**Week 10: Arduino Introduction Worksheet**

**Lab Deliverable**

You will need to turn in your INDIVIDUAL lab report worksheet (this document) with all code and questions completed by the due date listed on Canvas. Some of the tasks will be done as you go through the process, while some questions must be answered at the end of an activity. While your procedures will look identical to your team members, make sure that your answers are your own and you are able to explain every detail in this lab worksheet. DO NOT COPY your team members’ reports. If you are absent during the lab, you will be responsible for completing all of the work in this report yourself.

**Activity Description**

Automation is a huge field of advancement today to yield higher safety and awareness of everyday objects. This can exist in terms of digital or analog applications; where digital means that a value is either 0 or 1, mimicking the theory of digital logic seen in modern microcontrollers. Analog sensors however have a range of values that can indicate different situations for different values. In this lab, you will learn how to incorporate lights, switches, and buttons as digital sensors. You will also learn how to incorporate analog sensors through the use of light sensors. An important outcome of this lab is to familiarize yourself with the Arduino hardware and sensor uses and see the potential of future automation of everyday applications.

*NOTE: This lab can be completed virtually using TinkerCAD. All buttons, lights, switches, boards, etc. can be simulated using the Circuits tool. Please beware that the TinkerCad compiling tool can have issues. If you are working in TinkerCad and you get unexpected compilation errors, copy your code to a new file and retry, as this may solve your problems.*

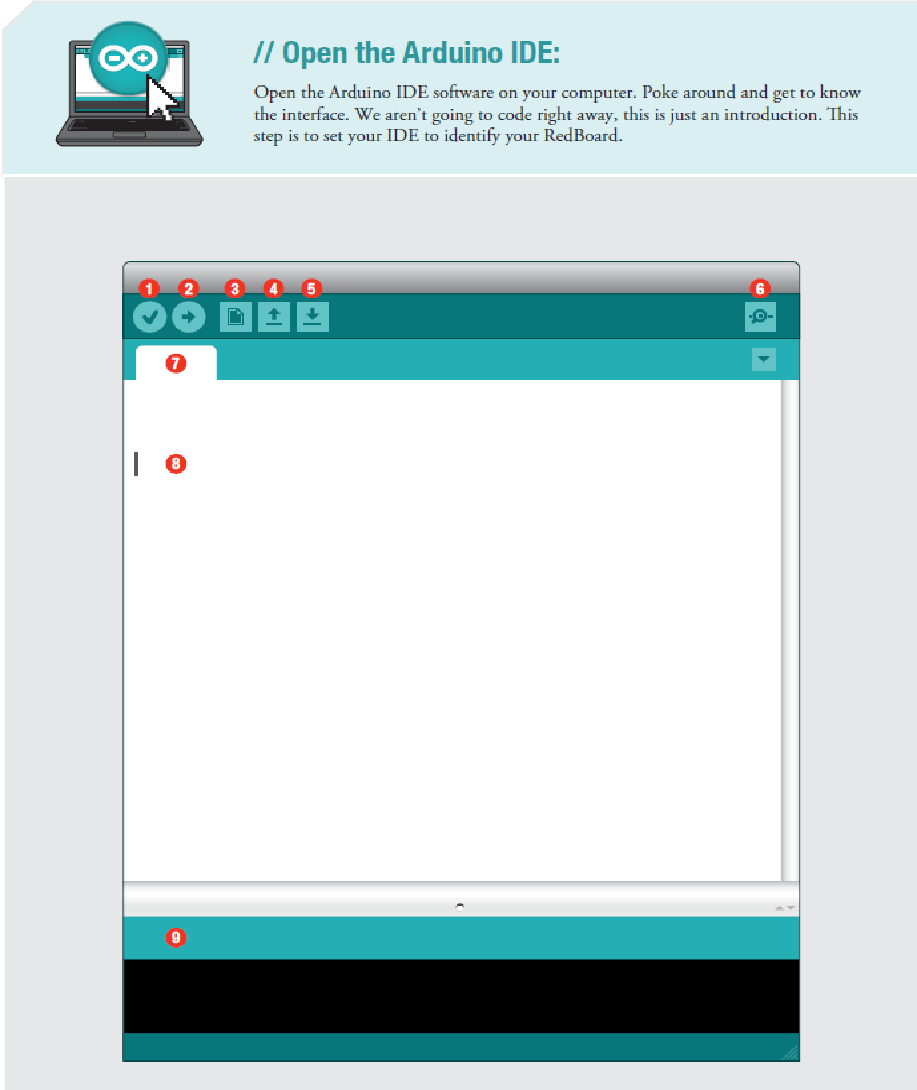
**Lab Procedure**

**Part 1: Setup (If Using Arduino Hardware)**

To begin, connect your Sparkfun RedBoard to your computer with the Arduino IDE installed via the USB cable - *be sure to have no other power source connected to the Arduino at this time, or you may accidentally fry your laptop!*



Now, open the Arduino IDE software on your computer. This step is to set your IDE to identify your RedBoard.





