PH 250

Intermediate Physics Laboratory for Physics and Physical Science Teaching Majors

R Todd Lines
David Oliphant

June 7, 2023

Contents

Preface

Experimental physics is the art of testing our physical models to see if they really work. Thousands of measuring devices have been designed and built to make the detailed measurements needed to perform this testing. We would like to interface these measuring devices to computers so data collection is all digital. Then the data can be analyzed, and displayed on our computers. To do this we need to understand computer interfacing.

Computer interfacing is a bit of engineering. We need to either design and build, or purchase instruments, and then get data from those instruments into a computer. But even though it is an engineering task, it is a necessary skill for experimental physicists. It is this skill that we wish to take on in this laboratory class. Of course we can't learn all that there is to know on computer interfacing in one semester. But we can make a good start.

We will use the open source Arduino microcontroler board as our computer interface and we will use Python and the Arduino C++ languages to write controlling software.

Students should leave this class with confidence that they can build a simple computer interface that will read in sensor data and save it on a computer.

Students in this class will also spend time investigating physical models from introductory electricity and magnetism theory.

Instrumentation is sometimes difficult, sometimes frustrating, but also a lot of fun. I hope you will find these lab experiences both informative and entertaining.

I would like to acknowledge Tyler Miller who has spent countless hours testing the codes and writing the Mac versions and helping to improve the text.

Introduction

As a PH250 student, you are probably taking PH220 concurrently with PH250. This lab course is designed to teach electronics and computer skills while your PH220 professor teaches electromagnetic field theory. Once you have a little bit of electric field theory under your belt from PH220, then our experiments designed to test out models of electric charge and electromagnetic fields begin in earnest. While we are waiting we will spend some time learning about how to control experiments with a computer, and how to import data from an experiment to a computer.

You should read the material for each lab before the lab begins. There will sometimes be practice problems to do to make sure you will be effective in lab. By preparing before lab you will have the full 2 hours and 45 minutes to make sure you can finish the lab work. Some labs may go fast, but most take the entire lab period. I also suggest you practice your computer and electronics skills a little. Build some blinking lights for your apartment, or measure how loud your roommates are, or something. The Arduino can be the data collection and control part of thousands fun projects.

This class is sometimes frustrating. But it is also a lot of fun. You will be introduced to computer instrumentation and will be able to perform an experiment that you and your lab group design. The student designed experiments are only limited by your imagination and our ability to find equipment. If you have concerns during the semester, don't hesitate to find your instructor or TA and ask.

Part I

Computer Interfacing

Chapter 1

Introduction to the Arduino and Computer Control

In PH250 we have a goal of getting our computers to talk to our physics experiments (see the course syllabus). This comes in two forms. One is to have the computer control the experiment. For example, we could have the computer turn on our apparatus, or turn it off. The second form is having the experimental device provide data to the computer that we can later analyze. In both cases, we need a piece of electronics that goes between the experimental measuring devices and the computer. These electronic pieces are often called Data Acquisition (DAQ) boards. There are many different forms of DAQ's. They can cost anywhere from a few dollars to many thousands of dollars.

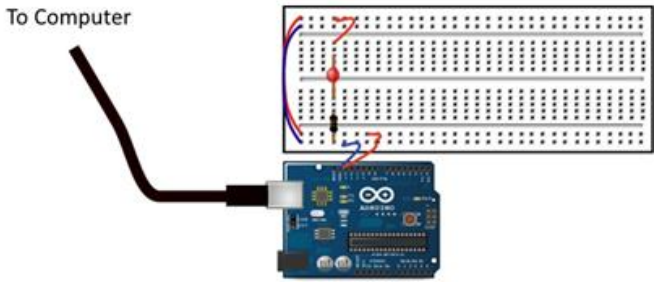
We will use a DAQ developed for learning. It is low cost because it's developers placed the plans for it in the open-source world so that no one would get royalties (including themselves) for the design. Thus board manufacturer's pay less, so we pay less. It is low cost, but not low quality. They named their DAQ "Arduino."

The Arduino is fragile, like all DAQ's. We will need to be careful with our Arduino's. They will be destroyed by putting too large a voltage or electrical current (things you will study in PH220 if you haven't already) into the input pins. They also can be destroyed by being dropped or squashed, so some caution is warranted.

Let's start our study of computer control by controlling something simple with our Arduino DAQ. Earlier we though about turning an apparatus on or off. We can practice doing this with any device. While it might be exciting to try this with a nuclear reactor, it will be easier (and safer!) to start off simple. Let's turn on and off Light Emitting Diode (LED) lights.

1.1 First Computer Control: LED blink (Step-by-step)

Your Super Starter Kit has a set of light emitting diodes (LEDs), a set of resistors, some wires, and a prototyping board. This last item helps hold the other things in place so we can make electricity go through them. Here is what we will build for our light blinking experience:



You are probably familiar with words like "voltage" and "current" even if you have not yet studied these in PH220. You have been plugging in things for many years now. And so you know that "voltage" has something to do with electrical energy. That is enough for now. Our voltage source is just a source of energy to make our circuits work.

A resistor is kind of what it sounds like. It tries to stop or slow down the electricity in the circuit. The LED is just a circuit element that lights up when electricity goes through it. You see them on computers and cell phones as indicators that something has happened.

Let's assemble our system one step at a time.

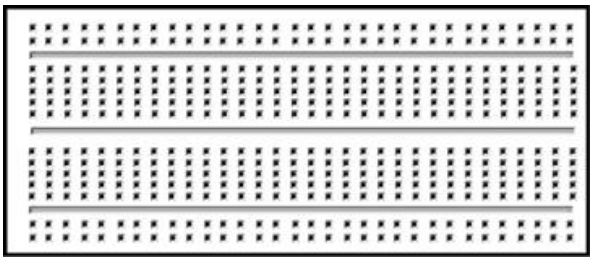
1.1.1 Prototyping boards

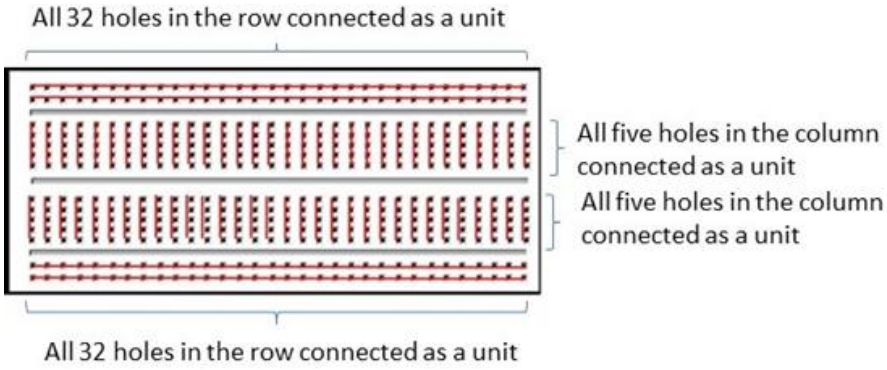
Let's start with your prototyping board.

You might notice that the LED and the resistor in the diagram seems to be stuck in some strange board full of holes. This is our prototyping board. You should have one in your Arduino kit. Your instuctor will have several that you can borrow if you need another one.

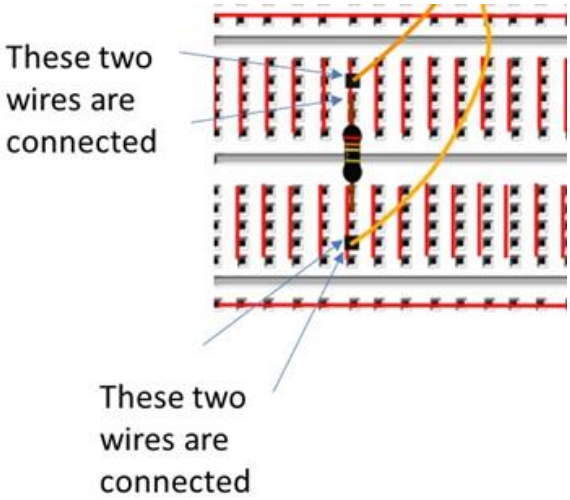
We will connect a lot of electrical wires together in this class. We have devices to make this job easier. Our porototyping board is one of these. They are officially called prototyping boards, but you might hear them called proto-boards, or even "breadboards." They are designed to allow you to put an electronic element on the board and then connect other things to that element. The boards look like this.

Notice that in the center of the board there are sets of five holes. Under the board surface, these holes are connected together as shown by the red lines in the next figure.



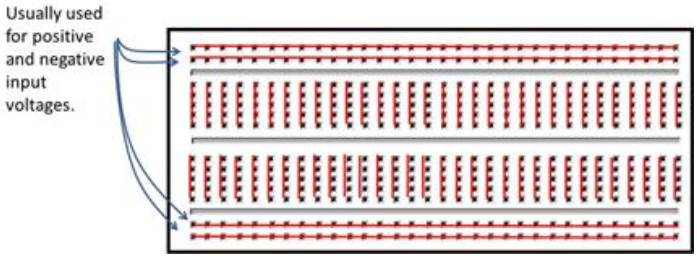


Any wire that goes in one of the connected holes is therefore connected to any other wire that goes in the set of five connected holes. So you can connect up to four other wires to the first wire. In the next figure, you can see an example of a resistor that is placed on the proto-board and is connected to two yellow wires.



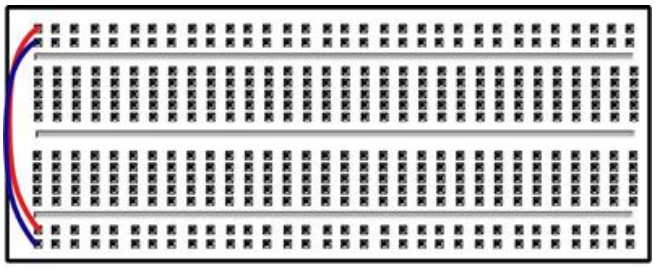
Notice that each end of the resistor is connected to a wire by placing the end of the resistor in one of a set of five connected holes and by placing a wire in another of the set of five connected holes. This would be a silly circuit to build. There is no connection to a source of electrical energy. But this gives the idea of how the prototyping board works.

There are also two long rows of holes on the side of the proto-board. These are usually all connected along the whole row.



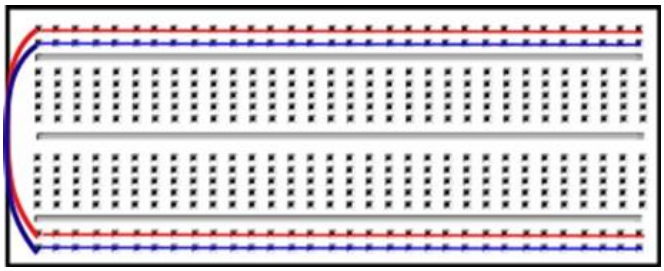
Not all proto-boards are alike when it comes to these long rows. But most are connected along the whole row. These long rows are often used as a convenient way to make input power available all along the whole board. Of course, once you know how the board is wired underneath, you can use the connections any way that is consistant with that design. We could skip using proto-boards and just connect everything with wires and alligator clips. But proto-boards often make holding everything in place much easier.

In today's lab experience, let's start by using the long rows as a place for electrical energy to be used. We have long rows on the top and bottom of the proto-board. For our first experience, we will use just 0V and +5V. Let's make these voltages available on both the top and the bottom of the board by wiring the two sets of rows together.



Looking at our wiring diagram for the inside of the proto-board we can see that how the top row and the second from the bottom row have been connected with a wire (it is red if you are reading this in color) and the second from the top and the bottom row are also wired together (the wire is blue if you are seeing this in color). And we know that these entire rows are wired underneath.

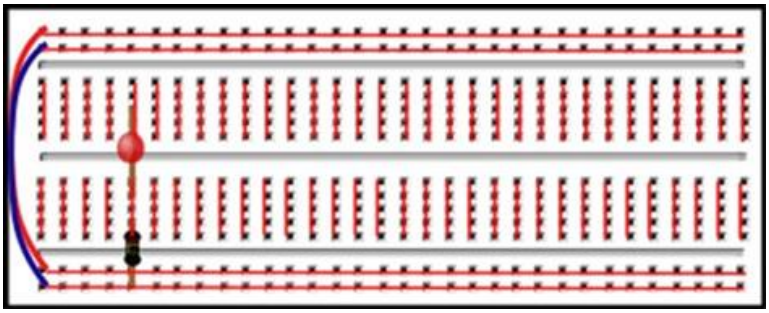
Some proto-boards even have red and blue lines to indicate that these rows can be used for electrical power. Some of ours do.



Next let's add our LED light and our resistor.

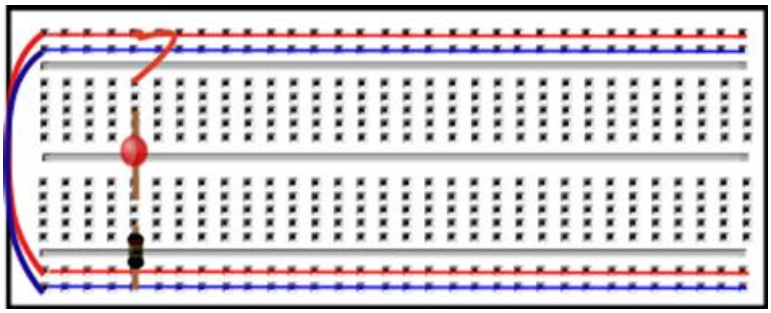
Notice that the LED spans the gap between the top and the bottom of the board.

In the next figure I have included our red connection lines so we can keep in mind which parts of the proto-board are connected underneath.

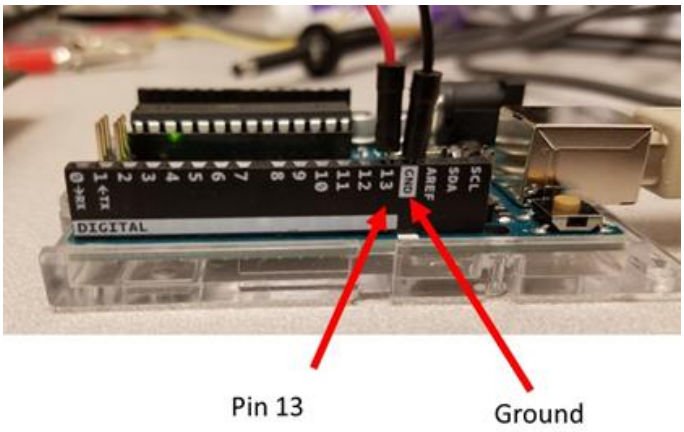


We placed one LED light wire in one column. On the other side of the board we place the other wire. The wires that come out of an electronic device are often called "leads" and I will use this term. So one LED lead goes in a group of five holes on one side of the board and the other lead goes into a group of five hole on the other side of the board. If you look closely, you will see that one side of your LED is flat. The lead on the flat side should be toward the bottom (toward our 0V connection that we will make). LED lights only work one direction. So when using LED's, if they don't work, turn them around.

Notice that the way we have done this one LED lead and one resistor lead are connected in one group of five holes. The resistor is connected to the very bottom row. The top of the LED is not connected to anything yet. It is in a set of five holes that are connected together, but we need another wire to connect the top of the LED to the +5V. Let's choose the top of our power rows to be +5V and add the wire to connect the LED.

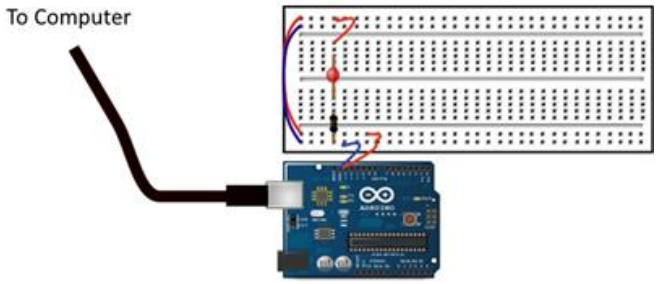


Now we can wire our light to our Arduino. This will take two wires. One should be wired to pin13. The pin numbers are given on the side of the wiring areas.



This pin 13 is a digital output. That means it will either have 5V or it will have 0V. The 5V will be our "on" value that could turn on an instrument. In our case it will turn on our LED light. The 0V is the "off" value. When pin 13 has a value of 0V our light will go off. We know that voltage is related to electrical energy. We will study voltage more next week. But for now we need to know that voltage is a comparison. So we need to know "5V compared to what?" We compare to the voltage of the ground. This is literal. Most buildings have a large copper rod pounded into the ground. All of the plugs in the building have one of their three wires connected to this rod. Thus they are all "grounded." This is our reference. Our computer is connected to a plug so it is grounded.

Our Arduino is connected to the computer so it is grounded. And one of our pins is a connection to the ground. It is marked "GND." This will be our 0V. So we need to connect pin 13 and the GND pin to our +5V and our 0V rows on our proto-board. The next figure shows one way to do this.



We now have our "hardware" built for blinking a light. But we need to give instructions to our Arduino that tells it what to do with pin 13 and pin GND so our light will blink. Our Arduino has a small computer on board, and we need a computer program to be uploaded to the Arduino for this small computer to run. We will write and upload that program next.

1.1.2 LED light blink Sketch

You wrote computer programs in Python in PH150. Our Arduino programs are similar. They have loops and mathematical statements. There are some differences as well. Our Arduino programs have special code "objects" for controlling our Arduino. Most Arduino programs are very short. We just need instructions to tell the Arduino what to turn on, turn off, or what data to collect. For today's lab, we just need to address the digital output pins.

Let me introduce the commands we will use. Each digital output has a number. The code defines a variable of type integer (int) and sets it equal to the pin number. That pin can be HIGH or LOW with HIGH = +5V and LOW = 0V. Each digital pin needs to be set up for output or input. This is done with a pinMode statement:

```
pinMode(ledPin, OUTPUT)
```

The variable ledPin is the pin number (for us 13). And the key word " OUTPUT" sets up our pin 13 to turn things on or off.

To set the pin to HIGH use a digitalWrite command

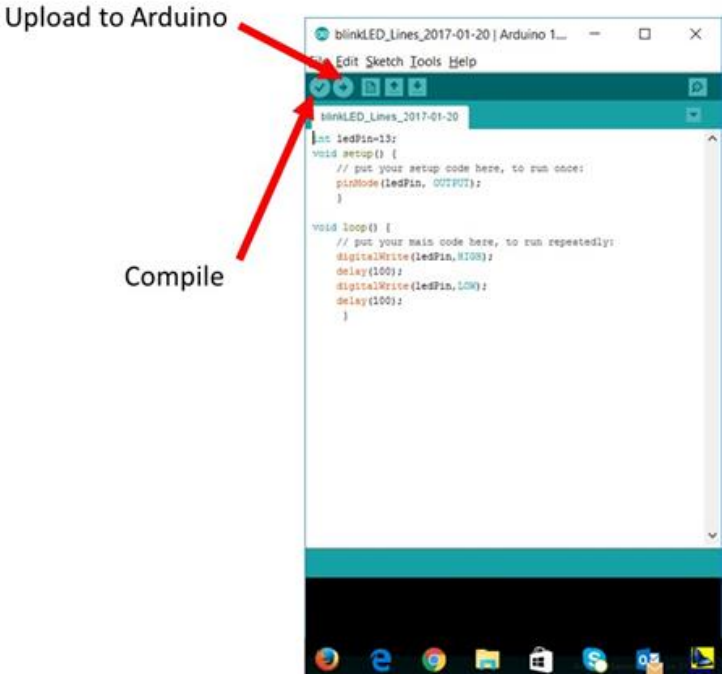
```
digitalWrite(ledPin,HIGH);
```

and finally, to let the light be on for a while, use a delay command where the delay time is given in milliseconds (ms)

```
delay(100);
```

These would not be normal Python commands. They are part of a specific code library for use in making things with Arduino's.

We have a special development environment for programing our Arduino as well. It looks like this.



We will need to install this development environment. To do so we go to https://www.arduino.cc/en/Guide/HomePage and choose to instal the Arduino IDE for your computer. IDE stands for "integrated development environment." This is the environment that you see in the last figure. It has two unusual buttons on its toolbar. One is the "compile" button. When you used Python in PH150, you could run your code by pushing on a green arrow button in most development environments. If you used Jupyter scripts, they also had a function to run your code. But our Arduino is not big enough to be able to run code this way. We need to convert our human-understandable code that looks mostly like Python into a digital format that the Arduino can understand. The compile button does this. As the code is converted, the Arduino software looks for errors in the code. If there are errors, it will tell you at this point. This is different than Python which only told you about errors as the code ran. But remember we don't want to run our program on our computer. We want it to run on our Arduino. So this check is good before we send the translated code to the Arduino.

We also need a way to send our code to our Arduino and that is what the upload to Arduino button does. Once the code is compiled without errors,

connect the USB cable to the Arduino and push the upload button.

There are also some syntax differences between the Arduino computer language and Python. If you have taken CS124 or know the "C" language, you will recognize some of these changes.

- Everything in a function needs to be in curly braces { }
- Indenting is a good idea, but not required
- comments can be started with two slashes //
- every line needs a semicolon at the end;

1.1.3 Arduino LED light blink sketch (program)

We are ready to write code to blink our LED light. Here is an example code. We write this code right in the Arduino development environment main window. You can download it directly here

```
/////////////////////////////////////////////////////////
//Arduino Sketch to blink one LED light
//   Written by Brother Lines
//  (place your name here in your code)
//   Feb 6, 2017
//
//   Define our Arduino Variables
//      We will call pin 13 "ledPin"
/////////////////////////////////////////////////////////
int ledPin=13;

/////////////////////////////////////////////////////////
// Arduino setup function comes next
//    Every Arduino sketch needs a setup function
//    We will set up our ledPin (pin 13) as an output pin
void setup() {
 // put your setup code here, to run once to set up:
 pinMode(ledPin, OUTPUT);
 }

/////////////////////////////////////////////////////////
// Arduino loop function
//    Every Arduino sketch has a loop function
//    This is where we put what we want the Arduino to do
//    The Arduino will do whatever is in the loop function
//    until the Arduino is unplugged.

void loop() {
 // put your main code here, to run repeatedly:
```

```
 // turn on the LED
 digitalWrite(ledPin,HIGH);

 // leave in on for 100ms
 delay(100);

 // turn off the LED
 digitalWrite(ledPin,LOW);

 // leave in off for 100ms
 delay(100);
 }
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
```

Notice that we used a lot of comments. There are only ten lines that are actually required to make the sketch run.

```
int ledPin=13;
void setup() {
   pinMode(ledPin, OUTPUT);
 }
void loop() {
  digitalWrite(ledPin,HIGH);
  delay(100);
  digitalWrite(ledPin,LOW);
  delay(100);
 }
```

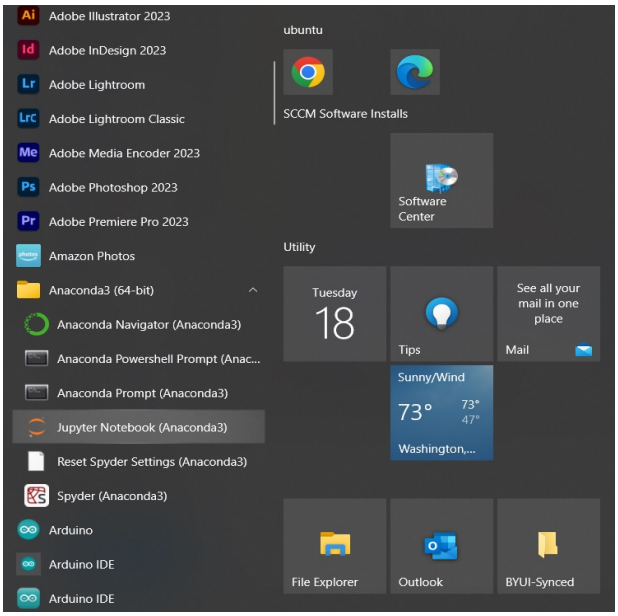
The comments are not required for the sketch to work, but they help us remember what we did later. In our class, **comments are required for full credit**. So don't leave out the comments. In fact, you can add any comments that might help you remember why the code works.

Also notice that Arduino programs are called "sketches." Most of the commands are special Arduino commands. And luckily, they make sense when we read them. Try it out. Type in this sketch including the comments and compile and upload it to your Arduino. If all goes well, the LED light will begin to blink. If all did go well, go on to the next section. If it did not, call over an instructor.

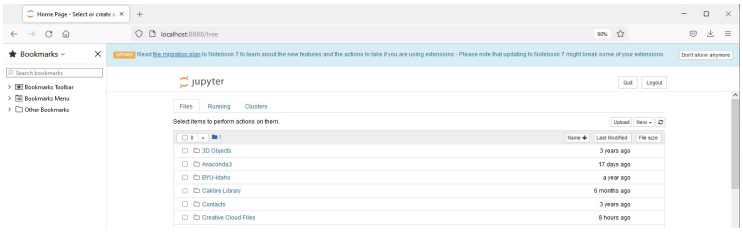
You will want to make a record of what you have done so you can look it up later when you need to repeat what we just did in a future project (like making a blinking light on your instrument so you know it is working!). Scientists record what they do in lab notebooks. Traditionally these are bound blank books that you write in with a pen. But in industry today companies use electronic lab notebooks. You are probably very familiar with Google Docs or other online document sharing tools. We will use such a tool that is widely used in both industry and academia. That is a Jupyter notebook. Note that Jupyter

notebooks on their own are not secure enough to protect intellectual property. They would need to be safeguarded withing a professional lab notebook system. But our homework for this class isn't considered a national security risk or even an intellectual property risk. So Jupyter notebooks are just fine for us. You may have seen them in other classes that use python. If not, don't worry, we can help you become comfortable with them.

Chances are you already have the Jupyter system because it comes standard as part of the Anaconda python system. If you installed Anaconda, you can just go to your list of programs and start Jupyter. If you don't find it, call over your instructor or TA. Here is how that looks in Windows (For those with Macs, it will look more Macish).

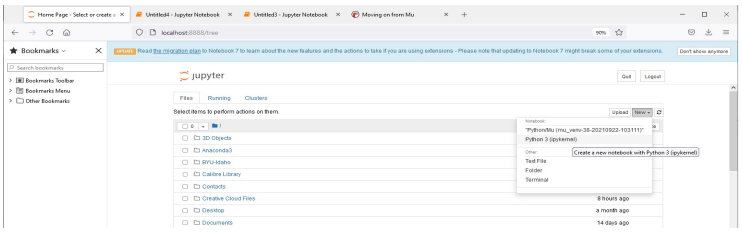


This is a little weird, but Jupyter will start in your web browser!

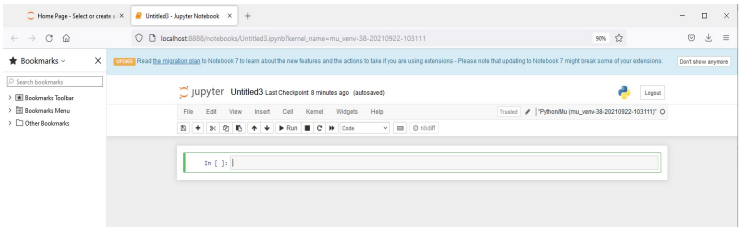


Jupyter notebooks use a browser as the program to display things and to let us input things into the notebook. You open a new notebook by going to the "New" button on the top right and selecting the Python3 option.

You get a new tab in your browser with an input window. The input window



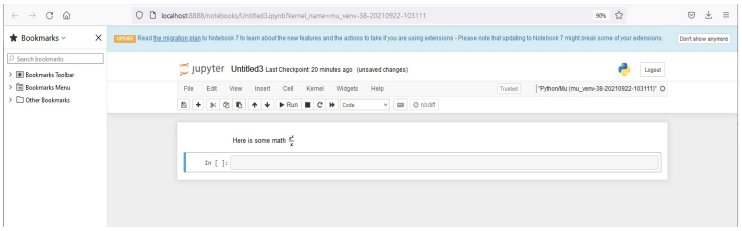
is called a "cell" and you can choose what kind of cell it is by going up to the Cell drop down menu and choosing "Cell Type." Notice that we get the choices "code" and "markdown" among others. Choose the "Markdown" option and you can input text, like a description of what we have been doing so far in this lab!



The "Code" type cells are a good place to put your Arduino code. And you can drop photos into the Markdown type notebook cells as well. You can even type math into the Markdown cells by placing the math symbols between dollar signs. That will take some practice, but the code

```
Here is some math $\frac {e^x}{x}$
```

will become this!



It looks like real math! This will take some getting used to. So start with what we have done so far today. Make sure you have saved sketch. Then write a description of what we have done so far in a Markdown cell and copy your code into a Code cell. Maybe take a photo of your hardware set up and include it in another Markdown cell. We will build on this sketch, so spend some time documenting what you did in your lab notebook so you can refer to it later.

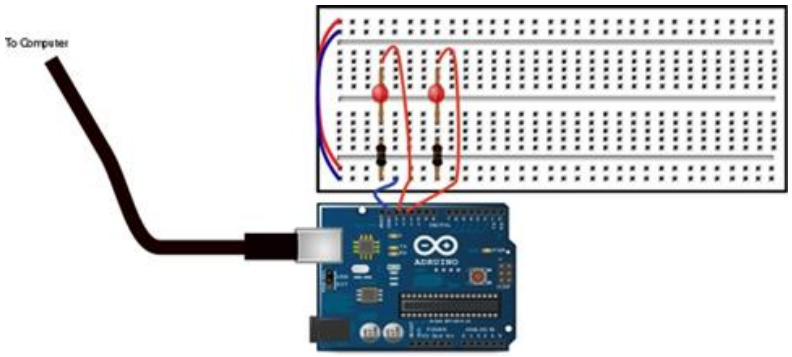
An example lab notebook can be downloaded here for reference.

1.2 Second Computer Control: Two LED blink

Now let's see if we can apply what we have learned. Let's change our hardware so that it has two LED lights. Most of the hardware setup will be the same. But We will wire them to two Arduino pins, say 12 and 13 (along with GND of course) and have them blink, but blink independently.

We will have to abandon our nice +5V top row as our connection to pin 13 because we need two pins, 12 and 13. The two pins must work independently. In fact, let's have one LED on when the other is off.

This is why we can't wire both LED lights to the top row and wire the top row to pin 13 as we did last time. That would make both lights blink at once, but would not let one be off and the other on. Instead, let's wire the top lead of one LED directly to pin 13 of our Arduino, and let's wire the top lead of the other LED to pin 12. The two resistors can share a connection to the GND pin, so let's keep using the 0V bottom row of the proto-board. Your hardware should look something like this.



We will need to modify our sketch (save the old one first so you have a separate copy for reference). Here is a suggestion of what the new sketch might look like.

```
////////////////////////////////////////////////////////
//Arduino Sketch to blink two LED light
//   Written by Brother Lines
//   Feb 6, 2017
//
//   Define our Arduino Variables
//      We will call pin 13 "ledPin1"
//      We will call pin 12 "ledPin2"
////////////////////////////////////////////////////////
int ledPin1=13;
int ledPin2=12;

////////////////////////////////////////////////////////
void setup() {
```

```
 // put your setup code here, to run once:
 // set both pins to OUTPUT
 pinMode(ledPin1, OUTPUT);
 pinMode(ledPin2, OUTPUT);
 }

//////////////////////////////////////////////////////
void loop() {
 // put your main code here, to run repeatedly:
 // turn on one LED and turn off the other
 digitalWrite(ledPin1,HIGH);
 digitalWrite(ledPin2,LOW);
 // wait
 delay(100);
 // now switch so the first LED is off
 // and the second is on
 digitalWrite(ledPin1,LOW);
 digitalWrite(ledPin2,HIGH);
 // wait
 delay(100);
 }
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
```

Again save your sketch and maybe take a photo of your hardware set up. Place both in your lab notebook along with notes on how you got it to work. Make sure others at your table are able to get their setup to work.

1.3 Third Computer Control: Two LED blink using math

Let's leave our hardware alone in the two LED setup from the last section. And let's make the LED's blink the same way. But this time, let's calculate when they should be on or off. Why would we do this? Because sometimes in computer control of experiments we need to turn something on or off based on a calculation. You may have your computer watching to make sure the experiment doesn't get too hot or cold. The Arduino can bring in temperature information, but you would have to write the code to tell it to turn off the heater when your experiments gets to hot and to turn it on when it gets too cold. This could be done with a mathematical comparison. We will use such a comparison in the next sketch.

Suppose we want to know if a number is even or odd. Even numbers are evenly divisible by 2. We could divide a number by 2 and see if the remainder is zero. Our Arduino language has a good set of mathematical functions. The remainder function is a "%" sign. For example

$$\begin{aligned} 3\%2 &= 1 \\ 6\%2 &= 0 \end{aligned}$$

Let's have one light turn on if a number is even, then switch to the other light if the number is odd.

In our code we will introduce a variable, i, that we will increment (add one to) every time the loop runs. So the first time the Arduino loop runs it will be zero (even) and the next time 1 (odd) and the next time 2 (even) and the next time 3 (odd) and so on. If you studied Python you would call such a variable an "integer" and might even know to call it a "loop counter."

In our Arduino sketch we will test to see if i is even in an if-statement. If-statements go like this

```
if (test condition ) {
  do something;
}
else {
  do something else;
}
```

Notice that the parts of the if-statement need curly braces. Our condition to test is

```
i % 2 == 0
```

Note that there are two equals signs. That makes it a test for equality rather than an assignment. So we will have an if-statement like this

```
if (i % 2 == 0 ) {
  digitalWrite(ledPin1,HIGH);
  digitalWrite(ledPin2,LOW);
  delay(1000);
  }
else {
  digitalWrite(ledPin1,LOW);
  digitalWrite(ledPin2,HIGH);
  delay(1000);
  }
```

One last addition, our Arduino language has a shortcut for the statement

```
i=i+1
```

It is simply

```
i++
```

We will use this to make i increase by one each time the loop runs. The whole code might look like this.

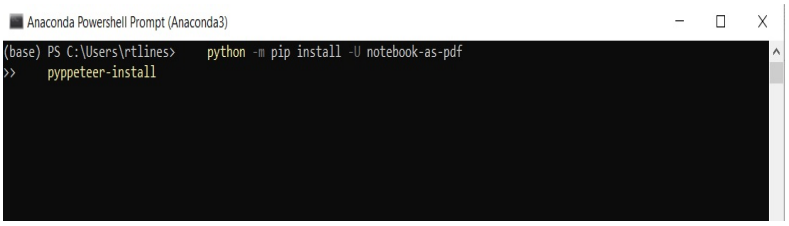
```
////////////////////////////////////////////////////////
// code to blink two LED's using a mathematical expression
// to determine when they should light. Note that the
// Arduino code is closer to C++ than python.
////////////////////////////////////////////////////////
int ledPin1=13;
int ledPin2=12;
int i=0; //loop counter

////////////////////////////////////////////////////////
void setup() {
 // put your setup code here, to run once:
 pinMode(ledPin1, OUTPUT);
 pinMode(ledPin2, OUTPUT);
 }

////////////////////////////////////////////////////////
void loop() {
 // put your main code here, to run repeatedly:
 // blink the LED's with the number of blinks being
 // the Fibonacci sequence.
 if (i % 2 == 0 ) {
    digitalWrite(ledPin1,HIGH);
    digitalWrite(ledPin2,LOW);
    delay(1000);
    }
  else {
    digitalWrite(ledPin1,LOW);
    digitalWrite(ledPin2,HIGH);
    delay(1000);
    }
    i++;   // increment the loop counter by adding 1 to i
 }

////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
```

Again save your sketch. You should probably say in your lab notebook that you used the previous hardware setup. You might want to describe in your lab notebook how the mathematical algorithm works.

