### Microcontroller Familiarization

### Due Date:

Consult schedule/syllabus. Submit code on Canvas.

The TAs will run and evaluate your code on a separate test circuit during office hours (see Canvas for the TA office hours).

### Purpose:

This assignment provides an introduction to programming, state machines, and basic Arduino libraries.

### Teamwork:

This assignment shall be completed individually.

### Materials:

The following materials will be provided to you:

* Arduino Uno
* USB A-B Cable
* Breadboard
* Button (2)
* RGB LED
* 10kΩ Rotary Potentiometer
* 150Ω Resistor
* 100Ω Resistor (2)
* 10kΩ Resistor (2)
* Wire

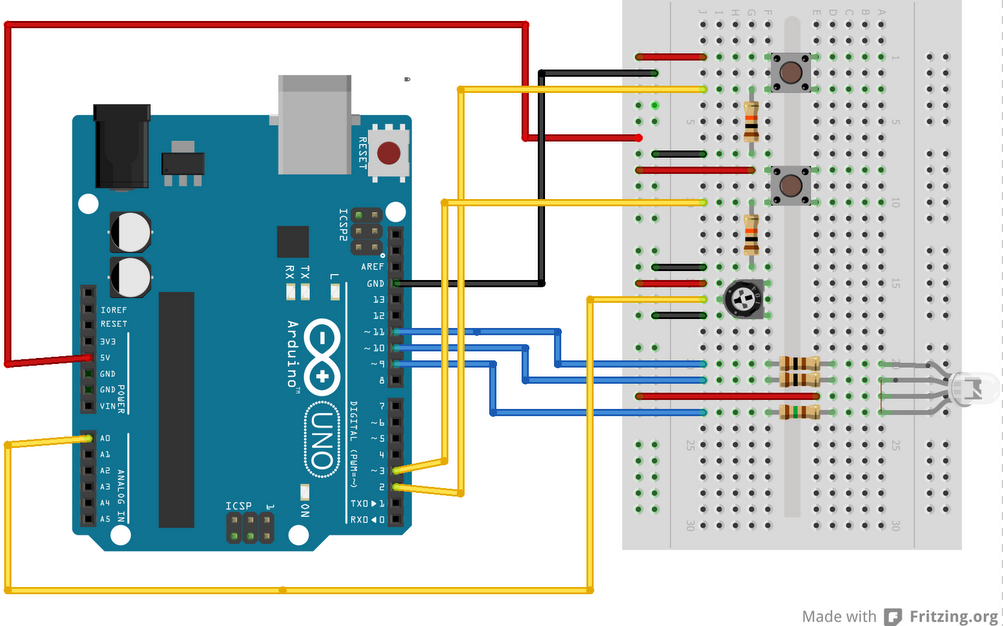
Upon completion of this assignment, please return all items **exactly how you found them** in the zip-lock bag.

### Overview:

Write a program for the Arduino to control the brightness and hue of a tri-color LED. This control will be implemented using a number of inputs: digital, analog, and serial. A simple finite state machine will be used to decide the source of control at any time.

### STEP #1: Wiring

The circuit has already been designed so you can focus on the software instead of the hardware. **Be sure to follow this circuit as shown below**, because your code will be evaluated by running it on a setup wired in this configuration by the TAs.



The sensors and LED channels will be connected to the following pins:

|  |  |
| --- | --- |
| **Arduino UNO Pin #** | **Component** |
| A0 | Potentiometer |
| 2 | Button 0 |
| 3 | Button 1 |
| 9 | Red LED channel |
| 10 | Green LED channel |
| 11 | Blue LED channel |

**Tip 1:** The buttons are not intended for breadboards. To insert buttons, try straightening the pins and giving them a slight axial twist.

**Tip 2:** These buttons need debouncing for reliable operation. You may not change the circuit, so you need to program a debouncing function in software.

### STEP #2: Programming

### State control:

Switching between states shall happen every time Button 0 is pressed. The states shall transition in order (0, 1, 2, 0, 1, 2, …). Every time Button 0 is pressed, one and only one state shall advance. Holding or releasing Button 0 shall have no further effect. This state machine switching shall be implemented using an interrupt. Button 0 has been wired to an interrupt pin so this is possible.

