Name: ___________________________

Class:____________________________

Module 4: Traffic Signal Design
Lesson 1: Traffic Signal (Arduino) Control System
Laboratory Exercise
Grade 6 - 8

Background

Traffic signals are used to control traffic that flows in opposing directions at intersections. Traffic signals use red, yellow and green indicators to permit or prevent vehicles from entering the controlled intersection at different times. The indicators, frequently referred to as "red lights," "yellow lights," and "green lights," organize traffic to prevent crashes and establish a right of way.

Traffic signals work by alternating the ability for vehicles to travel on each approach using different phases, which can have set or varying phase lengths. Phase lengths are the amount of time assigned to each green, yellow, or red phase.

Traffic Signal Design Activity

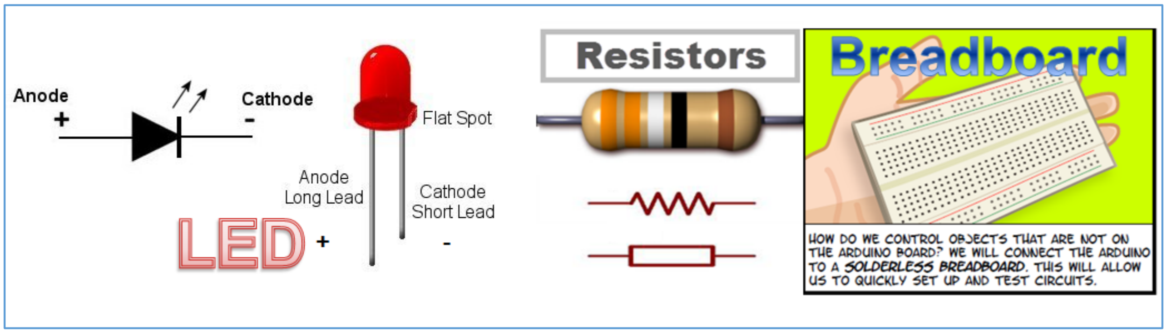
For this activity, you and your partner will act as transportation engineers and use an Arduino to construct a working traffic signal. In order for the Arduino traffic signal to operate properly, you will need to develop a computer program that tells each light (red, yellow, and green) when to come on, how long to stay on, and when to turn off.

Part 1: Constructing the Arduino Traffic Signal Circuit

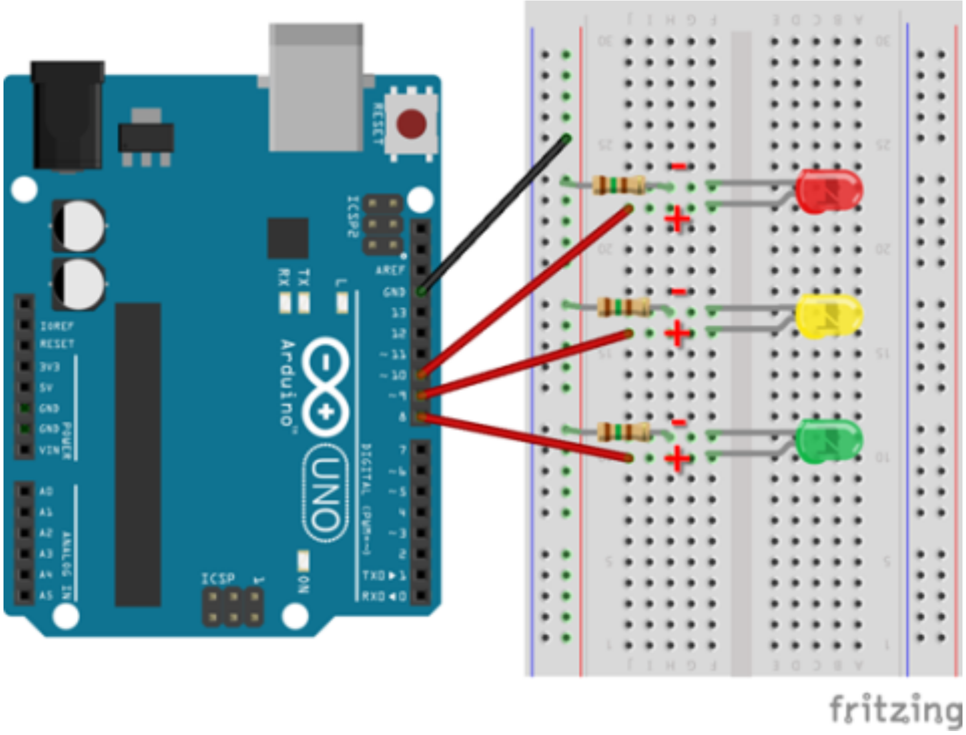
1. Break into groups of 2 or 3 students.
2. Obtain the following Materials:
   - **Arduino Microcontroller**
   - **3 – LEDs (1 Red, 1 Yellow, 1 Green)**
   - **3 – 330 Ω Resistors**
   - **Breadboard**
   - **4 – Wires**
   - **Computer with Arduino Programming Software**