**Verify:** Compiles and approves your code. It will catch errors in syntax (like missing semi-colons or parenthesis).

**Upload:** Sends your code to the RedBoard. When you click it, you should see the lights on your board blink rapidly.

**New:** The button opens up a new code window tab.

**Open:** This button will let you open up and existing sketch.

**Save:** This saves the currently active sketch.

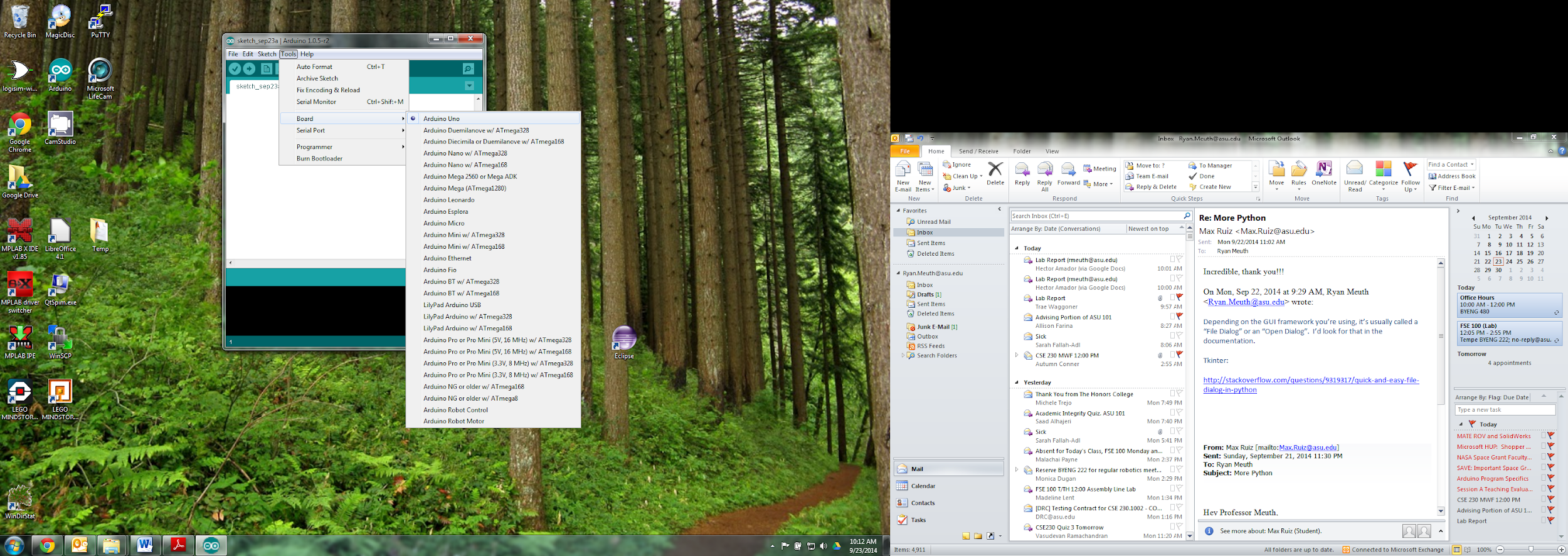
**Serial Monitor:** This will open a window that displays any serial information your RedBoard is transmitting. It is very useful for debugging.

**Sketch Name:** This shows the name of the sketch you are currently working on.

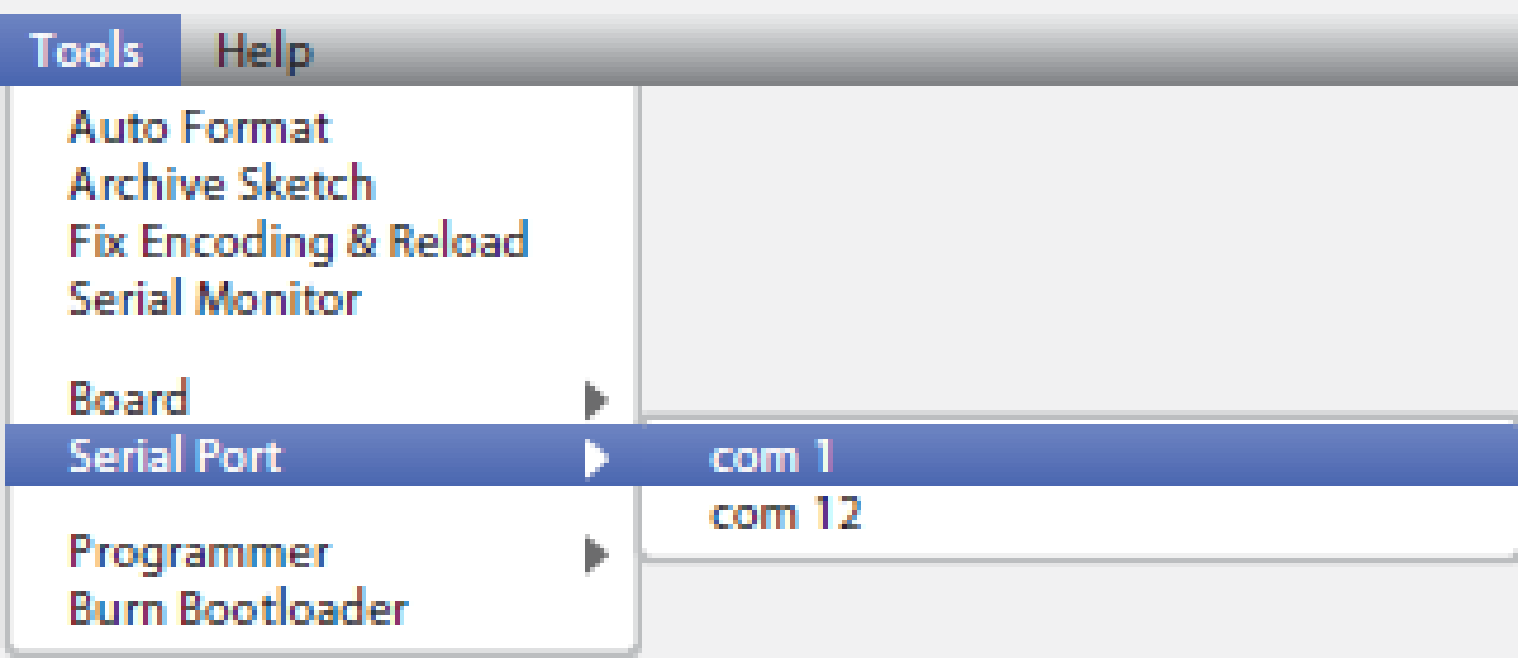
**Code Area:** This is the area where you compose the code for your sketch.

