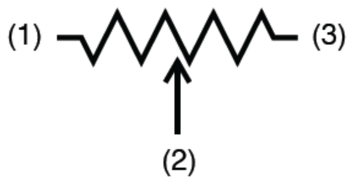
Arduino Lab 1 - The Voltage Divider

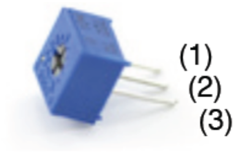
1. Introduction

In this lab, we will endanger a cute animal, create a portal to another dimension, and invent a new genre of music. Along the way, we will learn about breadboarding, voltage dividers, photoresistors, and using the TOMYUM SynthScope with MATLAB.

2. Some Basic Circuit Theory

Circuits are generally designed to accomplish some sort of fixed task, such as blinking a light or amplifying a sound signal. Often, however, you will want to add user input to your circuit, so that you can change its behavior on the fly (for example, setting the blink rate or adjusting volume). A common way of doing this is to use a *potentiometer*. A potentiometer looks like this:



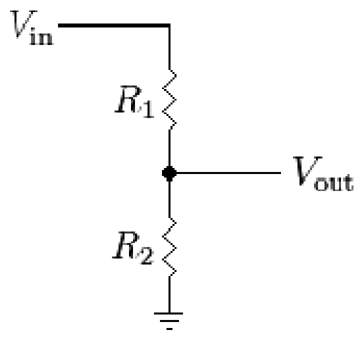


A potentiometer is a three-terminal device. The resistance between the two outermost terminals, $R_{1\text{-}3}$, is constant (this constant value is in fact written on the side of the potentiometer), but the center terminal (2) acts as a “wiper” that slides back and forth along the resistor as you rotate the potentiometer. Therefore, $R_{1\text{-}2}$ and $R_{2\text{-}3}$ change, but their sum $R_{1\text{-}2} + R_{2\text{-}3} = R_{1\text{-}3}$ is constant. Note that the potentiometer is symmetrical.

In this case, we will be using the potentiometer as an input to a MATLAB program, with the TOMYUM SynthScope as an intermediary. However, the SynthScope— like many devices— measures voltages, not resistance, so we must find a way to convert a change in resistance into a change in voltage.

Enter the voltage divider.

Here is a circuit schematic showing a potentiometer used as a voltage divider:



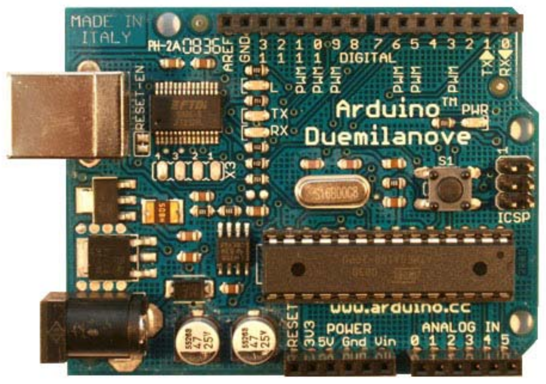
Through simple circuit analysis, it can be shown that:

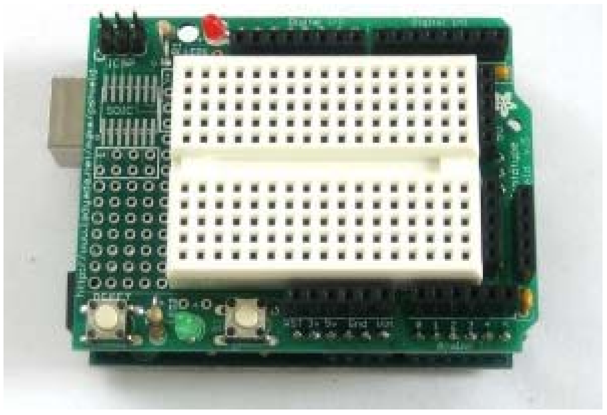
$$V_{out} = \left(\frac{R_{2\text{-}3}}{R_{1\text{-}2}+R_{2\text{-}3}}\right)V_{in} = \left(\frac{R_{2\text{-}3}}{R_{1\text{-}3}}\right)V_{in}$$

Now, since $R_{1\text{-}3}$ is constant, and $R_{2\text{-}3}$ will range from 0 Ω to $R_{1\text{-}3}$ Ω, we can see that V_{out} will vary linearly from 0 V to V_{in} V, proportional to $R_{2\text{-}3}$. In essence, we have created a simple way for the user to control the voltage V_{out} (by turning the potentiometer).

3. Introduction to the Arduino Platform

In front of you is a highly capable micro-controller – the Arduino Duemilanove:





Yours will look like the one on the right, as it has the *protoshield* attached. This is an extension board that extends the pin headers on the main board while also providing a breadboard prototyping surface. You'll find that the board has plenty of connections. In this lab, we will be connecting the board with the USB connection (no external power supply is required) and then using the Analog Inputs and power connections on the board.

PLEASE FOLLOW ALL DIRECTIONS!