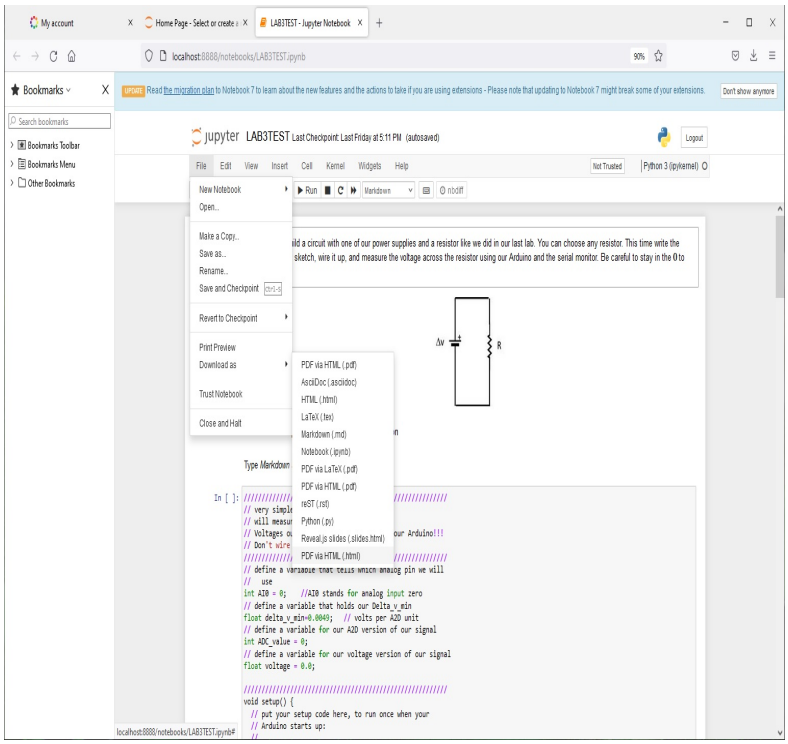
But at this point, you probably want to be able to submit your lab notebook for grading. This seems simple, but we need to modify your Anaconda system to make it work well. To do this, start an Anaconda Powershell Prompt.

This opens a big black box. In the black box type the following lines.

```
python -m pip install -U notebook-as-pdf
pyppeteer-install
```



A lot of words and symbols will go by if you are successful. After they are done you can try to save your Jupyter notebook as a pdf file for submission. From the File drop down menu choose “Download as” and then “PDF via HTML.” You should find that your browser downloaded a PDF version of your notebook into your download directory. Try this before you go on to the next part of the lab.



1.4 Fourth Computer Control: Two LED blink in the Fibonacci sequence

Suppose instead of LED lights we had large radio transmitters. And suppose we were part of the Search for Extra-Terrestrial Intelligence (SETI). We wish to send a message to any intelligent life that they would understand. Intelligent life probably would be able to do mathematics and would understand how mathematics occurs in nature. One sequence of numbers that occurs over and over again in nature was discovered by Fibonacci. Let's blink our LED lights (representing those powerful radio transmitters) in the Fibonacci sequence.

We need to know now to calculate the Fibonacci sequence. One method is to know that the sequence goes like this

$$0, 1, 1, 2, 3, 5, 8 \ldots$$

and that we can find the next number in the sequence by choosing $f_1 = 0$ first, then $f_2 = 1$ then using the formula

$$f(x-1) \; + \; f(x-2)$$

Let's see that this works. For the first of the sequence, we just write the 0. For the second we just write the 1. Then for the third

$$\begin{aligned} f_3 &= f_2 + f_1 \\ &= 1 + 0 \\ &= 1 \end{aligned}$$

So far so good. Let's try the next in the sequence

$$\begin{aligned} f_4 &= f_3 + f_2 \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

Again it worked. For the next one

$$\begin{aligned} f_5 &= f_4 + f_3 \\ &= 2 + 1 \\ &= 3 \end{aligned}$$

and though we won't prove it, it works for every member of the sequence. See if you can figure out how to write this code. An example is given below, but see if you can figure out what the code should be.

This example is a much more complex version of a mathematical based computer control.

```
//////////////////////////////////////////////////////////
// code to blink two LED's using a mathematical expression
// to determine when they should light. Note that the
// Arduino code is closer to C++ than python.
//////////////////////////////////////////////////////////
int ledPin1=13;
int ledPin2=12;
int i=0; //loop counter

//////////////////////////////////////////////////////////
int fib_count=0; // number of blinks based on Fibonacci
int i_max=10; // maximum Fibonacci number before
 // starting over

//////////////////////////////////////////////////////////
void setup() {
 // put your setup code here, to run once:
 pinMode(ledPin1, OUTPUT);
 pinMode(ledPin2, OUTPUT);
 }
```

```
int fib(int x) {
 // calculates the Fibonacci sequence using recursion
 if (x==0)
    return 0;
 if (x==1)
   return 1;
   return fib(x-1) + fib(x-2);
 }

////////////////////////////////////////////////////////
void loop() {
   // put your main code here, to run repeatedly:
   // blink the LED's with the number of blinks being
   // the Fibonacci sequence.
   fib_count=fib(i);
   if (i % 2 == 0 ) {
      // turn off one light
      digitalWrite(ledPin2,LOW);
      // now blink the second light fib_count times
      for (int n=0; n<fib_count; n++) {
         digitalWrite(ledPin1,HIGH);
         delay(100);
         digitalWrite(ledPin1,LOW);
         delay(100);
         }
      }
   else {
      // turn off the other light
      digitalWrite(ledPin1,LOW);
      // now blink the first light fib_count times
      for (int n=0; n<fib_count ; n++) {
         digitalWrite(ledPin2,HIGH);
         delay(100);
         digitalWrite(ledPin2,LOW);
         delay(100);
         }
      }
   // increment i
   i++;
   // limit our blinks to the first i_max Fibonacci numbers
   if (i>i_max) i=0;
   }

////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
```

Again save your sketch. You should probably say in your lab notebook that you used the previous hardware setup. You really should describe in your lab notebook how the mathematical algorithm works.

For next week, you should read the lab before coming to class. So your assignment is to read Lab 2.

Chapter 2

Introduction to Electrical Measuring Devices

(Not Step-by-step, read before class)

Last week we tackled very simple computer control, but we said we also want to transfer data from our experiment to our computer. In order to understand how to do that, we need to know what things we can measure with electronic devices. We will take on this question today, but we will get practice making these measurements with special equipment designed just for making these measurements. These devices won't send the data they measure to our computers, but they will display it so we can see the data.

Once we know how to make some basic measurements with these stand-alone instruments, then we can consider how we would make a new instrument to measure something else. We will build a current measuring device, an *ammeter* out of electrical components and a *voltmeter*. This will be something we do over and over again. We will build a new instrument using instruments we already know, and some electrical equipment.

This lab consists of a large pre-reading section that will give you background information, and then at the end an assignment where I will ask you to practice making these measurements with our equipment. Notice this is different than last week's lab reading. Last week the reading went step by step through the assignment. This week the assignment is at the end and you will have to think through how to do the problems as a group.

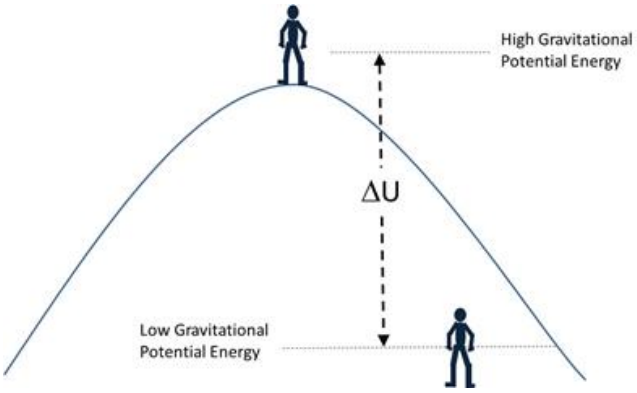
2.1 What we measure: Voltage (and Current)

The two easiest electrical measurements to make are voltage and current measurements. So physicists try to turn all other types of measurements into voltage or current measurements. You may want to measure relative humidity. But to

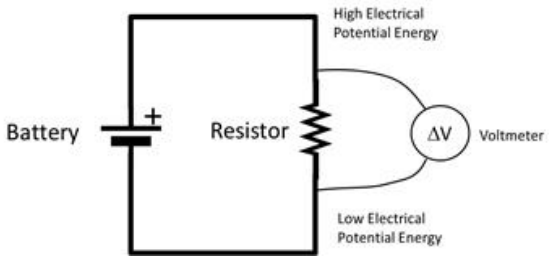
record relative humidity on our computer we need to convert relative humidity into a voltage or current! We will do experiments that do this type of conversion, but first, let's learn about voltage and current so we can see how electronic systems measure them.

Voltage is really "electrical potential difference," which is the difference between electrical potential energy per unit charge at two different circuit locations. Last lab we said voltage was a comparison, and this is the comparison. We compare the potential energy at two different circuit locations, only we divide the potential energy by the charge of an electron.

To get a feel for how this works, think of a change in gravitational potential energy, ΔU_g. If we wanted to measure the difference in potential energy between the top of a hill and the bottom of a hill, we would need to place some sort of device both at the top and at the bottom of the hill.



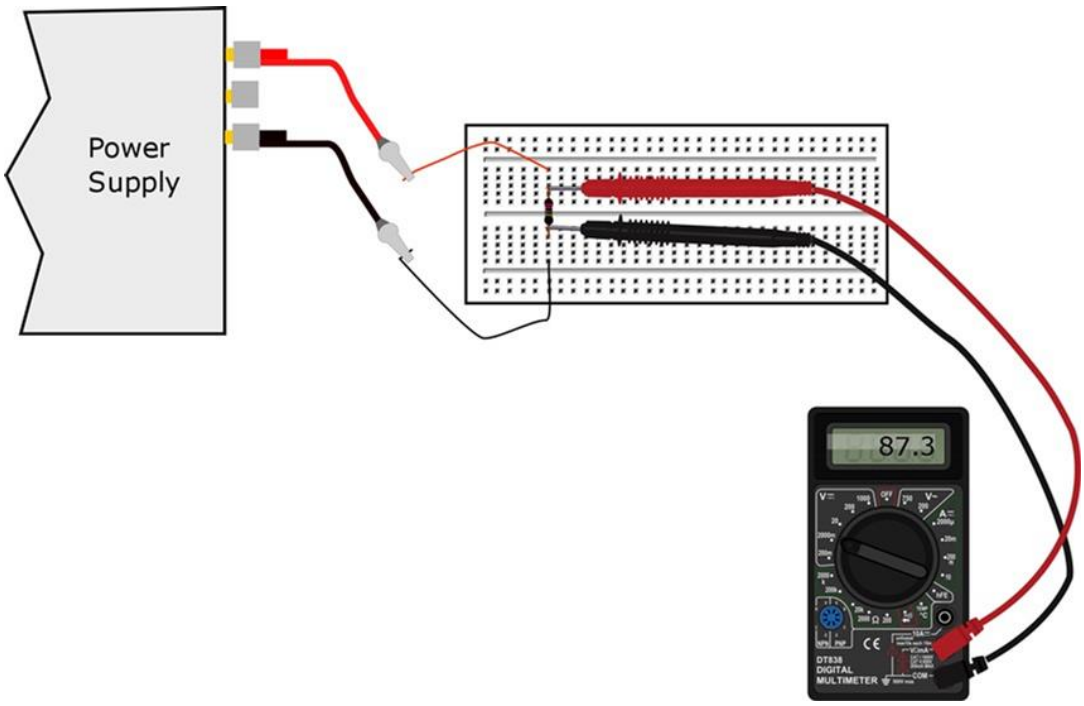
We have to do the same thing in our electrical case. We need two "probes," one placed at the high potential and one placed at the low potential. For example, we could have the circuit that you see in the next figure.



The positive end of the battery is like the top of the hill. It provides a high electrical potential energy. So we put one probe at the top of the "hill" or the plus side of the battery, and the other on the bottom of the "hill" or minus side of the battery. The negative side of the battery provides a low electrical potential energy. With this we measure how high our potential "hill" is. The

difference between these two measurements is called *voltage.* You should ask yourself "what would happen if you got the probes backward?"

In the next figure you can see how to actually perform this voltage measurement with one of our meters.



We say we measure voltage "across" a circuit element. This makes some sense if you consider that we very seldom stand batteries up so their electric potential is greater in the same direction as their gravitational potential. Batteries, resisters, capacitors, etc, often lie down, and we measure "across" them by putting the positive probe on the high potential side and the negative probe on the low potential side. Even though the battery is lying down, we are still measuring a higher and lower potential energy. Knowing a little about voltage, let's look at our devices that produce voltages and then the devices that measure voltages.

2.2 Stand-Alone Experimental Hardware

In today's lab we will study four hardware devices. It might be better to skim this part before lab, then read it in detail with your lab group in class when you have the equipment in front of you. **But do read these sections!** You can damage your equipment if you don't know how the equipment works. So read these sections as you work! And you need to read the sections about designing

new measuring devices below (starting in section 2.4), so skim the equipment sections, but not the rest!

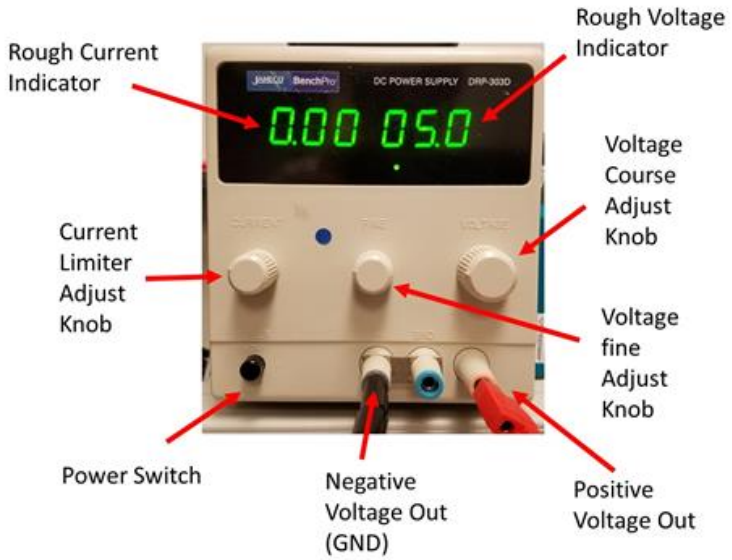
Here is a list of the devices we will study in this time's lab:

1. a power supply
2. a signal generator
3. a multimeter
4. an oscilloscope.

We will call these "stand alone" instruments because they are independent boxes that do their job of measuring or generating signals without a computer connected to them. The power supply and the signal generator make voltage signals. The other two devices measure them. Let's look at the power supply and signal generator first, then take on the measuring devices.

2.2.1 Power Supply

A power supply is like an adjustable battery. Batteries have fixed voltages. But a power supply may have an adjustable voltage.

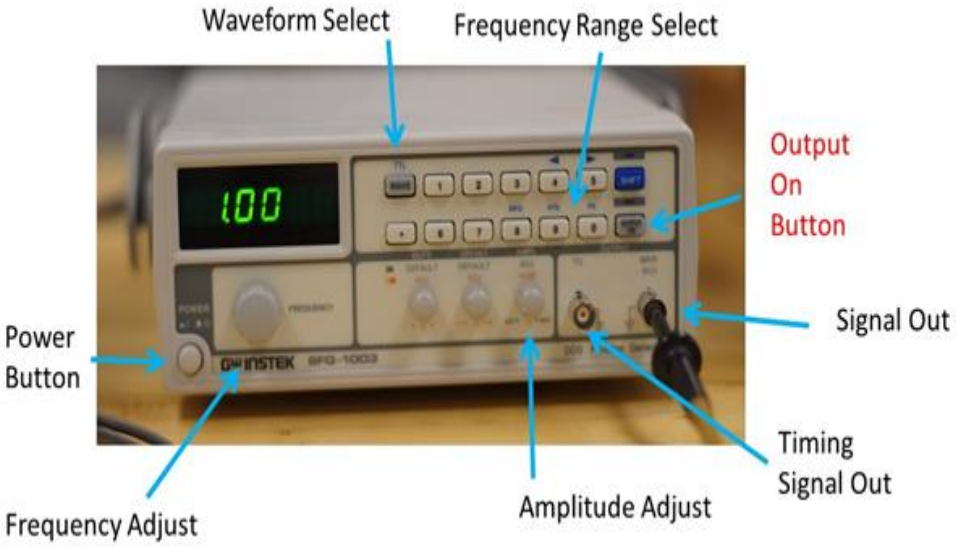


Usually a power supply takes electrical energy from the wall outlets and converts that energy into the specific voltage that we want for our experiment. So it is like a battery, but must be plugged into the wall. Our power supplies are designed to keep us safe. They are current limited, meaning that they try not to give too much charge flowing through our wires. Sometimes this is a

problem because they are too limited. There is a current limiting knob that you can turn to allow a little more current. **Be careful when you use this.** The voltage may jump wildly when you turn the current knob! It is best to turn all the knobs down as low as they will go before you turn on the power supply. Then, after turning on the power supply, increase the current knob about half a turn, and then slowly turn the voltage knob up to your desired voltage. If the voltage stops increasing, turn the voltage knob back down a bit, and turn up your current limiter knob some more. Then try your voltage knob again.

Some of our electrical devices are quite delicate, and will literally burn up if you apply too much current or voltage. In today's lab, we will practice using our power supply so we are prepared when the delicate components come out later.

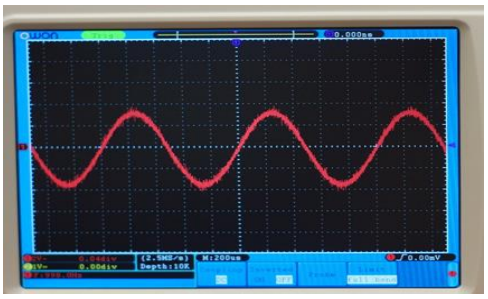
2.2.2 Signal Generator



The signal generator is a fancy power supply. It makes changing voltages. It can make voltages in sine, square, and triangle patterns. These time-varying signals have a maximum voltage (called the amplitude) . We will use both the wave output and a timing signal that the wave generator creates. Each has their own Bayonet Neill–Concelman connector (usually just called a BNC connector) on the front of the signal generator. You will need a cable with BNC connectors on one end (and maybe alligator clips on the other end) to use this device. There is an amplitude knob on the front of the signal generator. Because the signal generator makes a voltage that changes in time, the amplitude of the signal must be in voltage units. We should be careful not to set the signal amplitude (voltage) too high or we run the risk of destroying our measuring devices. Again

turn the amplitude (voltage) down before you connect the box to our electrical components. Then turn up the voltage to what you want in a safe way.

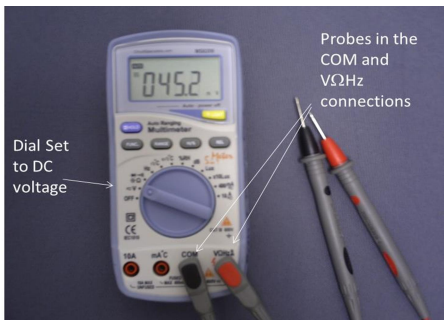
There are frequency range buttons (using the shift button) near the middle of the device panel. To change the frequency, you use the shift and range buttons to set which digit you are adjusting, then turn the frequency knob to make the change. An annoying feature of our frequency generators is that you must push the "output on" button or they don't output a signal. When everything is set up right, we get a sine wave (or square wave, or triangle wave) out. Here is a signal from the signal generator displayed on one of our measuring devices, the oscilloscope.



Of course, simple batteries are sources of voltage, and so are many other things.

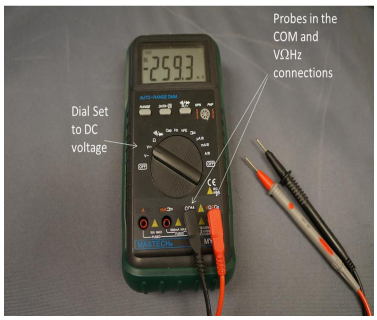
2.2.3 Voltmeter

Our first measurement device is voltmeter. It measures the electric potential difference (voltage) between its two leads (sometimes called "probes"). In the next figure you will see a picture of one of our multi-meters set to measure voltage.



The display is set to read voltage by turning the dial to the V position. There are often two voltage settings. The one that has a wavy line next to it is alternating voltage. The one that has a straight line with three dots under it is the direct current (DC) voltage. These words might not mean much to

you yet if you are just starting PH220. So for now, we will just use the DC voltage setting. As you learn more, we may use the alternating voltage setting. The leads (probes) should be connected to the COM (common) and VΩHz connectors. Note that connecting your probe leads to the wrong position can blow the fuse (or worse) in your meter and make subsequent readings very wrong. You should make sure you don't do this, and watch to make sure someone else has not done this before you. If the meter seems crazy, it just may be. We have several different voltmeter models. Here is a picture of a different model.

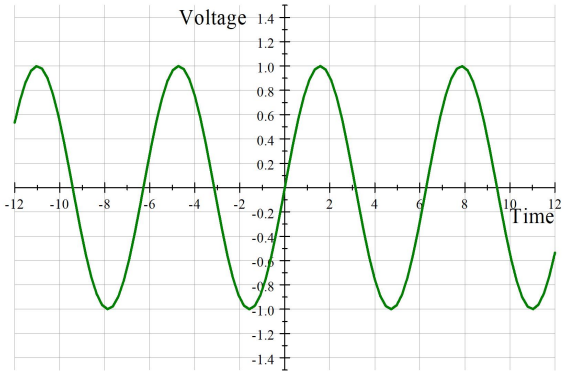


You will notice that there are other settings besides volts. Our meters are all multi-meters. That means that they can measure more than one thing. We will use several of the settings throughout the semester.

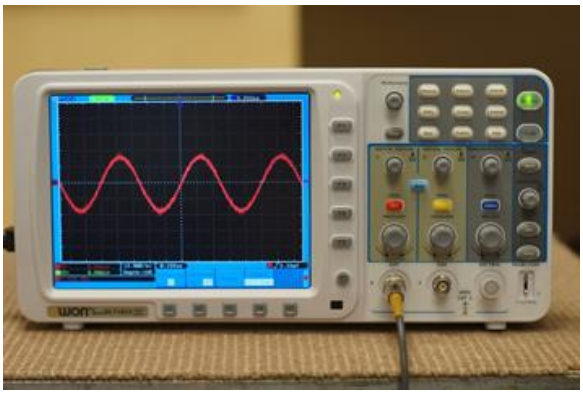
2.2.4 Oscilloscope

Our next device, the oscilloscope, is just a fancy voltmeter. Unlike the multimeter, it usually just measures voltage. But it does it with flare!

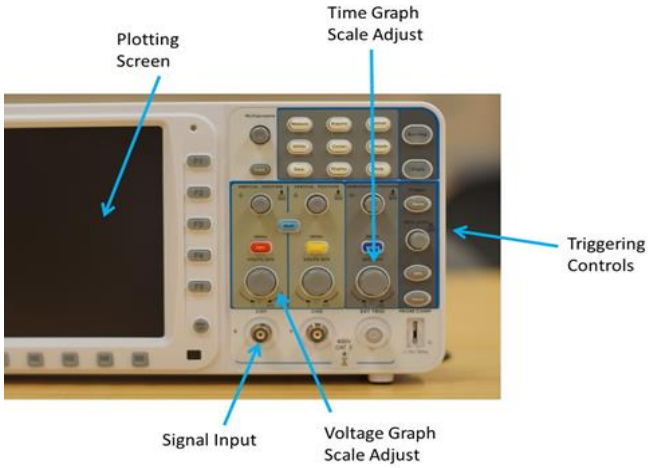
The oscilloscope can measure changing voltages very accurately and usually has a way to graph the changing voltage. The standard is a voltage vs. time graph. A sinusoidally varying voltage should look something like what you see in the next figure when plotted.



And that is what our oscilloscope does. We should see something like this on the oscilloscope screen. From our discussion of the signal generator, you know that this is just what we see.

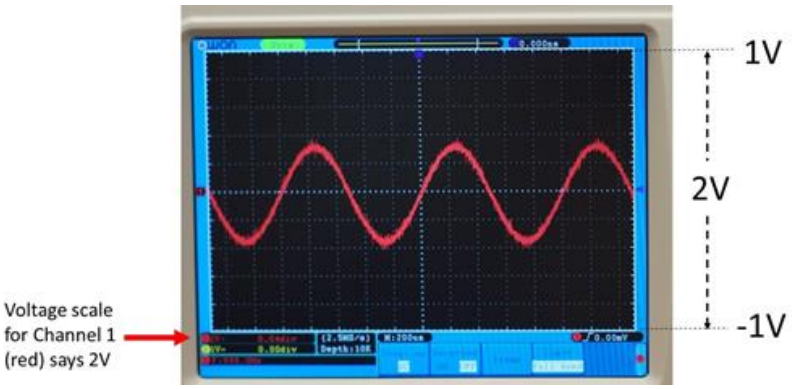


If the changing voltage is periodic, the oscilloscope has a way to use this fact to stabilize the graph so you can see the details more clearly. This stabilization is called "triggering" and on our oscilloscopes there are buttons and knobs on the right hand side of the oscilloscope that adjust the triggering to make the graph more stable (or less stable). The photograph of the sine wave above was taken by stabilizing a sine wave from our signal generator. The oscilloscope starts plotting at the same part of the wave each time, so the periodic signal seems to stand still. To do this we must "trigger" the graph at some good starting point. Our oscilloscopes have a build-in circuit that can watch for the same part of a signal and start the graph in the same place each time. One of the knobs adjusts the trigger point.

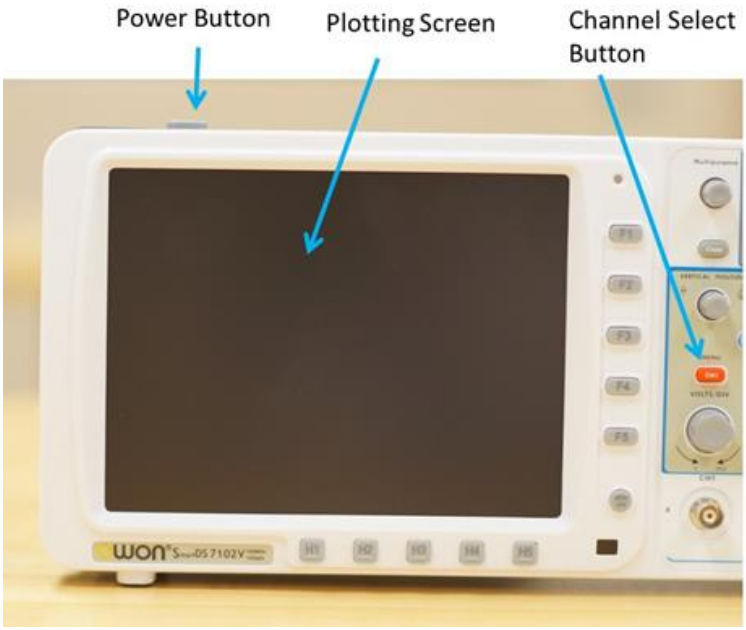


The other knobs adjust the horizontal and vertical axes. The vertical axis

is voltage, and the voltage axis control is next to the signal input toward the bottom middle of the front panel. To the right of this is the horizontal axis control, which is time. You can choose how many volts per division with one knob and how many seconds (or fractions of seconds) per division you have on your graph with another knob. In the next figure you can see a signal on the oscilloscope screen.



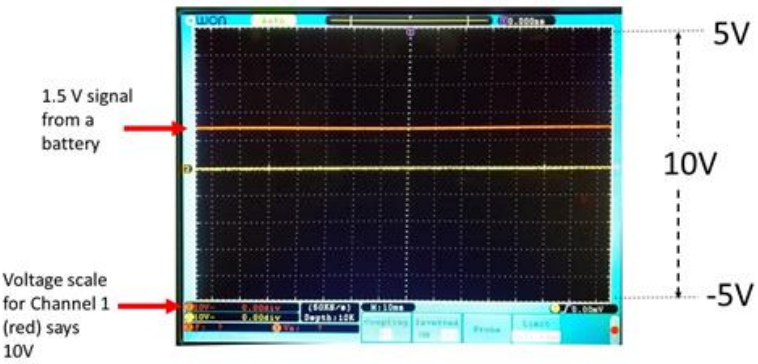
In the bottom left-hand corner there is a red dot and a voltage given. This is the voltage displayed across the whole screen. Since when this photo was take the voltage knob was set to 2V, this means that the bottom of the screen represents -1V and the top of the screen represents $+1$V.



There are two signal inputs because our oscilloscopes can look at two different voltage signals at the same time. Each signal input is called a "channel." Each channel has it's own voltage scale knob and voltage scale indicator in the bottom left-hand corner. They share the same time scale.

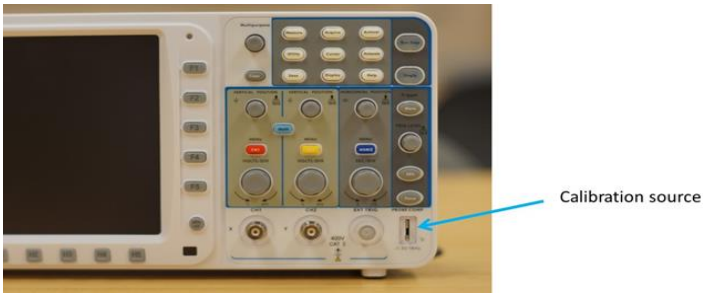
The channel inputs each have a BNC connector. We use oscilloscope probes connected to these connectors. Notice that since there are two channels, an oscilloscope can measure two voltages at once. But that means we may need two probes, each of which measure two electrical potentials!

To check that our oscilloscope is working correctly we can measure a known voltage, say, the voltage of a regular battery.

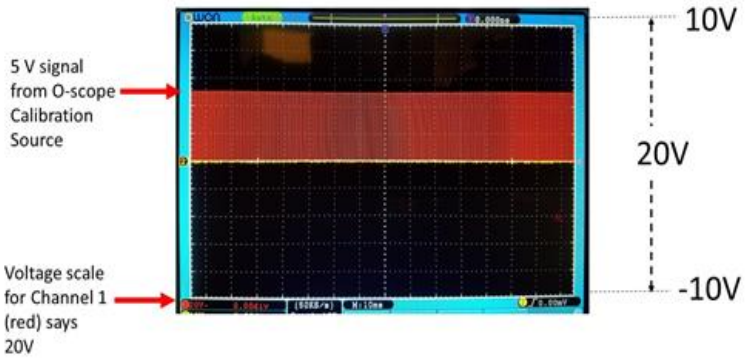


Notice that I changed the voltage scale knob position so that now the oscilloscope screen has a 10V total potential change. That means that we have 5V at the top of the screen and -5V at the bottom of the screen. The screen is divided into little boxes. There are five rows of boxes from the bottom to the top of the screen. Each box represents 1/10 of the total voltage. Since we have $\Delta V = 10$V, each box represents $\Delta V = 1$V. So our battery voltage should give us one and a half boxes. And that is just what we got.

But sometimes the oscilloscope does not get the right voltage. If this happens we need to calibrate the oscilloscope. Every time we use an Oscilloscope it is a good idea to check it to make sure it working well. Our oscilloscopes have a test voltage to use just for this purpose.



The calibration source makes a 5V square wave (try it to see what that looks like!). If we use this calibration source we should get something like what you see in the next figure.



If you don't get 5V, then something is wrong and you will need to go through the oscilloscope's calibration procedure. That is in the oscilloscope manual and you can find the manual on-line. Consult with your instructor.

2.3 Current

Our multimeters have a current setting as well as a voltage setting. Current is a flow of charge. This is like a water current, which is a flow of water. Only we have a different thing flowing. We have a flow of charge. In the wires in our Arduino, the moving charged things are (mostly) electrons. We can write the flow of something as

$$I = \frac{\Delta Q}{\Delta t}$$

where for us ΔQ is the amount of charge that has gone by in the time Δt. Physicists use the letter I for electrical current.

We should take a minute to think about what to expect when we allow charge to flow. Think of a garden hose. If the hose is full of water, then when we open the faucet, water immediately comes out. The water that leaves the faucet is far from the open end of the hose, though. We have to wait for it to travel the entire length of the hose. But we get water out of the hose immediately! Why?

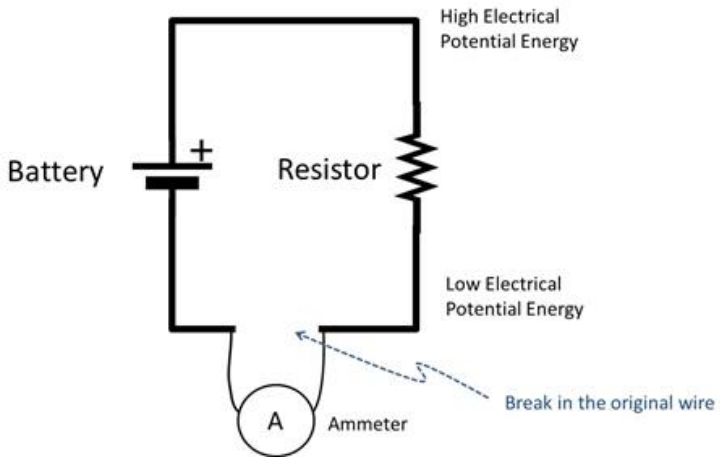
The new water coming in causes a pressure change that is transmitted through the hose. The water at the open end is pushed out. You can tell this is the case because the water immediately leaving the hose is warm and tastes like plastic hose. After a while, the water is colder and cleaner.

Current is a little bit like this. When we flip a light switch, the electrons near the switch start to flow. But there are already free electrons in the wire. These experience a push that makes the light turn on almost instantly. But

the electrons that turn on the light are not the ones that just went through the switch.

