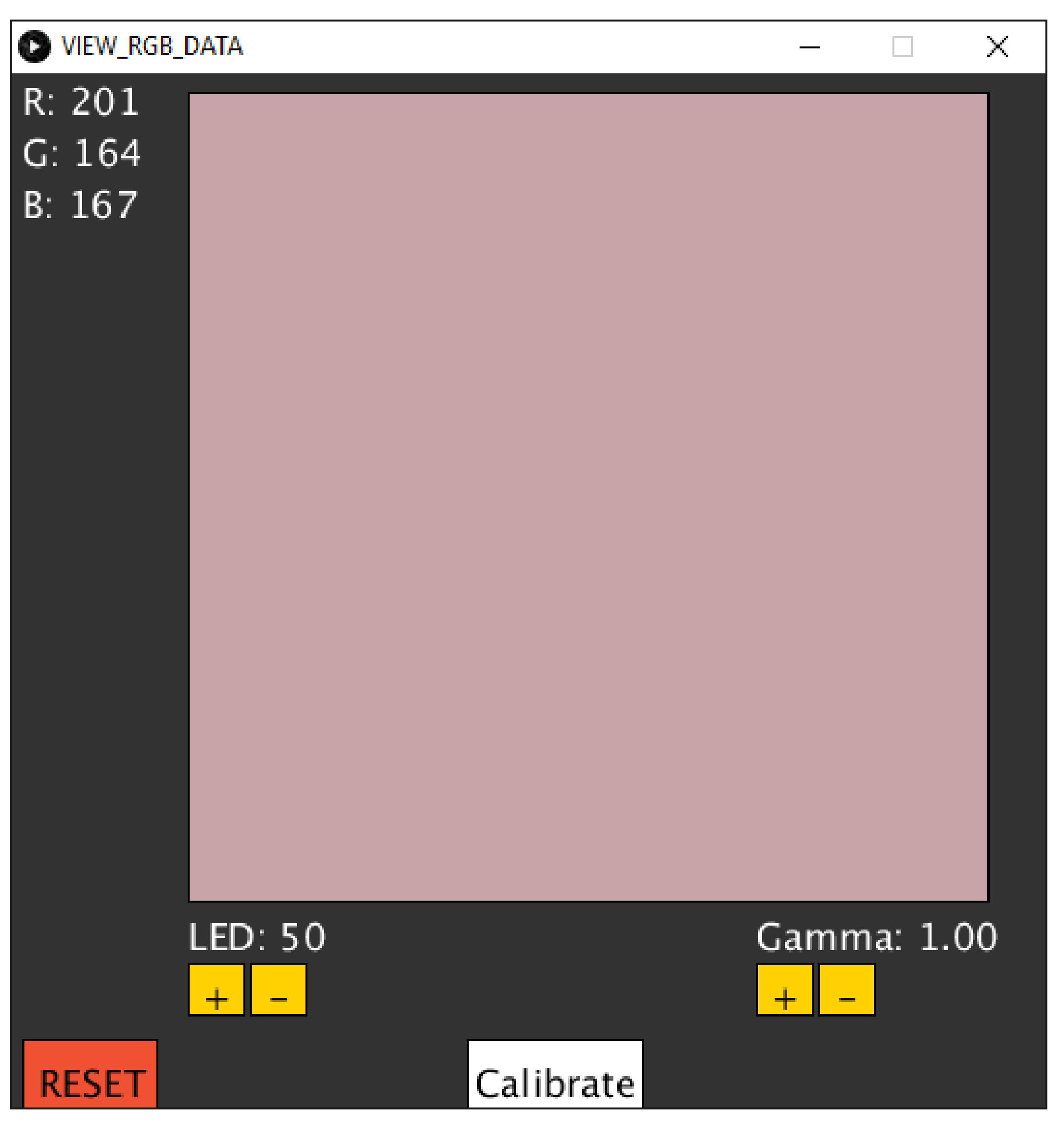
Atlas Scientific
Environmental Robotics

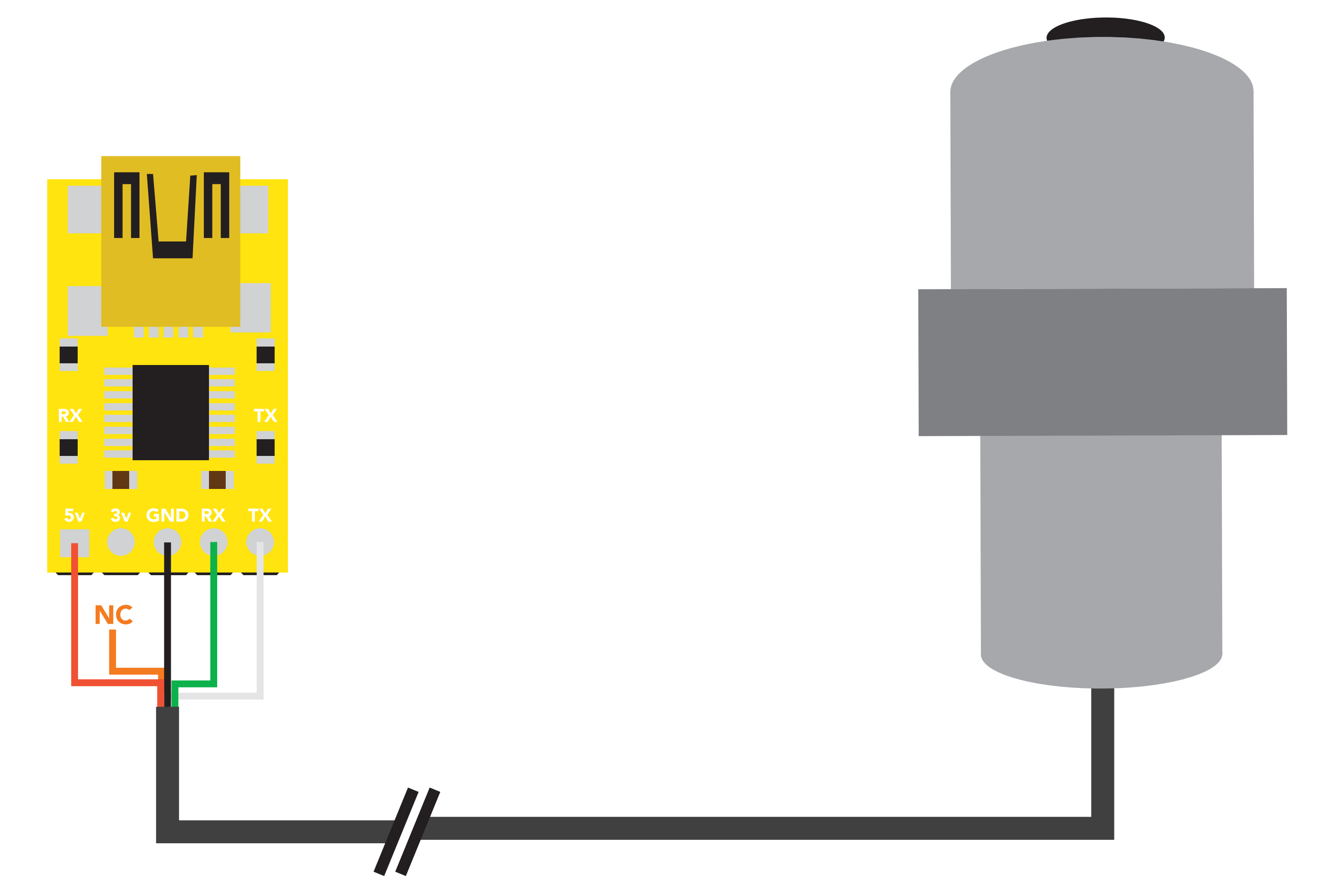
EZO-RGB™

How to view RGB output on your computer

In order to get this code working correctly, you will need to install Processing.