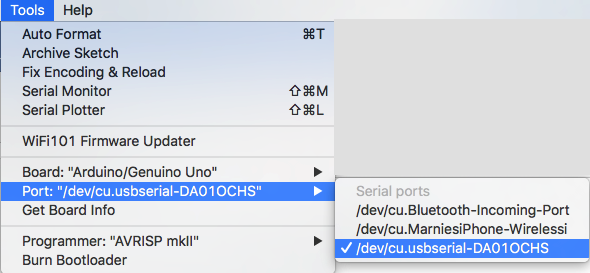
**Message Area:** This is where the IDE tells you if there were any errors in your code.

Next, select Tools->Board-> Arduino Uno (or Arduino/Genuino Uno). This will select the type of board which you are using.



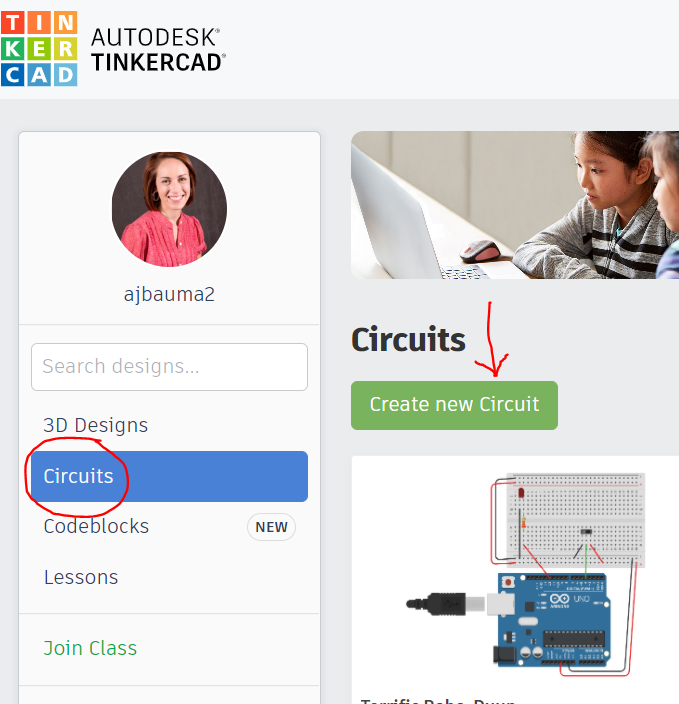
Then Tools->Serial Port-> COM3 (Or higher, whichever shows up). *(For Macs, Tools->Port->\*usbserial\*)*

Windows:

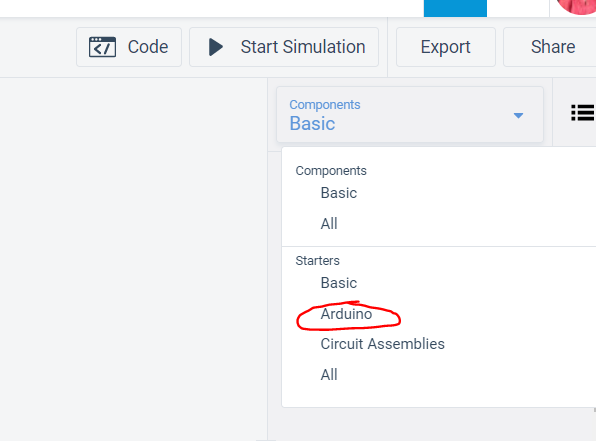
Mac:

**Part 1: Setup (If Using TinkerCAD Virtual Simulator)**

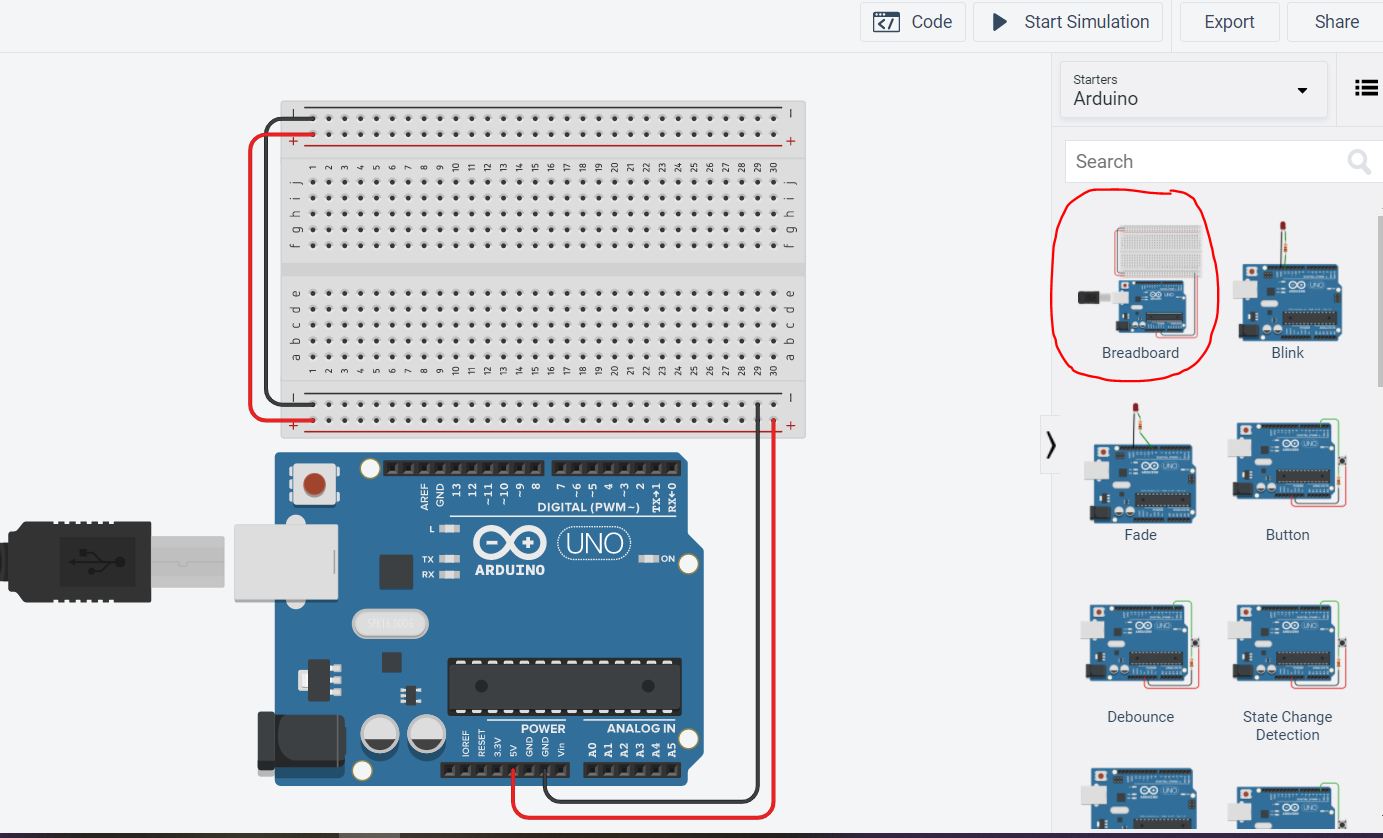
Log into Tinkercad.com with your personal account, or log into your class link with your ASURite username. From your main dashboard, on the left menu panel, select Circuits. Click Create New Circuit to open a fresh workspace.



Under the right drop-down menu of Components, select the Arduino option from the Starters menu.



Select the Breadboard option with the Arduino Uno and the breadboard already connected and place it on your fresh workspace by clicking in the open area. It should result in this view, and you are now ready to code:



This is the pin diagram of the Board.