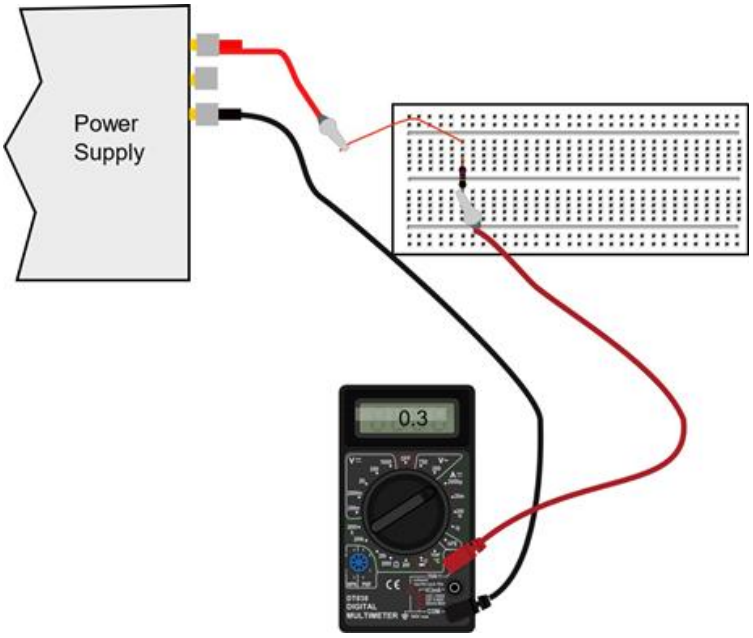
2.3.1 Measuring Current

Because current is a flow, to measure current we must put a meter into that flow. In a house, if you want to measure how much water is used, you connect the pipe from the city water system to a meter and then connect the meter to your house. The water flows through the meter and then goes into the pipe that brings water to the house. That way, the meter can't miss any of the water (and the city can't miss any of your payment!). The same is true for electrical current. To measure electrical current, we need to remove part of our circuit, and replace it with the meter to force the flow of electrical current to go through the meter. A schematic diagram of this might look like this:

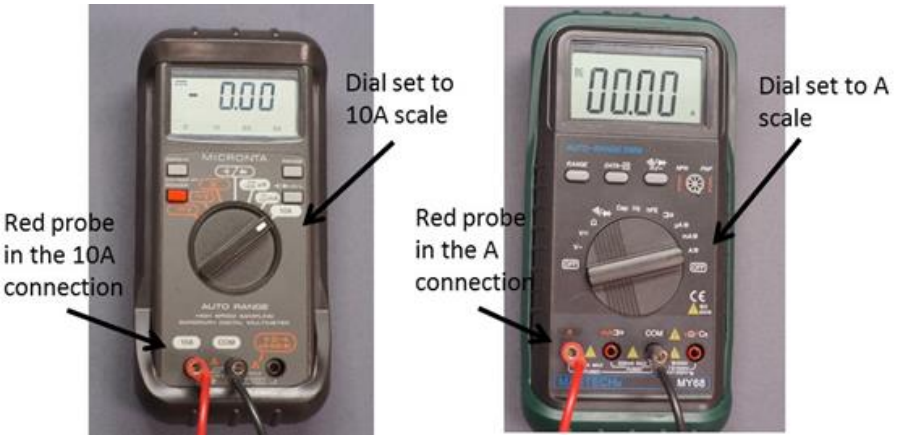


In this diagram, you can see that the electric current must go through the current meter. In fact, it couldn't go anywhere else because part of the original circuit wire is missing. This is just what we want. To actually perform this

measurement with one of our multimeters you could set up a circuit like the one in the following figure.



There is one more important thing to do to make this work. We need to change the meter settings. And there are two separate changes. The first is to switch the probe connections. One probe stays in the COM or common connector, but the other needs to move to the connector marked with an "A." Here is an example showing the changes with one of our kinds of multimeters.



The "A" stands for the standard unit of electrical current, the Ampere or Amp. With the multimeter set up like this we would call it an *ammeter*. Am-

meters measure electrical current.

2.4 Building a New Instrument

We said before that physicists like to change any measurement they can into a voltage measurement. That is because we have devices like our Arduino boards that measure voltage. We build new instruments by finding ways to turn the measurement that we want to make into a voltage.

In order to build a new instrument, we need to understand the quantity that we really want to measure. We will need to understand the physics of the quantity to make a good new instrument design. Let's take an example. Suppose we wish to measure current, but suppose we don't have a current setting on our Arduino (because we don't). Could we still make a current measurement?

The secret of instrument design is to understand the physics of the measurement we want to make (current) and then see if we can turn that measurement into a voltage.

2.4.1 Start with the Physics:

Let's keep thinking of current like water in a hose. Will there be any friction associated with the water traveling through the hose? Of course there will! We usually call friction in fluids *viscosity*. But it is a form of friction, and we can use our PH121 intuition about friction to see how it would work. Think of having two hoses, one twice as long as the other. Which would you expect to have more friction?

Our friction experience says that the longer the path, the more the friction. The current has longer to interact with the hose, so it experiences more friction. Electrical currents are like this. Longer wires give more friction.

George Simon Ohm noticed that with long metal wires, there seemed to be a linear relationship between the potential difference (voltage), the current, and the length of the wire. The longer the wire, the less the current. His work was confirmed and expanded on by others, who found that not only length mattered,

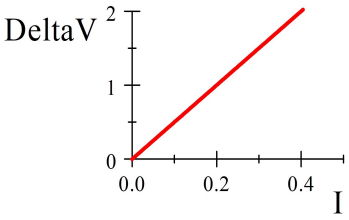
but also the diameter of the wire mattered. The relationship is now expressed as

$$\Delta V = IR$$

ΔV is our old friend, voltage, and we know I is the symbol for current, and R is the slope of the ΔV vs. I curve. The experiments showed this constant R depended on the material. It is like our viscosity in hoses. It is the friction. The more the friction, the harder it is to get the current through the wire. But like we don't call viscosity "friction," we also don't use the word "friction" for this friction-like term. We call it *resistance.* We could solve for this resistance

$$R = \frac{\Delta V}{I}$$

or we could plot ΔV vs. I and the slope of this line would be the resistance.



Either way, this relationship tells us that it takes more potential energy to get the same current if there is more resistance.

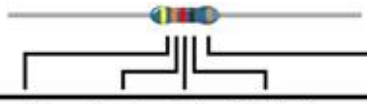
This is called *Ohm's law.* The relationship holds well for metals and many materials, but, like Hook's law, this "law" does not always hold. Devices that do provide a constant resistance coefficient, R, are called *resisters.* We will use this symbol

for resistors, but they often look more like this

Notice that this is important! We have found a way to relate our new quantity that we want to measure, current, to a voltage. We know how to measure a voltage! Our Ohm's law equation even tells us what extra part we need to convert our voltmeter into a current measuring instrument. We will need a resistor.

Resistor Code

Let's pause in our new instrument design for a moment and ask, "how would you know the resistance of a resistor?" Our multimeters have a resistance measuring setting, so you could measure the resistance directly using the meter. But many commercially produced resistors come conveniently marked with a color code that helps you identify their resistance. The basics of the color code are given in the following figure



COLOR	1st	2nd	3rd	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	1%
Red	2	2	2	100	2%
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	0.5%
Blue	6	6	6	1MΩ	0.25%
Violet	7	7	7	10MΩ	0.10%
Grey	8	8	8		0.05%
White	9	9	9	0.1Ω	
Gold				0.01Ω	5%
Silver					10%

To use the code:

1. Find the tolerance code band. This band is usually brown in our kit resistors and often is set off from the others a little more.

2. Read the first color band from the side opposite the tolerance band. This will be the first digit of your resistance. I think the example resister on the chart has a yellow first color band, so the first digit of our resistance is 4.

3. Read the second color band. This will be the second digit of your resistance. I think the second band of our example resistor is orange, so the second digit would be a 3, making our resistance so far 43

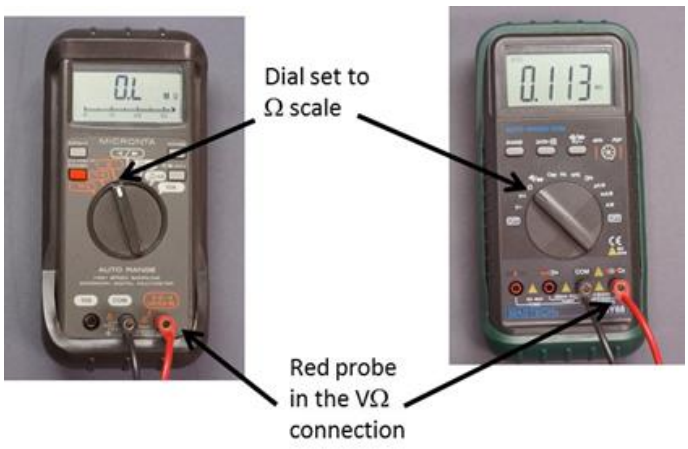
4. Read the third color band. This will be the third digit of your resistance. I think the second band of our example resistor is red, so the second digit would be a 2, making our resistance so far 432

5. Read the forth color band. This is a multiplier. You multiply the first three digits by this amount. For our example resistance, I think the third band is black. Then we multiply 432 by 1Ω to get 432Ω. This is our resistance.

6. The tolerance band gives the uncertainty in this value. Our example resistor seems to have a brown tolerance band, which tells us our value is good to $\pm 1\%$. For our example resistance, 1% would be $0.01 \times 432\Omega = 4.32\Omega$, so our resistance is $(432 \pm 4\Omega)$.

We won't memorize the resistor code, but you should be able to find a resistance using the code.

If you are in doubt about what color you see on a resistor, our multimeters can measure resistance directly.

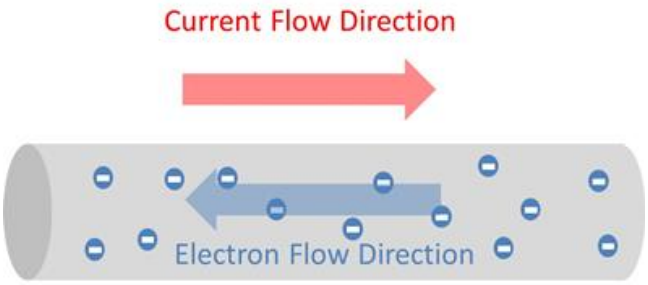


Place the red probe in the connector with a Ω marked on it and turn the dial to the Ω setting. Place the probes on either side of the resistance to be measured. Be careful/ You are a resistor too. If you touch your hands to the probes (common mistake while you try to hold the resistor on the probe ends) you may measure your resistance instead of the resistor's! You have a resistance of around half a megaohm. This is a general concern, every time you measure resistance with a meter you need to take the circuit element (resistor, light bulb, whatever) out of the circuit and measure it on it's own. Otherwise, you might be measuring the resistance of the rest of the circuit. Alligator clips are useful for this.

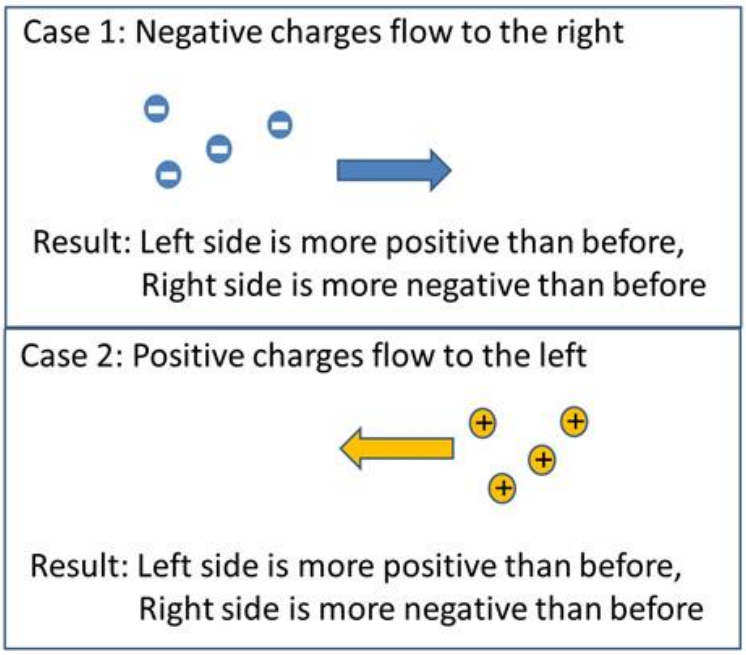
Direction of current flow

If you are half way through PH220 when you take this lab you will already know about current direction and can skip this section. **But if you are at the beginning of PH220, read on!** There is a historical oddity with current flow. That is that the current direction is the direction positive charges would flow. This may seem strange, since in good conductors electrons are doing the flowing and they are negative! The electrons go the opposite way the current goes.

The truth is that it is very hard to tell the difference between positive charge flow and negative charge flow the other direction. In fact, only one experiment



that I know of shows that the charge carriers in metals are electrons. And mathematically, the flow of electrons one direction is equivalent to the flow of positive charges the other direction.



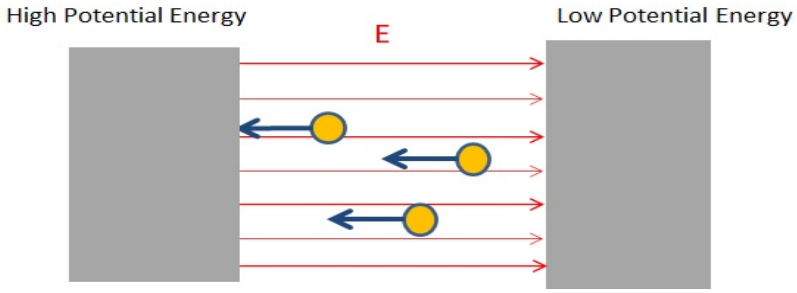
Worse yet, in biological things it *is* positive ions that flow. So for biology a positive charge carrier is just fine.

Ben Franklin chose the direction we now use. He had a 50% chance of making it easy for our electronics lab. But he got it backwards for us (but right for biology–and how many electronic things did Ben Franklin have anyway?). All this shows just how hard it is to deal with all these things we can't see or touch that we study in PH 220.

And even more importantly, in semiconductors–special electronic devices in all computers and in our Arduinos–it *is* positive charge that flows. In many elec-

trochemical reactions *both* positive and negative charges flow. So Mr. Franklin was not really so very wrong. We will stick with the convention that **the current direction is the direction that positive charges would flow regardless of the actual charge carrier motion.** If you are like me, this will seem a little backwards, but we all get used to it.

But what makes the electrons or positive charges want to flow in the first place? We know the answer to this from earlier in this lab reading. It is potential energy. When we connect a metal wire to the terminals of a battery we know that the charges in the metal wire ends will experience a difference in potential energy. The potential energy difference will set up an electric field inside the conductor.



This field makes the free charges move! It causes a force on the little electrons. We won't have to measure any fields in our lab today, but you should know they are there. The important thing is to realize that voltages produce currents. And the amount of current is proportional to the amount of voltage. This is just Ohm's law!

$$\Delta V = IR$$

The constant of proportionality is related to how much friction there is for the charges in the wire.

$$I = \frac{1}{R}\Delta V$$

It is just the resistance, R.

2.4.2 Knowing the Physics, Design the new instrument

Now that we understand electrical current, we have some hope of figuring out how to build an instrument to measure that electrical current. From what we learned, consider adding in an additional small resistor in our circuit. If we take a small resistance, one that is small compared to all the other resistances in the circuit, and we put it in the circuit it will slow down the current, but not by very much. If the resistance is small enough, we won't even notice the change. Then if we measure the voltage across that small resistor with a voltmeter,

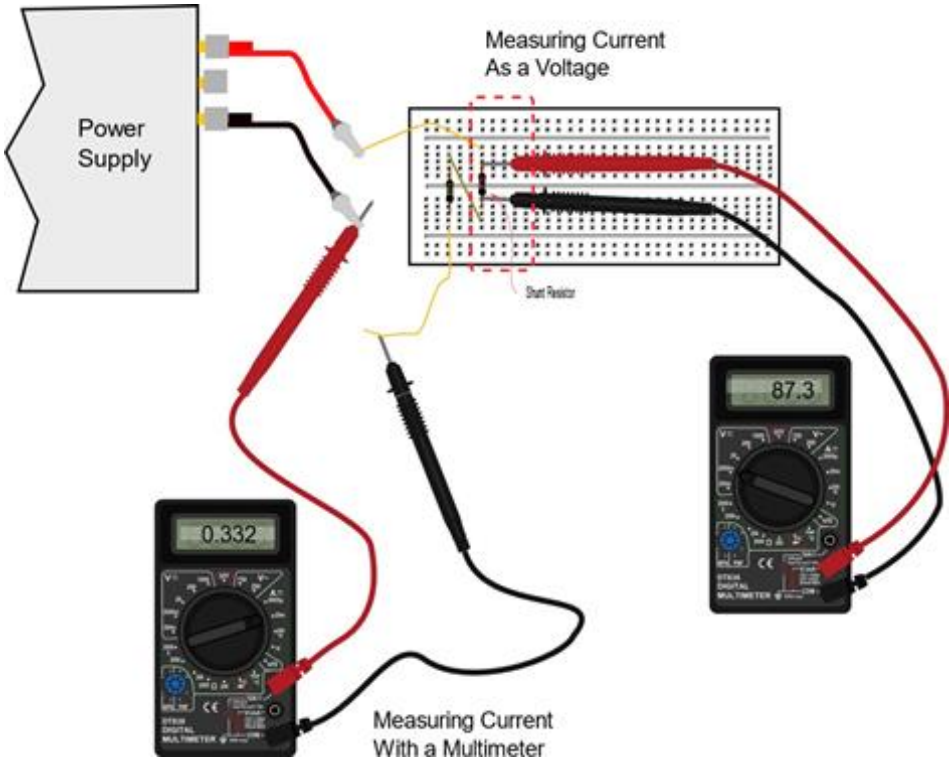
we could mathematically calculate how much current we have. Notice that this instrument design has two parts. The first is adding some new hardware to our voltmeter (a resistor) and the second is adding in some calculation to get our voltmeter reading converted into current. Here is our equation for the calculation

$$I = \frac{1}{R}\Delta V$$

And let's give this new additional small resistor a name. Let's call it the "shunt resistor."

$$I = \frac{\Delta V_{meter}}{R_{shunt}}$$

Today we will have to do the calculation part by hand. In future labs, we would carefully plan for this calculation in our Arduino sketch code.



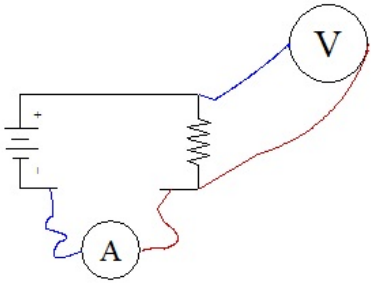
When we are done wiring our new instrument, we will have done something really cool. We have turned our current measurement into a voltage measurement plus a calculation. We measured something new in terms of a measurement we already knew how to make. We will generally try to do this for any type of measurement. That is because we are very good at measuring voltage, and not so good at measuring other things electronically.

2.4.3 Testing the new instrument

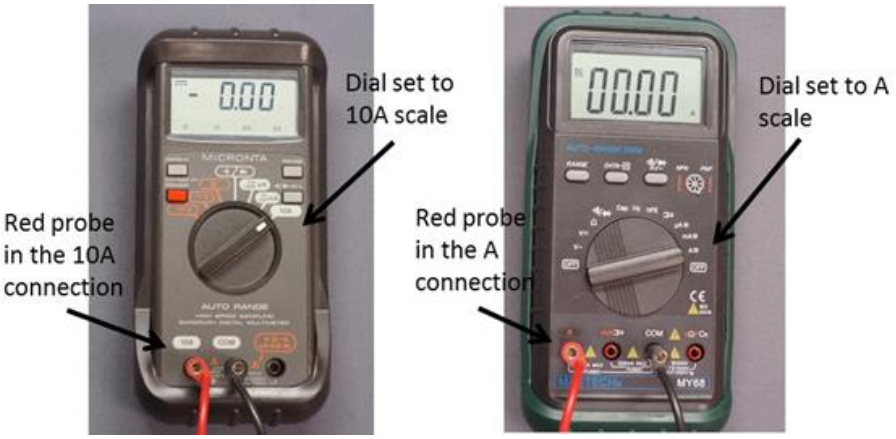
We will need a way to test how good our new instrument works. And fortunately we know our multimeters can also measure current. So we can build our new instrument an compare it to the measurement made by a multimeter. This test of a new instrument is an important part of designing a new instrument!

$$A = \frac{V}{\Omega}$$

Recall that to use an ammeter (the new instrument we build, or the one in our multimeter), you must break the electric circuit by disconnecting a wire. Then you replace that wire with the ammeter. Notice that in the diagram below that the bottom wire is now broken.

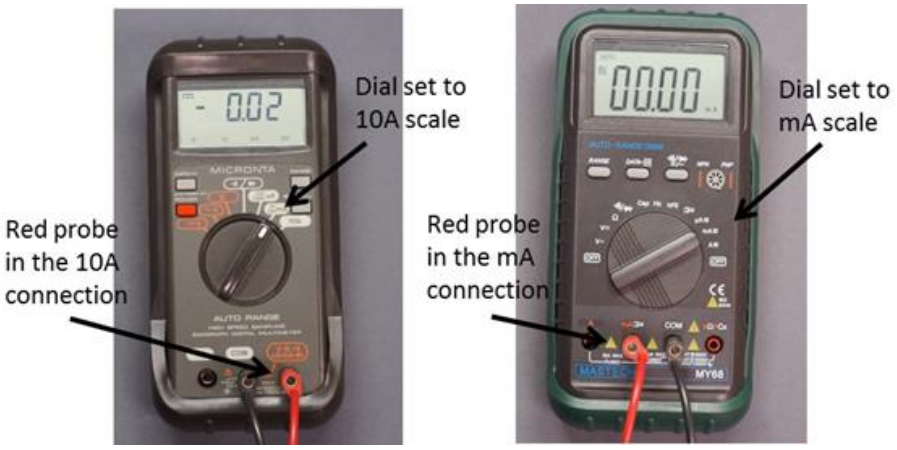


Where a wire was, I have drawn an ammeter. The current must flow through the ammeter for us to measure it.



Remember, to use our multimeters to measure current, we must turn the dial to the 10A setting AND move the red probe to the 10A connector. Failure to do this may result in the fuse blowing. Our meter does not warn you that it lost a fuse, it just pays you back by giving really wrong answers. You should be careful to connect it right, and be sure it is working (that someone else has

not blown the fuse before you). Since we have different kinds of multimeters, a second is pictured to the right. For this type of meter, put the red lead in the connector marked A and turn the dial to the A setting. If the currents you are measuring are very small, you might have to switch settings once again. Tiny currents can be measured by moving the dial to the mA setting AND changing the red probe to the mA connector.



Our multimeters really measure current in much the same way we are talking about for our new Arduino ammeter. They have a series of shunt resistors inside of them. When we choose a current measurement setting we are choosing a shunt resistor to put in the circuit (inside the meter, but the meter is in the circuit). Then the voltmeter will measure the voltage across that resistor and use that voltage to calculate the current.

If we create the current meter ourselves, we have to know the resistance that we used! That allows us to use the voltage meter to calculate the current,

$$I = \frac{\Delta V_{meter}}{R_{shunt}}$$

but our multimeters are programed to know their own shunt resistances and to do this calculation for us.

If you have time (and some won't) in today's lab, we will build the new instrument to measure current out of our stand-alone multimenters (in future labs we will use our Arduinos), and we will test it with a second multimeter set in ammeter mode. We will put both in the circuit at the same time (see figure 1 in the assignment section below) so we get readings from both. Then we can compare and see how well our new instrument works!

2.5 Calculating uncertainty, a review

That is all the new material for today's lab. But I wanted to remind you of something you already know.

Back in PH150 you should have gotten a good deal of experience in making measurements. We will be going back to experimentation soon, and we will need to remember what we learned in PH150 to take the measurements so that we can interpret our experimental results. You will remember that every measurement has an uncertainty. We have to estimate that uncertainty. It turns out that our voltage measurement schemes will introduce a new source of uncertainty! And we will have to include this in our uncertainty calculations. We will take that on next lab, but in this lab let's review how to calculate uncertainties (If you took our PH150 recently, you can skip this section!).

This is a "review." How much of a "review" it is may depend on where and when you took PH150 or its equivalent. If you are a chemist, you will note that our treatment of uncertainty goes beyond what you learned in Quantitative Analysis. Let's start by reviewing what a derivative is.

For our purposes, a derivative is a slope of a line. You should recognize the equation of a straight line as

$$y = mx + b$$

The slope m can be written as

$$m = \frac{dy}{dx}$$

This is nothing magic (or new). It is just a strange way to write m. With the slope written this way, the equation of the line could be written as

$$y = \frac{dy}{dx}x + b$$

But why dy/dx? Think of how we find a slope of a line. Back in junior high school we called the slope the "rise over run." That is, the change in y-value divided by the change in the x-value.

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

In physics, we write the change in a variable using the Greek letter delta, Δ. So we could write the slope as

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}$$

Just to jog your memory, let me write out Δy

$$\Delta y = y_2 - y_1$$

and Δx.

$$\Delta x = x_2 - x_1$$

So our straight line equation should be written

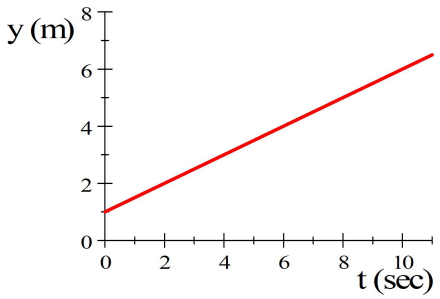
$$y = \frac{\Delta y}{\Delta x} x + b$$

but if we take Δx to be very, very small it is customary to write the Δx as just dx (I guess a "d" is smaller than a "Δ"). If this is not familiar from Math 112, is should be by now from PH121.

In PH121 you learned that the velocity is the slope of the plot of x vs. t, for example,

$$y = \frac{1}{2}\frac{\mathrm{m}}{\mathrm{s}} t + 1\mathrm{m}$$

is an equation giving the y position of an object as a function of time. Note that it is a straight line on a y vs. t plot.



The slope of the line is

$$\frac{dy}{dt} = \frac{1\mathrm{m}}{2\mathrm{s}}$$

We can verify that this works by looking at the plot and noting that for every two units of time, we go up one position unit. The slope is $1/2\frac{\mathrm{m}}{\mathrm{s}}$.

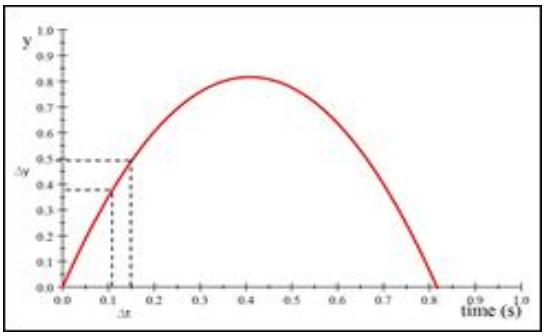
But not all curves are straight lines. What do we do with curves that, well, curve?

One idea is that we could split up the curve into little line segments, each with its own slope. We can think of dy/dt as an instantaneous slope, a slope of one of the tiny line segments that make up our curve. This is the sort of speed measurement that your speedometer gives. The speed might be different a short time later. But right now the speed is, say, 0.5m/s.

Really, in defining an instantaneous slope we have assumed that the slope near our point on the curve is essentially a straight line if Δt is small enough.

We can use this idea to interpret our error calculations. Suppose I throw a ball in the air with a initial speed of 4m/s straight up starting from $y_o = 0$. From PH121 you have learned that the equation for predicting how high the ball will go is

$$y = y_o + v_o t + \frac{1}{2} a t^2$$



It says that starting at y_o the ball will go higher depending on the initial velocity, v_o, and the acceleration, a. That makes sense.

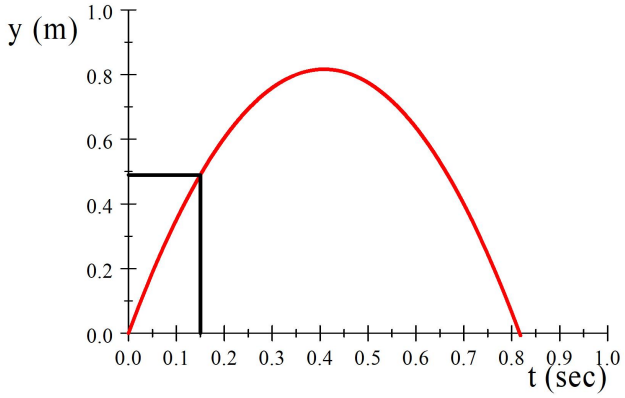
At a time, t, the ball should be at

$$y = 0 + 4\frac{\text{m}}{\text{s}}t - \frac{1}{2}\left(9.8\frac{\text{m}}{\text{s}^2}\right)t^2$$

where $a = -9.8\frac{\text{m}}{\text{s}^2}$ is the acceleration due to gravity. So, knowing this, I could predict how high the ball would go if I pick a particular time, say, 0.15s. The result should be

$$\begin{aligned} y &= 0 + 4\frac{\text{m}}{\text{s}}(0.15\text{s}) - \frac{1}{2}\left(9.8\frac{\text{m}}{\text{s}^2}\right)(0.15\text{s})^2 \\ &= 0.489\,75\text{m} \end{aligned}$$

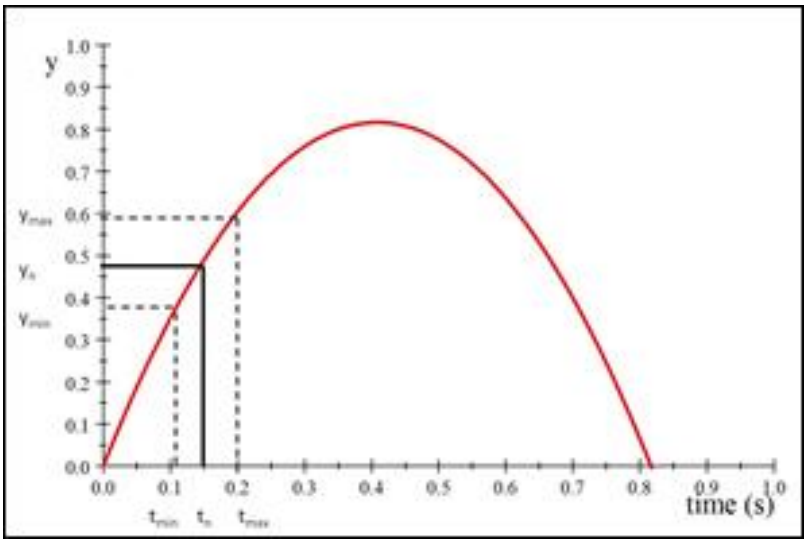
This is shown in the next figure with a black line. Solving the equation for y is equivalent to drawing a line up to the curve, then from our spot on the curve over to the y-axis to find the position.



For our case, we plot a line upward from 0.15s to the curve, and then plot a horizontal line from the intersection to the y-axis. We can see that we get 4.9m.

Suppose I try to verify this by taking a picture of the ball in flight at 0.015s, but my stop watch is only good to ± 0.005 seconds. I try to take the picture when the watch is at 0.015s, but I might have taken the picture at 0.01s or at 0.02s or anywhere in between. My time has some uncertainty. What does the uncertainty in my stop watch time mean for the uncertainty in my y value?

We can get a good approximation by graphically drawing vertical lines up from $t_{\min}$ and $t_{\max}$ to the curve, and then extending horizontal lines from the intersections to the y-axis. This gives us a $y_{\min}$ and $y_{\max}$. Our actual height could be anywhere in between these. This is a way to view our uncertainty in y.



We can use this idea to find a general way to calculate uncertainties. We could define $\Delta t = t_{\max} - t_{\min}$. If our Δt is small enough (so we can write it just dt), the curve is essentially a straight line in the region between $t_{\min}$ and $t_{\max}$. So if we knew the slope of that line (the derivative dy/dt) we could easily figure out the $y_{\max}$ and $y_{\min}$ points to get our uncertainty range, at least if we stay near our t_n part of the curve. Recall that our uncertainty in y is about

$$\delta y = \frac{y_{\max} - y_{\min}}{2} = \frac{\Delta y}{2}$$

Remembering that

$$y = \frac{dy}{dt}t + b$$

then

$$\begin{aligned}\Delta y &= y_{\max} - y_{\min} \\ &= \frac{dy}{dt}t_{\max} + b - \frac{dy}{dt}t_{\min} - b \\ &= \frac{dy}{dt}\Delta t\end{aligned}$$

From PH150, you will recognize this as almost the uncertainty in a function of one variable! But even if you don't recognize it, we can show that this is true using our definition of δy above. The quantity Δt is

$$\Delta t = t_{\max} - t_{\min}$$

so our uncertainty in t would be

$$\delta t = \frac{t_{\max} - t_{\min}}{2} = \frac{\Delta t}{2}$$

then

$$\begin{aligned}\delta y &= \frac{y_{\max} - y_{\min}}{2} \\ &= \frac{1}{2}\frac{dy}{dt}\Delta t \\ &= \frac{dy}{dt}\frac{\Delta t}{2} \\ &= \frac{dy}{dt}\delta t\end{aligned}$$

so

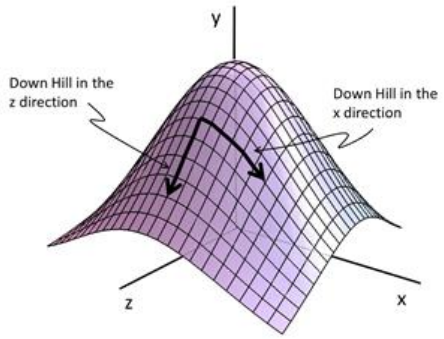
$$\delta y = \frac{dy}{dt}\delta t$$

So our uncertainty in y is just the slope at our point on the curve multiplied by our uncertainty in t.