Circuit Background

When working with LED (light emitting diode) lights, you will notice that one leg of the light is shorter than the other (as shown in the picture below). Because LEDs are polarized and only allow current to flow in one direction as their circuit symbol indicates, manufacturers make the legs different lengths to allow users to differentiate the polarization. As shown, the ***longer*** leg is called the **anode** and represents the **positive (+)** polarity and the ***shorter*** leg is called the **cathode** and represents the **negative (-)** polarity.


Anode
+
Cathode
-
Flat Spot
Anode
Long Lead
Cathode
Short Lead
LED
+
-
Resistors
Breadboard
HOW DO WE CONTROL OBJECTS THAT ARE NOT ON THE ARDUINO BOARD? WE WILL CONNECT THE ARDUINO TO A SOLDERLESS BREADBOARD. THIS WILL ALLOW US TO QUICKLY SET UP AND TEST CIRCUITS.

Using the picture below, assemble the traffic light circuit; notice that the polarity of the LEDs are shown on the breadboard. When constructing the traffic signal circuit using the LED lights, the anode (+) should be connected to the digital pins on the Arduino board to turn the light on and off (any numbered pin can be used – but you'll need to remember this number later). The cathode (-) will be connected to the resistor which is connected to the ground to complete the circuit. If the LED is positioned incorrectly, the current will be unable to flow and the LED will ***not*** work.



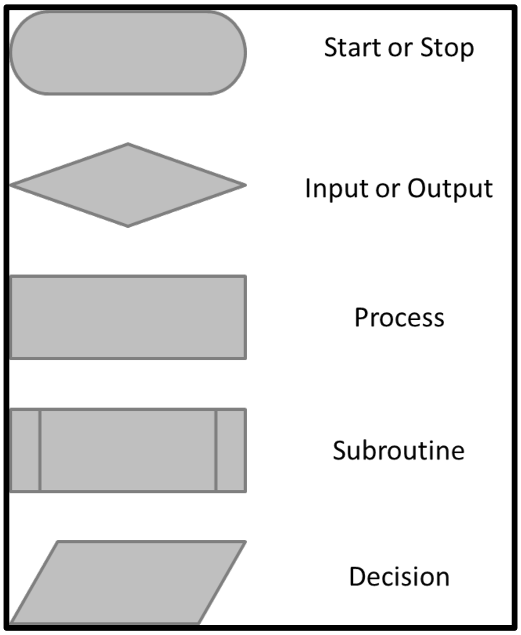
Part II: Algorithm Development with Pseudo Code

As a transportation engineer, you will need to write a computer program to allow your traffic signal to function properly. First, we will develop a flowchart to help us think about how we want the traffic signal to function. Consider the questions below before beginning your flowchart.

1. Will the lights in the traffic signal be inputs or outputs?
2. Will the lights in the traffic signal follow a specified order?
3. How long should each light remain on?
4. When one light turns on, what happens to the other two lights?

Now that you have identified how your traffic signal will function, use this information to develop a flowchart that shows the sequence the traffic signal will follow. Use the following symbols to represent the traffic signal processes.