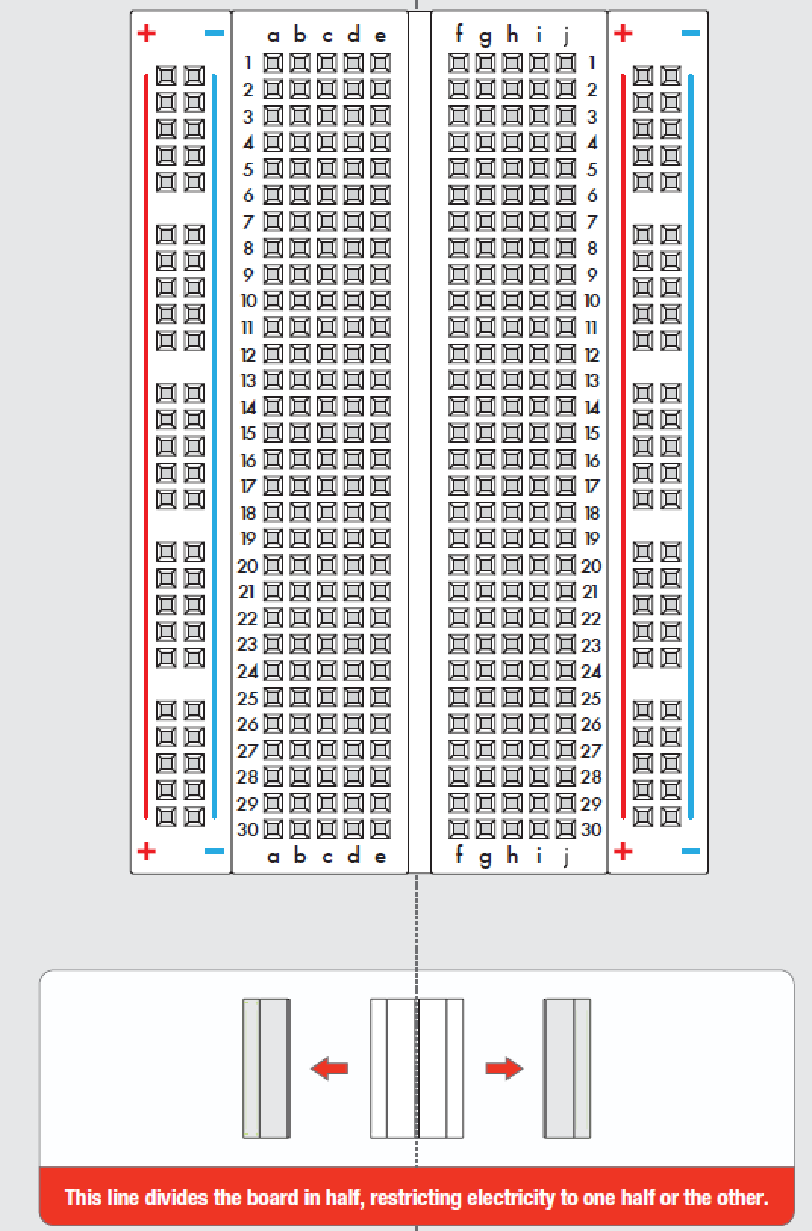


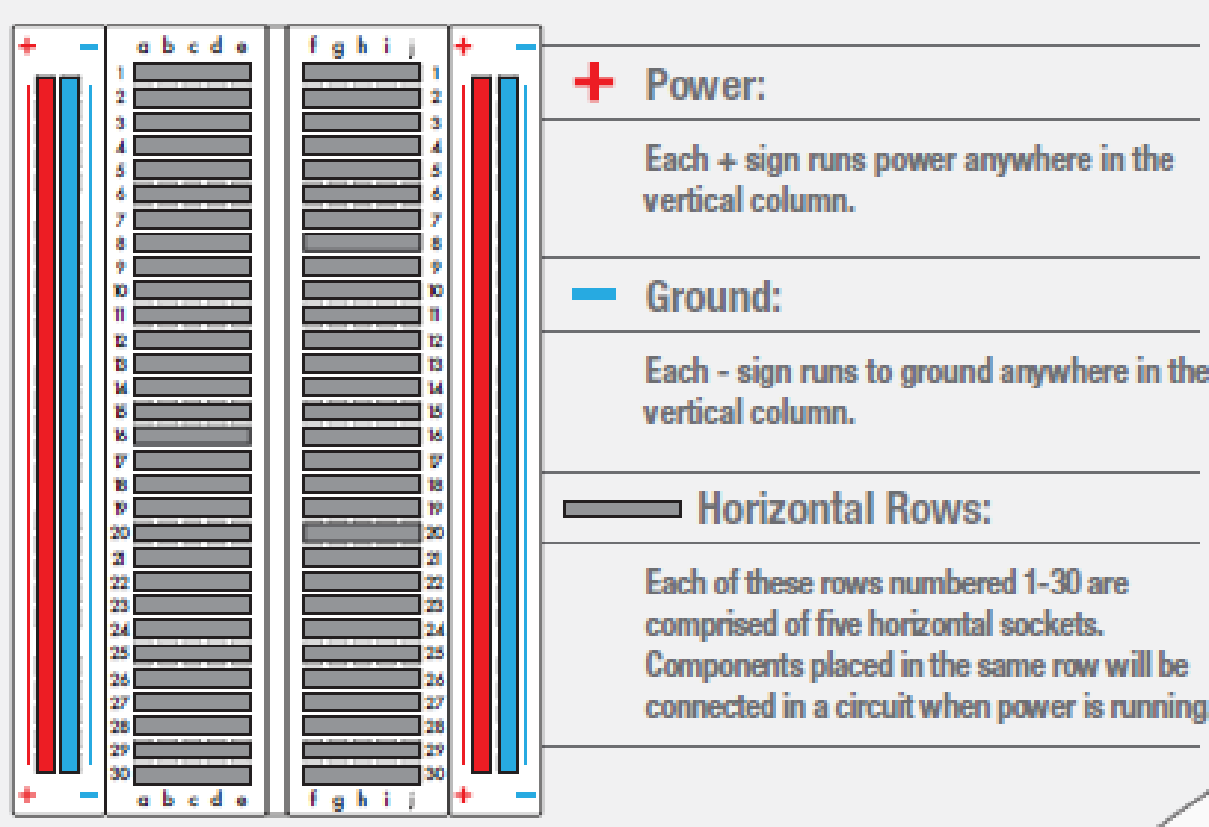
The digital pins (0 through 13) are either ON (HIGH/5V) or OFF (LOW/0V). Digital pins can be either inputs or outputs.

The analog pins (A0 through A5) can read any voltage between 0V and 5V. The microcontroller converts the analog voltage to a number between 0 (0V) and 1024 (5V). They can also act as additional digital inputs and outputs.

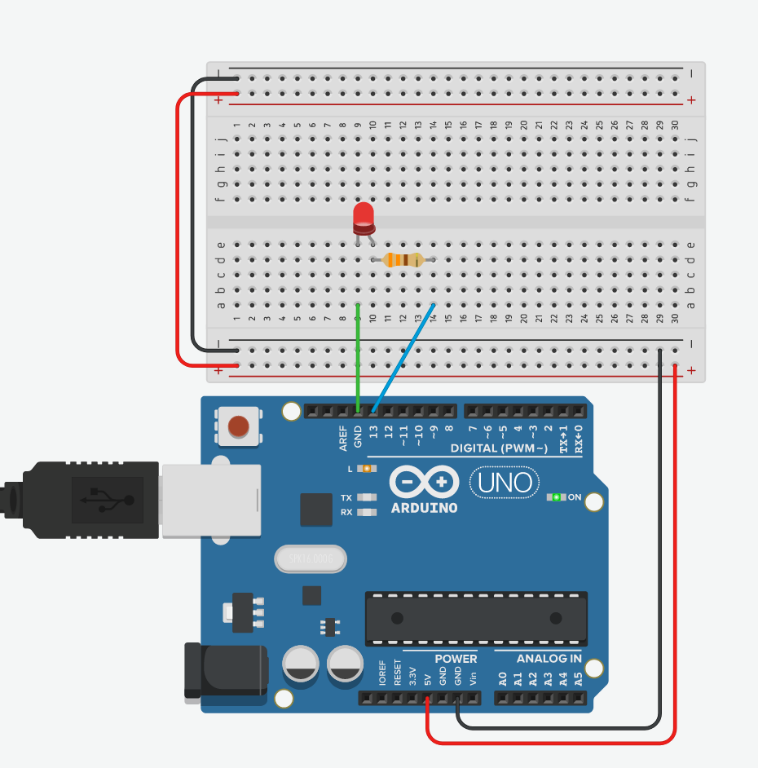
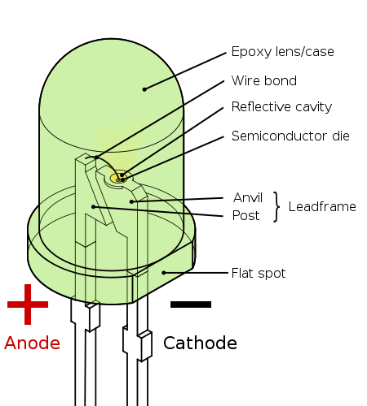
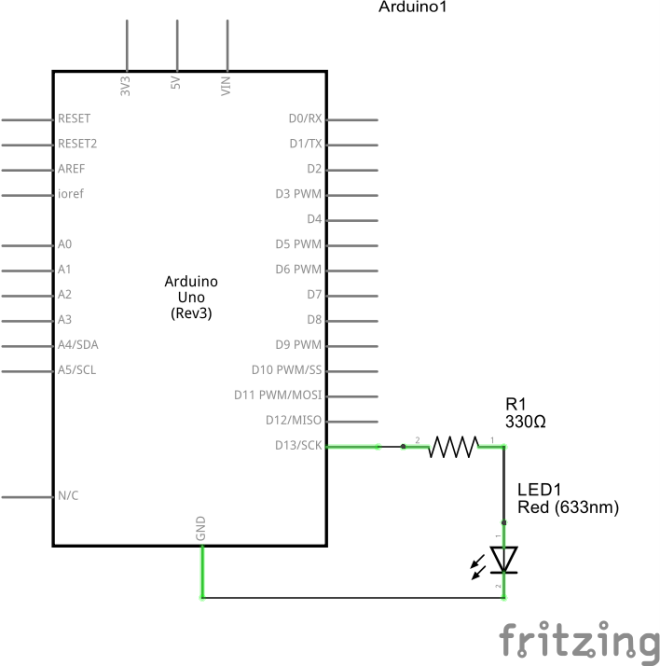
The power pins (VIN, GND, 5V, 3.3V, RESET, IOREF, RFU) are supplies, grounds, and references.