But what if we have more than one variable? Say, we have a function $y(x, z)$, we essentially have a two dimensional slope. Think of a hill, you can go down a hill in more than one direction. So we need slope parts for each direction we can go.

$$\Delta y = \frac{dy}{dx}x\hat{\imath} + \frac{dy}{dz}z\hat{k}$$

But there is a fix we need to make to this equation that you won't learn for several math classes to come. We want to have a slope in the x and z direction, but we want the slopes to be independent. Think back in PH121 when we did two dimensional motion problems, we split the problem into components. We want to do this for our two dimensional uncertainty! The notation for this is



$$\Delta y_x = \frac{\partial y}{\partial x} x$$

$$\Delta y_z = \frac{\partial y}{\partial z} z$$

where

$$\frac{\partial y}{\partial x}$$

means the component of the slope **just** in the x direction. We take a derivative of the function y, but assume only x is a variable (treat z and all z terms with no $x's$ as constants). This lets us separate the x and z parts. A special, one variable derivative like $\partial y/\partial x$ is called a *partial derivative* because you only take one dimension of the derivative at a time. So, if we wish to find the error in some general function $z\left(x,y\right)$ the error is given by

$$\delta y = \sqrt{\left(\frac{\partial y}{\partial x}\right)^2 \delta x^2 + \left(\frac{\partial y}{\partial z}\right)^2 \delta z^2}$$

This looks a lot like our slope equation. What we are doing is to assuming the function $y\left(x,z\right)$ is flat in a small region around the point we are studying. then the function has a slope $\partial y/\partial x$ in the x-direction, and $\partial y/\partial z$ in the y-direction. Each term like

$$\left(\frac{\partial y}{\partial x}\right) \delta x$$

gives how far off we could be in that direction (the x-direction in this case). Remember that we have assumed that $y\left(x,z\right)$ is essentially flat near our point of interest. The square root may be something of a mystery, but remember what you have learned about adding vectors in PH121. We add components of a vector to find the magnitude like this

$$V = \sqrt{V_x^2 + V_y^2}$$

This comes from the Pythagorean theorem. The x and y parts of the vector form two sides of a triangle. We want the remaining side. So we use the Pythagorean theorem to find the length of the remaining side.

We are doing the same for our small uncertainty lengths. We are just adding the x and the y components of the error. We could write our error formula for the general case of a function f, that depends on N different variables.

$$\delta f = \sqrt{\sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i} \right)^2 \delta x_i^2}$$

We will use this formula a lot, so make sure you understand what it means (ask your instructor for help if it is not clear).

2.5.1 How do we find the slope?

But now we have an equation in terms of slope written as $\partial y/\partial x$ or $\partial y/\partial z$, but how would we ever find these slopes? Your calculus class has or will teach you how to take a derivative. They might not have yet taught you how to take this type of derivative. The symbol ∂ means that our derivatives are "partial" derivatives. This means that we assume all the variables other than the one that shows up in the derivative symbol are constants for our derivative.

Let's take an example. What is the slope of the function $y = 5zx^3$? if we calculate the slope only going in the x-direction (that is, if we take a partial derivative with respect to x) we get

$$\frac{\partial}{\partial x}\left(5zx^3\right) = 5z\,(3)\,x^{3-1} = 15zx^2$$

notice that we treated z as a constant! That is what we mean when we user the symbol ∂ and when we say "partial derivative." Let's try another. How about finding the slope of $f = 7yx^2 - 2x + z$ with respect to the x-direction.

$$\frac{\partial}{\partial x}\left(7yx^2 - 2x + z\right) = 7y\,(2)\,x^1 - 2(1)x^0 + z\,(0)$$

The last term illustrates that the slope of a constant is zero, and as we go just in the x-direction, z is constant. That makes sense. So the change in f just due to the last term (z) should be zero. We also remember $x^0 = 1$. So we are left with

$$\frac{\partial}{\partial x}\left(7yx^2 - 2x + z\right) = 14yx - 2$$

We could also find

$$\frac{\partial}{\partial y}\left(7yx^2 - 2x + z\right) = 7x^2$$

and

$$\frac{\partial}{\partial z}\left(7yx^2 - 2x + z\right) = 1$$

In our equation for calculating uncertainties, we want to find the uncertainty in each dimension (for each variable) and to add these uncertainties like components of vectors, so this partial derivative is just what we want, the slope of our function in just one direction.

Tie to statistics

Back in PH150 you should have learned that for experiments where we repeat the same experiment over and over again, our outcome can be given by the mean value an our uncertainty can be given by the standard deviation. We need to tie our statistical ideas into what we have learned about error propagation. Lets go back to our function $f(x, z)$ the error is given by

$$\delta f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 \delta x^2 + \left(\frac{\partial f}{\partial z}\right)^2 \delta z^2}$$

but now we know we could express this in terms of standard deviations (provided you don't need to ensure every bit of your data are within your uncertainty range). We can write our uncertainties as

$$\sigma_f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial z}\right)^2 \sigma_z^2}$$

So one way to get an estimate of uncertainty like δx or δz above is to make many measurements, and use the standard deviation σ_x as an estimate for δx and σ_z for δz. This is usually not too far off (we will refine this analysis in PH336 for those lucky enough to take the course).

We can use connection between δx and σ_x to show that the standard deviation of the mean (the best estimate of our uncertainty) is given by

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{N}}$$

(a result you should have learned back in PH150). Think of calculating a mean value

$$\bar{x} = \frac{x_1 + x_2 + \cdots x_N}{N}$$

We can find the uncertainty in this function $\sigma_{\bar{x}}$

$$\sigma_{\bar{x}} = \sqrt{\left(\frac{\partial \bar{x}}{\partial x_1}\right)^2 \sigma_{x_1}^2 + \left(\frac{\partial \bar{x}}{\partial x_2}\right)^2 \sigma_{x_2}^2 + \cdots + \left(\frac{\partial \bar{x}}{\partial x_N}\right)^2 \sigma_{x_N}^2}$$

You see we just take the partial derivative of our function $\bar{x}$ with respect to each of the variables x_i and multiply by the uncertainty in that variable written now as a standard deviation σ_i.

For this special case, all of the x_i are the same (we are measuring the same value over and over in taking an average) and all of the σ_i are the same so we just have

$$\sigma_{\bar{x}} = \sqrt{N\left(\frac{\partial \bar{x}}{\partial x_1}\right)^2 \sigma_{x_1}^2}$$

and we can take the derivative using our rule. Only x_1 is a variable, so we can write the average $\bar{x}$ as

$$\bar{x} = \frac{x_1}{N} + \frac{x_2 + \cdots x_N}{N}$$

This is a polynomial! The first term is $\frac{1}{N}x_1$ and the whole second term is a constant if we take a partial derivative with respect to x_1. The derivative is

$$\begin{aligned}
\frac{\partial \bar{x}}{\partial x_1} &= \frac{\partial}{\partial x_1}\left(\frac{x_1}{N} + \frac{x_2 + \cdots x_N}{N}\right) \\
&= \frac{1}{N}x_1^0 + 0 \\
&= \frac{1}{N}
\end{aligned}$$

so our statistical error function is just

$$\begin{aligned}
\sigma_{\bar{x}} &= \sqrt{N\left(\frac{1}{N}\right)^2 \sigma_{x_1}^2} \\
&= \sqrt{\frac{\sigma_{x_1}^2}{N}} \\
&= \frac{\sigma_{x_1}}{\sqrt{N}}
\end{aligned}$$

or, since all the σ_{x_i} are the same, we can just write this as

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{N}}$$

Notice that in this example we had many x_i and that to find the uncertainty we just extended our equation from two variables

$$\sigma_f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial z}\right)^2 \sigma_z^2}$$

to N variables

$$\sigma_f = \sqrt{\sum_{i=1}^{N}\left(\frac{\partial f}{\partial x_i}\right)^2 \sigma_i^2}$$

In this special case, we were trying to show a special result, but we can do this for any function with any number of variables. If your function is complicated, you just need to take more partial derivative terms under the square root.

I'm sure you remembered all of that from PH150. But now it is time to move on to today's lab assignment.

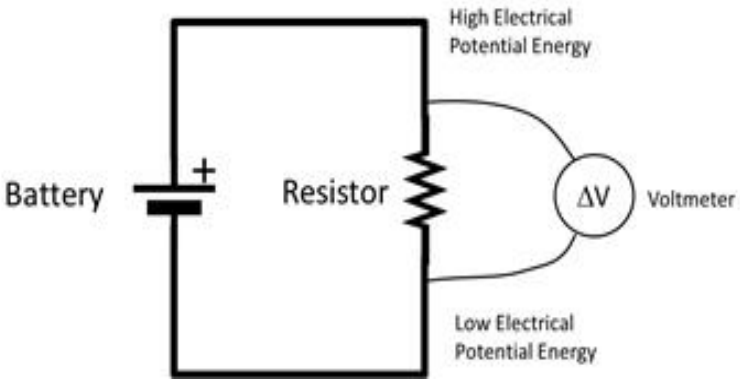
2.6 Practice Measurements

Here are some measurements to try with the instruments we learned about in this reading.

2.6.1 Use a Multimeter

1. Measure the voltage of a D-Cell battery with a voltmeter. Report the value you get from the measurement and the uncertainty.

2. Set up the circuit described in section (2.1) only use our power supply in place of the battery. The figure is repeated below. Measure the voltage with a multimeter. **The indicator on the power supply is not very accurate**, but it can serve as a check to see if we are way off. So compare the power *supply voltage indicator to the meter indicator. Are they the same (you should think of the uncertainty in both meters and answer using the uncertainties)



3. Modify your circuit described in section (2.1) (still using the power supply in place of the batery) and change the settings of your multimeter so that you measure the current in the circuit. Compare your ammeter measurement to the current shown on the power supply indicator.

2.6.2 Use an Oscilloscope

1. Predict what you will see on an oscilloscope if you measure the voltage of a D-Cell battery. Make sure your lab table agrees on your prediction before you go on.

2. Use the oscilloscope to measure the voltage of a D-Cell battery.

3. Practice interpreting oscilloscope screens

   (a) Figure out what the markings on the Oscilloscope screen mean.

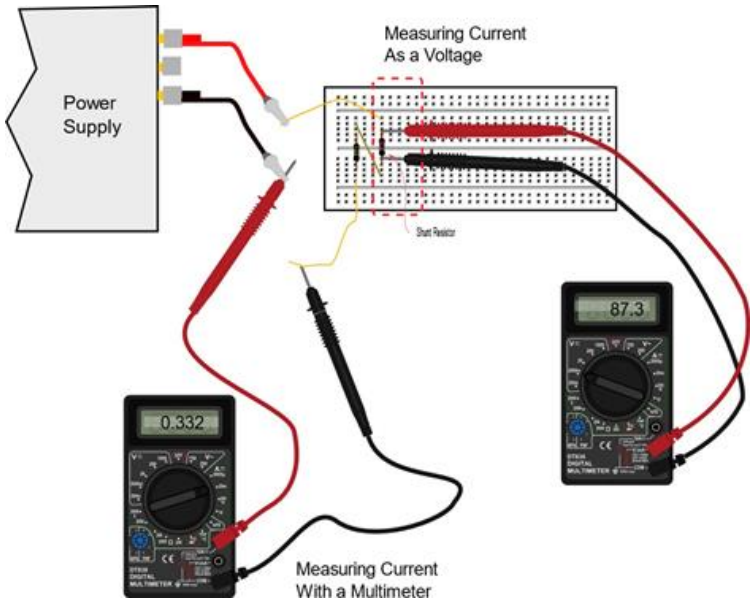
(b) Figure out what the voltage scale is and how to change it

(c) Figure out what the time scale is and how to change it

4. Generate a sinusoidal signal using the stand alone Signal Generator.
5. Measure and display this signal using the stand alone digital oscilloscope.
6. Report the amplitude and the frequency (and their uncertainties) of the measured signal and compare to the signal generator settings.

2.6.3 Build a new instrument from an old instrument

1. IF THERE IS TIME, choose a resistor in the 20kΩ range (say, from about 10kΩ to about 30kΩ). Choose a shunt resistor in the 200Ω range. Then set up the circuit to measure a current.



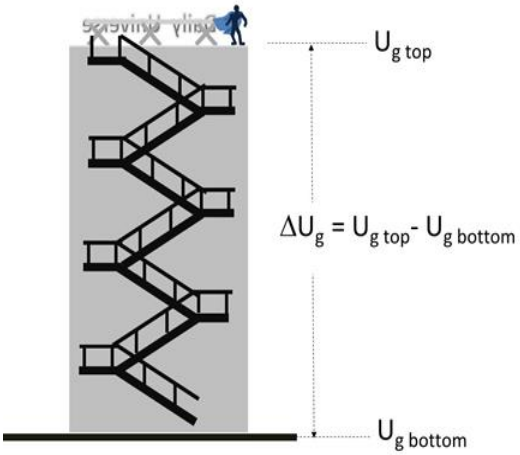
Use an additional multimeter set to measure current to check our voltage-current measurement. Calculate a percent difference to see how well this worked.

Chapter 3

First DAQ Measurements: Voltage

Let's review what we learned last week. In physics equipment, we try to measure voltages. If your data is not a voltage, we try to convert it into a voltage. Already we converted current into a voltage (using a shunt resistor). But what is a voltage? We said last week that it is a measure of electrical potential energy. It is also likely that you know the word "voltage" because we live in a world that has electricity everywhere. You probably know that your house or apartment has wires in the walls that carry "110 Volts." And you probably realize that "voltage" is a measure of how much energy there is in the wires.

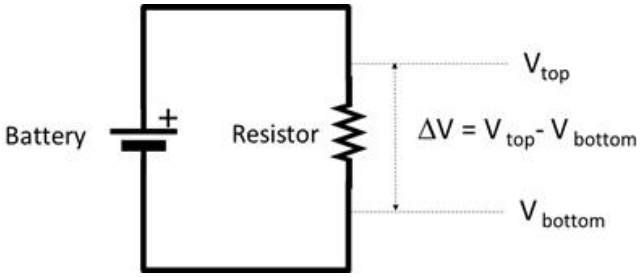
If it hasn't already, soon your PH220 class will explain that "voltage" is proportional to the electrical potential energy difference. But for us, now, we just need to know that we are measuring something proportional to energy and we need to learn how to measure it.



Because voltage is proportional to a *difference* in electrical potential energy, a voltage measurement really is a combination of two measurements. Think of gravitational potential energy. If we ask for the potential energy difference as Super Person jumps from the bottom of a building to the top we need two measurements, one at the bottom and one at the top. Then

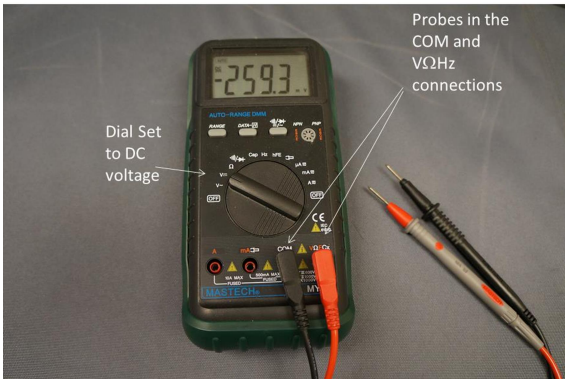
$$\Delta U_g = U_{g_{top}} - U_{g_{bottom}}$$

We will do something very similar in measuring voltages. We will measure the potential energy at two places. For example, suppose we have an electric circuit as shown in the next figure.



The circuit is very simple, just a battery and a resistor. You have experience with batteries, and resistors now. A resistor is just a piece of material that has lots of electrical friction, or "resistance" that makes it hard for electrons to go through it. If we want to measure the voltage across the resistor, we have to measure on the top and bottom of the resistor. That will give us a measurement proportional to the potential energy difference from one side to the other of the resistor.

Most meters that measure voltage have two "probes" and do the difference calculation internally. These meters are called *voltmeters* and we used them last week.



In today's world, voltmeters are usually just one part of a device that can measure many things. We call these devices multimeters. To measure voltage

we will set the multimeter on the DC Voltage setting. Notice the two probe wires in the figure. We need two probes to make the two measurements and the device does the subtraction for us.

We learned to use a sand-alone voltmeter in the last lab. But we also need to read in voltages in a way that the data shows up on our computer. To get the data into our computer we will use a different set of pins on our Arduino board. They are called *analog* pins.

Even before we begin, we need a warning. We absolutely must not wire up the analog pins on our Arduino backwards! This can (and probably will) destroy the pin circuitry inside our Arduino. So we will need to be careful in wiring for this part of our lab. Where this could be a problem a warning sign will appear in the text, just to remind you to be careful!

You may see quite a few of these in this lab.

3.1 Building a voltmeter

Our Arduino has what we call and Analog to Digital converter (ADC). That is, it takes analog voltage signals that could have any value, and it maps them into a set of discrete values and sort of rounds to the nearest whole discrete value.

The word "analog" might not be familiar. Think of our power supply. It has a knob that adjusts the voltage. The knob can produce any voltage from 0 to about 30V. This is an analog signal. The voltage can take on any value in a range. So we represent an analog signal with real numbers and we might have a voltage of exactly

$$4.32768543255325734 57\text{V}$$

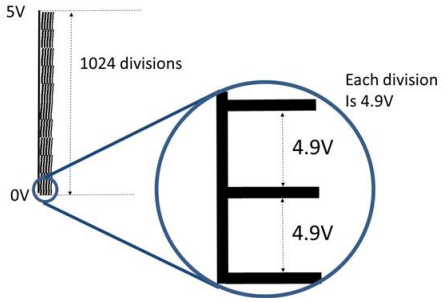
and this would be perfectly valid for an analog signal.

A battery, on the other hand, is not this way. It has a fixed voltage, say, 1.5V like the D-Cell batteries that we used in our last lab. Two D-Cell batteries could be used together to make 3V. But you can't use D-Cell batteries to get 2.25V. The batteries come in discrete units.

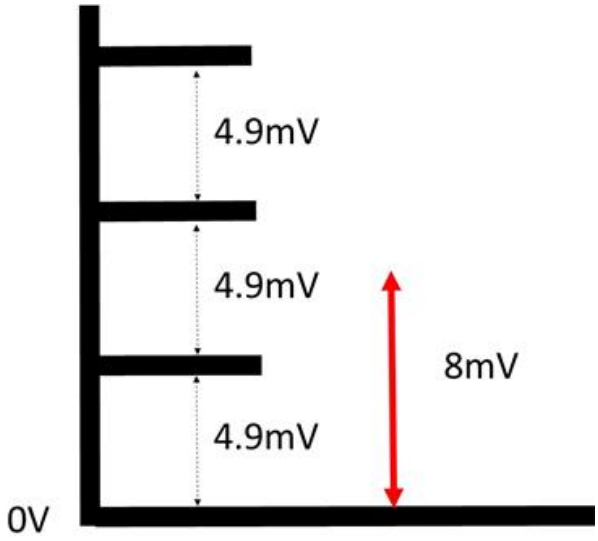
Our Arduino analog pin is designed to measure voltages in the range 0 to 5V. Don't set your power supply to more than 5V! But there is more to the

ADC than just a voltage range. The Arduino chops the voltage range into 1024 discrete voltage divisions. Each division is then

$$\Delta v_{\min} = \frac{5\text{V}}{1024} = 4.9\text{mV}$$



This means that changes in voltage that are less than 4.9mV won't be seen, since it takes a whole 4.9mV to get a different division. So if we give our Arduino 8mV this is not enough to fill the second 4.9mV division, so our Arduino would still read only 4.9mV. If we gave it 11mV it would then read 9.8mV because $9.8\text{mV} = 2 \times 4.9\text{mV}$ and 9.8 is the closest whole unit of 4.9mV.



This is called "discretization" or more commonly "digitization" or even *"quantization."* We have taken a signal that might have any value between 0 to 5V and we output a signal that will be rounded to the nearest $n \times 4.9\text{mV}$.

As a second example, 3.793V would be reported as 3.792 6V since

$$\frac{3.793\text{V}}{4.9\text{mV}} = 774.08$$

but we need even units of Δv_{min}, so the 0.8 would be dropped by the A2D converter giving

$$774 \times 4.9\text{mV} = 3.792\,6\text{V}$$

and our first voltage from our power supply, 4.3276854325532573457V would be reported as $4.326\,7$V (make sure you can see how we got this result!).

This means that we can be off in our voltage measurements as much as 4.9mV! In dividing up our voltage range into 1024 pieces we have introduced some error, but we have divided our 0 to 5V into numeric values that we can use in our computer, so it is worth the cost of some error.

The amount of error depends on how many different values the ADC converter has. Since breaking an analog signal into discrete values is called *quantization,* we call this source of error *quantization error.* It is the source of much of the error we see in electronic measuring devices. We could say that our new voltmeter has an uncertainty of at least the voltage resolution

$$\delta V_{signal} = \Delta V_{\text{min}} = 4.9\text{mV}$$

but of course it could be larger if there are other sources of error.

The ADC sends the measured value through our USB cable to our computer's serial port. But it doesn't send it in units of volts. It sends it in ADC units. If we have a signal voltage of 9.8mV we don't get out 9.8, we get 2 because $9.8\text{mV} = 2\Delta V_{\text{min}}$. The ADC units are the number of ΔV_{min} sized units that are in our signal voltage. To get back to voltage units, we need to multiply by ΔV_{min}. In our code we will do this before reporting the value.

Of course we would like to see the voltage that we measure. There is a simple way to do this. The voltage values we calculate can be sent to our computer through the serial cable. We will need an Arduino sketch with some additional setup and some additional loop commands. One of these commands will turn our ADC units into volts.

Before we look at the entire sketch, let me introduce the new commands that we will need. To get the Arduino to communicate with the computer we use the command

```
Serial.begin();
```

and in the loop function we use the command

```
Serial.print();
```

We also need to know that computers make a distinction between integer and real numbers. Our voltages will be real numbers, so we need to tell the Arduino that we want a real number. The command for this is the word "float." For example,

float delta_v_min=0.0049;

defines a variable named "delta_v_min" and sets it to the value 0.0049. If we want an integer number we use the word "int." For example

int value = 0;

defines a variable named "value" and sets it equal to 0. All this is a little like listing your variables back in PH121. Only here if you don't do it, it doesn't just cost you points, it confuses the Arduino software and the Arduino software will give you an error.

We also need special commands to read our Arduino analog pins. The special Arduino command

analogRead()

will do this.

The whole Arduino sketch might look like this: Download here

```
////////////////////////////////////////////////////////
// very simple voltmeter
// will measure 0 to 5V only!
// Voltages outside 0 to 5V will destroy your Arduino!!!
// Don't wire this backwards!
////////////////////////////////////////////////////////
// define a variable that tells which analog pin we will
//   use
int AI0 = 0;    //AI0 stands for analog input zero
// define a variable that holds our Delta_v_min
float delta_v_min=0.0049;   // volts per A2D unit
// define a variable for our A2D version of our signal
int ADC_value = 0;
// define a variable for our voltage version of our signal
float voltage = 0.0;

////////////////////////////////////////////////////////
void setup() {
  // put your setup code here, to run once when your
  // Arduino starts up:
  //
  // Initiate Serial Communication, so we can see the
  // voltage on our computer
  Serial.begin(9600);    //9600 baud rate
}

////////////////////////////////////////////////////////
void loop() {
```

```
  // Read in the voltage in A2D units form the serial port
  //   remember that AI0 is the pin number to read from
  ADC_value = analogRead(AI0);
  // Let's print out our A2D version of our signal
  Serial.print("␣A2D␣");
  Serial.print(ADC_value);
  // Now convert to voltage units using delta_v_min
  voltage = ADC_value * delta_v_min;
  // And print out our voltage version of our signal
  Serial.print("␣voltage␣");
  // Print the voltage with 4 significant figures)
  Serial.println(voltage, 4);
}
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

Make sure you understand every line of this code. Write it in the Arduino IDE and run it to help see what the lines do. Lines that begin with two slashes, "//," are comments. The Arduino will ignore these lines. But you shouldn't! The comments tell you, the programmer, what the code is doing. I will ask you in lab to input comments for every line. If there is any part of this sketch that is mysterious, work with your group to resolve the mystery and if it is still mysterious, call your instructor over to discuss the sketch with you.

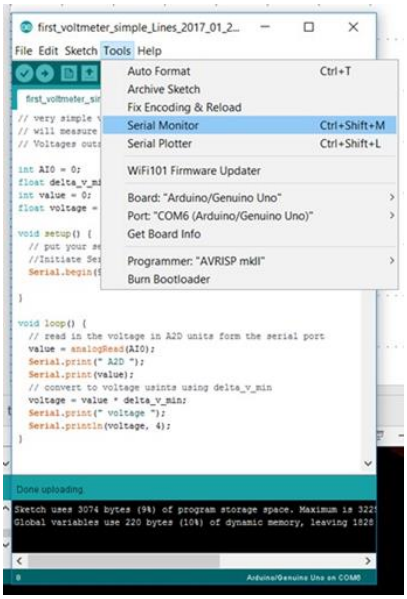
3.1.1 Wiring the simple voltmeter

You knew that was coming, didn't you! We must be very careful to wire our Arduino correctly. Our Arduino can measure 0 to 5V. But if we switch the 5V and the 0V by plugging them into the wrong pin, our Arduino may be damaged and never work the same way again (probably won't work at all!). So wire first, then before you connect the Arduino have group member check your wiring, then check the wiring with a stand-alone meter (that is why we learned to use them last week). Also remember, more than 5V will damage the Arduino. So only put in voltages in the range 0V to 5V.

We need one wire attached to the pin marked A0. We need another wire attached to one of our Arduino ground pins marked GND. And we connect the first wire to the positive output of our signal source (say, our power supply) and the GND wire to our negative output of our signal source (say the negative or ground connection on our power supply). That is all there is to it!

3.1.2 Seeing the data

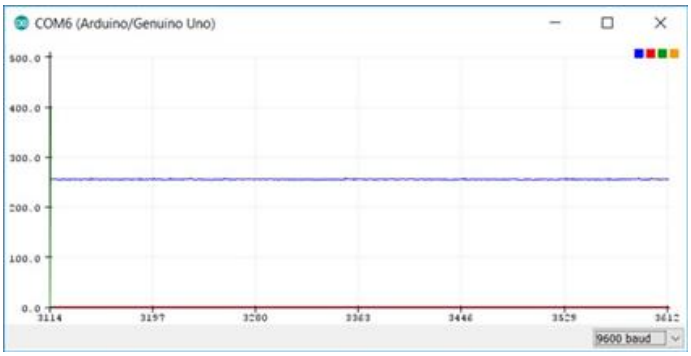
Once the code is compiled and uploaded, the Arduino will send data to the serial port. The serial monitor can display the data. The serial monitor is found under the Arduino Software Tools menu on the legacy IDE, and down at the bottom hiding behind a tab in the new IDE.



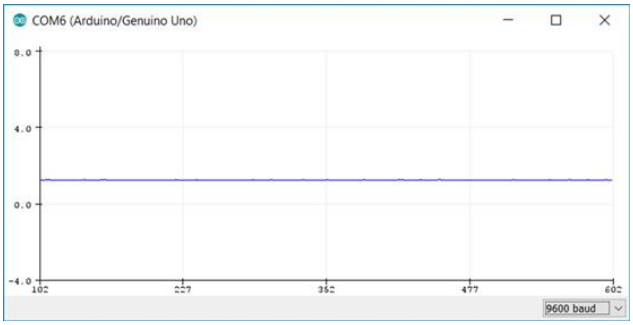
You should see something like this:

The Arduino Software can also plot the data from the serial port. In the next figure there is a plot of the same data that we saw on the serial monitor.

Notice that it plotted our voltage values and it also plotted our ADC values.



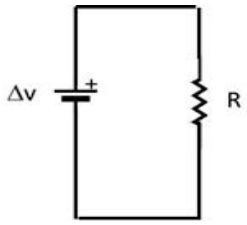
This makes the voltage values hard to see. We could fix this by commenting out the lines that print the ADC values (putting "//" at the beginning of the line). Then those lines won't be executed by the Arduino. Then we get just the voltage.



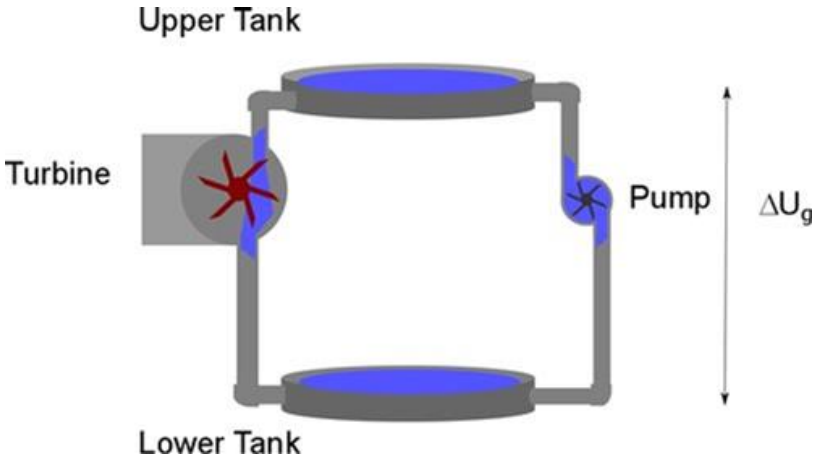
Notice that the horizontal axis is not exactly time. It is just the data point number. We could convert this to time with some calculation if we know how often the Arduino sends us a data point. I will leave this as an exercise.

3.2 Extending our voltmeter with a voltage divider

This Arduino-based voltmeter that we have built is great, but will only let us measure voltages in the range 0 to 5V. That seems a little restrictive. We would like to extend our voltmeter to a larger range, say, 0 to 20V. To do this, we will need to add some electronic components and think about what we have learned about voltage, resistance, and current. Let's consider this circuit.

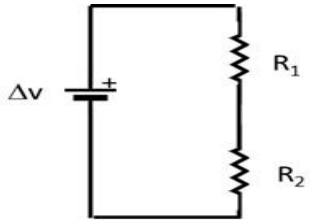


We have a battery, That will make the current flow much like a pump makes water move through pipes.



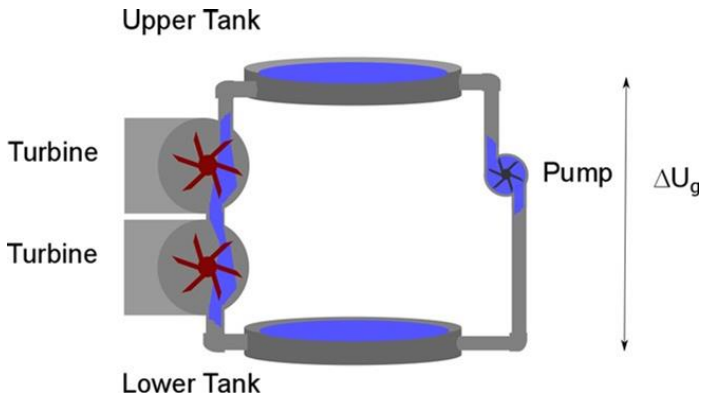
The water in a pipe system gains potential energy as it moves up. In our circuit we will find that electric charge gains potential energy as we move it across a battery. Then the charge will move down the wire like water moves down a pipe until it is out of potential energy. Notice that the water in a pipe system will lose all the potential energy that it gained when the pump raised it to the upper tank (see previous figure). That is true of electric charge too. The electric current travels from the battery through the resistor, but in doing so it loses all the potential energy that the battery gave it by the time it returns to the battery.

Now suppose we have two resistors in a circuit.

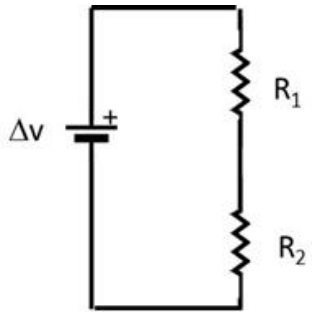


Our water analogy can still help us understand what will happen. Suppose that

we have two turbines in our pipe system.



The water leaves the high potential energy part of the pump, and is put to work turning the first turbine. The resistance of the turbine will slow the water current. So when the water leaves the turbine, it will have lost some potential energy. Since we have a second turbine the current will again be slowed and more potential energy will be lost. How much potential energy do we lose as the water falls? All of the potential energy that the pump gave it! We must end up with the water at the bottom back at the low potential energy. We will find this to be true for our electric circuit as well. We will loose some potential energy as the electrical energy "falls" from the high electric potential "down" the first resistor. After the second resistor, we can guess that we must be back at the low electric potential we started with.



We know that electric potential is a potential energy per unit charge. And energies just add up. If

$$\Delta V = RI$$

is satisfied, then we would expect that adding two resistors would just linearly add the effects of the two resistors together

$$\begin{aligned} \Delta V_{total} &= \Delta V_1 + \Delta V_2 \\ &= R_1 I + R_2 I \end{aligned}$$

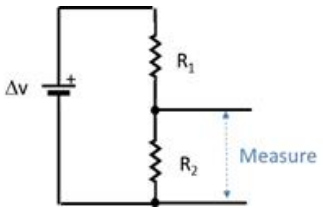
Note that the same current must flow through each of the resistors, since the current leaving R_1 is the current flowing into R_2. Then

$$\Delta V_{total} = (R_1 + R_2)\, I$$

Our current will be

$$I = \frac{\Delta V_{total}}{R_1 + R_2}$$

But suppose we measure the potential change across just resistor R_2.



what would we expect to get? We lost voltage across both ΔV_1 and ΔV_2 so

$$\Delta V_{total} = \Delta V_1 + \Delta V_2$$

because we must loose all the ΔV_{total} given to the current by the battery. And

$$\Delta V_2 = I R_2$$

from Ohm's law. So

$$\Delta V_2 = \left(\frac{\Delta V_{total}}{R_1 + R_2} \right) R_2$$

This is only part of the total voltage. And if we have two different resistors so that $R_1 \neq R_2$ then we can choose for ΔV_2 to be nearly as much as ΔV_{total} or nearly as little as 0 by carefully choosing our two resistances. We call a set of two resistors like this a "voltage divider" because it divides the battery voltage between the two resistors. If R_1 is bigger than R_2 then ΔV_1 is bigger than ΔV_2.

Remember that the input can only withstand 0 to 5V. More than that can destroy the board! But we want to measure a voltage that varies from 0 to 20V. We now have a way to do this. We will use a voltage divider. The voltage across both resistors will be as much as 20V, but we will measure the voltage across only one of the resistors. And we will choose our resistor so that when the total voltage is 20V but the voltage across our resistor is 5V (or less). Since

we will know the resistances, we can use a little math to calculate what the total voltage was using the voltage measurement from just one of the resistors.

This is like what we did to measure current last lab. We used a voltmeter and a resistor and some math to make an ammeter. Today we will use two resistors, our Arduino voltmeter, and some math to make a new voltmeter that can measure higher voltages. We just need to choose our resistors so that we map our 0 to 20V to 0 to 5V. Once choice might be

$$\begin{aligned} R_1 &= 40\text{k}\Omega \\ R_2 &= 10\text{k}\Omega \end{aligned}$$

Let's try it. We would get

$$\begin{aligned} \Delta V_{2\,\max} &= \left(\frac{20\text{V}}{40\text{k}\Omega + 10\text{k}\Omega}\right)(10\text{k}\Omega) \\ &= 4.0\text{V} \end{aligned}$$

when $\Delta V_{total} = 20\text{V}$ and

$$\begin{aligned} \Delta V_{2\,\min} &= \left(\frac{0\text{V}}{40\text{k}\Omega + 10\text{k}\Omega}\right)(10\text{k}\Omega) \\ &= 0\text{V} \end{aligned}$$

when $\Delta V_{total} = 0\text{V}$. Notice that this really didn't work. We only got a maximum voltage of 4V. But this gives us a margin of safety. If we give our Arduino more than 5V we can burn it up. If we plan our circuit so we don't get to close to 5V we are safer. So this set of resisters is not a terrible choice.

To report out our voltage we need to do this conversion backwards. Say we have $\Delta V_{total} = 10\text{V}$ that we are measuring with our new instrument. Then

$$\begin{aligned} \Delta V_2 &= \left(\frac{10\text{V}}{40\text{k}\Omega + 10\text{k}\Omega}\right)(10\text{k}\Omega) \\ &= 2\text{V} \end{aligned}$$

The 2V is what we actually measure at the A0 input. But we know that this represents 10V across both resistors, so we want the Arduino program to print out 10V. So we report

$$\Delta V_{reported} = \frac{\Delta V_2}{R_2}\left(R_1 + R_2\right)$$

or for our case, since we measured 2V across our resistor,

$$10\text{V} = \frac{2\text{V}}{(10\text{k}\Omega)}\left(40\text{k}\Omega + 10\text{k}\Omega\right)$$

We will have to write this math in our code. There is a further complication. The Arduino A0 input is giving us a number that represents 0 to 4V for our setup. But that is not what we see on the serial port. We see a number from 0 to 1024. We know the 1024 represents 5V and the 0 represents 0V. So we need to multiply the number that comes from our Arduino by $\delta V = 4.9\text{mV}$ once again to get our Arduino output into voltage units. So our reported voltage equation is something like this.

$$\Delta V_{reported} = A2D \times \delta V_2 \times \frac{1}{R_2}\left(R_1 + R_2\right)$$

All this calculation to get our reported voltage must do something to our measurement uncertainty. We could do our usual math to find the reported uncertainty, but instead, let's think. Every small voltage ΔV_2 would be multiplied by $\left(\frac{1}{R_2}\left(R_1 + R_2\right)\right)$ to map it into our original 0V to 20V range. That should work for our smallest voltage that we can detect, namely $\delta V = 4.9\text{mV}$. That is the smallest value ΔV_2 could have. So in our 0V to 20V range the smallest value this can map to is

$$\delta V_{reported} = (\delta V)\left(\frac{1}{R_2}\left(R_1 + R_2\right)\right)$$

The first term in parenthesis is essentially 1 (digitizer unit) multiplied by ΔV_{min} and the second term in parenthesis converts the ΔV_{min} value into actual volts measured across both resistors.

The quantity $\delta V_{reported}$ gives us our quantization error value for our new instrument. Our output will be in multiples of

$$V_{reported} = n \times \delta V_{reported}$$

Putting in numbers gives

$$\begin{aligned}
\delta V_{reported} &= \left(4.884 \times 10^{-3}\text{V}\right)\left(\frac{1}{10\text{k}\Omega}\left(40\text{k}\Omega + 10\text{k}\Omega\right)\right) \\
&= 0.024\,42\text{V} \\
&= 24.42\text{mV}
\end{aligned}$$

This is much bigger than our 4.9mV uncertainty for the simple voltmeter. And this is the cost of using a voltage divider to extend our voltage range. For the bigger voltage range we get a bigger uncertainty.

Let's try another example. Suppose we wish to measure 0 to 20V and we look in our case of resistors and find we have the following two resistors to use:

$$\begin{aligned}
R_1 &= 98\text{k}\Omega \\
R_2 &= 15\text{k}\Omega
\end{aligned}$$

We would expect that our 0 to 20V would be mapped to a smaller range. Let's find that range.

$$\begin{aligned} \Delta V_{2\,\max} &= \left(\frac{20\text{V}}{98\text{k}\Omega + 15\text{k}\Omega}\right)(15\text{k}\Omega) \\ &= 2.654\,9\text{V} \end{aligned}$$

So our voltage range at the Arduino A0 input will be 0V to 2.65V. This set of resistors won't use the full Arduino 0V to 5V range. But it will measure 0 to 20V. The minimum detectable voltage for this new instrument design for our 0 to 20V source will be

$$\begin{aligned} deltaV_{reported} &= (\delta V_2)\left(\frac{1}{R_2}(R_1 + R_2)\right) \\ &= \left(4.880\,3 \times 10^{-3}\text{V}\right)\left(\frac{1}{(15\text{k}\Omega)}(98\text{k}\Omega + 15\text{k}\Omega)\right) \\ &= 3.676\,5 \times 10^{-2}\text{V} \\ &= 37\text{mV} \end{aligned}$$

This uncertainty is much bigger than the uncertainty for our last choice of resistors. So 98kΩ and 15kΩ are not great choices even though they technically work.

For your version of the voltmeter in lab, you will choose the resistor values to use. Here is an Arduino sketch to implement this extended volt meter. In it are the not-so-good 98kΩ and 15kΩ, but of course **you should change the sketch to have your resistor values**.

Download here