Click to download **Processing**






```
import processing.serial.*;                    //import the Processing  serial data library

Serial myPort;                                 //name the serial port
String inString;                               //input string from serial port
char cr = '\r';                                //ASCII carriage return
boolean data_received = false;                 //data received flag
char c1=0;                                     //used to look at the first char of the incoming string
int r=0;                                       //holds the red color data
int g=0;                                       //holds the green color data
int b=0;                                       //holds the blue color data
int led_brightness=50;                         //used to display the LED brightness
float gamma=1.00;                              //used to display the gamma correction
boolean device_reset=false;                    //device reset flag

BUTTON led_up_btn;                             //declaring the LED + button object
BUTTON led_dn_btn;                             //declaring the LED - button object
BUTTON gamma_up_btn;                           //declaring the gamma + button object
BUTTON gamma_dn_btn;                           //declaring the gamma - button object
BUTTON reset_btn;                              //declaring reset button object
BUTTON cal_btn;                                //declaring the calibration button object

void setup() {                                 //program setup
  size(500, 500);                              //set window size
  textSize(18);                                //set text size
  printArray(Serial.list());                   //list possible serial ports
  myPort = new Serial(this, Serial.list()[0], 9600);   //by default we use the first port in the list. YOU MAY NEED TO CHANGE THIS
  myPort.bufferUntil(cr);                      //read the serial stream until a carriage return is seen

  led_up_btn = new BUTTON( color(255, 210, 0), "+", 85, 430, 27, 25);          //we create 6 new button objects
  led_dn_btn = new BUTTON( color(255, 210, 0), "-", 115, 430, 27, 25);         //the first two are for LED brightness.
  gamma_up_btn = new BUTTON( color(255, 210, 0), "+", 360, 430, 27, 25);       //Then two for gamma correction, a calibrate button
  gamma_dn_btn = new BUTTON( color(255, 210, 0), "-", 390, 430, 27, 25);       //and a reset button
  reset_btn= new BUTTON( color(240, 81, 51), "RESET", 5, 467, 65, 35);         //the first 3 numbers "color(255, 210, 0)" sets the color
  cal_btn =new BUTTON( color(255, 255, 255), "Calibrate", 220, 467, 85, 35);   //of the button in RGB.
                                                                               //the next bit is the text that goes in the button.
                                                                               //In this button the text is "-"
                                                                               //the final block of numbers is the location of the
                                                                               //button X,Y, width, height
                                                                               //personally I find this coordinate system to be difficult
                                                                               //to work with but... that's life.

  myPort.clear();                    //clear out the serial port before we begin
  myPort.write("c,0\r");             //take the sensor out of continuous mode
  delay(60);                         //wait 60ms for the instruction to be acted upon
  myPort.write("l,50,t\r");          //set the LED brightness to 50%
  delay(60);                         //wait 60ms for the instruction to be acted upon
  myPort.write("g,1.00\r");          //set gamma correction to 1.00
  delay(60);                         //wait 60ms for the instruction to be acted upon
  myPort.write("c,1\r");             //put it back in continuous mode
  delay(60);                         //wait 60ms for the instruction to be acted upon
  myPort.write("c,1\r");             //we do this twice just for good measure
  delay(60);                         //wait 60ms for the instruction to be acted upon
}

void serialEvent(Serial p) {         //we use an interrupt even to consume the incoming serial data
  inString = p.readString();         //store the incoming data in "inString"
  data_received = true;              //set a flag saying that we got some new data
  println(inString);                 //this is just for debugging
}

void mousePressed()                                      //so, you clicked the mouse button, you might be pressing a button?