This is the layout of the breadboard. (Please refer to the Breadboard video in Module 10 for more detailed instructions).





**Part 2: Blinky Test**

Using the hookup wires, a 330-ohm resistor, and an LED, construct the following circuit:

Remember LED polarity. The cathode is the side that has the flat edge on the bulb, and that indicates the negative side of the light which should be connected to Ground. The anode is the round side which should be connected to the resistor.

In the Editor, enter the following code to test the circuit connection and programming setup:

/\* Hello World! \*/

void setup**()** **{** 

pinMode**(**13**,** OUTPUT**);** 

**}**

void loop**()** **{**

digitalWrite**(**13**,** HIGH**);** 

delay**(**1000**);** 

digitalWrite**(**13**,** LOW**);** 

delay**(**1000**);**

**}**

Run the simulation and observe the effect on the Arduino.

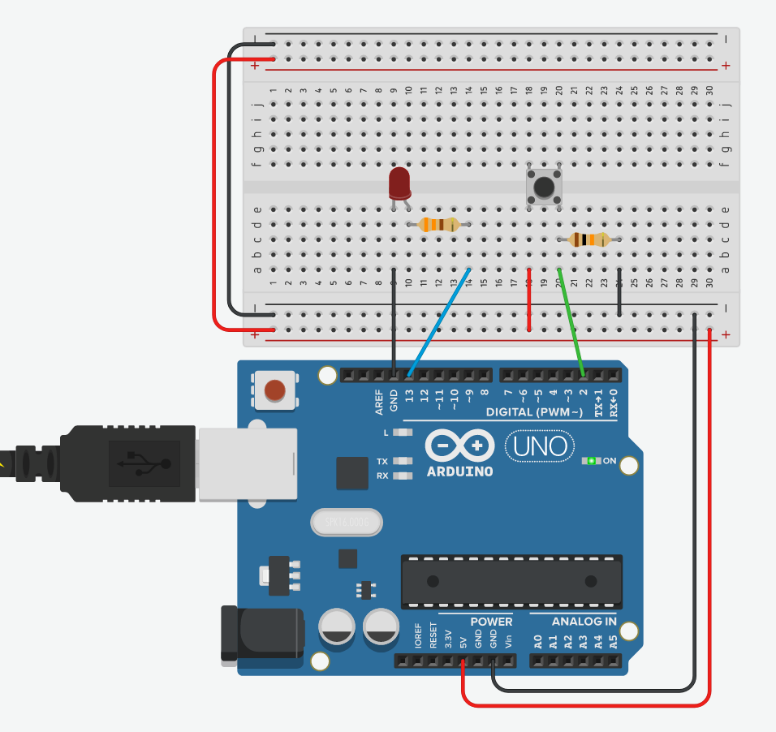
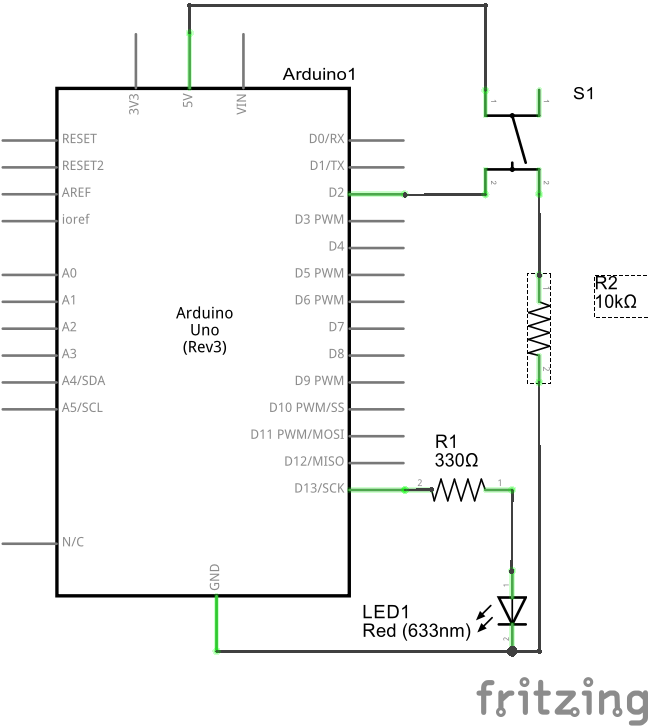
**ON YOUR OWN:** Modify the program to blink “S-O-S” in Morse Code. Make sure there is a clear long pause between the end of the message and the start of the next one. **Paste your code on your worksheet and demo your code to your instructor, TA, or team members.** *Hint: if you aren’t sure what S-O-S is in Morse code, look it up online.*