```
////////////////////////////////////////////////////////
// Extended Voltmeter
// This voltmeter with the values given below
// is designed to measure a 0 to 20V range with 1024
// discrete values of with an uncertainty of about 0.02V
////////////////////////////////////////////////////////
 //set up a variable to represent Analog Input 0
int AI0 = 0;
// Resistance of R1(put in your actual value here)
float R1 = 98000.0;
// Resistance of R2(put in your actual value here)
float R2 = 15000.0;

int ADC_value = 0;     // Place to put the A2D values
float voltage = 0.0;  // calculated signal voltage
//mV Arduino's minimum detectable voltage
```

```
float delta_v_min = 0.0049;

////////////////////////////////////////////////////////
void setup() {
  //Initiate Serial Communication
  Serial.begin(9600);    //9600 baud rate
}

////////////////////////////////////////////////////////
void loop() {
  // read the serial data from AI0
  ADC_value = analogRead(AI0);
  // if you want to, print out the channel A2D values.
  // Uncomment if you want them.
  //Serial.print("analog channel value ");
  //Serial.print(ADC_value);
  // calculate the signal voltage
  voltage=ADC_value*(delta_v_min)*(R1+R2)/R2;
  // print out the signal voltage
  Serial.print("␣voltage␣");
  Serial.println(voltage, 4);
}
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
```

Of course you will want to have another person check your math and wiring, and you should check your output voltage with a stand-alone meter before you plug into your Arduino.

3.3 Practice Problems

Here is an example for you to work out on your own before class. Do this and compare your result to the results of the other people in your lab group as you come into class on lab day.

Suppose we wish to measure 0 to 15V and we look in our case of resistors and find we have the following two resistors to use:

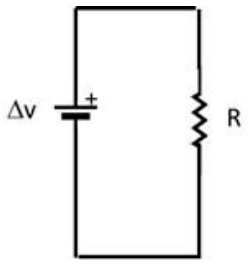
$$\begin{aligned} R_1 &= 43.2\text{k}\Omega \\ R_2 &= 15.2\text{k}\Omega \end{aligned}$$

What range of voltages would we see at the Arduino, and what is the quantization error for our measurement?

3.4 Lab Assignment

1. **Simple Arduino Voltmeter**

   (a) Build a circuit with one of our power supplies and a resistor like we did in our last lab. You can choose any resistor. (see section (2.1).



   This time write the simple voltmeter sketch, wire it up, and measure the voltage across the resistor using our Arduino and the serial monitor. Be careful to stay in the 0 to 5V range!

   (b) Calculate the uncertainty due to quantization error for your Arduino simple voltmeter

   (c) Compare your calculated uncertainty to the measured uncertainty that you see in your device output. (This is tricky, does the power supply give a truly constant voltage?)

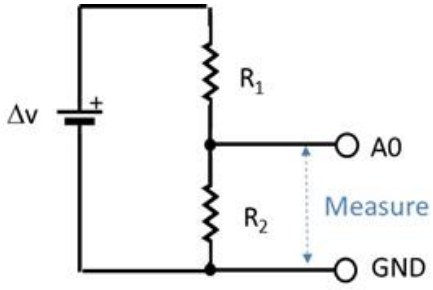
2. **Build a new extended Voltmeter Using a Voltage Divider**

   (a) Build the voltage divider using two resistors as described in section. (3.2). You will have to think about which resistors from our set will work best. Discuss this with your group, or have group members try the calculations with different combinations.

   (b) Use a multimeter to verify that the output of the voltage divider is never more than 5V and never less than 0V. Take your power supply all the way from 0V to 20V and watch the multimeter to ensure it stays in the 0 to 5V. Range. **Do this with a multimeter before**

you hook up your Arduino. You are making sure everything works so you won't destroy your Arduino!

(c) Write the sketch and then hook the output of your voltage divider to the A0 pin and the other side of R_2 to a GND pin.



Your voltmeter should now be set up. Compile and load the sketch and use the Serial Plotter to watch the voltage values as you take the power supply from 0 to 20V using the serial monitor or plotter. **Don't go over** 20V!

(d) What is the quantization error for this voltmeter? Check to see if this matches your values on the serial monitor.

(e) Design a voltage divider that will allow the full 0 to 30V range of our power supply to be measured using the Arduino's 0 to 5V analog input. What would the quantization error be for this new circuit?

3. Together with your group, fill out a "brainstorming sheet" in preparation for your design project later in the semester.

Chapter 4

Getting data to the Computer

We now have voltage data coming from our Arduino into the computer serial port, and we can see that data with the Arduino serial port monitors. But suppose we want to analyze our data in another program. How would we get the data into Excel or LoggerPro, or even our beloved Python that we learned to use in PH150?

There are lots of ways to do this, but an easy way is to use our Python skills and write a simple code using the Python serial port library. That is what we are going to do in this lab.

4.1 How to get Python – Skip if you have python

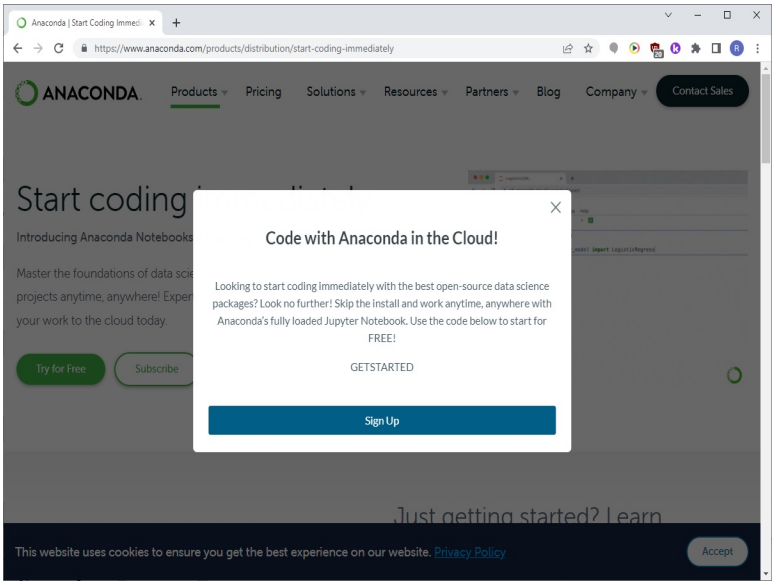
But wait, you might say, I didn't use snakes in PH150. What are you even talking about? Or maybe you got rid of Python because you never thought you would use it again. Whatever the case, Python is another way to make computer code like our Arduino app. Only, this code is designed to work on our computer, itself. This is just what we want for getting the data ready for analysis on our computer. If you already have Python, that is fine, you could skip ahead. If you don't, the next steps show how to get Python on your computer and how to get the Python serial library so you have the commands to read the serial port. Our physics department uses two different distributions of Python. Canopy, and Anaconda. But both work fine. But if you don't have a distribution, I encourage you to try the Anaconda version. Instructions for downloading and installing are given next.

4.1.1 Getting Anaconda Python

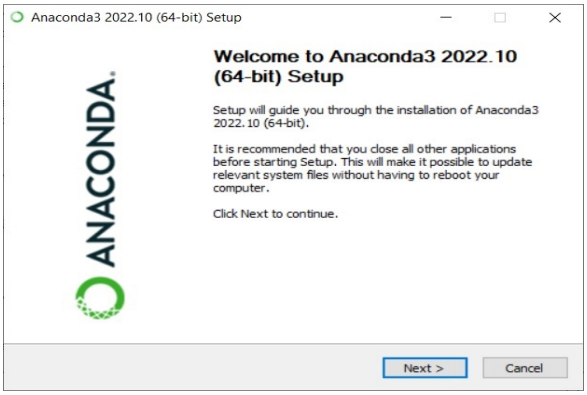
The Anaconda distribution of Python is designed for scientific work (so it has most of the science libraries of functions already installed) It can be found at

https://www.continuum.io/downloads. There is an install link for Windows, Mac, and Linux. Choose the one for your operating system[1]

If you are runing Windows or Mac, when you click on the green download button it will download an executable file. Meanwhile the website tries to get you to subscribe, and offers you the opportunity to code with Anaconda in the cloud. I encourage you to ignore these opportunities and to just use the executable file to install python. In windows it looks like this

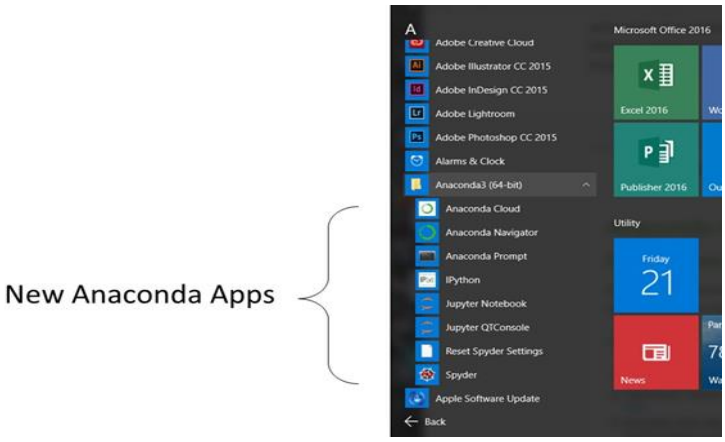


Choose "Save File" and when the file is downloaded open it to start the installation. You should see something like this:

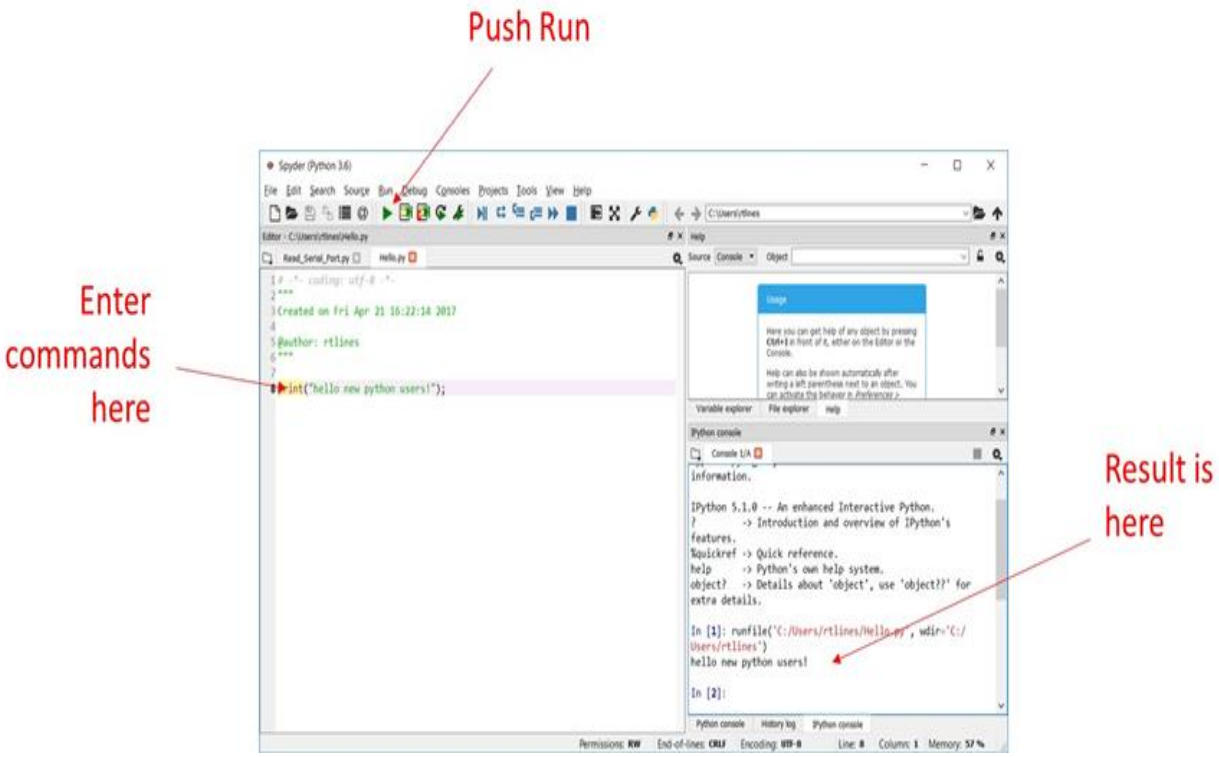


[1]If you have a Linux computer, it is likely that you will want to actually see if Anaconda is in your distribution's repository. If you are not a Linux user and you have no idea what that means you can ignore it.

Choose "Next" and follow the installation instructions. If all goes well, you should have a new set of apps. Here is what mine looked like on my Windows 10 computer.



The list of the apps has an app called Spyder. Let's launch it to see what it does.



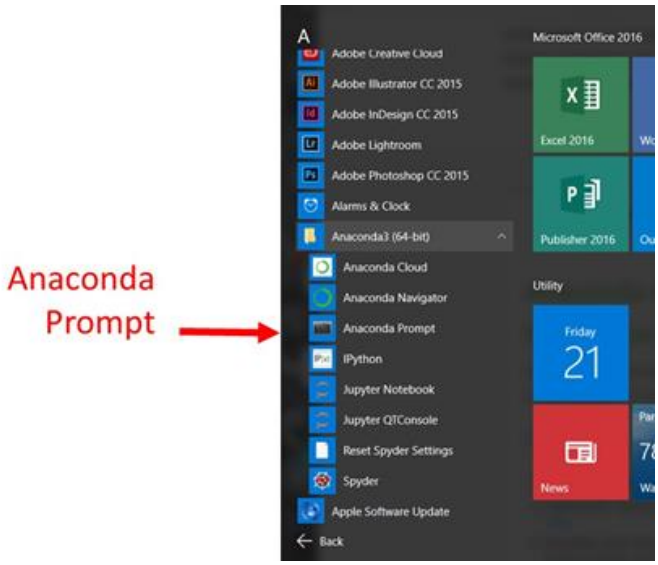
What we get is something like the Arduino program that we have been using to write Arduino sketches. Spyder is a place to write and run Python commands. Only this time there is no checking and uploading the code, because the Python code will run on our computer, not on an Arduino.

In the figure above, the command just says to print "hello new python users!" and that is all. When it runs, it prints our message on the small window to the right. But of course Python can do much more than print silly messages in little windows. We will have our Python system read a serial port and save our data to a file. But we need an additional piece of Python to do this. We need functions that can handle serial ports. These functions are already written by someone, and put together in a package called a "library." But this library isn't included in what we have downloaded. So we need to fix this next.

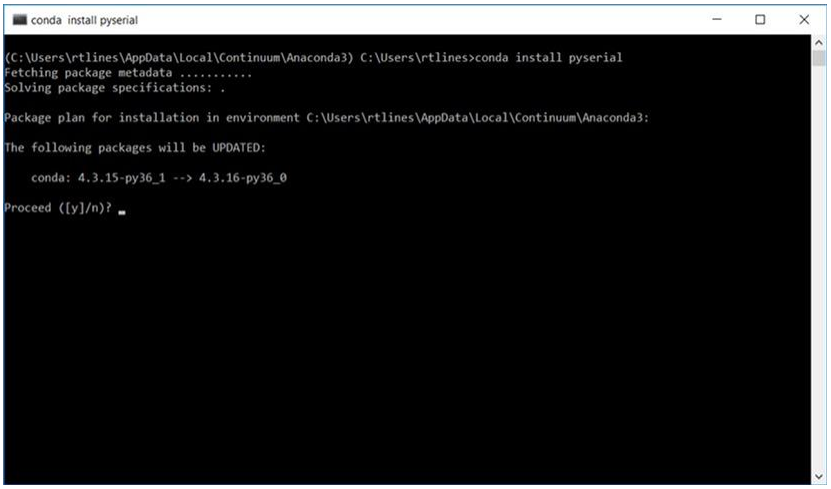
4.2 Getting the PySerial library for Anaconda – Don't skip this

Now that we have Python, we need to enhance it with the PySerial library so that we can read our data from the computer serial port. If you have installed the Anaconda package as your Python distribution, follow along here. If you have a different distribution of python, ask your instructor for help.

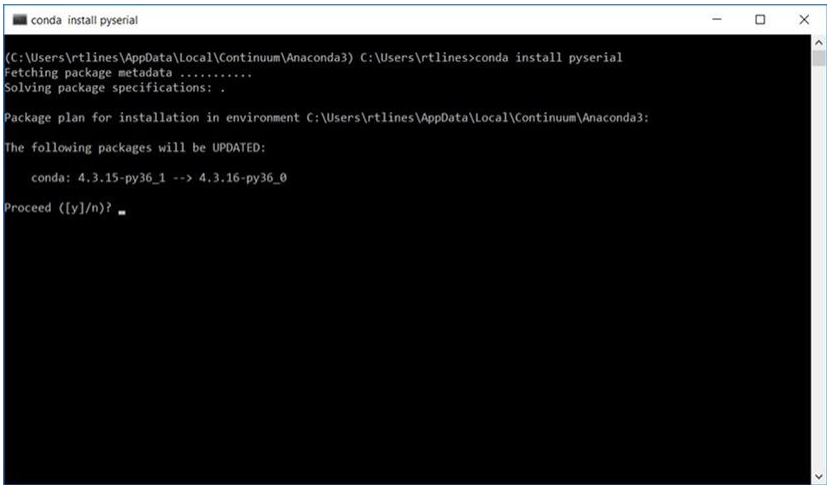
We will use the Anaconda prompt to get the PySerial library. The Anaconda prompt is an app that let's us modify the Python libraries that we have installed. The Anaconda prompt is found in the list of apps that were installed in Anaconda. If you are using Windows, you might find it in your app list like this.



Once you launch the Anaconda prompt it just looks like a big black box. We can type commands in that box that will modify the Anaconda Python programs that we installed. In our case we want to type in the command `conda install pyserial`.

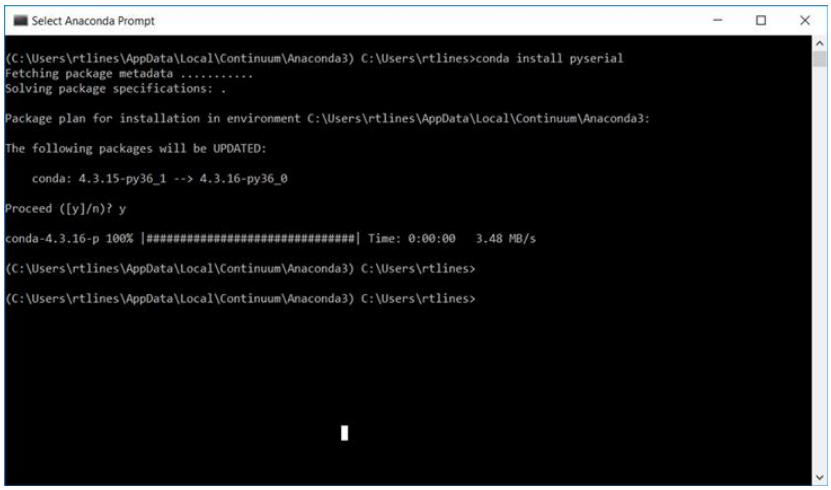


Notice that the Anaconda prompt app responds to our command. What it responds will depend on your computer and your Python distribution. You need a network connection to install new libraries, so if you get an error, you may just not be connected to a network.



If you already have PySerial installed, you will be told so, or asked if you wish to update it if an update is available. If you don't have PySerial, it will ask you if you want to install it. Answer "yes" and PySerial will be installed. If any error messages are generated, ask for help from your instructor. Hopefully

you will see a happy end result like this



and you will be all ready to start writing code to get data from the serial port.

4.3 Getting data from the Arduino

It might help to know how a serial port works. The serial port in the computer takes in data from the serial cable. It stores the data in a temporary place in memory called a *buffer*. Whatever data comes into the port goes into that memory location.

Suppose that we set up our Arduino to send data to the port. The Arduino sends data every few milliseconds. But, suppose we only want data every few seconds. We only want some of the data that the Arduino has sent.

The computer has placed every data point sent by the Arduino into it's buffer, and the buffer must be read in sequence. You can't skip data values that are in the buffer. But this is just what we want to do, skip some data values and take a value every few seconds. A way to do this is to continuously read in data, but only store it every so often. We will use this technique in the code that follows.

Just for fun, let's think of an actual data collection. Suppose we want to measure a voltage from our Arduino, but we want to measure it many times, each time five seconds apart. This way we are looking at how the voltage changes over time. We might want 10 total measurements.

You might be an expert Python programmer, and if so you can probably see how to write this code. If so, go ahead and do so. But if not, let me introduce some of the Python code elements we will need, and then give an example code.

The first new code piece is making files for our data. We create a file with a line like this (see the actual code below):

```
fileObject=open("C:\\Users\\rtlines\\Documents\\data.txt","w")
```

The "fileObject" is a variable that contains all the file information like the path and file name. It is way easier to type than to include all that information each time we use a file. So we will use fileObject variables. In the code below I named the fileObject variable "dataFile." Of course, you will have to choose your own path where you will place the data file (you can't use mine, because you don't have my computer!) and you will need to choose your own file name. Notice the weird double slashes \\." These are a Python thing on Windows computers and you probably need to write the path this way if you have a Windows based computer. It is different in Mac and Linux (more on this below).

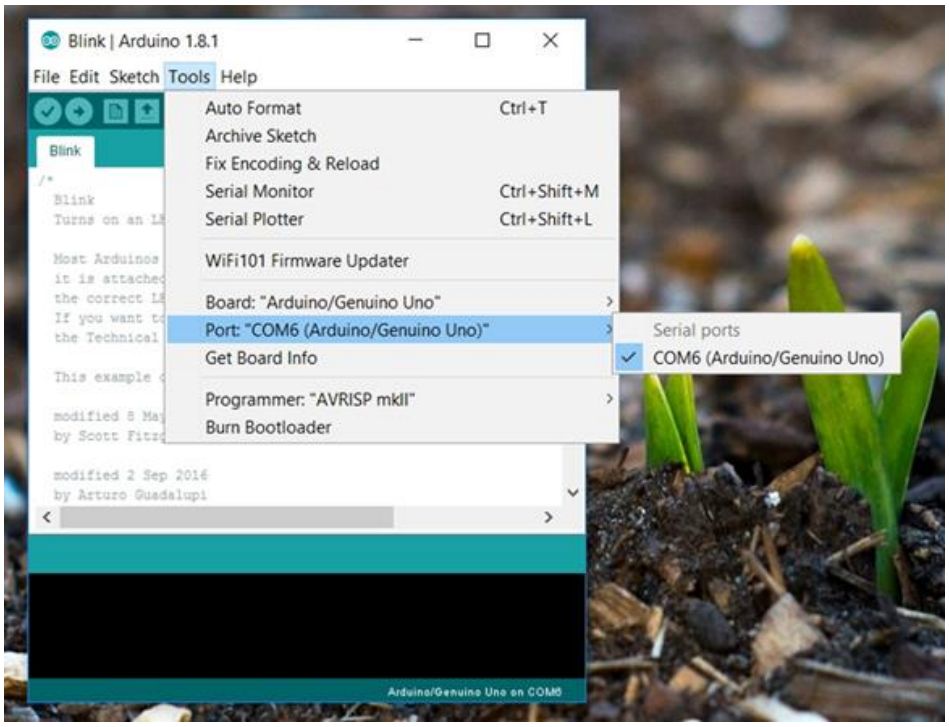
The other new code piece is dealing with serial ports. Like with our Arduino code, we have to set up the serial port. We do that with the line like the this (also see the actual code below).

```
ser=serial.Serial('COM6', baudrate = 9600, timeout=1)
```

Note that when I was using this code my Arduino was in COM6. But that might not be true for you. Use the Arduino app to find out where your Arduino is connected.

The COM port number must match. This looks a little uglier on a Mac, but is much the same. Once the serial port is set up, a line like

```
arduinoData=ser.readline().decode('ascii')
```



will read data from the serial port and "decode it" so that it is text that we can use in another program. The rest of the example code writes the data point to a file. I have the Areduino code calculate a value in miliseconds for when the data points are taken. This is passed along with the voltage through the serial port. From this we could calculate the time since the beginning of our data collection (we might just need that in a future lab). This time in milliseconds is output into the file as well.

We are going to want a name for the amount of time in between data points. In the example code, the amount of time to delay in between data points is named `sleeptime` and the number of data points is named `N`.

At the end of the program we close the file

```
fileObject.close()
```

```
ser.close()
```

so that it will be ready to use next time. Your complete code might look something like this.

Download here

```
#---------------------------------------------------------------------
# Python Code to read a stream of data from the serial port
```

```
#   and save it to a file
#----------------------------------------------------------------------
#   The idea is to read a series of voltages from an Arduino
#   connected to the serial port where the Arduino is being
#   used as the Analog to Digital converter. Both the voltage
#   and the time the voltage was taken are sent to the serial port.
#
# We will use two libraries, serial and time
#   The serial library is used to read the serial port
#   The time library is used to space out our data collection by
#   adding a delay in between data points. The amount of time
#   to wait in between data points is called "timeBetween."
#
# We may have to install the serial library. If you have the
#   Anaconda Python for Windows, you can open an Anaconda
#   window and use the command 'conda install pyserial'
#   This must be done before the code can run.
#
# Debugging issues:  The Anaconda Python distribution tends to
#   hang on to the serial port even if the program does not run.
#   If this happens, try sending the python command ser.close()
#   at the command prompt. If this doesn't work, You may have to
#   restart Python.
#   In windows, closing (after saving) the IDE and reopening it
#   might be enough.
#----------------------------------------------------------------------
# import libraries
import serial
import time

# define variables for the delay time we wait between data points
timeBetween=5 #seconds

# define the number of data points to collect
N=20

#the next line opens a file and creates a pointer or handle for that
                                    file
#  to use as a reference. You have to use double slashes in the path.
#  The pointer, "dataFile" takes the place of all the path and file
#  name so it is easier to use in the code below
#  This line worked for Brother Lines, but won't work for you as it is.
#  You need to replace "rtlines" with your username at a minimum.
dataFile=open('C:\\Users\\rtlines\\Documents\\data2.txt','w')

#the next line opens the serial port for communication. You need to
#  set the COM port based on where the Arduino IDE says the Arduino
#  is plugged into your computer. On Windows, this is usually COM1
#  through COM 8.
ser=serial.Serial('COM3', baudrate = 9600, timeout=1)

#there will be a delay before the first data point comes from the
#  serial port, warn the user so they don't worry.
print('\n getting started...\n')

# set our index to zero
i=0
```

```
# Now for N points (set above), collect data and write it to a file
while (i<N):    #Begin data collection loop
    #We will take data every "timeBetween" seconds. We need to know
    #   when we start waiting so we can tell if it is time to collect
    #   data yet. Use the time.time() to get the current time in
                                        seconds
    #   since Jan 1,1970. Yes that is a weird way to measure time, but
    #   computers do it this way.
    waitStart=time.time()

    #Data comes to the serial port fast. We will continually read
    #  the data as fast as it comes, but only save it every timeBetween
    #  seconds. The next while loop keeps us reading in data, but only
    #  when the current time - waitStart >= timeBetweem will we use
    #  the data. All the data points inbetween get thrown away.
    while (time.time()-waitStart<timeBetween): #Begin Data read loop
         # Get data from the serial port
         # it should have a time and a voltage
         arduinoData=ser.readline().decode('ascii')
         # end of the Data read loop

    # the next line just prints the time and voltageon the console
    #   so the user feels like something is happening.
    print(arduinoData)
    # This next line makes sure our Arduino data is in a python
    #   string format
    writeString=str(arduinoData) #+ " \n"
    # The next line writes our Arduino data value to the file.
    dataFile.write(writeString)
    # And finely we increment the loop counter so we can see if we have
    #  collected data N times adnd are ready to quit.
    i=i+1      # end Data collection loop

# Print out a message saying we are done
print("done with data collection, closing the  file and the serial port
                                    ")
# Close the file
dataFile.close()
# Close the serial port
ser.close()
#---------------------------------------------------------------------
#---------------------------------------------------------------------
```

Make sure you understand what each line does. Discuss each line with a group member or with the instructor. Python code runs one line at a time. That is different than our Arduino code that must be checked and translated before it goes to the Arduino. Errors in Python code show up as the code runs. If you use the Spyder IDE, the errors show up in the little output box to the right. If you use Canopy they show up in the lower box of the editor.

I modified my simple voltmeter to give the time that the data was taken and to send both the time and the voltage to the serial port. Here is my sketch (remember since it is a simple voltmeter it can only handle 0V to +5V.)

Download here

```
////////////////////////////////////////////////////////
// very simple voltmeter that also returns the time since
// the data collection started with the voltage.
// will measure 0 to 5V only!
// Voltages outside 0 to 5V will destroy your Arduino!!!
////////////////////////////////////////////////////////
  int AI0 = 0;
  float delta_v_min=0.0049;   // volts per A2D unit
  int value = 0;
  float voltage = 0.0;

////////////////////////////////////////////////////////
   void setup() {
      // put your setup code here, to run once:
      //Initiate Serial Communication
      Serial.begin(9600);    //9600 baud rate
    }

////////////////////////////////////////////////////////
   void loop() {
      // read in the voltage
      // in A2D units form the serial port
      value = analogRead(AI0);
      Serial.print("␣time␣in␣milliseconds␣");
      // the millis() function gives the time
      // in milliseconds since the sketch started
      Serial.print(millis());
      // convert to voltage units using delta_v_min
      voltage = value * delta_v_min;
      Serial.print("␣voltage␣");
      Serial.println(voltage, 4);
  }
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
```

You can't use the Arduino serial monitor and get data from the serial port using Python at the same time. So turn off the serial monitor if it is running.

Now that we have our data in a file, we can analyze it. You might try opening the file in Excel or another spreadsheet program and plotting the data. If you know Python, you could add more code to plot the data right in the code that takes it from the serial port. But, if you are new to Python, you could plot the data in something like a Excel or LoggerPro. Ask for help if you don't know how to do this. We will plot data in future labs.

4.4 Getting Pyserial if you have a Mac – skip if you don't have a Mac

Of course, we have a diversity of computers on campus. The instructions I gave above are for a PC type computer.

4.4.1 Anaconda Mac Users 4.1.2

Go to the Anaconda Prompt and type in: *conda install -c anaconda pyserial*. This will install pyserial -v3.4.

If this does not work for some reason, try going to the Anaconda Navigator, on the left hand side, click Environments. Next, on the right hand side, change the drop-down from Installed to Not Installed. Then enter serial in the search box. Check the box for pyserial and click the Apply button. If you have Spyder running, then relaunch Spyder.

4.4.2 Canopy Mac Users 4.2.1 – skip if you don't have Canopy

The steps are the same as the Microsoft Version.

4.4.3 Manually install pyserial through the terminal on a Mac. Codingin emacs/VI

This is kind of a "last resort" approach, so only try this if the other methods above fail.

1. First, go to *https://pypi.python.org/pypi/pyserial* and download *pyserial-3.4.tar.gz* or the version that correlates with the version of python you are using. If you are using python 2 then instead of 3.4, you should look for a file that starts with 2.x; however, if you are using python 3 then 3.4 is the correct version of pyserial you are looking for (but please consider using python 3.x!).
2. Be sure that you downloaded to your Downloads folder.
3. Go to your search bar and type in *terminal* and open that application.
4. Type into the command line *cd Downloads* then press enter.
5. Next, type in *tar -xzf pyserial-3.4.tar.gz* then press enter. Type in *cd pyserial-3.4*, press enter
6. Then type in *sudo python3 setup.py install*.
7. If you are using python 2 then only type in *python* where it says *python3*.

After that you are ready to use the serial library in python.

4.4.4 Mac Pathway and Port Notation – Skip if you don't have a Mac

Mac computers list paths to files differently than PC computers.

Mac Pathway

In fact, Mac computers don't make file locations obvious to users at all! But our python code needs to know where to put the files we build, so we will need to understand how to do this.

On the line of code that starts with *dataFile* you will replace it with *dataFile = open("/Users/rtlines/Documents/data.csv","w")*. This is in the form of /Users/username/folder name/(optional if you have a folder within the previous folder) folder name/file name + extension (e.g. .csv or .txt).

You can find your username by going to your download folder then right click on any file in there and select **Get Info**. Next make sure that the arrow to the left of *General* is pointing down. Once it is pointing down look at the line that reads, **Where: Macintosh HD → Users → your username will be here → Downloads**.

Mac Port

Mac computers also deal with serial ports differently. So we need to change that next line of code after the dataFile line that begins with *ser = serial....* The line of code will need look something like *ser = serial.Serial('/dev/cu.usbmodem1411', baudrate = 9600, timeout = 1)*. However, yours may vary slightly by a different usbmodem number. To find out what to place between the apostrophes is by going to your arduino code, click on **Tools**, scroll down to **Port** and write down what is written there minus what is in the parenthesis.

4.4.5 Mac version of the python code

Download here

```
#-------------------------------------------------------------------
# Python Code to read a stream of data from the serial port
#   and save it to a file
#-------------------------------------------------------------------
#   The idea is to read a series of voltages from an Arduino
#   connected to the serial port where the Arduino is being
#   used as the Analog to Digital converter. Both the voltage
#   and the time the voltage was taken are sent to the serial port.
#
# We will use two libraries, serial and time
#   The serial library is used to read the serial port
#   The time library is used to space out our data collection by
#   adding a delay in between data points. The amount of time
#   to wait in between data points is called "timeBetween."
#
# We may have to install the serial library. If you have the
#   Anaconda Python for Windows, you can open an Anaconda
```

```
#   window and use the command 'conda install pyserial'
#   This must be done before the code can run.
#
# Debugging issues:  The Anaconda Python distribution tends to
#   hang on to the serial port even if the program does not run.
#   If this happens, try sending the python command ser.close()
#   at the command prompt. If this doesn't work, You may have to
#   restart Python.
#   In windows, closing (after saving) the IDE and reopening it
#   might be enough.
#-----------------------------------------------------------------------
# import libraries
import serial
import time

# define variables for the delay time we wait between data points
timeBetween=5 #seconds

# define the number of data points to collect
N=20

#the next line opens a file and creates a pointer or handle for that
                                        file
#  to use as a reference. You have to use double slashes in the path.
#  The pointer, "dataFile" takes the place of all the path and file
#  name so it is easier to use in the code below
#  This line worked for Brother Lines, but won't work for you as it is.
#  You need to replace "rtlines" with your username at a minimum.
#  PC version next:
#  dataFile=open("C:\\Users\\rtlines\\Documents\\data.txt","w")
#  MAC version next
dataFile = open('/Users/rtlines/Documents/data.csv','w')


#the next line opens the serial port for communication
#  PC version next
#ser=serial.Serial('COM3', baudrate = 9600, timeout=1)
#  MAC version next
ser = serial.Serial('/dev/cu.usbmodem1411', baudrate = 9600, timeout =
                                        1)


#there will be a delay before the first data point comes from the
#  serial port, warn the user so they don't worry.
print('getting started...')