{
  if (led_up_btn.mouse_over()) {                         //if your mouse curser is over the LED + button
    led_brightness+=5;                                   //increase the LED brightness by 5%
    if (led_brightness > 100) {                          //let's make sure we don't go over 100%
      led_brightness=100;                                //if we do go over 100% set it back to 100%
    }
    myPort.write("l," + str(led_brightness) + ",t\r" );  //now we send the new brightness level to EZO-RGB
  }

  if (led_dn_btn.mouse_over()) {                         //if your mouse curser is over the LED - button
    led_brightness-=5;                                   //decrease the LED brightness by 5%
    if (led_brightness <= 0) {                           //let's make sure we don't go under 0%
      led_brightness=0;                                  //if we do go under 0% set it back to 0%
    }
    myPort.write("l," + str(led_brightness) + ",t\r" );  //now we send the new brightness level to EZO-RGB
  }

  if (gamma_up_btn.mouse_over()) {                       //if your mouse curser is over the gamma + button
    gamma+=0.05;                                         //increase the gamma by 5%
    if (gamma > 4.99) {                                  //let's make sure we don't go over 4.99
      gamma=4.99;                                        //if we do go over 4.99 set it back to 4.99
    }
    myPort.write("g," + str(gamma) + ",t\r" );           //now we send the new gamma correction value to EZO-RGB
  }

  if (gamma_dn_btn.mouse_over()) {                       //if your mouse curser is over the gamma - button
    gamma-=0.05;                                         //decrease the gamma by 5%
    if (gamma <0.01 ) {                                  //let's make sure we don't go under 0.01
      gamma=0;                                           //if we do go under 0.01 set it back to 0.01
    }
    myPort.write("g," + str(gamma) + ",t\r" );           //now we send the new gamma correction value to EZO-RGB
  }

  if (cal_btn.mouse_over()) {                            //if your mouse curser is over the calibration button
    myPort.write("cal\r");                               //send the calibration command
  }

  if (reset_btn.mouse_over()) {                          //if your mouse curser is over the reset button
    device_reset=true;                                   //we set a flag and reset the device in the main code
  }
}

void draw() {                                  //this is where the main code processed


  if (data_received == true) {                 //if we have received new data

    c1 = inString.charAt(0);                   //let's have a look at the first character in the string

    if (Character.isDigit(c1)) {               //is that character a number? because if it is, we have an RGB reading
      inString=trim(inString);                 //remove any white space in the string

      String[] q = splitTokens(inString, ","); //the string is a comma separated value, lets split the string into 3 parts at the comma
      if (q.length >= 2) {                     //but only if the string actually contains more than 2 characters (this stops errors)
        r=int(q[0]);                           //extract the RED value
        g=int(q[1]);                           //extract the GREEN value
        b=int(q[2]);                           //extract the BLUE value
      }

      data_received = false;                   //reset the data received flag
      }

    background(50);                            //set the background of the window to some sort of gray color
    fill(255, 255, 255);                       //set the next color we are working with to be white, this will be the color of the
                                               //text in the window


    text("R: " + str(r), 5, 20);               //output R: XXX (where XXX is the value for RED) the 5, 20 is the X,Y coordinate
    text("G: " + str(g), 5, 45);               //location for the text
    text("B: " + str(b), 5, 70);               //output G: XXX (where XXX is the value for GREEN) the 5, 45 is the X,Y coordinate
    text("LED: " + str(led_brightness), 85, 424);   //location for the text
    text("Gamma: " + nf(gamma, 1, 2), 360, 424);    //output B: XXX (where XXX is the value for BLUE) the 5, 70 is the X,Y coordinate
                                               //location for the text
                                               //output LED: XXX (where XXX is the value for the LED brightness) the 85, 424 is the
                                               //X,Y coordinate location for the text
                                               //output Gamma: XXX (where XXX is the value for the gamma correction) the 360, 424
                                               //is the X,Y coordinate location for the text (NF is number formatting so we don't
                                               //have an ugly floating point number)



fill(r, g, b);                                 //set the next color we are working with to be the color that the EZO-RGB sensor sees
    rect(85, 9, 387, 391);                     //draw a big rectangle that is the color the EZO-RGB sensor sees. The numbers are the
                                               //X,Y,width,height coordinates
    led_up_btn.display_btn();                  //display the LED + button
    led_dn_btn.display_btn();                  //display the LED - button
    gamma_up_btn.display_btn();                //display the gamma + button
    gamma_dn_btn.display_btn();                //display the gamma - button
    reset_btn.display_btn();                   //display the reset button
    cal_btn.display_btn();                     //display the calibration button
  }

 if (device_reset== true) {                                    //if you clicked on the reset button
       myPort.write("factory\r");                              //send the factory reset
       delay(1200);                                            //wait 1.2 sec for the instruction to be acted upon
       led_brightness=50;                                      //set the LED brightness back to 50%
       myPort.write("l," + str(led_brightness)+ ",t\r");       //now we send the new brightness level to EZO-RGB
       gamma=1.0;                                              //update the gamma correction var to 1.00 (the EZO-RGB gamma is reset
       device_reset=false;                                     //to 1.00 after a factory reset)
    }                                                          //reset the flag
}

class BUTTON {                       //in order to make all the buttons we had to make a button object.