Just like in MATLAB, syntax matters. If you don’t use the correct syntax, the Arduino will not understand what you are trying to do.

| **Part 2: SOS Code** |
| --- |
| void setup() {  pinMode(13, OUTPUT);  }  void loop() {  // Blink "S" which is three dots in morse code  digitalWrite(13, HIGH);  delay(1000); // dot is generally 1 second  digitalWrite(13, LOW);  delay(1000); // delay like in blink program  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);    delay(2000); // delay between letters    // Blink "O" which is three dashes in morse code    digitalWrite(13, HIGH);  delay(3000); // dash is generally 3 seconds  digitalWrite(13, LOW);  delay(1000);  digitalWrite(13, HIGH);  delay(3000);  digitalWrite(13, LOW);  delay(1000);  digitalWrite(13, HIGH);  delay(3000);  digitalWrite(13, LOW);  delay(1000);    delay(2000); // delay between letters    // Blink "S" again  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);    delay(4000); // Delay before repeating  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Instructor, TA, Teammate approval (for Demo)** |

**Part 3: Digital Inputs with Buttons**

In this part, you will build a circuit that turns on a light when a button is pressed, to only act when an input has been activated. Build the following circuit:



In the sketch editor, enter the following code:

int buttonPin **=** 2**;**

int ledPin **=** 13**;**

int buttonState **=** 0**;**

void setup**()** **{**

pinMode**(**ledPin**,** OUTPUT**);**

pinMode**(**buttonPin**,** INPUT**);**

**}**

void loop**(){**

buttonState **=** digitalRead**(**buttonPin**);**

if **(**buttonState **==** HIGH**)** **{**

digitalWrite**(**ledPin**,** HIGH**);**

**}**

else **{**

digitalWrite**(**ledPin**,** LOW**);**

**}**

**}**

Upload the code to your Board and observe the effect of pressing and holding the button. **Record your observation on the Worksheet.**

Using your existing 1 button, 1 LED circuit, run the following program:

int buttonPin **=** 2**;**

int ledPin **=** 13**;**

int buttonState **=** 0**;**

int pressCount **=** 0**;**

void setup**()** **{**

pinMode**(**ledPin**,** OUTPUT**);**

pinMode**(**buttonPin**,** INPUT**);**

Serial**.**begin**(9600);**

**}**

void loop**(){**

buttonState **=** digitalRead**(**buttonPin**);**

if **(**buttonState **==** HIGH**)** **{**

digitalWrite**(**ledPin**,** HIGH**);**

pressCount= pressCount + 1**;**

**}**

else **{**

digitalWrite**(**ledPin**,** LOW**);**

**}**

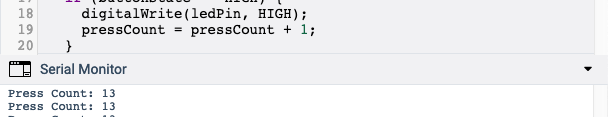
Serial**.**print**(**"Press Count: "**);**

Serial**.**println**(**pressCount**);**

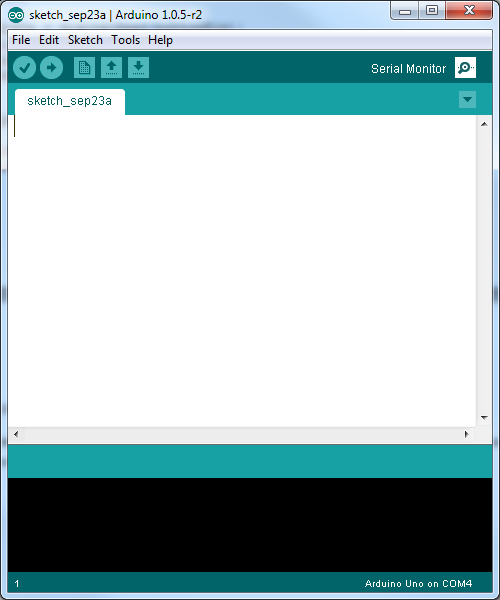
**}**

Upload the program to the Arduino code, run the simulation, then turn on the Serial Monitor.

If you are using TinkerCAD online, the Serial Monitor is below your code editor window.



If you are using the Arduino IDE software with the hardware kit, your Serial Monitor is in the upper right corner of your editor window:



Press the button on your breadboard and observe the effect. **Record your observations in the Worksheet**.

**ON YOUR OWN:** Write a program that implements the following functionality:

* On every 5 presses of button 1, the LED toggles (turns on after 5 clicks, turns off after another 5 clicks).
* The number of presses is output to the Serial Monitor.