# set our index to zero
i=0


# Now for N points, collect data and write it to a file
while (i<N):    #Begin data collection loop
    #We will take data every "timeBetween" seconds. We need to know
    #   when we start waiting so we can tell if it is time to collect
    #   data yet. Use the time.time() to get the current time in
                                            seconds
    #   since Jan 1,1970. Yes that is a weird way to measure time, but
```

```
    #   computers do it this way.
    waitStart=time.time()

    #Data comes to the serial port fast. We will continually read
    #  the data as fast as it comes, but only save it every timeBetween
    #  seconds. The next while loop keeps us reading in data, but only
    #  when the current time - waitStart >= timeBetweem will we use
    #  the data.
    while (time.time()-waitStart<timeBetween): #Begin Data read loop
         # Get data from the serial port
         # it should have a time and a voltage
         arduinoData=ser.readline().decode('ascii')
         # end of the Data read loop

    # the next line just prints the voltage point on the console so the
                                        user
    # feels like something is happening.
    print(arduinoData)
    # This next line makes sure our Arduino data is in the proper
    #   python string format
    writeString=str(arduinoData) #+ " \n"
    # The next line writes our time plus Arduino value to the file.
    dataFile.write(writeString)
    # and finely we increment the loop counter
    i=i+1       # end Data collection loop

# Print out a message saying we are done
print("done with data collection, closing the file and the serial port"
                                      )
# Close the file
dataFile.close()
# Close the serial port
ser.close()
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
```

Make sure you understand each line of this code. It is a little fancy in that it tries to check to make sure the SD card file is working properly and warns you if something is wrong. But most of the code is comments. So don't be discouraged by the length.

4.5 Lab Assignment – Really don't skip

1. Finish any part of the last lab that you haven't done.
2. Using Python, Save Arduino data to a file on your computer
   (a) Wire up a simple voltmeter and load its sketch. Build a simple circuit to test like we did back in section (2.1). Start your Arduino and check to make sure voltages are going to the serial port by looking at the serial monitor or plotter.
   (b) Check with your group members to make sure their simple voltmeters are working. Help if they are not.

(c) Start your Python system (Spyder or Canopy if you are following the instructions given above) and write the program to read the serial port and save the data to a file.

(d) Close the serial monitor and/or serial plotter. Then run your Python code. Check to make sure that the file of data is written properly. The voltages written in the file should match what your power supply and circuit provided.

(e) Make sure you save this program and record what you did in your lab notebook.

(f) Make sure your lab group members all have Python programs that run and save data correctly. Help if they do not.

(g) Rejoice! This is a major outcome of the course and you just did it!

Part II

Testing Models

Chapter 5

Validation of Ohm's Law

What physicists do is to try to understand how the universe works. To do this we use the Scientific Method. And what makes the scientific method different from philosophy is the use of experimentation to verify our ideas.

So in a physics lab class, we need to test ideas about how the universe works. We call these ideas "mental models" or just "models." We have been using one of these models in making voltage measuring devices already. It was called Ohm's law. Let's start out by testing Ohm's law to see if it really works.

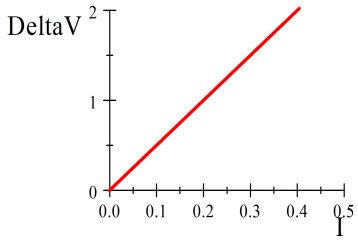
5.1 Ohm's Law Revisited

We learned several labs ago that voltage and current are linearly related to each other. This is what we would call a *model*, a mental understanding of how part of the universe works. Usually in physics we distill the model into an equation. We call this equation a law. In this case, Ohm's law.

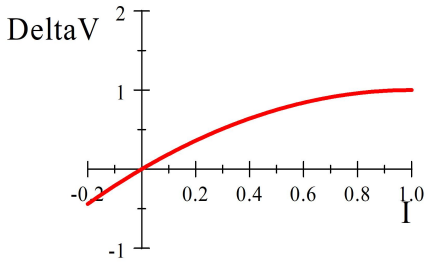
$$\Delta V = IR$$

where R is the slope of the ΔV vs. I curve. We can see the model relationship between ΔV and I reflected in the equation. Note that being a "law" doesn't mean the equation is always true. The word "law" generally implies that the equation is true at least some of the time, but really it is telling us we have distilled our model into math.

We might plot our equation to show the ΔV vs. I relationship.



But as scientists, we should ask, does our model work for all materials? What if we graphed ΔV vs I for some device and found a graph that looks like this?



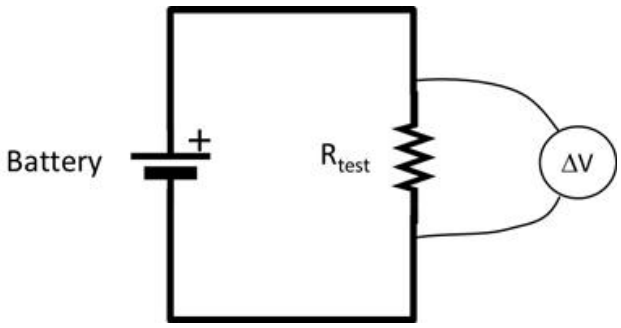
Such a device would *not* follow Ohm's law. We would say that such a device is *nonohmic*.

Today we will test our model by taking ΔV and I measurements and seeing if the equation $\Delta V = IR$ describes the data well. Of course, this means we need to measure two things at once with our Arduino. We need both ΔV and I. But this isn't a problem because our Arduninos have five analog inputs. So we just need to have one measurement attached to, say, pin A0 and another to, say, pin A1. Of course both will need to be connected to GND as the second measurement because ΔV measurements take two leads.

But wait, if we are testing Ohm's law, we don't want two ΔV measurements, we want ΔV and I. How do we get I measured by an Arduino?

5.1.1 Measuring current with our Arduino

Arduinos and other DAQs only measure voltages. Let's review how we measure the voltage across a resistor, and then review how to turn that voltage measurement into a current measurement.



We put the two leads of a voltmeter (shown as a circle with a ΔV in it) on either side of the resistor that we are testing. If the voltmeter is our Arduino, the leads on the side of the resister connected to the positive side of the battery should go to A0 and the lead connected to the negative side of the battery

should be connected to GND. This is all what an Arduino (or any other DAQ) can do.

To measure a current with our Arduino we have to somehow turn that current into a voltage. This is true of most measurements we will do. We need to turn temperature, or humidity, or magnetic field, or light intensity into a voltage. Turning magnetic field into a voltage is a little tricky, but we already know all we need to know to handle current.

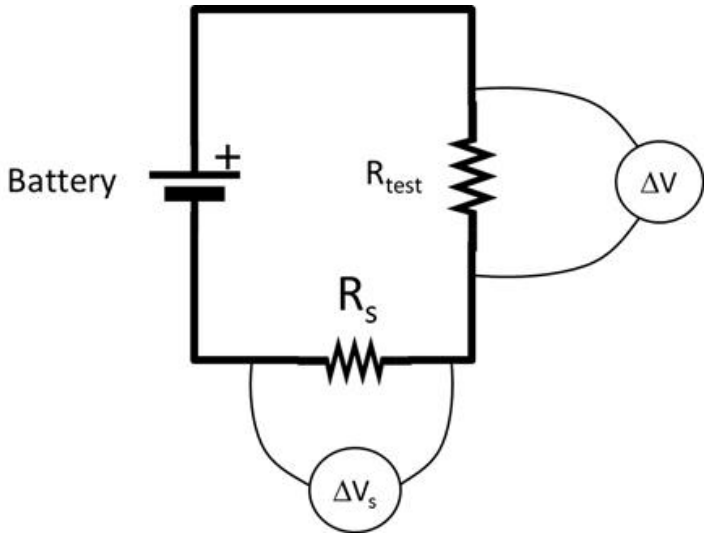
To turn a current into a voltage, think of Ohm's law again.

$$\Delta V_s = I R_s$$

we can solve for I

$$I = \frac{\Delta V_s}{R_s}$$

so if we add a new resistor, R_s to the circuit,



and measure the voltage across that circuit, we will be able to calculate the current. This is familiar from an earlier lab. We called this extra resistor a "shunt" resistor.

Of course, if R_s is very large, then R_s, itself, will slow down the current. So we want to choose a R_s that is much less than R_{test}.

$$R_s \ll R_{test}$$

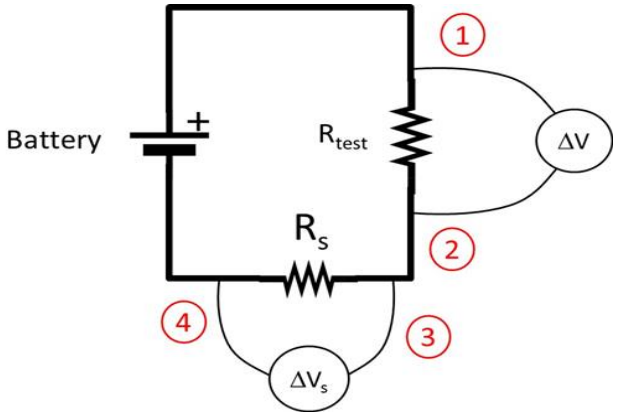
But so long as this is true, our R_s won't change the current much, and since we know R_s we know the current

$$I = \frac{\Delta V_s}{R_s}$$

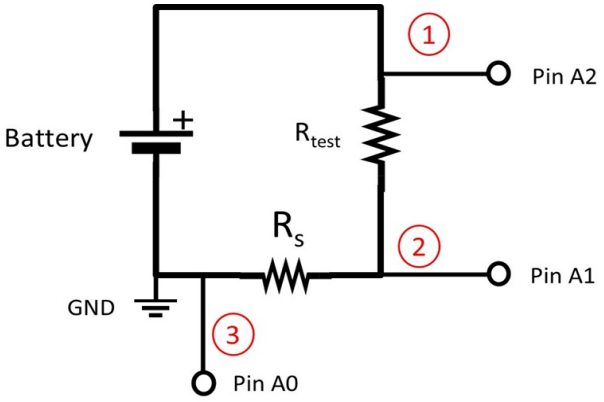
We have turned our current measurement into a voltage measurement!

5.1.2 Actually making an Arduino measure current

This idea is great, but let's talk a little bit about how to wire this dual measurement. Think again about our two voltage measurements, ΔV and ΔV_s. Each Δ implies two measurements. That means we need a total of four measurements to make this work! Let's see where these measurements would be on our circuit diagram.



By drawing the diagram, we realize that we can create both ΔV measurements with only three individual voltage measurements because the voltage at point 2 and the voltage at point 3 should be the same. So we could wire our circuit like this:



and of course we need to wire the negative pole of the battery or power supply to the GND pin. Then our two voltage difference measurements will be formed from

$$\begin{aligned} \Delta V &= V_{A2} - V_{A1} \\ \Delta V_s &= V_{A1} - V_{A0} \end{aligned}$$

If we keep our voltage from our battery or power supply in the 0V to +5V range, then we can use our simple voltmeter sketch. We do need to modify it to take three different voltage measurements. And we need to add the math to make ΔV_s into I. We could even modify this so that our code would report out

$$R = \frac{\Delta V}{I}$$

and we might as well. Here is an example sketch.

Download here

```
 ////////////////////////////////////////////////////////
// very simple voltmeter and equally simple ammeter
// will measure 0 to 5V only!
// Voltages outside 0 to 5V will destroy your Arduino!!!
// Delta_V_shunt must therefore be much much less than 5V
// The shunt resistor should be much less than the
// resistance of the rest of the circuit.
////////////////////////////////////////////////////////
// Shunt resistor value goes here:
  float R_shunt= 1000;
    //ohms  -  remember you have to replace this with your
    // actual shunt resistor value

// make some integer variables that identify
//the analog input pins we will use:
   int AI0 = 0;
   int AI1 = 1;
   int AI2 = 2;

// you also need a place to put the analog to
//digital converter values from the Arduino
   int ADC0 = 0;
   int ADC1 = 0;
   int ADC2 = 0;

// Remember we will have to convert from Analog to
// digital converter(ADC) units to volts. We need
// our delta_V_min just like we did in lab 3
   float delta_v_min=0.0049;   // volts per A2D unit

// We need a place to put the
// calculated voltage and current.
```

```
float voltage = 0.0;
float amperage = 0.0;

////////////////////////////////////////////////////////
 void setup() {
  // put your setup code here, to run once:
  //Initiate Serial Communication
  Serial.begin(9600);    //9600 baud rate
}
////////////////////////////////////////////////////////
 void loop() {
  // Read in the voltages in A2D units form the
  // Arduino Analog pins
  ADC0 = analogRead(AI0);
  ADC1 = analogRead(AI1);
  ADC2 = analogRead(AI2);

  // Convert the voltage across the
  //   test resistor to voltage
  //   units using delta_v_min
  voltage = (ADC2-ADC1) * delta_v_min;

  // Convert the voltage across R_shunt to voltage units
  //   using delt_v_min, then convert to
  //   current using R_shunt
  amperage = (ADC1-ADC0)* delta_v_min / R_shunt;

  // output the voltage, amperage, and resistance
  Serial.print("␣voltage␣");
  Serial.print(voltage, 6);
  Serial.print("␣amperage␣");
  Serial.print(amperage,6);

}
```

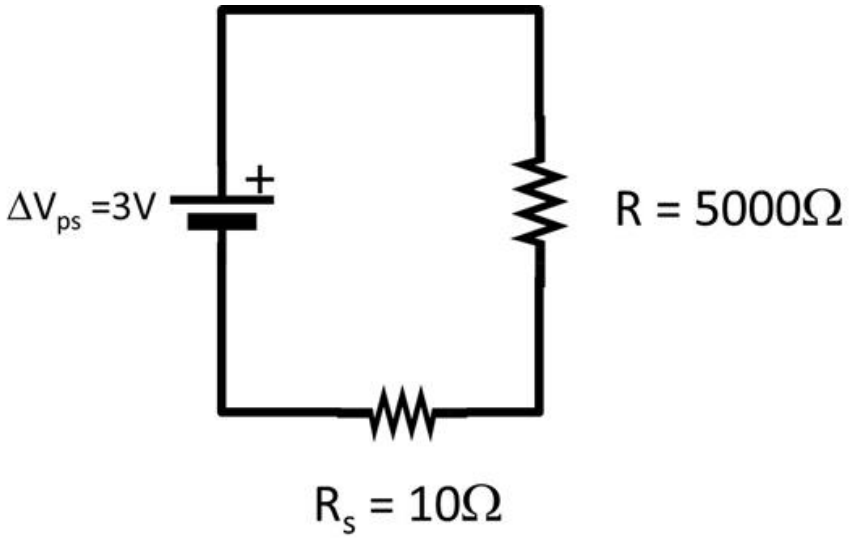
Choosing shunt resistors

It's harder than you might think to choose a good shunt resistor. The shunt resistor resistance shouldn't be big enough to cause too much error in our ΔV measurement for the resistor we want to measure (Think of our voltage divider, the two resistors will share the voltage from the battery, so every bit of ΔV_{shunt} is error in our measurement of ΔV_{test}).

The shunt resistance should also be small enough that it doesn't effect the current much. But suppose we find the smallest resistor that we can, say, 10Ω. Surely that will not affect the actual ΔV or I measurements. But still, we may

have a problem. Let's consider an actual circuit to see why.

Suppose we have a circuit where the input voltage from the power supply is $\Delta V_{ps} = 2\text{V}$ and our test resistor is $R = 5000\Omega$. And suppose we try to use $R_s = 10\Omega$.



The two resistors together are a voltage divider. So we expect the voltage drop across each resistor to sum to the voltage given by the power supply

$$\Delta V_{ps} = \Delta V_R + \Delta V_s$$

and we know the current will be the same in the entire circuit. We can use Ohm's law to find the current.

$$\Delta V_{ps} = IR_{total}$$

so that

$$\begin{aligned} I &= \frac{\Delta V_{ps}}{R_{total}} \\ &= \frac{\Delta V_{ps}}{R + R_s} \end{aligned}$$

Now we can find the voltage drop across just R_s

$$\begin{aligned} \Delta V_s &= IR_s \\ &= \left(\frac{\Delta V_{ps}}{R + R_s}\right) R_s \end{aligned}$$

and let's put in numbers

$$\begin{aligned} \Delta V_s &= \left(\frac{2\text{V}}{5000\Omega + 10\Omega}\right)(10\Omega) \\ &= 3.992 \times 10^{-3}\text{V} \\ &= 3.992\text{mV} \end{aligned}$$

Remember that for our simple voltmeter,

$$\Delta V_{\text{min}} = \frac{5\text{V}}{1024} = 4.88\text{mV}$$

and this is larger than ΔV_s so once we use our Arduino analog to digital converter ΔV_s will appear to be zero! Our current meter that we built will say our current measurement will be zero even thought there is a current flowing. That is a 100% error!

We might try to improve things by increasing the power supply voltage. Even if we increased the voltage from the power supply to, say, 5V (our maximum) we would only have

$$\begin{aligned} \Delta V_s &= \left(\frac{5\text{V}}{5000\Omega + 10\Omega}\right)(10\Omega) \\ &= 9.98\text{mV} \end{aligned}$$

We should compare this value to our ADC minimum detectable value

$$\frac{\Delta V_s}{\Delta V_{\text{min}}} = N$$

the number of ADC units that will be used. We can see that for $R_s = 10\Omega$

$$N = \frac{9.98\text{mV}}{4.88\text{mV}} = 2$$

ADC units. With our entire value of ΔV_s split into only two numbers our uncertainty in our ΔV_s would be something like 50%. That won't make a very good current measurement.

Suppose instead, we use $R_s = 170\Omega$. This is much bigger, so it will affect the voltage measurement of the test resistor a little. But in the end will work better. If we set our power supply back to $\Delta V_{ps} = 2\text{V}$ the $R_s = 170\Omega$ gives.

$$\begin{aligned} \Delta V_s &= \left(\frac{2\text{V}}{5000\Omega + 170\Omega}\right)(170\Omega) \\ &= 65.764\text{mV} \end{aligned}$$

This would give

$$N = \frac{65.764\text{mV}}{4.88\text{mV}} = 13.476$$

or about 13 ADC units spread across our 65.764mV. Then each of our ADC units would be worth

$$\delta\Delta V_s = \frac{65.764\text{mV}}{13} = 5.1\text{mV}$$

This is very near the $\Delta V_{\min} = 4.88\,\text{mV}$ value, but a little bit higher. If $\delta\Delta V_s$ from our calculation is larger than $\delta V_{\min,}$ then we have to use the larger value as our uncertainty in ΔV_s. So we would say $\delta\Delta V_s = 5.1\text{mV}$. But still this is not a terrible error. Now our percent error is

$$100 \times \frac{5.0588\text{mV}}{65.764\text{mV}} = 7.7\%$$

which is much better than 50% or 100% error. You might guess that we can do a little better by trying other resistance values. And you would be right. But if you only need an 8% error, this value would be fine.

Our stand-alone meters have lots of shunt resistors inside of them. You are changing shunt resistors when you change the dial setting, trying to balance these errors. By changing shunt resistors in our circuit we are doing the same thing as turning the dial on the current settings of a multimeter.

5.1.3 Finding Uncertainty in a calculated value

In the last section I gave errors in ΔV_s, but didn't finish the error in the current, I. Of course, since we had to calculate the current, we also need to find the uncertainty in our current using error propagation. Fortunately we "remember" how to do this from PH150. We use our basic form for standard error propagation. If we have a function $f(x, y, z)$ then the uncertainty in f would be

$$\delta f = \sqrt{\left(\left(\frac{\partial f}{\partial x}\right)(\delta x)\right)^2 + \left(\left(\frac{\partial f}{\partial y}\right)(\delta y)\right)^2 + \left(\left(\frac{\partial f}{\partial z}\right)(\delta z)\right)^2}$$

In our current case, our function f is the current I and it is a function of ΔV_s and R_s

$$f = I = \frac{\Delta V_s}{R_s}$$

so we will have an uncertainty like this

$$\delta I = \sqrt{\left(\left(\frac{\partial I}{\partial \Delta V_s}\right)(\delta \Delta V_s)\right)^2 + \left(\left(\frac{\partial I}{\partial R_s}\right)(\delta R_s)\right)^2}$$

and we can find the partial derivatives

$$\frac{\partial I}{\partial \Delta V_s} = \frac{1}{R_s}$$
$$\frac{\partial I}{\partial R_s} = -\frac{\Delta V_s}{R_s^2}$$

so we have

$$\delta I = \sqrt{\left(\left(\frac{1}{R_s}\right)(\delta \Delta V_s)\right)^2 + \left(\left(-\frac{\Delta V_s}{R_s^2}\right)(\delta R_s)\right)^2}$$

Let's try this for our example in the last section. We have $R_s = 170\Omega$. We know that $\delta \Delta V_s = 5.058\,8\text{mV}$ and our resistors are only good to 1% so that would be $\delta R_S = 1.7\Omega$

$$\begin{aligned}\delta I &= \sqrt{\left(\left(\frac{1}{170\Omega}\right)(5.058\,8\text{mV})\right)^2 + \left(\left(-\frac{65.764\text{mV}}{(170\Omega)^2}\right)(1.7\Omega)\right)^2} \\ &= 3.000\,8 \times 10^{-5}\text{A}\end{aligned}$$

This looks small. Is it a good uncertainty? We can't tell until we compare it to our expected current. We expect for our example

$$\begin{aligned}I &= \frac{\Delta V_{ps}}{R} \\ &= \frac{2\text{V}}{5000\Omega} \\ &= 0.000\,4\text{A} \\ &= 4 \times 10^{-4}\text{A}\end{aligned}$$

so the fractional uncertainty in the current will be

$$100\frac{3.000\,8 \times 10^{-5}\text{A}}{4 \times 10^{4}\text{A}} = 7.502$$

This still isn't great, it's about what we got for the error in $\delta \Delta V_s$ (but it's better than 50%). We might be able to do better. But if 8% is OK for our application, then we stop here!

Let's try to figure out what the biggest contributor to our uncertainty might be. To do this we look at the terms in our uncertainty calculation separately

$$\begin{aligned}\left(\left(\frac{1}{R_s}\right)(\delta \Delta V_s)\right)^2 &= \left(\left(\frac{1}{170\Omega}\right)(5.058\,8\text{mV})\right)^2 = 8.855\,2 \times 10^{-10}\text{A}^2 \\ \left(\left(-\frac{\Delta V_s}{R_s^2}\right)(\delta R_s)\right)^2 &= \left(\left(-\frac{65.764\text{mV}}{(170\Omega)^2}\right)(1.7\Omega.)\right)^2 = 1.496\,5 \times 10^{-11}\end{aligned}$$

The first term is about sixty times the second. So to make an improvement we would want to first concentrate on the first term. We could change our $\delta\Delta V_s$ or change our R_s value. Changing ΔV_s is harder than changing R_s. Maybe we could even make R_s a little bigger to improve our current measurement. *Notice that this was not the obvious solution!* At first it seemed that smaller R_s values would give better uncertainties. But after doing the uncertainty calculations, we find that there is an optimal range for R_s. Big R_s is still bad, but very small R_s is also bad. You have to do the math to find this out.

Iterate to find an optimal value

Since there is an R_s in the bottom of both terms in our current uncertainty, let's try changing the R_s value and see if the uncertainty gets better. We have to start all the way back at the top with ΔV_s. We will have to go through all our calculations again! A symbolic math processor or python might be a good way to go so you aren't putting the same things in your calculator over and over.

We start by finding the current in the circuit

$$I = \left(\frac{\Delta V_{ps}}{R + R_s}\right)$$

Then an estimate for ΔV_s across the shunt resistor would be

$$\begin{aligned} \Delta V_s &= IR_s \\ &= \left(\frac{\Delta V_{ps}}{R + R_s}\right) R_s \end{aligned}$$

and then the number of ADC units we used will be

$$N_{ADC} = \frac{\Delta V_s}{\Delta V_{\min}}$$

rounded to the smallest integer, which gives a new estimate of our uncertainty in ΔV_s

$$\delta\Delta V_s = \frac{\Delta V_s}{N_{ADC}}$$

and now we need can find the uncertainty in I

$$\delta I = \sqrt{\left(\left(\frac{1}{R_s}\right)(\delta\Delta V_s)\right)^2 + \left(\left(-\frac{\Delta V_s}{R_s^2}\right)(\delta R_s)\right)^2}$$

and its fractional uncertainty

$$f_I = \frac{\delta I}{I}$$

As you can see, it is probably best to put all this in a symbolic package (like Mathmatica or Maple, or Sage, or python's sympy, or whatever your favorite

symbolic math processor might be). That way, you can change values of, say, R_s and ΔV_s without redoing everything. At least consider using a spreadsheet program or even in Python!.

Let's try this once more with $R_s = 500\Omega$ just to see what would happen.

$$\begin{aligned} I &= \left(\frac{2\text{V}}{5000\Omega + 500\Omega}\right) \\ &= 3.636\,4 \times 10^{-4}\text{A} \end{aligned}$$

so

$$\begin{aligned} \Delta V_s &= IR_s \\ &= \left(\frac{2\text{V}}{5000\Omega + 500\Omega}\right)(500\Omega) \\ &= 0.181\,82\text{V} \end{aligned}$$

and then the number of ADC units we used will be

$$N_{ADC} = \frac{0.181\,82\text{V}}{4.88\text{mV}} = 37.258$$

which gives a new estimate of our uncertainty in ΔV_s

$$\delta\Delta V_s = \frac{0.181\,82\text{V}}{37} = 4.914\,1 \times 10^{-3}\text{V}$$

and now we need can find the uncertainty in I. We will need $\delta R_s = 500\Omega \times 0.01 = 5.0\Omega$

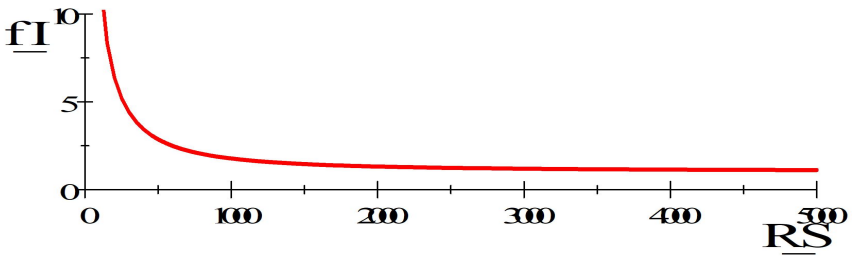
$$\delta I = \sqrt{\left(\left(\frac{1}{500\Omega}\right)(\delta\Delta V_s)\right)^2 + \left(\left(-\frac{\Delta V_s}{(500\Omega)^2}\right)(\delta R_s)\right)^2}$$

$$\begin{aligned} \delta I &= \sqrt{\left(\left(\frac{1}{500\Omega}\right)(4.914\,1 \times 10^{-3}\text{V})\right)^2 + \left(\left(-\frac{0.181\,82\text{V}}{(500\Omega)^2}\right)(5.0\Omega)\right)^2} \\ &= 1.047\,9 \times 10^{-5}\text{A} \end{aligned}$$

and its fractional uncertainty

$$f_I = 100 \times \frac{1.047\,9 \times 10^{-5}\text{A}}{3.636\,4 \times 10^{-4}\text{A}} = 2.881\,7$$

This was a bit of an improvement! We could continue to iterate. I had my computer do this for $R = 5000\Omega$ and $\Delta V_{ps} = 2\text{V}$ I asked it to plot f_I as a function of R_s



Notice that after about 500Ω we are not going to get much of an improvement. So our choice of $R_s = 500\Omega$ seems good for this situation. Once you have a symbolic package or spreadsheet version of this calculation, picking different shunt resistors becomes fairly easy.

We should check, though. What did our 500Ω resistor do to our ΔV measurement? We found our current in the circuit to be

$$I = 3.636\,4 \times 10^{-4}\text{A}$$

and our test resistor is 5000Ω so

$$\Delta V_{test} = \left(3.636\,4 \times 10^{-4}\text{A}\right)(5000\Omega) = 1.818\,2\text{V}$$

We know the power supply was providing $\Delta V_{ps} = 2\text{V}$. So we have introduced an error. We can find the percent difference

$$(100)\,\frac{1.818\,2\text{V} - 2\text{V}}{1.818\,2\text{V}} = -9.998\,9$$

which means the ΔV measurement will be 10% low due to our inserting the shunt resistance. If we can live with a 10% error, then we are fine. If not, it is back to iteration to find a better shunt resistance.

Of course, so far we have just found uncertainty in ΔV and I. These are the uncertainties in our measuring devices that we built. Since you are the manufacturer of these devices, you have had to calculate what their uncertainties will be. When we design our own measuring devices, we always have to do this. Of course you could have built the devices and then watched the output to see where the digits fluctuate like we did with our stand-alone multimeters. But the risk is that it might take a long time to find a value for each part of our device that works together with the other parts, and in the mean time we might burn up our equipment if we don't plan for what we want first. You can check your uncertainty calculations by looking at the fluctuation of the digits to see if we are right (or if some other uncertainty factor has crept in that we haven't handled yet).

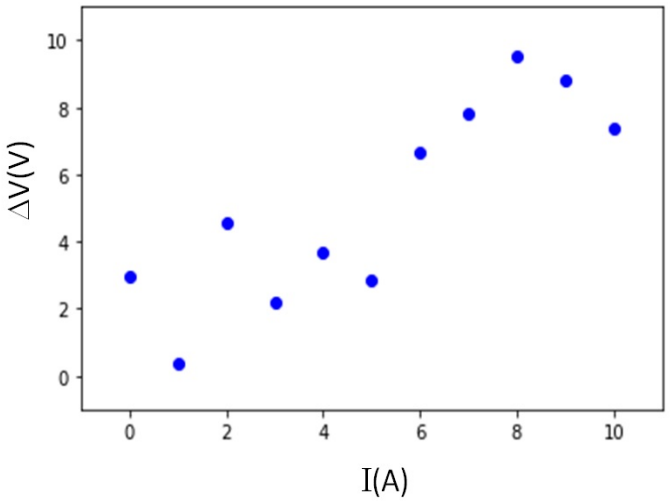
When we started this lab we said we were testing Ohm's law, and we wanted to find R_{test} with it's uncertainty δR_{test} to see if Ohm's law really works. We can find $R_{test} = \frac{\Delta V}{I}$. But how do we find δR_{test}? In finding the uncertainty

in our measuring devices we haven't found δR_{test}. We could use standard error propagation again to do this. But let's not! We will review a different way to find δR_{test}.

5.1.4 Using statistics to calculate uncertainty

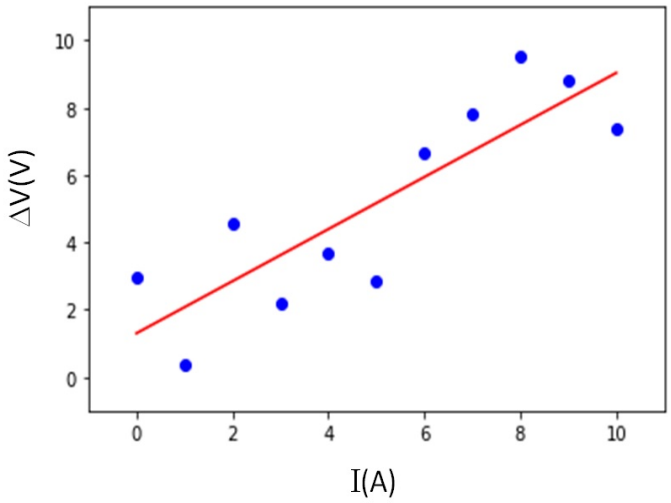
By now in a experimental design, I am usually at my tolerance limit for calculating uncertainties. You might ask, can't we get our powerful computers to help us out a little bit with finding the uncertainty? After all, we went to all the trouble to get the data on the computer. The answer is, yes!

Let's suppose we have done our experiment and we have some data that look like this:



This looks pretty good. It seems to be sort of linear. We might guess from this that Ohm's law is begin obeyed. We could calculate R_{test} from each pair of ΔV and I points and find it's uncertainty using standard error propagation. But it seems that it would be better to take all the points into our analysis to find R_{test}. More data should give us a better estimate for R_{test}. We want R_{test}, and R_{test} is the slope of this line. So we could use this plot and it's slope to find R_{test}! We can use this! Back in PH150 you may have done such a thing using a *curve fit*. In the next figure, a curve fit is shown for the data from the last figure.

Notice that I performed this curve fit in python, but if you are a not a great Python programer you could do this in Excel. You could also do this in LoggerPro or many other data analysis programs. If you need help finding a way to perform a curve fit, talk to your instructor. Here is an example code for doing the curve fit in python.



Download here