<u>**Flow Chart Symbols**</u>

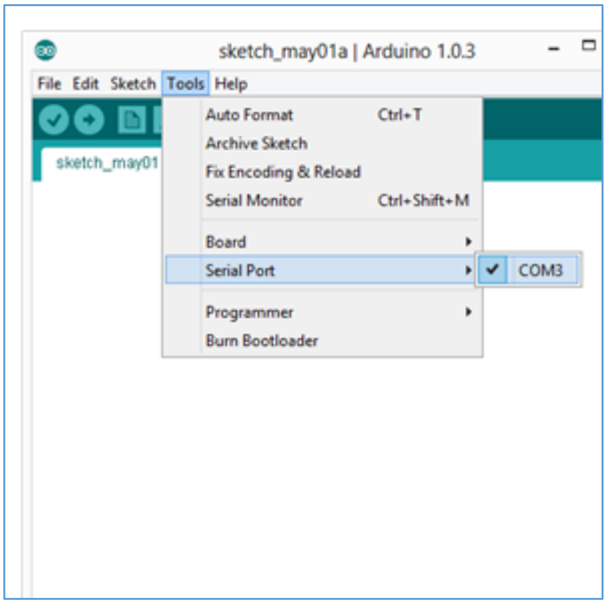


<u>**Traffic Signal Logic Flow Chart**</u>

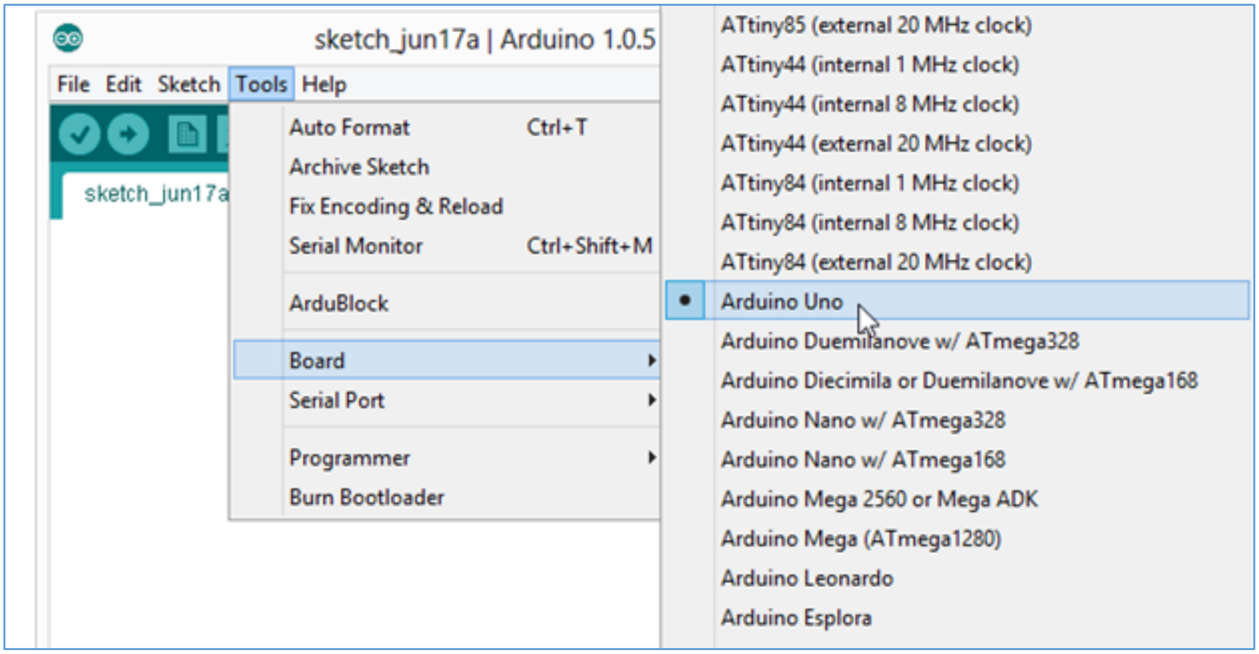
Develop the traffic signal flow chart below