*Hint: you’ll notice that the press count increases faster than you can lift your finger off the button. Add a delay after you initially recognize that the button is HIGH to slow the code down for your press and depress.*

**Record your counting code on the Worksheet and demo your code to the Instructor, TA, or team member.**

| **Part 3: Digital Inputs with Buttons** |
| --- |
| **What did you observe when you press and depress the button?**  **When the button is pressed the light is lit, when not pressed the light is off** |
| **What did you observe when monitoring the Serial port?**  **Everytime the button is pressed the counter increased** |
| **5 Count Press Code**  **int buttonPin = 2;**  **int ledPin = 13;**  **int buttonState = 0;**  **int pressCount = 0;**  **bool ledState = false;**  **const int toggleThreshold = 5;**  **void setup() {**  **pinMode(ledPin, OUTPUT);**  **pinMode(buttonPin, INPUT);**  **Serial.begin(9600);//bitrate of serial monitor**  **}**  **void loop() {**  **buttonState = digitalRead(buttonPin);**  **if (buttonState == HIGH) {**  **delay(50);**  **if (buttonState == HIGH) {**  **pressCount++;**  **Serial.print("Press Count: ");**  **Serial.println(pressCount);**    **if (pressCount % toggleThreshold == 0) {**  **ledState = !ledState;//changes state of led**  **digitalWrite(ledPin, ledState ? HIGH : LOW);**  **}**  **}**  **while (digitalRead(buttonPin) == HIGH)//stops looping till led low**  **{**  **}**  **}**  **}**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Instructor, TA, Teammate approval (for Demo)** |

**Part 4: Analog Inputs with Light Sensors**

In this part of the lab, you will learn how to incorporate an analog sensor and use a range of values to trigger a reaction. Using the photoresistor and a 10K Ohm resistor, build the following schematic:

# C:\Users\rmeuth\Google Drive\ASU\CourseWare\FSE100 - Intro Engineering\Alternate Version\Labs\Media\Photoresistor_schem.jpg

Upload the following code to test your photoresistor circuit:

void setup()

{

Serial.begin(9600);

}

void loop()

{

int val = analogRead(0);

Serial.print("Sensor Value: ");

Serial.println(val);

delay(100);

}

View the Serial Monitor. Adjust the light value on the photoresistor.

Add an LED circuit to your breadboard similar to Part 2, but use the Photoresistor as the switch.

**ON YOUR OWN:** Write code to turn on the LED when the Photoresistor reads a low level of light by covering the sensor with your hand.

**Copy your night-light code to the Worksheet and demo your code to the Instructor, TA, or teammate.**

| **Part 4: Analog Inputs with Light Sensors** |
| --- |
| **Night Light Code**  **void setup()**  **{**  **Serial.begin(9600);**  **pinMode(9, OUTPUT);**  **}**  **void loop()**  **{**  **int val = analogRead(0);**  **Serial.print("Sensor Value: ");**  **Serial.println(val);**  **delay(100);**    **if (val < 900) {**  **digitalWrite(9, HIGH);**  **} else {**  **digitalWrite(9, LOW);**  **}**  **}**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Instructor, TA, Teammate approval (for Demo)** |
| **Part 5: Individual Reflection** |
| **1. In your S-O-S code, it is possible to reduce the number of lines of code needed within the “void loop” structure by creating your own functions based on the Morse Code letter of choice. Create your own functions called “morse\_S” and “morse\_O” so that when either function is used within the “void loop”, it will perform the LED actions associated with that letter of Morse Code.**  void blinkDot() {  digitalWrite(13, HIGH);  delay(1000); // dot is generally 1 second  digitalWrite(13, LOW);  delay(1000); // gap  }  void blinkDash() {  digitalWrite(13, HIGH);  delay(3000); // dash is generally 3 seconds  digitalWrite(13, LOW);  delay(1000); //  }  void morse\_S()  {  blinkDot();  blinkDot();  blinkDot();  }  void morse\_O()  {  blinkDash();  blinkDash();  blinkDash();  } |
| **2. Your traffic light sensor will be a “Sparkfun RGB Light Sensor – ISL29125” found here:** [**https://www.sparkfun.com/products/12829**](https://www.sparkfun.com/products/12829)  **Research and describe the pin setup required for this board to connect to your Arduino Redboard.**  **Then, research and describe the output data that is measured by this sensor. How can you use this output data from the sensor to help you with your traffic light requirements?**  To connect to the arduino board we need to connect 3.3v to the arduino 3.3vpin. We also need to connect the GND to arduinoGND pin. Then connect SDA to the arduino A4 pin. Next connect SCL to Arduino A5 pin.  The output data represents RGB which is red, green, blue. These can be used to assess ambient light conditions, enabling adaptive control of devices such as traffic lights. With analyzing these values, adjustments can be adjusted to the brightness and color calibration of lights, changing energy efficiency, visibility, and safety. |
| **3. The following code has 3 errors in it causing the program to fail. Read the comments of the program as it is intended to run, and identify the 3 errors within the code.**  /\*  ReadAnalogVoltage  Reads an analog input on pin A0, converts it to voltage, and prints the result to the Serial Monitor. Graphical representation is available using Serial Plotter (Tools > Serial Plotter menu). Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.  This example code is in the public domain.  http://www.arduino.cc/en/Tutorial/ReadAnalogVoltage  \*/  // the setup routine runs once when you press reset:  void setup() {  pinMode(A0, OUTPUT);  // initialize serial communication at 9600 bits per second:  Serial.begin(9600);  }  // the loop routine runs over and over again forever:  void loop() {  // read the input on analog pin 0:  int sensorValue = analogREAD(A0);  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):  float voltage = sensorValue \* (5.0 / 1023.0);  // print out the value you read:  Serial.println(Voltage);  }  **Error #1:**pinMode(A0, OUTPUT); Should be pinmode(A0, INPUT)  **Error #2:**analogREAD(A0); should be analogRead(A0)  **Error #3:**Serial.println(Voltage); should be Serial.println(voltage) |