### States:

Each state (0, 1, and 2) shall behave as described below. Each state also has a “bonus challenge” in case you want an additional task for more practice. For each state, either behavior is acceptable and will not lose or gain points (assuming it works).

The settings and functionality of the states shall be completely independent between the states. For example, the brightness set using the potentiometer in State 1 shall not affect the functionality of State 0.

In each state, you shall print to Serial whenever the state changes (including the current state value). It is also helpful to print the values being read from the sensors and being written to the LED to help with debugging.

|  |  |
| --- | --- |
| **STATE** | **DESCRIPTION** |
| 0 | **Blinking LED:** Each time Button 1 is pressed, the color of the LED shall toggle once between on and off (if the LED is off, it should turn on; if the LED is on, it should turn off). Holding or releasing Button 1 shall have no effect. In this state, the LED can be treated as a binary switch, where the controlling pin (for each color) can be set LOW to turn it on and set HIGH to turn it off. The key here is that you are reading a digital input and writing to a digital output each time Button 1 is pressed.   * 1. Bonus: If you desire, you may instead have the LED switch between colors by changing which of the colors are on or off each time the button is pressed. For example, cycling through red-green-blue, or red-yellow-green-cyan-blue-magenta. |
| 1 | **Controlling Brightness:** The brightness of the LED is controlled by turning the potentiometer. As the potentiometer is turned to the right (clockwise), the LED shall become brighter, and as it is turned to the left, the LED shall become dimmer. The key here is that you are reading an analog input and writing to an analog output.   * 1. Bonus: If you desire, you may instead change the hue of the LED (by dimming the red, green, and blue together, but at individual values, such as out-of-phase sine waves). |
| 2 | **Handling Serial Commands:** The color of the LED is controlled by sending serial commands. The commands shall be structured in the following way: a letter followed by a number. The letter will be ‘r’, ‘g’, or ‘b’, and the number will be an integer in range [0, 255]. The letters indicate which channel of the LED is changing (red, green, and blue respectively), and the number indicates the desired brightness of that channel. For example: “r255” means that the red LED channel shall be set to full brightness; “g0” means the green LED channel shall be set to off, and “b128” means the blue LED channel shall be set to half brightness. Writing commands to a channel shall not override the state of other channels. For example, if “r255” was the previous command and “b255” is the next command entered, then the LED shall turn “purple” and not only blue (i.e., r255 and b255 are both ON). Don’t worry about nonlinearity in the LED brightness levels. A newline shall be present after each command. If a “garbage” command is received (something other than the above specified format), just discard the line. The key here is that you are reading serial inputs and writing to an analog output.   * 1. Bonus: Accept more than one command on a line (separated by spaces). For example, to set all the LED channels to off at the same time, the command “r0 g0 b0” can be sent.   2. Bonus: Create other possible commands, such as an automated sequence fading/flashing different colors. Document these in your code. |

### STEP #3: System Verification

### The TA shall run and test your code to ensure that the system is built and programmed as specified by running your code on the TA’s separate test circuit. All states must be working as specified to receive full credit.

### Students should return the resistors, RGB LED, and potentiometer at the end of this project. Students may keep the Arduino UNO, USB cable, breadboard, and the supplied pushbuttons for the upcoming Sensors & Motor lab.

The test rig in Figure 2 below is a depiction of what the TAs shall use for testing the students’ implementations. Only the test rig shall be used.

### IMG_1013.JPGFigure 2.  TA test rig for software implementation.

### Deliverables

### System operation tested and verified by the TAs by the specified due date. See TA hours on Blackboard.

### Submit your code on Canvas as a single file by the specified due date. Name your file as follows: Team[Letter]\_[andrewid]\_TaskN. If your code is contained in one Arduino file, submit your file with the “.ino” extension. If you have multiple files, submit a zip file instead.

### References:

Arduino Tutorials: <http://arduino.cc/en/Tutorial/HomePage>

Arduino language reference: <http://arduino.cc/en/Reference/HomePage>

Finite State Machine tutorial: <http://www.mathertel.de/Arduino/FiniteStateMachine.aspx>