Part III: Writing Arduino Code

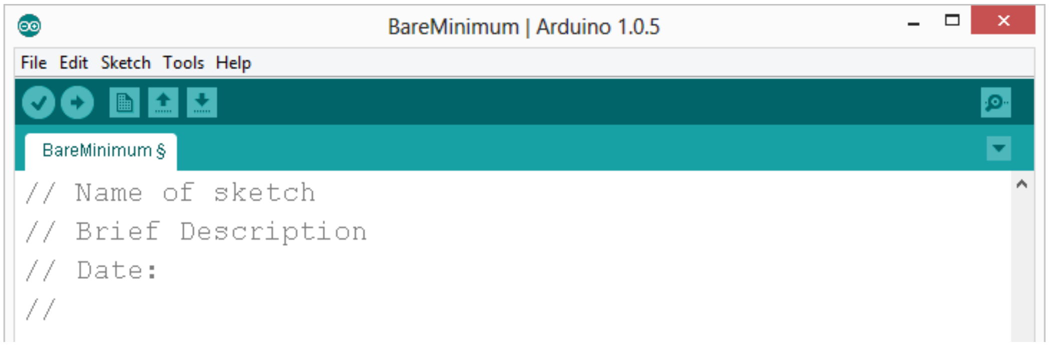
1. Connect Computer to USB port on the Arduino Uno boards.
2. Open the Arduino program:
3. Find the Arduino connection on the computer: Tools → Serial Port
   Your computer communicates to the Arduino microcontroller via a serial port → through a USB-Serial adapter. Check to make sure that the drivers are properly installed.



4. Double-check that the proper board is selected under the Tools→Board menu. (You are using an Arduino Uno).


sketch_jun17a | Arduino 1.0.5
File Edit Sketch Tools Help
sketch_jun17a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
ArduBlock
Board
Serial Port
Programmer
Burn Bootloader
ATtiny85 (external 20 MHz clock)
ATtiny44 (internal 1 MHz clock)
ATtiny44 (internal 8 MHz clock)
ATtiny44 (external 20 MHz clock)
ATtiny84 (internal 1 MHz clock)
ATtiny84 (internal 8 MHz clock)
ATtiny84 (external 20 MHz clock)
● Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora

5. Just like any work you do – put your name and date at the top in a comment along with a brief description:



```
// Name of sketch
// Brief Description
// Date:
//
```

6. When you first open the Arduino program, you will be looking at an Arduino "sketch." A sketch is what the Arduino calls the program or code. You should see two required functions on your screen:
void setup() {

// put your setup code here, to run once:

}

****The void setup() function is where you will define your inputs and outputs.***

void loop() {

// put your main code here, to run repeatedly:

}

****The void loop() function is for your main code that the Arudino will continually loop through.***

Both of these functions are automatic and are required to run the Arduino program.

****Remember: When you are writing a computer program, the language is very sensitive, so spaces, punctuation, lower and uppercase letters make a difference! When you use codes that Arduino recognizes, you will notice them change color.**

7. When you begin writing a computer program, a good practice is to first define the variables. You will need to do this before the void setup function. In this case, there will be 3 variables – green, yellow, and red. You will define your variable related to the pin number.

   Ex. int GREEN = 2; (this would mean that the green light is in pin 2)

8. Once you have defined your variables, you are ready to write the program. Under the ***void setup function, you will need to setup whether each pin is an output or input***. To do this, Arudino has a function that it already understands.

   Ex. pinMode (GREEN, OUTPUT);

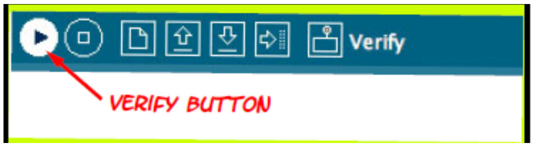
9. Now, ***under the void loop function***, you will need to tell the traffic signal exactly how to function. Keep in mind you will need to tell the lights when to come one, how long to stay on, when to turn off, and how long between each light. (Remember the language is very sensitive so be careful).

Arduino Coding

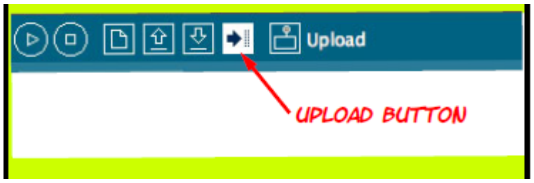
digitalWrite (GREEN, HIGH) – This function tells the light to turn on or off. A HIGH reading turns the light on and a LOW reading would turn the light off.

Delay (400) – This function delays the Arudino for a specific length of time. The Ardunio reads milliseconds so this function would delay the program for 400 milliseconds. 1000 milliseconds = 1 second.

10. Once you have written the code for the traffic signal, you will need to verify the code to ensure that there are no mistakes. To do this, click on the checkmark. If it produced errors look for missing parenthesis and semicolons.



11. If your code didn't produce any errors on the bottom, upload the code to the Arduino (click the right pointing arrow) and see if it works.


Upload
UPLOAD BUTTON