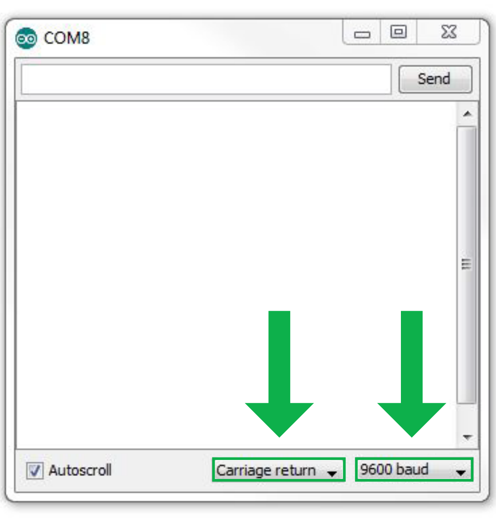
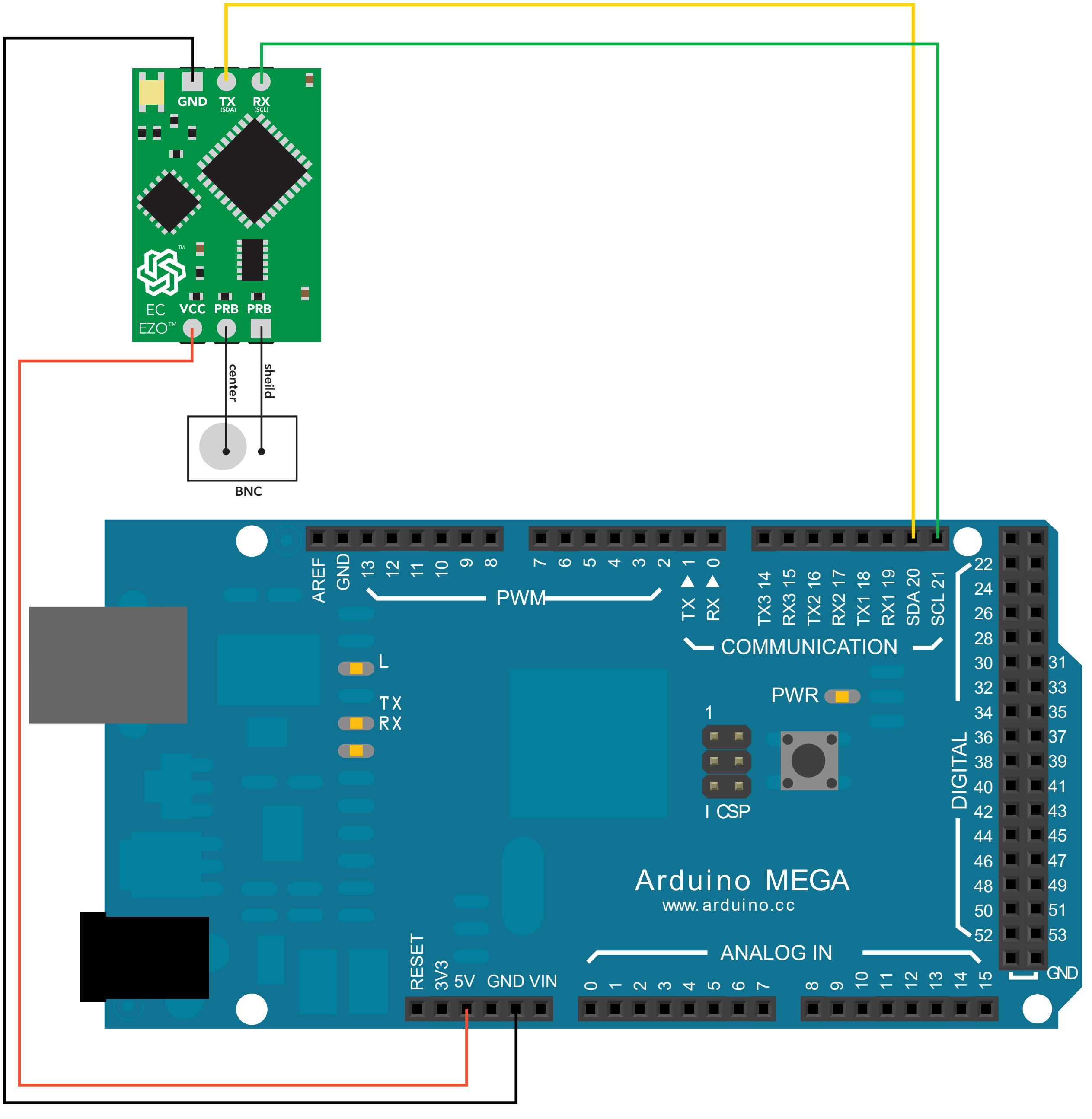
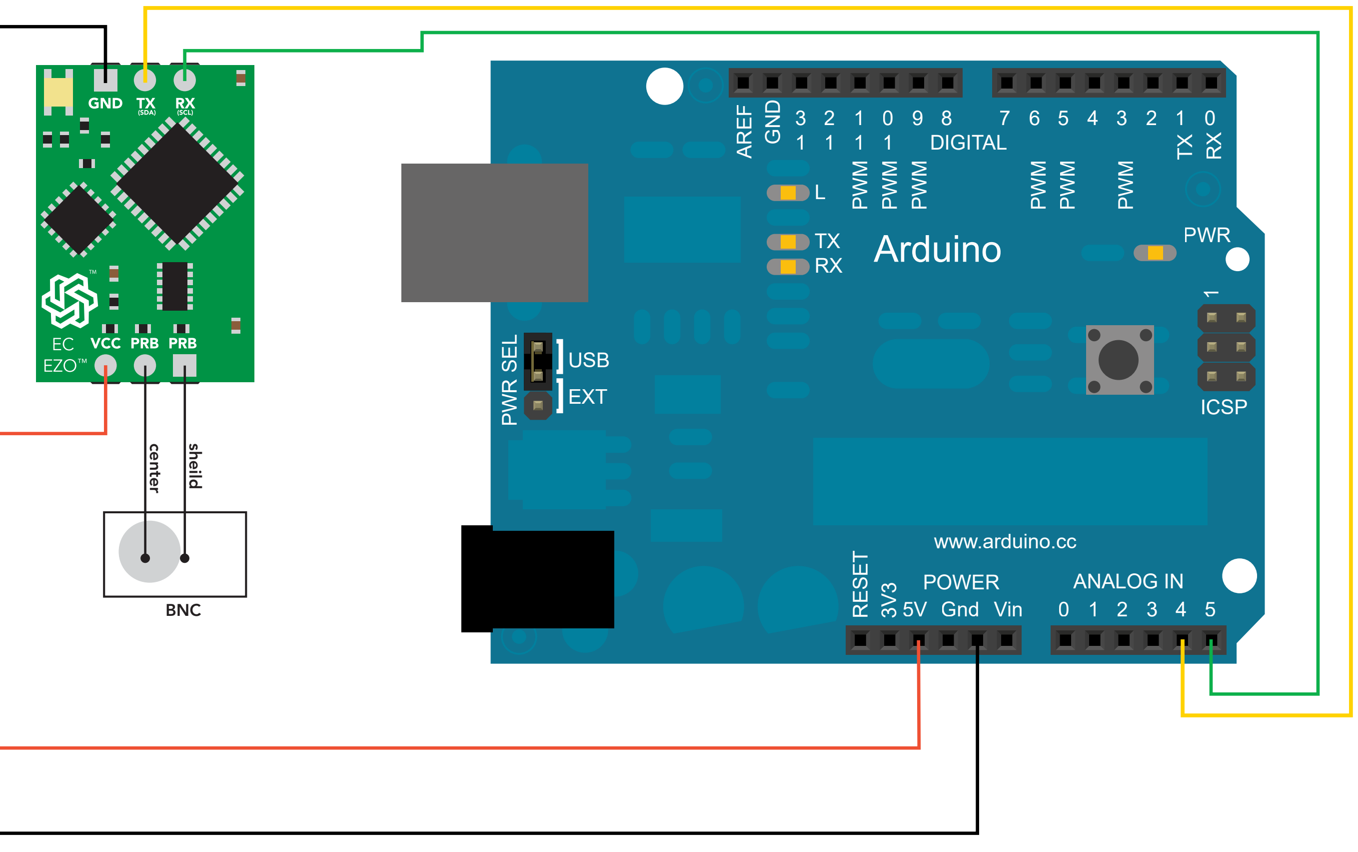
AtlasScientific™
Environmental Robotics

I²C
Conductivity Sample Code








```
//**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 EC Circuit in I2C mode.