```
#########################################################
# Program to do a simple linear curve fit in python
#  with data supplied by the user.
#  The user types the data into numpy arrays by hand.
#
# Author:  Todd Lines
# Date:    2023-02-20
#########################################################
# linear_fit.py
#########################################################
# libraries to import
import scipy as sp
import numpy as np
import matplotlib.pyplot as plt

# Type in your data into arrays
x=np.array([0.001274, 0.001764, 0.001029, 0.00049, 0.000735, 0.002058,
            0.000049, 0.000735, 0.001127, 0.00147  ])
y=np.array([2.8824 , 3.807, 2.3275, 1.1907, 1.7003, 4.3659, 0.3087,
            1.7787, 2.5333, 3.1164 ])

# Plot the data, you may have to adjust the plot axis
#  limits
#plt.axis([-1,11, -1,11])
plt.plot(x,y,'bo')
```

```
# now let's do the curve fit
result= sp.stats.linregress(x, y)

# result holds all of the data from the fit in this order
#   slope, intercept, rvalue, pvalue, stderr (error on the
#   slope), and #  intercept_stderr (error on the
#   y-intercept) Let's print out the slope and intercept
print ("slope is  ",result.slope, "+- ", result.stderr)
print ("intercept is  ", result.intercept, "+- ",
       result.intercept_stderr)

# Now let's try the curve fit to see how it does
xf=np.linspace(0, max(x), 11)
yf=np.zeros(11)
for i in range(11):
    yf[i]=result.slope*xf[i] + result.intercept

# and add the fit line to our graph
plt.plot(xf,yf,'r')

# and we are done. Tell the user we have finished
print('program ended successfully')
```

The scipy linregress() function uses the equation for a line

$$y = result.slope * x + result.intercept$$

Our code gives the result

```
slope is   0.7737292152727273 +-  0.1612215125332822
intercept is   1.2996514309090914 +-  0.9537993308988922
program ended successfully
```

which tells us that we have a slope of about 0.774. Since our graph has ΔV on the vertical axis and I on the horizontal axis we recognize

$$\begin{aligned} \Delta V &= R_{test} I + 0 \\ y &= mx + b \end{aligned}$$

that R_{test} must be the slope. We can see that the resistance for this example is about a tiny 0.774Ω because that is the slope of our fit line. But we know we need an uncertainty along with this nominal value. And low and behold! the scipy lineregress() function has already given us the data we need to find the errors on the slope and intercept as well! This was pretty easy!

Notice that in this analysis technique, we can often afford some error in our ΔV and I values. So maybe a 9% error (like we found in one of our instrument

designs) is not so bad. We may not have to work too hard to get a wonderful choice of shunt resistor if we are going to use many data points and the power of statistics to analyze the data in the end!

5.1.5 Philosophical warning

There was a lot to this reading. We talked about designing and building an instrument. We used some physics, Ohm's law, in our design process. Then we tested a physical model, Ohm's law, with our instrument. All these were good things, but maybe you wondered along the way if it is acceptable to test Ohm's law with an instrument that depends on Ohm's law. And the answer is a big NO!

This lab is practice, but it is imperfect practice. We do know that Ohm's law works, so we are going to use it in designing instruments. But you really need a different instrument, one that does not depend on Ohm's law, to test Ohm's law in a credible way. In next week's lab, we will test a different physical model with the same basic instrument that we built today. The instrument will depend on Ohm's law, but the new physical model must not if the experiment is to be valid.

5.2 Proposals

It's time to start thinking of what experiment you and your group will design. If you are taking PH250 as a block class we have to do this right at the start because it may take time to order in parts for your instrument that you will design. If you are taking PH250 as a full semester class, it is not as much of a hurry, but we still need lead time to order things. So even if it feels early, we need to think about this. You are required to write a proposal for this experiment. This is a document that is intended to persuade someone (your professor, funding agency, yourself, etc.) that you should be given the resources and support to perform the experiment. The proposal consists of the following parts:

1. Statement of the experimental problem
2. Procedures and anticipated difficulties
3. Proposed analysis and expected results
4. Preliminary List of equipment needed

Since you be writing each of these sections, let's discuss what should be in them.

5.2.1 Statement of the experimental problem

This is a physics class, so our experiment should be a physics experiment. The job of an experimental physicist is to test physics theory. So your statement of

the experimental problem should include **what theory you are testing** and a brief, high level, overview of what you plan to do to test this theory.

5.2.2 Procedures and anticipated difficulties

Hopefully, your reader will be so excited by the thought of you testing your theory that he or she will want to know the details of what you plan to do. You should describe in some detail what you are planning. If there are hard parts of the procedure, tell how you plan to get through them.

5.2.3 Proposed analysis and expected results

You might think this is unfair. How are you supposed to know what analysis will be needed and what the results should be until you take the data? But really you both can, and should, make a good plan for your data analysis and figure out what your expected results should be. After all, you have a theory your are testing! You can encapsulate that theory into a predictive equation for your experiment. You can design your experimental apparatus, and put in the numbers from your experimental design. From this you can calculate what should be the outcome.

If you don't do this first, you don't know what equipment you will need or how sensitive that equipment needs to be. If you are trying to measure the size of your text book, an odometer that only measures in whole miles may not be the best choice of equipment. To know what you need, do the calculations in advance.

You should also do the error analysis. You will want to predict the uncertainty. A measurement of your laptop computer length that is good to $\pm$3m is not very satisfying in most cases. Uncertainty in your result is governed by the uncertainty inherent in the measurements you will take. The uncertainty calculation tells you what sensitivity you will need in your measurement devices. Since you are choosing those measurement devices as part of your proposal, and you are choosing the inputs to your model equation (like the resistance and the capacitance in today's lab) you will know how much uncertainty they have, so you can do the calculation in advance.

You should do all of this symbolically if you can, numerically if you must, but almost never by hand (meaning don't use your calculator) giving single value results. Some measurements will come back poorer than you anticipated, or some piece of equipment will be unavailable. You don't want to have to redo all your calculations from scratch each time this happens. For example, in the event of an equipment problem, your analysis tells you if another piece of equipment is sufficiently sensitive, or if you need to find an exact replacement. When I perform an analysis like this, I try for a symbolic equation for uncertainty. I like to program these equations in Mathmatica, or Maple, or SAGE, or MathCAD, or whatever symbolic math processor I have. Alternately, you could code it into Python. Then, as actual measurements change, I instantly get new predictions. In the absence of a symbolic package, a spreadsheet program will do fine. A

numerical program also is quick and easy to re-run with new numbers when no symbolic answer is found.

5.2.4 Preliminary List of equipment needed

Once you have done your analysis, you are ready to list the equipment you need and the sensitivity of the measurement equipment you need. Final approval of the project and the ultimate success of your experiment depend on the equipment you choose or are granted. You want to do a good job analyzing so you know what you need, and a good job describing the experiment so you are likely to have the equipment granted.

5.2.5 Designing the Experiment

Of course, as part of your proposal, you will have to design your experiment. In PH150 we learned that to design an experiment we needed the following steps. Some evidence of these steps should be found in your lab notebook:

1. Identify the system to be examined. Identify the inputs and outputs. Describe your system in your lab notebook.
2. Identify the model to be tested. Express the model in terms of an equation representing a prediction of the measurement you will make. Record this in your lab notebook.
3. Plan how you will know if you are successful in your experiment. Plan graphs or other reporting devices. Record this in your lab notebook. This usually requires you to calculate the predicted uncertainty and to evaluate the relative size of the terms in the uncertainty equation (see below).
4. Rectify your equation if needed. Record this in your lab notebook.
5. Choose ranges of the variables. Record this in your lab notebook.
6. Plan the experimental procedure. Record this in your lab notebook.
7. Perform the experiment . Record this in your lab notebook (see next section).

Let's just take a moment to recall what we learned from PH150 about group projects and lab notebooks. You won't do all the work by yourself in a group project (we aren't in high school anymore!). So you won't have an entry in your own lab notebook for everything that happens in your group project. For example, you might do the wiring, but someone else does the coding. But if you say nothing about the coding, you have a huge hole in your record of what happened. To solve this, you mention that the item happened (coding in our example) but just put in a reference to the person who did that part of the work. So in our example you would put in a reference in your lab notebook to

the lab notebook of the person who wrote the code. That way, in your work group there is a clear path to what was done in everyone's lab notebooks. This is what you do in real jobs! So we will practice it here (and yes, it gets graded, because this isn't a real job, just a class).

5.2.6 Using Uncertainty to refine experimental design.

Suppose you plan to test our model for resistance from your PH220 text book. The equation for resistance is

$$R = \rho \frac{\ell}{A}$$

where ρ is the resistivity, the material properties of the material that makes wire or resistor have friction. The length of the wire or resistor is ℓ, and A is the cross sectional area. We could find the uncertainty in R

$$\delta R = \sqrt{\left(\frac{\partial R}{\partial \rho}\delta\rho\right)^2 + \left(\frac{\partial R}{\partial \ell}\delta\ell\right)^2 + \left(\frac{\partial R}{\partial A}\delta A\right)^2}$$

The first term in the square root is

$$\left(\frac{\partial R}{\partial \rho}\delta\rho\right)^2 = \left(\frac{\ell}{A}\delta\rho\right)^2$$

and the other two terms are

$$\left(\frac{\partial R}{\partial \ell}\delta\ell\right)^2 = \left(\frac{\rho}{A}\delta\ell\right)^2$$

$$\left(\frac{\partial R}{\partial A}\delta A\right)^2 = \left(-\rho\frac{\ell}{A^2}\delta A\right)^2$$

And suppose that our design is to have a copper wire with

$$\begin{aligned} \rho &= 1.68 \pm 0.03 \times 10^{-8}\Omega\mathrm{m} \\ \ell &= 5.0 \pm 0.1\mathrm{m} \\ A &= 5.0 \times 10^{-10}\mathrm{m}^2\mathrm{m}^2 \end{aligned}$$

This would give a resistance of

$$\begin{aligned} R_{new} &= 1.68 \times 10^{-8}\Omega\mathrm{m}\frac{5\mathrm{m}}{5.0 \times 10^{-10}\mathrm{m}^2} \\ &= 168.0\Omega \end{aligned}$$

We can calculate each of our terms from the δR equation.

$$\left(\frac{\ell}{A}\delta\rho\right)^2 = \left(\frac{5\text{m}}{5.0\times10^{-10}\text{m}^2}\left(0.03\times10^{-8}\Omega\text{m}\right)\right)^2 = 9.0\Omega^2$$

$$\left(\frac{\rho}{A}\delta\ell\right)^2 = \left(\frac{1.68\times10^{-8}\Omega\text{m}}{5.0\times10^{-10}\text{m}^2}\left(0.1\text{m}\right)\right)^2 = 11.290\Omega^2$$

$$\left(-\rho\frac{\ell}{A^2}\delta A\right)^2 = \left(-\left(1.68\times10^{-8}\Omega\text{m}\right)\frac{(5\text{m})}{\left(5.0\times10^{-10}\text{m}^2\right)^2}\left(0.1\times10^{-9}\text{m}^2\right)\right)^2 = 1129.0\Omega^2$$

The overall uncertainty then would be

$$\delta R = \sqrt{9.0\Omega^2 + 11.290\Omega^2 + 1129.0\Omega^2} = 33.901\Omega$$

So with this design we predict a fractional uncertainty of

$$\frac{33.901\Omega}{168.0\Omega} = 0.201\,79$$

or a little over 20%. This is not a great design. We would like a much lower uncertainty, something that gives a fractional uncertainty more like 1%. It is clear that the last term has the highest contribution to the uncertainty, so this is the term that needs fixing. One method of fixing the problem would be to decrease A. We could try $1.0\pm0.1\times10^{-9}\text{m}^2$. In order to have the same resistance we will also have to change the length of the wire from 10m to 5m.

$$\begin{aligned}\rho &= 1.68\pm0.03\times10^{-8}\Omega\text{m}\\ \ell &= 10.0\pm0.1\text{m}\\ A &= 1.0\pm0.1\times10^{-9}\text{m}^2\end{aligned}$$

Checking we see we do get the same resistance

$$\begin{aligned}R &= 1.68\times10^{-8}\Omega\text{m}\frac{10\text{m}}{1.0\times10^{-9}\text{m}^2}\\ &= 168\Omega\end{aligned}$$

But now for the last term we would get

$$\left(-\rho\frac{\ell}{A^2}\delta A\right)^2 = \left(-\left(1.68\times10^{-8}\Omega\text{m}\right)\frac{(10\text{m})}{\left(1.0\times10^{-9}\text{m}^2\right)^2}\left(0.1\times10^{-9}\text{m}^2\right)\right)^2 = 282.24\Omega^2$$

which is better. But we have to check to make sure our design change didn't cause a large rise in the other two terms.

$$\left(\frac{\ell}{A}\delta\rho\right)^2 = \left(\frac{10\text{m}}{1.0\times10^{-9}\text{m}^2}\left(0.03\times10^{-8}\Omega\text{m}\right)\right)^2 = 9.0\Omega^2$$

$$\left(\frac{\rho}{A}\delta\ell\right)^2 = \left(\frac{1.68\times10^{-8}\Omega\text{m}}{1.0\times10^{-9}\text{m}^2}\left(0.1\text{m}\right)\right)^2 = 2.8224\Omega^2$$

The first term was hurt by our new design change, but not badly. So with the new design the overall uncertainty would be

$$\delta R = \sqrt{9.0\Omega^2 + 2.8224\Omega^2 + 282.24\Omega^2} = 17.148\Omega$$

So with this new design we predict a fractional uncertainty of

$$\frac{17.148\Omega}{168.0\Omega} = 0.102\,07$$

which is about 10.%. This is much better. From our uncertainty terms, we can see that to do better we need to improve both the δA term and the $\delta\ell$ terms because they are now about the same size. Probably no one in our class is really testing the resistance equation. This is just an example. But the point we need to take away from this example is that **the terms in our uncertainty calculation tell us how to modify our experimental design.**

There is a refinement we could make to our process. really there are no area measurement devices available, so what we would do is measure the diameter of the wire and calculate the area.

$$A = \frac{1}{4}\pi D^2$$

We could find δA by using our propagation of uncertainty equation again, or we could modify our resistance equation so that it is in terms of what we actually measure.

$$R = \rho\frac{4\ell}{\pi D^2}$$

and calculate our uncertainty in terms of ρ, ℓ, and D. That is preferred and usually less work. The general rule is to express your model equation in terms of what you will actually measure before you calculate the uncertainty terms.

The moral of this long story is that we must calculate the uncertainty ***as part of the design process*** . It is probably best to use a symbolic math processor or at lease a python program or spreadsheet so that as the design changes your uncertainty estimate will change too without having to manually recalculate it.

5.3 Lab Assignment

1. Build the instrument

   (a) Choose a test resistor in the 1kΩ to 10kΩ range and a shunt resistor. You will have to check your values using the math we discussed above to make sure they will work. If your first shunt resistor choice works, use it. If not, iterate until you have a shunt resistance that will work.

   (b) Modify your voltmeter sketch to measure both the voltage and the current. (Check the voltage, currents, and their uncertainties with the serial monitor to make sure things seem good).

   (c) Build your voltmeter and ammeter so your Arduino is taking $(I, \Delta V)$ pares and reporting them. Reporting to the serial monitor is fine for a start. if you based your sketch on the simple voltmeter, make

   sure you don't use voltages outside the 0V to +5V range! Include expected uncertainties for your ΔV and I measurements.

   (d) Check your lab group's instruments to see if they work, and have your lab group members check yours.

2. Test Ohm's law

   (a) Take 10-15 measurements of ΔV and I. For each ΔV measurement change the ΔV setting on the power supply a small amount (don't go over 5V if you are using the simple voltmeter!).

   (b) Plot voltage vs. current and fit a curve to the data.

   (c) Determine the resistance from this curve fit and its uncertainty.

   (d) Finally, determine if your results support the Ohm model for how potential and current are related?

   (e) Compare your data and conclusions to the data and conclusions of your lab group members. Have them look at your results as well.

3. Using the same new instrument, repeat part 2 for a diode by replacing your test resistance with the diode. Do your results support the Ohm model for how potential and current are related?

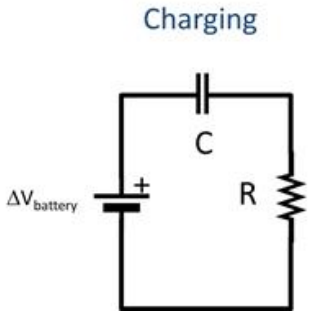
4. Could you have your Arduino sketch report the calculated uncertainties for ΔV, I, and R? If you have time (you probably won't) give this a try.

Chapter 6

Resistors and Capacitors

In this lab, we will not build a new instrument. We will use an instrument we built in a previous lab (or at least only a new version of that instrument adjusted for today's resistance values). You will notice a pattern in what we do from now on in PH250. We will build an instrument and then test a model with that instrument. The instrument must be designed so that it can take the data needed to test the model. In today's lab, we will test the model of how capacitors work in a circuit. If your PH220 class is moving along nicely, this model will be familiar.

6.1 The Model to Test



Let's start by thinking of hooking up a capacitor and a resistor in series with a battery. The capacitor will become charged. The voltage across the capacitor as a function of time will be given by

$$\Delta V_C(t) = \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) \qquad \text{Charging}$$

where

$$\tau = RC \tag{6.1}$$

is the product of the resistance, R, and the capacitance, C. The current in the circuit as a function of time will be given by

$$I(t) = I_{\max} e^{-\frac{t}{\tau}} \qquad \text{Charging}$$

while we charge up the capacitor. The quantity

$$\tau = RC$$

is called the time constant.

We should review what a time constant is. Think of a particular case, say,

$$\begin{aligned} \Delta V_{battery} &= 1.5\text{V} \\ R &= 2\Omega \\ C &= 10\text{F} \end{aligned}$$

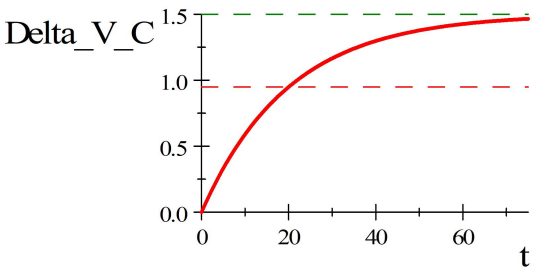
then

$$V_C(t) = (1.5\text{V})\left(1 - e^{-\frac{t}{(2\Omega)(10\text{F})}}\right)$$

and

$$\begin{aligned} \tau &= (2\Omega)(10\text{F}) \\ &= 20.0\text{s} \end{aligned}$$

We can plot this



Notice, that by about $t = 70\text{s}$ we essentially have $\Delta V_C = \Delta V_{battery}$. But up to that point, the voltage across the capacitor changes in a very non-linear way. The part of the equation that looks like

$$\left(1 - e^{-\frac{t}{RC}}\right)$$

is interesting. What is e^0?

$$e^0 = 1$$

So at $t = 0$ we do have $\Delta V_C = 0$ on the capacitor because

$$\left(1 - e^{-\frac{t}{RC}}\right) = (1 - 1)$$

For any positive time, $e^{-\frac{t}{RC}}$ will be less than 1. For large positive times $\frac{t}{RC}$ gets to be a big number. So $e^{-\frac{t}{RC}}$ gets very small. Then $\left(1 - e^{-\frac{t}{RC}}\right)$ gets very close to 1. That means that

$$\lim_{t\to\infty} \Delta V_C = \lim_{t\to\infty} \Delta V_{battery}\left(1 - e^{-\frac{t}{RC}}\right) = \Delta V_{battery}\,(1) = \Delta V_{battery}$$

just as we saw in the graph and as we know it must.

But what if $t = \tau = RC$? Then

$$\begin{aligned} \Delta V_C &= \Delta V_{battery}\left(1 - e^{-\frac{RC}{RC}}\right) = \\ &= \Delta V_{battery}\left(1 - e^{-1}\right) \\ &= 0.632\,12\Delta V_{battery} \\ &\approx 63\%\Delta V_{battery} \end{aligned}$$

The time τ is the time it takes for the capacitor to be 63% charged!

The quantity τ is called the *time constant* because it tells us something about how long it takes for ΔV_c to go from 0 to get to $\Delta V_{battery}$. The "t-looking-thing" is a Greek letter "t." It is pronounced "tau." This quantity will be useful in planning your experiment.

Notice what we have done. We have used our model to form an equation, and we have used part of that equation to understand how much time it will take to perform a test (experiment) of the model. This is typical, get an idea of how to make the measurement from the model we are testing.

But! you say, I don't really remember where all of these equations came from. Or maybe your PH220 class hasn't gotten to allowing current to flow yet so you have not done this. If any of this is mysterious, please read the section of our PH220 book that covers RC circuits. But if it is vaguely familiar or seems to make sense, really we can test our model of how capacitors work just knowing a little about capacitors and the equations that came from the model.

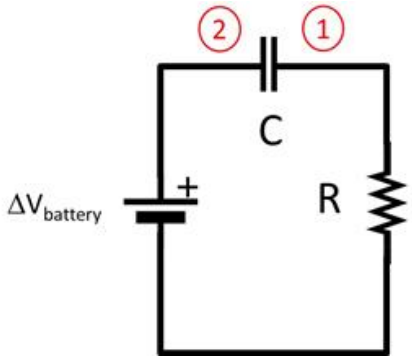
6.2 The Instrument

To test our capacitor model we need to measure the voltage across the capacitor as a function of time. We could also measure the current in the circuit as a function of time. One of these is sufficient to test the model. I am going to

describe measuring the voltage across the capacitor as a function of time. But you know from a previous lab how we might add current as a function of time.

We need a device that measures voltage and how it changes as a function of time. But that is just what our Arduino's do! We already know how to build this device. Suppose we can live with a 0V to +5V range of $\Delta V_{battery}$. Then even our simple voltmeter will work. Since it is a function of time that we are testing, we need to output both voltage and time from our Arduino. We can't guarantee that either of our capacitor leads will be at ground, so we will have to be careful in wiring this voltmeter to give ΔV_C.

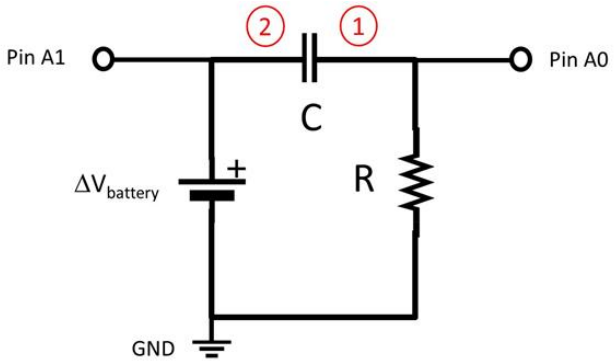
Remember that ΔV_C is the difference between two voltage measurements.



so

$$\Delta V_C = V_2 - V_1$$

neither of which will be ground, so we really have to measure both with our Arduino. We also need a ground connection. The wiring diagram might look like this:



and our sketch will be a little like the one from our last lab.

Download here

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
// Voltmeter with a time stamp
// will measure 0 to 5V only!
// Voltages outside 0 to 5V will destroy your Arduino!!!
////////////////////////////////////////////////////////////
// we want to have voltage vs time,
// so make a place to store a time value
unsigned long time;

// make some integer variables that identify the
// analog input pins we will use:
int AI0 = 0;
int AI1 = 1;

// you also need a place to put the analog to
// digital converter values
// from the Arduino
int ADC0 = 0;
int ADC1 = 0;

// Remember we will have to convert from Analog to
// digital converter(ADC) units to volts.
// We need our delta_V_min just like we did in lab 3
float delta_v_min=0.0049;   // volts per A2D unit

// We need a place to put the calculated voltage
float voltage = 0.0;

////////////////////////////////////////////////////////////
 void setup() {
    // put your setup code here, to run once:
    //Initiate Serial Communication
    Serial.begin(9600);    //9600 baud rate
    }

////////////////////////////////////////////////////////////
 void loop() {
    // Read in the voltages in A2D units form the
    // serial port
    ADC0 = analogRead(AI0);
    ADC1 = analogRead(AI1);

    // Convert the voltage across the
    // test resistor to voltage
```

```
    //   units using delta_v_min
    voltage = (ADC1-ADC0) * delta_v_min;

    // output the time since we started and the voltage
    Serial.print("␣time␣in␣sec,␣");
    // this next line uses millis() which gives time in
    // ms since we started
    time = millis();
    // convert to seconds and print.
    Serial.print(time/1000.0, 6);
    Serial.print(",␣voltage␣across␣C,␣");
    Serial.println(voltage,6);
    }
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

This is just a voltmeter, but one with two A2D pins and a ground. This sketch also gives us time using the millis() function. This function gives the number of milliseconds since our experiment began. We can use our python code from a previous lab to save the data into a file. I modified the previous code just a bit, so here is an updated version.

Download here

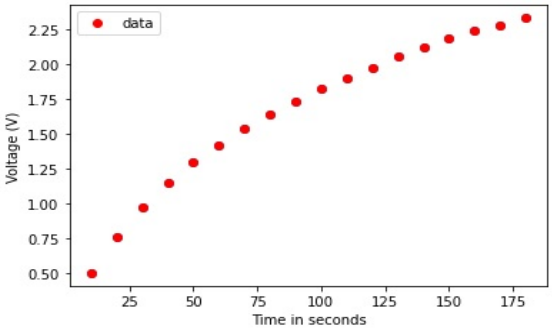
```
#-------------------------------------------------------------------
#-------------------------------------------------------------------
# Python Code to read a stream of data from the serial port
#   and save it to a file
#-------------------------------------------------------------------
#   The idea is to read a series of voltages from an Arduino
#   connected to the serial port where the Ardunio is being
#   used as the Analog to Digital converter. Both the voltage
#   and the time the voltage was taken are sent to the serial port.
#
# We will use two libraries, serial and time
#   The serial library is used to read the serial port
#   The time library is used to space out our data collection by
#   adding a delay in between data points. The amount of time
#   to wait in between data points is called "timeBetween."
#
# We may have to install the serial library. If you have the
#   Anaconda Python for Windows, you can open an Anaconda
#   window and use the command 'conda install pyserial'
#   This must be done before the code can run.
#
# Debugging issues:  The Anaconda Python distribution tends to
#   hang on to the serial port even if the program does not run.
#   If this happens, try sending the python command ser.close()
#   at the command prompt. If this doesn't work, You may have to
#   restart Python.
#   In windows, closing (after saving) the IDE and reopening it
#   might be enough.
#-------------------------------------------------------------------
# import libraries
```

```
import serial
import time


# define variables for the delay time we wait between data points
timeBetween=4 #seconds

# define the number of data points to collect
N=40

#the next line opens a file and creates a pointer or handle for that
                                    file
#  to use as a reference. You have to use double slashes in the path.
#  The pointer, "dataFile" takes the place of all the path and file
#  name so it is easier to use in the code below
dataFile=open("C:\\Users\\rtlines\\Documents\\RCdata.csv","w")

#the next line opens the serial port for communication
ser=serial.Serial('COM3', baudrate = 9600, timeout=1)
#the next line clears out the serial port so we get clean data.
ser.flushOutput()

#there will be a delay before the first data point comes from the
#  serial port, warn the user so they don't worry.
print('getting started...')

# set our index to zero
i=0


# Now for N points, collect data and write it to a file
while (i<N):    #Begin data collection loop
    #We will take data every "timeBetween" seconds. We need to know
    #   when we start waiting so we can tell if it is time to collect
    #   data yet. Use the time.time() to get the current time in
                                        seconds
    #   since Jan 1,1970. Yes that is a weird way to measure time, but
    #   computers do it this way.
    waitStart=time.time()

    #Data comes to the serial port fast. We will continually collect
    #  the data as fast as it comes, but only save it every timeBetween
    #  seconds. The next while loop keeps us reading in data, but only
    #  when the current time - waitStart >= timeBetweem will we use
    #  the data.
    while (time.time()-waitStart<timeBetween): #Begin Data read loop
         # Get data from the serial port
         # it should have a time and a voltage
         arduinoData=ser.readline().decode('ascii')
         # end of the Data read loop

    # the next line just prints the voltage point on the console so the
                                        user
    # feels like something is happening.
    print(arduinoData)
    # This next line writes the time since we started and the Arduino
    #   value from the serial port into one string.  That seems simple
```

```
    #   enough. But there is a historical issue that makes this harder.
    #   Windows puts in two characters to say we have reached the end
    #   of a line.  They are no printing characters so lets represent
    #   as as <CR> and <LF>. Linux just uses one <LF>  and some macs
    #   use <LF> and some <CR>. Sigh.  If we arn't careful, we can use
    #   the wrong one and get an extra blank line on some systems.
    #   The next line gives the option to add in one or two of these
    #   non-printing characters.  Usually just printing what we read
    #   in is fine.  You might need to add a "new line" version of the
    #   non printing new line characters by adding a \n to the end of
    #   your data text data that you read in.  Or a \r might work, or
    #   both!  If you are having difficult output, try this.
    #   The next line also makes sure our data line is in string
    #   format.
    writeString=str(arduinoData) #+ " \n"
    # Of course it could be that you have a \n and don't need one.
    #   The next line removes the end of line characthers in case you
    #   have extras.
    writeString = writeString.replace("\n", "")
    # The next line writes our time plus Arduino value to the file. If
    #   it doesn't have a new line character, it will supply one (well,
    #   sometimes.  If you don't need this line, comment it out)
    dataFile.write(writeString)
    # And finely we increment the loop counter so we can go get the
                                        next
    #  line of data.
    i=i+1      # end Data collection loop

# We are done!  Print out a message saying we are done
print("done with data collection, closing the file and the serial port"
                                      )
# Close the file
dataFile.close()
# Close the serial port
ser.close()
# Note if you code does not finish running, your file will still be
                                     open
#  and your serial port will still be locked to the code.  That means
#  runing the code again won't work.  So if this happens, type the last
#  two commands in the console windown to close the file and the
#  serial port.
#------------------------------------------------------------------------
#------------------------------------------------------------------------
```

The resulting data could be plotted in python. It might look something like this.

I plotted this in python. You might plot it in Excel or LoggerPro or many other plotting tools. Before we fit a line to data. But looking at the data, a linear fit doesn't seem appropriate for this data. So we need to go beyond what we have done so far. To do this we need a curve fit tool that can use whatever equation that might fit the data. Python can do this, and so can LoggerPro. But Excel can't. We will have to be more picky about what tool to use to match our data.



6.2.1 Python Advanced Curve Fitting.

In a previous lab we fit a line to a set of data in python. That wasn't very hard. But looking at the sample data in the previous graph, it is clear that a straight line won't do. The data has a definite curve to it. So what equation do we use for our fit? The only equation that makes sense is the equation we developed from our reasoning about RC circuits.

$$\Delta V_C\left(t\right) = \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) \qquad \text{charging}$$

We can use a different python routine to find a curvy equation like this. We will use the scipy optimize curve_fit() routine. But we can help the curve_fit() routine a little by adding in an extra part. Let's code our fit equation like this

$$y = A * (1 - \exp(-B * x)) + C$$

where we have added a constant C. Of course C should be equal to zero or at least should be close to zero because there is no constant term in our theoretical equation. We will need to check for this in our curve fit solution.

Download here

```
#############################################################
# Program to do a curve fit in python with a user defined
#  fit equation and %  with data supplied by the user.
# The user types the data into numpy arrays by hand,
#  and supplies a function with the fit equation.
#  In this example, the function is called RC_Charging.
#
# Author:  Todd Lines
# Date:    2023-02-20
#############################################################
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

```
# Define the Gaussian function
print("␣␣")
print("Find␣a␣curve␣fit␣to␣a␣user␣defined␣function")

#Here is where you define the function to use in the fit
def RC_Charging(x, A, B, C):
    y = A*(1-np.exp(-B*x)) + C
    return y

#here is where you give the data to fit. Put it into numpy
#  arrays. Say our data is voltage vs. time. Put the
#  time #  values in the xdata array and the voltage
#  values in the #  ydata array. For the exmple the units
#  are seconds and volts
xdata = np.array([10, 20, 30, 40, 50, 60, 70, 80,
                  90, 100, 110, 120,  130, 140, 150,
                  160, 170,  180])
ydata = np.array([0.5, 0.76, 0.97, 1.15, 1.3, 1.42,
                  1.54, 1.64, 1.73, 1.82, 1.9, 1.97,
                  2.05, 2.12, 2.18, 2.24, 2.28, 2.33])

#we also need the ydata uncertainty for error bars.
#  Say it is 0.1  (you sould calculate what your
#  value should be)
ydata_err = 0.1 #V

#Now plot the data so we can see the data points
#plt.plot(xdata, ydata, 'o')


#Now perform the curve fit. We can't␣just␣use␣a␣linear
#  fit. The data is very much not linear. So we
#  will use a more robust curve fit rotine from scipy.
#  The sciepy optimize curve_fit() routine needs the
#  equation to use for the fit as a function (here
#  RC_Charting()) and it needs the fit parameters and
#  for the error on the fit parameters we need the
# covariance matrix to be output
parameters, covariance = curve_fit(RC_Charging,
                                   xdata, ydata)

#Pull out the fit prameters
fit_A = parameters[0]
fit_B = parameters[1]
fit_C = parameters[2]
```

```
#Pull out the uncertanty from the diagonal elements
#  of the covariance matrix. Remember that the
#  diagonal elements #  are the error squared.
SE = np.sqrt(np.diag(covariance))
SE_A = SE[0]
SE_B = SE[1]
SE_C = SE[2]

#Use the fit parameters to make a set of estamated
#  y values #  from the fit equation. We can pass
#  in the whole xdata array #  and get out all the
#  y value estimates at once using our #  function
#  with our fit equation.  I called the new y-values
#  fit_y.
fit_y = RC_Charging(xdata, fit_A, fit_B, fit_C)

#Plot the data (as dots) and the fit (as a line) to
#  see if the equation makes sense as a good fit.
plt.xlabel('Time␣in␣seconds')
plt.ylabel('Voltage␣(V)')
plt.errorbar(xdata, ydata, yerr = ydata_err,
             label ='data', fmt='ro')
plt.plot(xdata, fit_y, 'b-', label='fit')
plt.legend()
plt.show()

#and print out our fit parameters and their
#  uncertainties
print("␣␣")
print('The␣value␣of␣A␣is␣',fit_A, end ="␣")
print ('with␣standard␣error␣of␣', SE_A)
print('The␣value␣of␣B␣is␣', fit_B, end ="␣")
print('with␣standard␣error␣of', SE_B)
print('The␣value␣of␣C␣is', fit_C, end ="␣")
print('with␣standard␣error␣of', SE_C)
print("␣␣")

#sometimes the curve fit routine throws a
#  math warning, let the user know that the program
#  ended and not to be upset about the warning
print('successful␣end␣of␣program')
print('warning␣about␣overflow␣may␣follow')
print("␣␣")
```

We have to match our variables with the ones we have in our theory equation.