If your computer already has ArduinoUploader please skip to “Flashing the Arduino”

The first step in this lab is to download the software required. The necessary files can be found on the **ESE 205 website under the Arduino link**. You will need the compiled firmware file (*main.hex*), the Arduino Uploader folder (*ArduinoUploader*), and the folder containing the MATLAB functions (*synthscope-lab*).

1. Save both the firmware file and the Arduino Uploader folder to the C drive (C:\users).
   a. **NOTE: RCA computers only need *main.hex* if not present in C:\users.**
2. Cut or copy *main.hex* and place it in the *ArduinoUploader* folder.
   a. **NOTE: Not required on RCA computers**
3. Save the MATLAB functions to your S drive (remember this location).
   a. **NOTE: MATLAB Functions are found on 205 website, not BlackBoard.**

-Flashing the Arduino- (Windows XP/Vista)

1. Connecting Arduino to computer
   a. Right-click on “My Computer” on the desktop
   b. Click “*Properties*”
   c. Change the tab to “*Hardware*”
   d. Click “Device Manager”
   e. **Click “*OK*” when error message opens**
   f. Open up *“Ports (COM & LPT*)” by clicking the “+” sign
   g. Knowing which ports are currently displayed, plug USB cable into Arduino and then a free USB socket on computer
   h. The new COM port number is the Arduino board
   i. If this new COM number is greater than 9 (ie, contains 2 digits)
      Example, COM10, COM 24, etc
      i. Right-click on the port
      ii. Select “*Properties*”
      iii. Click the “*Port Settings*” tab
      iv. Click “*Advanced*”
      v. Change “*COM Port Number*” to a lower, single digit port that is open (not present In the list)
         Example: if COM12, change to COM5
      vi. Click “*OK*”
      vii. Click “OK”
      viii. After this, even though the port number did not change in the list the Arduino will be assigned this new number

j. Make a note of this number

2. Opening command prompt
   a. Click “Start” button
   b. Click “Run”
   c. Type “cmd” for the command prompt
   d. Type in (if not already in C:\users) :

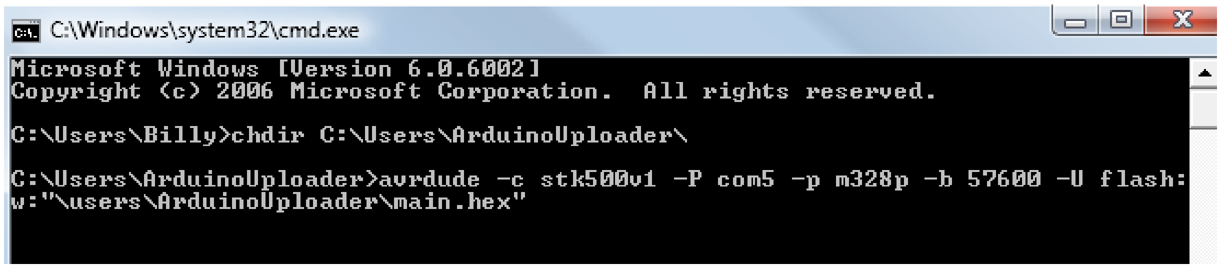
      ```
      chdir C:\users\
      ```

      i. If installed on home computer, use directory where Arduino Uploader is located
   e. In the command prompt, type the following with a single space in between:

`avrdude`	(the program to flash the board)
`-c stk500v1`	(the programmer used)
`-P comX`	(the COM port – replace X)
`-p m328p`	(the microcontroller model)
`-b 57600`	(the controller baud rate)
`-U flash:w: main.hex`	

(flash the board by writing the file to the chip)

This should look like the following:



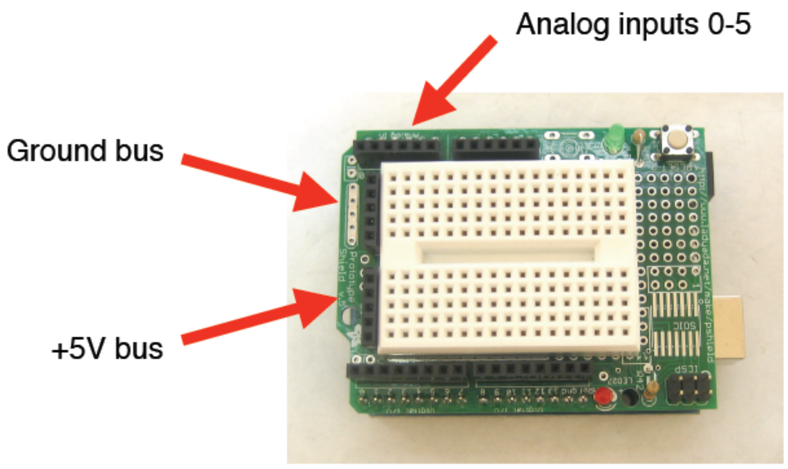
```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\Billy>chdir C:\Users\ArduinoUploader\

C:\Users\ArduinoUploader>avrdude -c stk500v1 -P com5 -p m328p -b 57600 -U flash:
w:"\users\ArduinoUploader\main.hex"
```

(Remember: replace X with your COM port –if Arduino is on COM port 8, use “com8”)

4. Moving from Hardware to Software

Let's try building this voltage divider circuit and measuring V_{out} from MATLAB.

