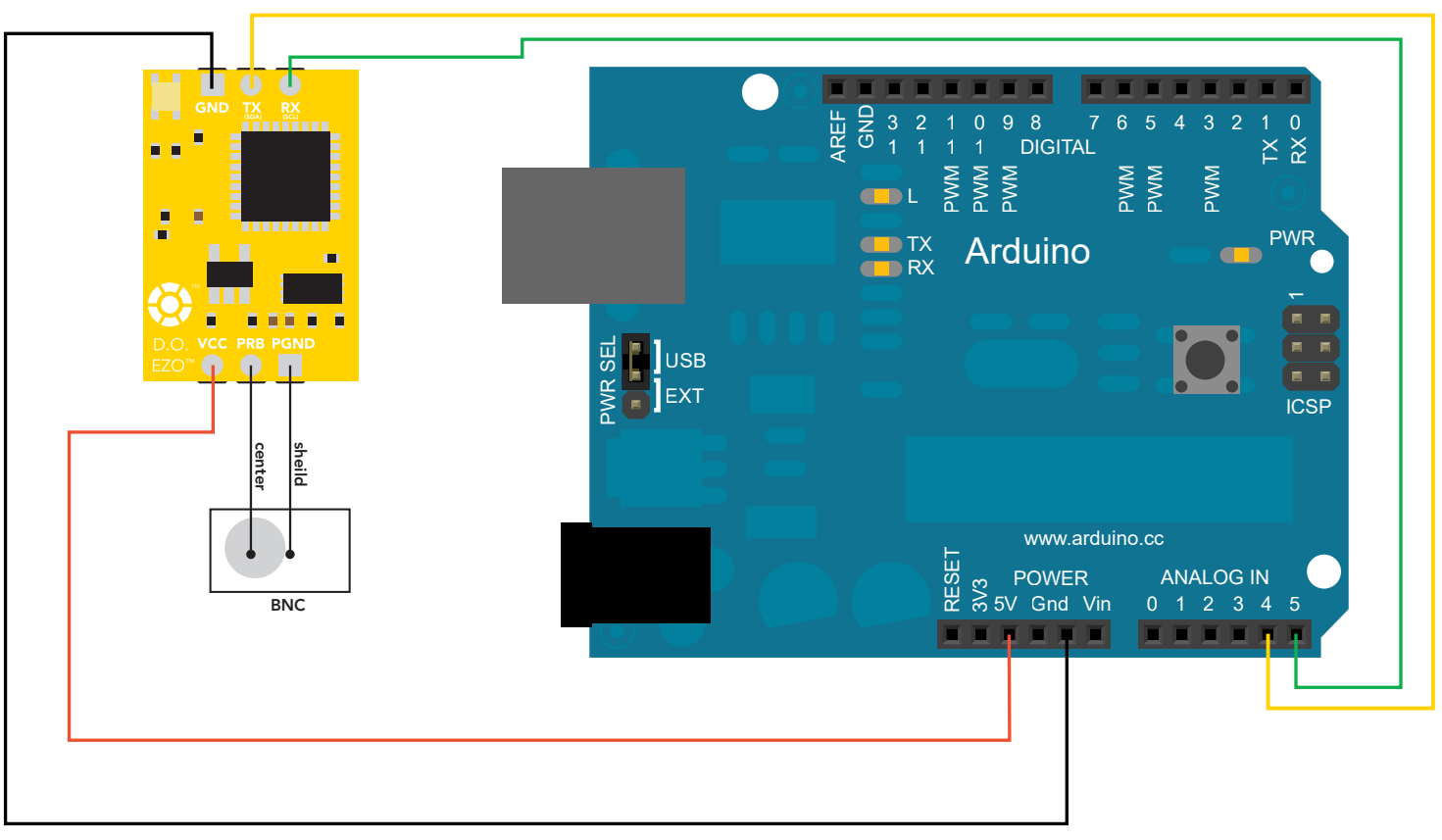
Atlas Scientific™
Environmental Robotics

I²C
Dissolved Oxygen Sample Code

Revised 7/26/16








```
//**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 DO Circuit in I2C mode.
//this code was last updated 7-26-2016


#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 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.
int time_ = 1800;                 //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 loop() {                     //the main loop
  byte i = 0;                     //counter used for DO_data array.

  if (Serial.available() > 0) {
    received_from_computer = Serial.readBytesUntil(13, computerdata, 20); //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.
    computerdata[received_from_computer] = 0;                             //stop the buffer from transmitting leftovers or garbage.
    computerdata[0] = tolower(computerdata[0]);                           //we make sure the first char in the string is lower case.
    if (computerdata[0] == 'c' || computerdata[0] == 'r')time_ = 1800;    //if a command has been sent to calibrate or take a reading
                                                                          //we wait 1800ms so that the circuit has time to take
                                                                          //the reading.
    else time_ = 300;                                                     //if not 300ms will do


    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 than 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.
      }
    }


    if (isDigit(DO_data[0])) {
      string_pars();                  //If the first char is a number we know it is a DO reading, lets parse the DO reading
    }
    else {                            //if it's not a number
      Serial.println(DO_data);        //print the data.
      for (i = 0; i < 20; i++) {      //step through each char
        DO_data[i] = 0;               //set each one to 0 this clears the memory
      }
    }
  }
}

void string_pars() {                  //this function will break up the CSV string into its 2 individual parts, DO and %sat.
  byte flag = 0;                      //this is used to indicate is a "," was found in the string array
  byte i = 0;                         //counter used for DO_data array.

  for (i = 0; i < 20; i++) {          //Step through each char
    if (DO_data[i] == ',') {          //do we see a ','
      flag = 1;                       //if so we set the var flag to 1 by doing this we can identify if the string being sent
                                      //from the DO circuit is a CSV string containing tow values
    }
  }

  if (flag != 1) {                    //if we see the there WAS NOT a ',' in the string array
    Serial.print("DO:");              //print the identifier
    Serial.println(DO_data);          //print the reading
  }

  if (flag == 1) {                    //if we see the there was a ',' in the string array
    DO = strtok(DO_data, ",");        //let's pars the string at each comma
    sat = strtok(NULL, ",");          //let's pars the string at each comma
    Serial.print("DO:");              //print the identifier
    Serial.println(DO);               //print the reading
    Serial.print("Sat:");             //print the identifier
    Serial.println(sat);              //print the reading
    flag = 0;                         //reset the flag
  }

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

Click here to download the *.ino file