#include <Wire.h>                    //enable I²C.
#define address 100                  //default I²C ID number for EZO EC 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.
bool serial_event = false;           //a flag to signal when data has been received from the pc/mac/other.
byte code = 0;                       //used to hold the I2C response code.
char ec_data[48];                    //we make a 48 byte character array to hold incoming data from the EC circuit.
byte in_char = 0;                    //used as a 1 byte buffer to store in bound bytes from the EC Circuit.
byte i = 0;                          //counter used for ec_data array.
int delay_time = 1400;               //used to change the delay needed depending on the command sent to the EZO Class EC Circuit.

char *ec;                            //char pointer used in string parsing.
char *tds;                           //char pointer used in string parsing.
char *sal;                           //char pointer used in string parsing.
char *sg;                            //char pointer used in string parsing.

float ec_float;                      //float var used to hold the float value of the conductivity.
float tds_float;                     //float var used to hold the float value of the TDS.
float sal_float;                     //float var used to hold the float value of the salinity.
float sg_float;                      //float var used to hold the float value of the specific gravity.


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 = true;                                                 //(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.
                                                                       //set the serial event flag.


void loop() {                                            //the main loop.
  if (serial_event == true) {                            //if a command was sent to the EC circuit.

    for (i = 0; i < received_from_computer; i++) {       //we need to check each character in the array.
      computerdata[i] = tolower(computerdata[i]);        //if a character in the array is uppercase we change it to lowercase.
    }

    i = 0;                                               //reset the counter i to 0.

    if (computerdata[0] == 'c' || computerdata[0] == 'r')delay_time = 1400;  //if a command has been sent to calibrate or take a
    else delay_time = 300;                                                   //reading we wait 1400ms so that the circuit has enough
                                                                             //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.


    if (strcmp(computerdata, "sleep") != 0) {            //if the command that has been sent is NOT the sleep command,
                                                         //wait the correct amount of time and request data.
                                                         //if it is the sleep command, we do nothing. Issuing a sleep command
                                                         //and then requesting data will wake the EC circuit.

      delay(delay_time);                                 //wait the correct amount of time for the circuit to complete its instruction.
      Wire.requestFrom(address, 48, 1);                  //call the circuit and request 48 bytes (this is more than we need)
      code = Wire.read();                                //the first byte is the response code, we read this separately.


    while (Wire.available()) {                           //are there bytes to receive.
      in_char = Wire.read();                             //receive a byte.
      ec_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.
      }
    }


    while (Wire.available()) {                           //are there bytes to receive.
      in_char = Wire.read();                             //receive a byte.
      ec_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.
      }
    }


    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.
    }


    Serial.println(ec_data);                             //print the data.
   }
   serial_event = false;                                 //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 into its individual parts.
  }
}


void string_pars() {                                     //this function will break up the CSV string into its 4 individual parts.
                                                         //EC|TDS|SAL|SG.
                                                         //this is done using the C command "strtok".

  ec = strtok(ec_data, ",");                             //let's pars the string at each comma.
  tds = strtok(NULL, ",");                               //let's pars the string at each comma.
  sal = strtok(NULL, ",");                               //let's pars the string at each comma.
  sg = strtok(NULL, ",");                                //let's pars the string at each comma.

  Serial.print("EC:");                                   //we now print each value we parsed separately.

  Serial.println(ec);                                    //this is the EC value.

  Serial.print("TDS:");                                  //we now print each value we parsed separately.
  Serial.println(tds);                                   //this is the TDS value.

  Serial.print("SAL:");                                  //we now print each value we parsed separately.
  Serial.println(sal);                                   //this is the salinity value.

  Serial.print("SG:");                                   //we now print each value we parsed separately.
  Serial.println(sg);                                    //this is the specific gravity.


  //uncomment this section if you want to take the values and convert them into floating point number.
  /*
  ec_float=atof(ec);
  tds_float=atof(tds);
  sal_float=atof(sal);
  sg_float=atof(sg);
  */
}
```

Click here to download the *.ino file

Atlas-Scientific.com