Like many digital devices, the maximum voltage that the SynthScope can safely be connected to is 5V. For that reason, let's use the 5V power bus on the SynthScope as our V_{in}. To measure the output voltage V_{out}, it must be connected to one of the analog inputs.



Using a potentiometer and some spare wires, construct the voltage divider circuit from the previous page on your SynthScope's breadboard.

Once you've set up your circuit on the breadboard, you can connect your SynthScope to the computer with a USB cable and begin reading voltages into MATLAB. You will need to download the SynthScope MATLAB files for this lab from BlackBoard, and change your working directory in MATLAB to be this downloaded folder.

Once you've set up MATLAB, try running the following commands. Replace **CHANNEL** with the number of the analog input that you connected V_{out} to.

```
s = SynthScope();
```

If this doesn't work, try : `s=SynthScope( 'COMX' );`

```
voltage = readAnalog(s, CHANNEL)
```

The readAnalog command tells MATLAB to measure the voltage at the SynthScope's input pin. To view visually how this value changes over time, try running the following command:

```
showScope(s, CHANNEL)
```

Notice how the value changes as you move the potentiometer. Is this what you expected? What happens if you flip the potentiometer around?

5. Practical Example #1: Rotating a Marmoset

Here, on the left, we have included for you an ordinary marmoset.

Problem is, he's been sitting in this lab manual for as long as he can remember and has really become quite bored. Notice his concerned expression and unnatural tenseness.

Let's use what we've learned so far to entertain him.

We will now use the voltage output from the potentiometer to control the behavior of a software application, in effect creating a new input device for the computer. With the same circuit setup as before, try running the program in `marmoset.m` and rotating the potentiometer around.

Now, you may find that rotating your marmoset is more fun than you could ever have anticipated. Just be careful not to rotate him too far, lest he slip off and injure himself.

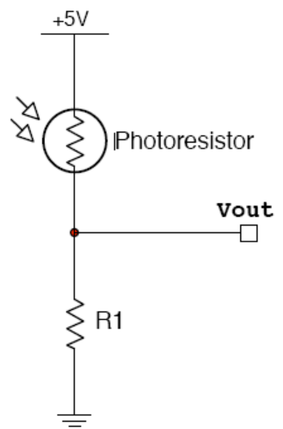
What does each command in the above program do? Can you add to the program to make MATLAB display a warning message when the marmoset reaches a dangerous incline? (Hint: try using the `msgbox` command)

What other things could you control with the potentiometer?

On second thought, the marmoset actually doesn't seem too happy with all this moving around. Can you construct a voltage divider using fixed resistors such that he will constantly remain at a comfortable 30 degrees?

6. Practical Example #2: A Dimension-Piercing Flashlight

Voltage dividers are also very useful when using *photoresistors*, a special type of resistor that changes resistance according to the amount of light that hits it. The more illuminated the photoresistor is, the lower its resistance; the less light hits it, the higher its resistance. In order to sense light intensity, we again wish to convert this change in resistance into a change in voltage, so we will use a voltage divider as shown in the following diagram:

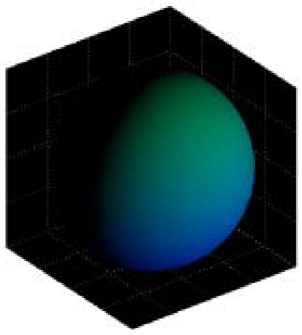


Using the circuit above, construct two separate light sensors (one facing leftwards and one facing rightwards) on your breadboard and connect the output voltage of each to an analog input on the SynthScope. For R1, choose any value of resistor (100Ω to 10kΩ) for now; we will fine-tune it momentarily.

Use the `analogRead` command again to see what voltages are being output by your voltage dividers in ambient light. Since we will want to sense both increases and decreases in light, with the maximum possible dynamic range, we ideally would like the ambient light voltage to be 2.5 V (why?). Try calculating a more appropriate value for R1 to satisfy this design goal, and replace the resistor in your circuit.

Now try using the showScope command again to see how these voltages change as you play around with a flashlight:

```
showScope(s, [CHANNEL1, CHANNEL2]);
```



Try running the `flashlight.m` example program to see an example of how to use these two light sensors in tandem to locate the source of a light (you will need to change **CHANNEL1** and **CHANNEL2** in the program to match your circuit setup). Can you figure out how the program is working? For a brief non-educational distraction, try changing the `colormap` command's argument. (Hint: type `help graph3d` into the MATLAB prompt to find a list of colormaps.)

7. Design Challenge: Synthesizer

Try using what you have learned so far to make your own new kind of musical instrument, using the potentiometer and/or light sensors. To generate sound, connect a buzzer to the Digital Out 9 pin, and use the following two commands:

`writeWave(s, 'SINE')` will start sound output; you only need to run this once

`setWaveFreq(s, FREQUENCY)` will change the frequency of the sound output

Remember that frequencies around 3kHz sound best with the buzzer we are using.

Go for it!