                                     //The first part of the object is to declare the class (declare we are calling BUTTON)

  color btn_color;                   //what color is the button
  color txt_color;                   //what color will the button text be
  float x_location;                  //what is its X location
  float y_location;                  //what is its Y location
  float how_wide;                    //what is its width
  float how_high;                    //what is its height
  String btn_txt="";                 //what will the text on the button be
  int mouse_x_location;              //when you mouse over the button, what X location is that
  int mouse_y_location;              //when you mouse over the button, what Y location is that

  boolean button_enabled = false;    //is your mouse directly over a button?

  BUTTON(color c, String t, float x, float y, float w, float h) {  //to make a new button you need to send it the variables we just declared
    btn_color= c;                                                  //set the color
    txt_color= color(0, 0, 0);                                     //the button text color is fixed -black
    btn_txt= t;                                                    //set the text
    x_location=x;                                                  //set the X location
    y_location=y;                                                  //set the Y location
    how_wide=w;                                                    //set the width
    how_high=h;                                                    //set the height
  }

  void display_btn()  {                                                         //ok,lets draw the buttons
                                                                                //this is the color we are going to use to when we
    fill(btn_color);                                                            //draw the rectangel that is the button
    rect(x_location, y_location, how_wide, how_high);                           //the location and size of the button
    fill(txt_color);                                                            //this is the color we are going to use for the text
    textAlign(CENTER, CENTER);                                                  //make sure the text in the button is centered
    text(btn_txt, ((x_location+1)+ (how_wide/2)), (y_location+1)+(how_high/2)); //this is the location of the text inside the button
    textAlign(LEFT, BASELINE);                                                  //move the centered text again,...very tedious!
  }

  boolean mouse_over() {                       //this tells us if your mouse is over a button
    mouse_x_location= mouseX;                  //store the mouse's X location
    mouse_y_location= mouseY;                  //store the mouse's Y location

    boolean mouse_x_over_btn=false;            //used as a flag to show the mouse is over the X location of a button
    boolean mouse_y_over_btn=false;            //used as a flag to show the mouse is over the Y location of a button

    if ((mouse_x_location >= x_location) &&  (mouse_x_location <= (how_wide+ x_location))) {  //if the mouse if over a buttons X location
      mouse_x_over_btn=true;                                                                  //set the flag to be true
    }

    if ((mouse_y_location >= y_location) &&  (mouse_y_location <= (how_high+ y_location))) {  //if the mouse if over a buttons Y location
      mouse_y_over_btn=true;                                                                  //set the flag to be true
    }

    button_enabled = ((mouse_x_over_btn==true) && (mouse_y_over_btn==true));   //if the mouse is over both the X and Y
    return button_enabled;                                                      //location of a button, set button enabled
  }                                                                             //to true
}                                                                               //return said truth
```

Click here to download the *.pde file