Let's compare the equations.

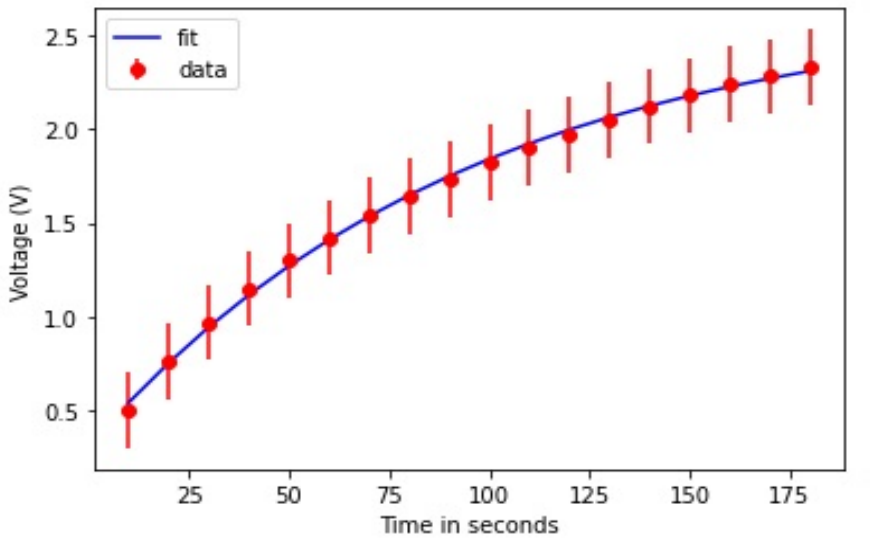
$$\begin{aligned} y &= A*(1-\exp(-C*x))+B \\ \Delta V_C(t) &= \Delta V_{\max}\left(1-e^{-\frac{t}{\tau}}\right)+0 \end{aligned}$$

We can see that

Theory	python
$\Delta V_{\max}$	A
0	B
τ	$\frac{1}{C}$

I suppose we could have just used the theory variables like ΔV_{max} in python. you might choose to do that.

Our curve fit might look like this



The program output looks like this:

```
Find a curve fit to a user defined function

The value of A is 2.35419 with standard error of 0.03356.
```

```
The value of B is 0.01051 with standard error of 0.00047.
The value of C is 0.30913 with standard error of 0.02340.

successful end of program, warning about overflow may follow
```

If you look at the code you will notice that there is a big difference between the linregress() function and our curve_fit() function when it comes to the uncertainties in the coefficients. The linregress() function just gave the uncertainteis. But the curve_fit() function returns a covariance matrix. The diagonal elements of this matrix are our uncertainties on our coefficents – squared! So we had to take a square root to get the uncertainties.

The curve fit looks nice and that is comforting. For my data, it looks like our capacitor model might be correct, an important thing to check is to see if the curve fit goes through the error bars on all of the data. If it does, it is a good sign that the curve fit might represent the data well. Beware! you have to put your own estimate for the uncertainties of your voltage measurements into the code! I overestimated the size of the error bars on purpose to make sure you couldn't miss them in the plot. Yours might be smaller (or bigger!) depending on how you design your instrument.

But now let's go back to our little list of curve fit parameters. We identified

$$\tau = \frac{1}{\mathtt{B}}$$

and for my data I have

$$\mathtt{B} = 0.01051 \pm 0.00047$$

We need units, and looking at the equation we know τ has units of seconds, so B must have units of inverse seconds.

$$\mathtt{B} = (0.01051 \pm 0.00047)\,\frac{1}{\mathrm{s}}$$

so we can find a value for τ. For my data, I have

$$\begin{aligned} \tau_{measured} &= \frac{1}{0.01051\frac{1}{\mathrm{s}}} \\ &= 95.15\,488\mathrm{s} \end{aligned}$$

Note that we will have to calculate the uncertainty in τ. I will leave that for an exercise. But I can compare this $\tau_{measured}$ to the $\tau = RC$ value I started with. If they are within each other's error range, this is a powerful confirmation of our capacitor model.

6.3 Lab Assignment

We have two equations for charge as a function of time for a RC circuit. They are

$$\Delta V_C(t) = \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) \qquad \text{charging}$$

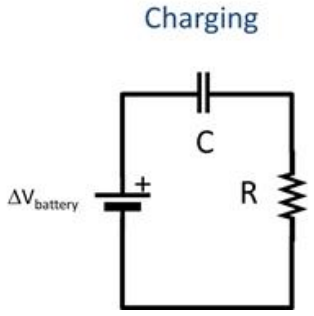
$$I(t) = I_{\max} e^{-\frac{t}{\tau}} \qquad \text{charging}$$

where

$$\tau = RC \tag{6.2}$$

is the time constant.

1. Using a capacitor with a capacitance of about 20μF and a resistor of about $1\text{M}\Omega$, create a circuit as shown. This will be our system that we will use to test our capacitor model.



   You can use one of our power supplies, but be careful to either stay in the 0V to +5V range, or to use a voltage divider to achieve this range at the Arduino input. Follow good lab notebook procedures by recording the model you are testing and your test setup in your lab notebook. Just a note, we will be using directional capacitors. You haven't studied directional capacitors in class. For today's lab, the only difference is that these capacitors only work one direction. This is a little like our diodes. If the circuit doesn't work, try turning your capacitor around.

2. Build your instrument. and write the sketch and Python collection codes. Test every part of the instrument before you start collecting capacitor data. Don't forget to find your uncertainties. If you start your python code but it doesn't finish, remember to close the serial port and the file before trying again. Follow good lab notebook procedures by recording your instrument design in your lab notebook.

3. Now get ready to collect data for a capacitor charge. Work with a lab partner from your group to achieve the data collection. Compare your data among your group to make sure things went well. Follow good lab notebook procedures by recording your data or giving a location of the stored data in your lab notebook.

4. Take the data from your file and graph it. Graphing is easy in python, but you could use LoggerPro if you have this on your computer. You should include this graph in your lab notebook (but might also include the curve fit described in next item on the same graph).

5. Perform the curve fit. As in the last lab, having the proper curve fit the data is a validation of our model! So if the theoretical curve fits the data, it makes sense that something about the model might be right. Note that we have an equation from our theory, equation (6.1). If our data fits a different curve, that could be evidence that our theory is wrong! We hope the data fits this curve and no other. We know python can do this and so can LoggerPro. Python might be a good choice. Once you have the curve fit and the fit parameters (and their uncertainties) make a graph of the data and the fit. Error bars would be a good idea (but don't use my uncertainty values, calculate your own!). Include the graph of the curve fit and the data in your lab notebook as well as the fit equation and fit parameters (don't forget their uncertainties).

6. Find the time constant and compare to your predicted value. If these compare within their uncertainties, we have a further validation of the model. Record the time constants and their uncertainties in your lab notebook.

7. Draw a conclusion, is our capacitor model good?

Chapter 7

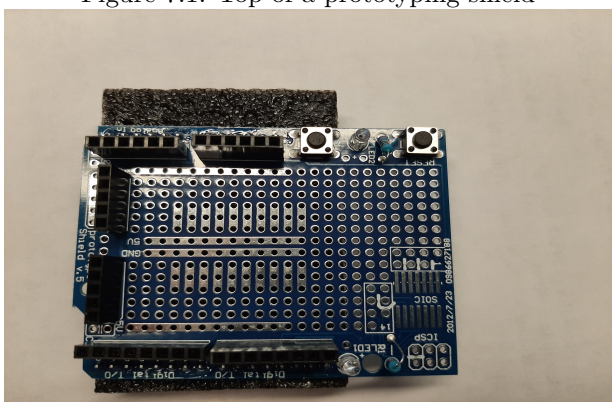
DataLogging

It today's lab, we are going to measure temperature. We will use a transducer that turns temperature (energy of the air molecules around us) into a voltage. "Transducer" is a general word for a device that turns some energy measurement into a voltage (really any form of energy to a different form of energy). But really our temperature measurement is just to have something useful to measure. We will take a break from testing physics models today. And we will learn about a second way to collect data so that we can analyze it on a computer.

Often we need to have a data collection device that can operate far from our computer, but still save data for later use. For example, if you were to launch your Arduino-instrument on a high altitude balloon.

Today we all need to be able to have our Arduino collect data and save it with no computer attached. We will do this using a shield. An Arduino shield fits over the top of your Arduino, connecting to the necessary pins to do its job. The other pins are still available to use for other measurements.

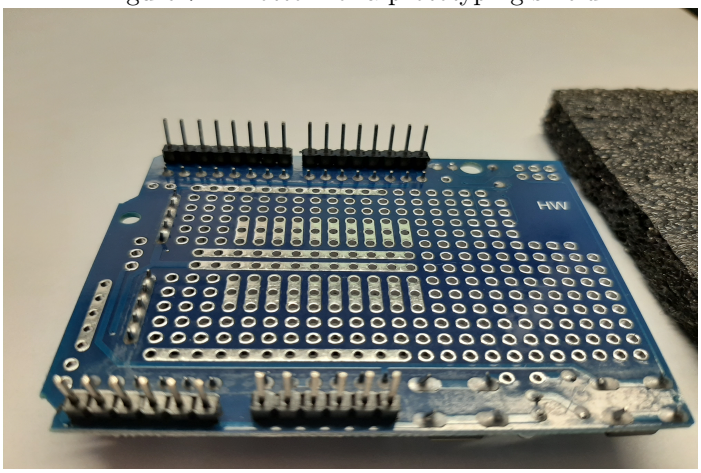
Figure 7.1: Top of a prototyping shield



7.1 Arduino Shields

Arduino shields are a simple way to add specific functionality to your Arduino. There are many different shields with many different features. Some shields have additional integrated circuit (IC) chips on them that can add additional computational power. Others are extremely simple and are designed so you can have a more solid electrical connections with your experiment. The previous figure and the next figure are the top and bottom of an example of a simple prototyping shield.

Figure 7.2: Bottom of a prototyping shield



Notice the long wire pins on the bottom. These are designed to be plugged in directly into the Arduino. When not in use these should be pushed into the conductive foam. This will not only protect them from getting bent, but will also protect the electrical circuits on the shield from static shocks.

Many shields are compatible with other shields such that they can be stacked onto each other. This will depend on what pins each shield is using and on the physical placement of both the bottom and top pins.

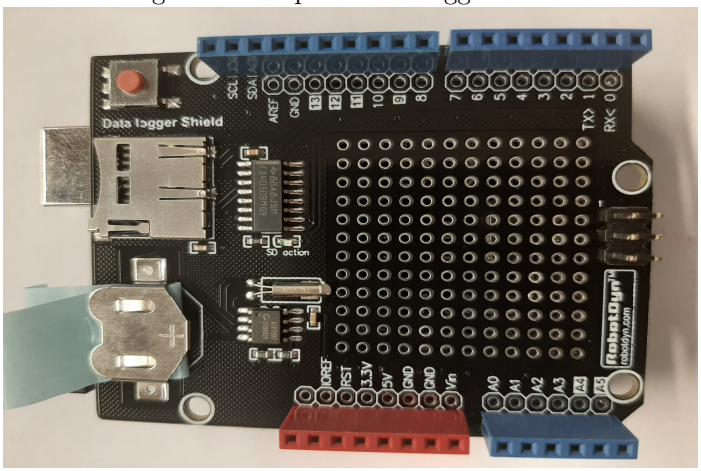
Because shields are designed for a specific functionality the manufactures will often provide example code that utilizes those features. This code can then be modified and/or combined with other code to get your Arduino to do what ever you want. While working with shields be aware of the pins that the shield is using. So you don't interfere with its operation. For example pin 9 or 10 will be a special pin for our particular shield that we will use today. You can't use one of these special pins for your instrument because it is already being used for the shield, itself.

7.2 Data Logger Shield

Data logging is a fancy name for storing data. Usually data logging involves the time that the data point is collected as well as the data, itself. And this is often one of the most important parts of any experiment. We will use a data logger shield that has a real time clock(RTC) and a SD card slot built into it. The data logger shield allows us to not only save the data to an SD card but we can also save the time along with it. This is just what we want!

Take a close look at the figure 7.3. You will notice that the SD card slot (right next to the words "Data logger Shield") is close to an integrated circuit (IC). That IC (a black rectangle) controls the SD card slot.

Figure 7.3: Top of Data Logger Shield



There is also a battery holder. It has a battery in it, but the image also shows a piece of blue paper covering both sides of the battery. This is for shipping so that the battery isn't drained while not in use. The battery is needed for the real time clock. So that it keeps time even if there is no other power to the board. If your shield has the blue paper, you can take it out now.

But why do we need a clock on our data logger? Often we just need to know what time the data was collected. But more importantly, we might wish to compare our data to data collected on another instrument. If we know what time the data points were gathered, the comparison is easier. For one example, suppose I am measuring the temperature from the ground to $30km$. My instrument measures temperature (like the one we will build today) but I want to know the temperature as a function of altitude. I will need an altimeter to collect data along with my temperature instrument. To match up the data from my instrument and the altimeter, both can record the time for each data point and then I can match the times from my temperature data and the times from the altitude data to get my temperature as a function of altitude. Our BYU-I High Altitude Research Team uses this method for most of their instruments.

On our data logger shield the electronics for the RTC are right next to the battery. Note the smaller IC and the silver cylinder. The cylinder holds a small quartz crystal tuning fork that keeps oscillating. These oscillations are counted by the IC and are used to keep time.

Just like on your Arduino the shield pins are all labeled. But notice that one one of our data logger shield types some of the pins are labeled with a white square and a back number. These pins (A4, A5, 9,11,12 and 13) are the pins that the shield uses for its functions. Make sure that your experiment doesn't use these pins. Pins A4 and A5 are used for the RTC and pins 9,11,12 and 13 are used for the SD card. The bottom of shield gives additional labels to these pins that describe their function.

We have two types of data logger shields. And the second type does not have the nice little white warning boxes to tell you to not use the pins it uses. But most of the pins that you can't use are the same (A4, A5, 10,11,12 and 13). The big difference is that for the second type of shield pin 9 got changed to pin 10. These second kind of shield have a brown battery holder.

7.3 Set up

The Data Logger shield uses a library. Libraries are extensions to the computer language we are using. We used one to extend python so we could get data from the serial port (pyserisl). We will do something similar to get our data logger shield to work. But this time we have to be very specific. Our real time clock was made by a company named Adafruit. If you think about it, they are the only people who for sure know how the clock works. After all, they designed it! We need to use their real time clock library and no one else's. To get that library follow these steps in the Arduino software.

1. Under "Tools" go to "Manage Libraries ..."
2. Search for "RTClib"
3. Find the library by Adafruit and install it.

The first time the software is run on the Arduino with the Data Logger shield the RTC needs to be set. Pull out the paper that keeps the battery from contacting and put the battery back (if the paper is still there). Then use the code below in an Arduino program (like the Adafruit RTC example code that they provided with the library!). Check the serial monitor. If it says that the "RTC has not been set!" or if the date and time are incorrect then you will need to remove the comment back slashes on the line

```
rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
```

but change the numbers to match the current date and time. Then upload the code again. This will set the clock. This only needs to be done once, as the battery will keep time from then on. So put the comment back slashes back in

and upload the code again. Here is an example data logger code that does all this.

Download here

```
// ================================================
// RobotDyn SD/RTC Arduino shield example
// Nolan Chandler
// Last revised: 2020-11-18
//
// Meant to serve as a starting point for data logging
// using both a real-time clock (RTC) and a microSD
// card.
//
// Collects data at a given rate, and saves them with
// a timestamp on the SD card to files called
// "LOG000.TXT", "LOG001.TXT", etc. A new file is
// opened every N samples to collect another N samples.
// ================================================

#include <SPI.h>
#include <SD.h>

// by Adafruit. Must be manually installed
#include "RTClib.h"

# We have two types of dataloggers. If the battery
#    holder is black use pin 9.  If the battery holder
#    is brown then use pin 10.
const int SD_PIN = 9;

File logfile;

RTC_DS1307 rtc;

float data;
int i = 0;
int N = 50; // Number of samples per file
int waittime_ms = 500; // milliseconds between samples

// ================================================

void setup()
{
  // Open serial communications
  Serial.begin(9600);
```

```
  init_RTC();

  // (note 24-hour time: 3pm -> 15)
  // This line sets the RTC with an
  // explicit date & time, for example:
  // to set January 21, 2014 at 3:18pm
  // you would use the following line:
  // rtc.adjust(DateTime(2014, 1, 21, 15, 18, 0));

  init_SD();

  logfile = open_next_logfile();
}

// ================================================

void loop ()
{
  if (i < N) {
    // generate a random number
    data = random(-100, 100);

    DateTime now = rtc.now();

    // Write the date, time and data to log file
    // Same as printing to Serial!
    logfile.print(now.year());
    logfile.print('-');
    logfile.print(now.month());
    logfile.print('-');
    logfile.print(now.day());
    logfile.print(' ');
    logfile.print(now.hour());
    logfile.print(':');
    logfile.print(now.minute());
    logfile.print(':');
    logfile.print(now.second());
    logfile.print(',');
    logfile.print(data);
    logfile.println();

    // write same data to serial
    Serial.print(now.year());
    Serial.print('-');
    Serial.print(now.month());
    Serial.print('-');
```

```
    Serial.print(now.day());
    Serial.print('␣');
    Serial.print(now.hour());
    Serial.print(':');
    Serial.print(now.minute());
    Serial.print(':');
    Serial.print(now.second());
    Serial.print(',');
    Serial.print(data);
    Serial.println();

    delay(waittime_ms); //ms
    i++;
  }

  // Reached N samples, open the next log
  // file to record N more
  else {
    logfile.close();

    // comment out the next two lines to stop
    // recording after the first file
    i = 0;
    logfile = open_next_logfile();
  }
}


// ========================================
// initializes the RTC,
// and checks to see if it has been set
// ========================================

void init_RTC()
{
  Serial.print("Initializing␣RTC...");

  if (!rtc.begin()) {
    Serial.println("␣failed!");
    while (1);
  }

  Serial.println("␣done!");

  if (!rtc.isrunning())
    Serial.println(
```

```
    "WARNING: RTC has not been previously set");

}

// ================================================
// attempts to initialize the SD card for reading/writing
// ================================================

void init_SD()
{
  Serial.print("Initializing SD card...");

  if (!SD.begin(SD_PIN)) {
    Serial.println(" failed!");
    while (1);
  }

  Serial.println(" done!");
}

// ================================================
// Opens the next available log file in SD:/LOGS/
// Write to the file using logfile.print() or println(),
// just like Serial
// ================================================

File open_next_logfile()
{
  char filename[24];

  // Create folder for logs if it doesn't already exist
  if (!SD.exists("/LOGS/"))
    SD.mkdir("/LOGS/");

  // find the first file LOGxxx.TXT that doesn't exist,
  // then create, open and use that file
  for (int logn = 0; logn < 1000; logn++) {
    sprintf(filename, "/LOGS/LOG%03d.TXT", logn);

    if (!SD.exists(filename)) {
      Serial.print("Opened \'SD:");
      Serial.print(filename);
      Serial.println("\' for logging.");
      break;
    }
  }
```

```
  return SD.open(filename, FILE_WRITE);
}
```

The code above saves random data to the SD card. When you have it working check to see that the data file is on the SD card by putting the SD card in your computer and opening the file. Note that by default it is using pin 9 for the SD card reader/writer. But if you have the data logger shield with the brown battery holder you will need to change the code to use pin 10.

Of course, we don't want random data. You will need to modify the code such that it saves the temperature. But just getting the time information to save to the SD card is a great first step!

Before we go on to add the sensor, spend a few moments looking at the example code. Notes that at the end there are some special functions that do things like initialize the RTC and the SD card reader. These are used by the setup() function to get things working. You should not have to change these much. But do note that they need to be part of your code for the data logger shield to work.

Also notice up at the top of the example code there are more include statements and some special variables. Like the

```
  RTC_DS1307 rtc;
```

line or the

```
  SD_PIN = 9;
```

line. These are also needed by the data logger. So we will have to include them in our data logger codes.

One strategy might be to start with the example code so we are sure we have all of these new parts included.

7.4 Adding in your sensor

So now that you have your time being recorded (plus a random number) we want to make the modification so that our data logger is recording temperatures. We will start with the thermistor from your Arduino kit. This thermistor was produced by the Elegoo company that made your kit. It might be tempting to google thermistors to find out how to use it. **But don't do this!** This thermistor was made by a specific company, and we need to go to that specific company to learn how to use it. We might find similar instructions for other thermistors from other companies, but they almost certainly won't be right for our specific thermistor. We need the Elegoo instructions.

And these instructions are hidden on the CD that came with your kit. But who has a CD player in their computer anymore? To get around this we will go to the Elegoo web site and download their instructions. Here is a link:

Link to Elegoo website

Once you are there look for their Support area and choose "Arduino kits files downloads." For any sensors you get (including the ones for your group design projects) the procedure will be similar. You will need to get information for your Arduino code from the manufacturer of your sensor. We have an UNO R3 starter kit, so choose that option on the website. And download the files for the Super Starter Kit. The Elegoo web site changes from time to time, so you may have to do some hunting. Once you have downloaded the zip file, you will find it includes a PDF file that tells you how to use all of the things in your kit. We want the thermistor. That shows up in "Lesson 15" but there is a catch. Elegoo's lesson 15 has an LCD display that is nice, but not what we want. So you can't just take their wiring diagram without thinking. you need to look for the part that is just for the thermistor. A comparison with Elegoo's Lesson 14 might be helpful, because Lesson 14 shows how to hook up the LCD display. You don't want any of that. So take that away and see what is left! You will see that you need your thermistor and a resistor. The wiring is pretty easy. Does it matter which resistor? Of course it does. We need to follow the manufacturers directions if we want our temperature measurement to work.

What about the code? We know we need to add something to our data logger code. If you look at the Elegoo files you downloaded, you will see example code for all of the lessons. Under Lesson 15 there is code called Thermometer.ino. That looks promising! But we know we will need just the part about measuring temperature, not the part about LCD displays. When you open the code you will find something like this:

```
int tempReading = analogRead(tempPin);
// This is OK
double tempK = log(10000.0 * ((1024.0 / tempReading - 1)));
tempK = 1/(0.001129148+(0.000234125+(0.0000000876741*tempK*tempK))*tempK);
float tempC = tempK - 273.15;            // Convert Kelvin to Celcius
```

This looks like the part that we need! And it fits our model for making a new instrument! We are converting temperature into a voltage using a piece of equipment (the Thermistor) and then using math to convert from voltage back to temperature so we can print the temperature. Of course there must be some physics behind this math. And a good book on thermal resistance could give you that physics. The specific values in the math are usually determined by a curve fit for the specific thermistor. Because this isn't a class on thermal resistance, we won't study this physics. But we will use the results of other scientists who have by using their mathematical formula to recreate temperature from our voltage measurement.

You will need to add this math into your data logger code. Be careful. I showed the math for the conversion, but you might need other parts of the code (e.g. do you need to set tempPin?). This math should replace the code that makes random numbers in our data logger code. Your challenge is to make this integration of the new and old code happen.

7.5 Digital Data

You might be tempted to complain about how Elegoo built this temperature sensor. Couldn't they have worked a little harder and incorporated all this math into their sensor? Of course we would have to pay more for such a sensor, but if it's not too much more it might be worth it!

The answer is "yes!" And it turns out we have such an upgraded temperature sensor in our kit. It is the DHT11. It is blue and looks like a little blue box attached to a small circuit board.

But what does it look like to get the data already formatted for us? To do this computer engineers have developed ways to transfer data from one piece of electronics to another. We have already leaned about one of these, the Universal Serial Bus or USB. But there are other ways of doing this and our DHT11 uses the digital pins on our Arduino to transfer its data. You can see this in the Elegoo Lesson 11. Again we have to go to the manufacturer to find out how to use the new sensor! So we look at the Elegoo documentation. And it has a wiring diagram! The DHT11 needs power and a ground and a digital pin.

The Elegoo data files also include an example code. This time the code is much simpler. There is no LCD display part to ignore. Your job is to integrate this into a new copy of your data logger example code and to save the temperature data to the SD card.

If you have time, it might be good to try to write out the data to the serial monitor and the SD card in a nice format that is easy to read.

7.6 Difficulties with Larger Data Sets and Sensor Libraries

So far logging temperatures has been a challenge, but not too bad. But there can be some difficulties. Arduino microcontrollers have very little memory. Our DHT11 only gives two data points, temperature and humidity. But some sensors have more. The Adafruit BME680 also measures temperature, humidity, atmospheric pressure, altitude, relative humidity, and air quality. Our data loggers can handle all this, but in our Arduino code we have to become much more frugal about memory use. In the DHT11 code we could make the data easy to understand by putting in words and units in the output. For the BME680 we don't have that privileged. We won't use this today (and depending on your group design project, maybe not at all), but here is an example you can download of a much more memory frugal code for the BME680 sensor.

Download here

You might have to revisit this as you write codes for our group design projects.

7.7 Lab Assignment

Work in groups of three to five for this set of problems. We have enough equipment for you to each build your own data logger, but work together and don't go on to another step until each te]am member has completed the previous step.

1. Get the data logger shield up and running and set the correct date and time if needed.

2. Modify the code to take data from a thermistor (included in your kit) and do the math to turn the thermal resistance into a temperature. You will have to look at your Arduino kit manual to know how to write this code. You can start with the example code, but you will have to modify it for the thermistor measurement. Record the temperature on the SD card.

3. Remove the SD card after the data collection is complete, and make sure the data makes sense (compare to a thermometer in the room) and that the SD card writing is working.

4. If there is time, try powering your sensor system on a battery to make sure it can operate independently.

5. If there is time, switch to the digital temperature and humidly sensor. Modify your sketch to read in and output both temperature and humidly values. Again you will have to look at the Arduino kit manual to figure out how to do this. Check your data file to make sure all is working

Chapter 8

Inductance and Series RLC Circuits Part 1

Hopefully by now you have learned in your PH 220 class that magnetism is related to current. Oersted discovered this by accident (so all those accidents we have experienced in our lab could be telling us something!). If you don't know yet, you will soon learn that changing magnetic fields can cause currents. This is called induction. Electronic circuits often use the ability of currents to cause magnetic fields and the ability of changing magnetic fields to cause currents. Devices that use magnetic fields are called *inductors*. Some of you will know about inductance already. You can safely skip the next sections on inductance. But a review might be a good idea since our theory of inductance is the model we will be testing today.

We're not going to use our Arduino's today. Instead, we are going to use a device that measures voltage as a function of time and plots it. We studied this device briefly back in our second lab. It is called an oscilloscope. We will get some experience with this device today, and then try to build such a device with our Arduino in our next lab.

8.1 The Model: Self Inductance

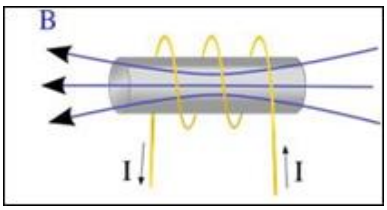
When we put capacitors and resisters in a circuit, we found that the current did not jump to it's maximum value all at once. There was a time dependence. But really, even if we just have a resister (and we always have some resistance!) the current does not reach it's full value instantaneously. Think of our circuits, they are current loops. It turns out that electric and magnetic fields are involved both generated by current loops. For those of you who have gotten this far in PH220, we are going to use Lenz's law. As the current starts to flow, Lenz's law tells us that there will be an induced potential energy difference $\mathcal{E}$ that will oppose the flow. The potential energy drop across the resister in a simple battery-resister circuit is the potential drop due to the battery potential energy

($\mathcal{E}_{bat}$), *minus the induced potential energy drop.*

$$\mathcal{E}_R = \mathcal{E}_{bat} - \mathcal{E}_{induced}$$

If you are remembering (or currently learning) your PH220 you will know that we stop calling a potential energy change per unit charge a voltage (ΔV) and start calling it an emf ($\mathcal{E}$). This is tradition. The letters "emf" don't sand for anything. There is a subtle difference in meaning between voltage and emf. But we will let your PH220 class go into that.

We can use the fact that there is an induced emf to control current in circuits. To see how, we can study a new case



Let's take a coil of wire wound around an iron cylindrical core. In the picture, we start with a current as shown in the figure above. If you have studied inductance, you know that there will be a magnetic field created by the current in the wire. And if you have gotten this far in PH220, we can find the direction of the B-field using our right hand rule number 2. But we now will allow the current to change. As it gets larger, we know

$$\mathcal{E} = -N\frac{d\Phi_B}{dt}$$

and we know that as the current changes, the magnitude of the B-field will change, so the flux through the coil will change. So we will have an induced emf. The induced emf is proportional to the rate of *change* of the current.

$$\mathcal{E} \equiv -L\frac{\Delta I}{\Delta t}$$

You might ask if the number of loops in the coil matters. The answer is yes. Does the size and shape of the coil matter. Yes, but we will include all these effects in the constant L called the *inductance*. It will hold all the material properties of the iron cored coil. And in designing circuits, we will usually just look up the inductance of the device we choose, like we looked up the resistance of resisters in our labs.

For our special case, we can calculate the inductance, because we know the induced emf using Faraday's law (or you will soon be able to do this in PH220).

$$\mathcal{E} = -N\frac{d\Phi_B}{dt} \equiv -L\frac{dI}{dt}$$

so for our case

$$-N\frac{d\Phi_B}{dt}\frac{dt}{dI} \equiv -L$$
$$L = N\frac{d\Phi_B}{dI} \approx N\frac{\Delta\Phi_B}{\Delta I}$$

if we start with no current (so no flux)

$$L = N\frac{\Phi_B}{I}$$

8.1.1 Inductance of a solenoid

We will use a coil much like the one above, but with no iron bar in the middle. We will try to find the inductance of this coil. Fortunately, this is one easy case we can do by hand. So let's do it! We call a coil a *solenoid.* Take a solenoid of N turns with length ℓ. We will assume that ℓ is much bigger than the radius r of the loops. We can use Ampere's law to find the magnetic field in this case

$$\begin{aligned} B &= \mu_o n I \\ &= \mu_o \frac{N}{\ell} I \end{aligned}$$

where $n = N/\ell$ is the number of turns of the coil per unit length. The flux through each turn is then

$$\Phi_B = BA = \mu_o \frac{N}{\ell} IA$$

then we use our equation for inductance for a coil

$$\begin{aligned} L &= N\frac{\Phi_B}{I} \\ &= N\frac{\left(\mu_o \frac{N}{\ell} IA\right)}{I} \\ &= \frac{\mu_o N^2 V}{\ell^2} \\ &= \mu_o n^2 V \end{aligned}$$

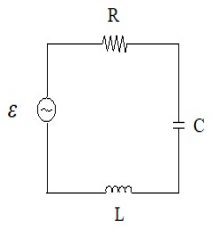
$$L = \mu_o n^2 V \tag{8.1}$$

where V here is the volume of the solenoid, $V = A\ell$. We have reduced our inductance model to a simple equation. And we can test simple equations! We are getting towards something we can test with an experiment.

8.2 RLC Series circuits

Today we will use a coil of wire, a solenoid, as your inductor. We will also use a capacitor and will have some resistance–because there is no way to have real

wire without some resistance. We will connect these in series with a source of alternating voltage (or alternating emf). We will use our signal generator to get a sinusoidally changing voltage. The circuit diagram should look like what you see in the previous figure where the small coil shaped element is the inductor.



Some of you will remember from PH123 that when a harmonic oscillator was driven at the natural frequency we had resonance. Let's look at the current of our *RLC* circuit. It has an equation very like a harmonic oscillator. The current is given by

$$I_{rms} = \frac{\Delta V_{rms}}{Z}$$

or

$$I_{rms} = \frac{\Delta V_{rms}}{\sqrt{(R)^2 + (X_L - X_C)^2}}$$

where $X_L = 2\pi f L$ and $X_C = \frac{1}{2\pi f C}$. When $X_L = X_C$ this will be a maximum. This is a form of resonance. You will remember resonance from swinging as a child. When your parent pushed you at just the right time, you went higher. We would say your swing "amplitude" got bigger. The same thing is happening here. The signal generator is "pushing" the current through the capacitor. If it pushes at just the right frequency, the current will get large.

Starting with $X_L = X_C$, we can find the frequency that will be the resonant frequency, the frequency of the "push" that will make the current biggest.

$$\begin{aligned} X_L &= X_C \\ 2\pi f L &= \frac{1}{2\pi f C} \end{aligned}$$

then

$$f^2 = \frac{1}{4\pi^2 LC}$$

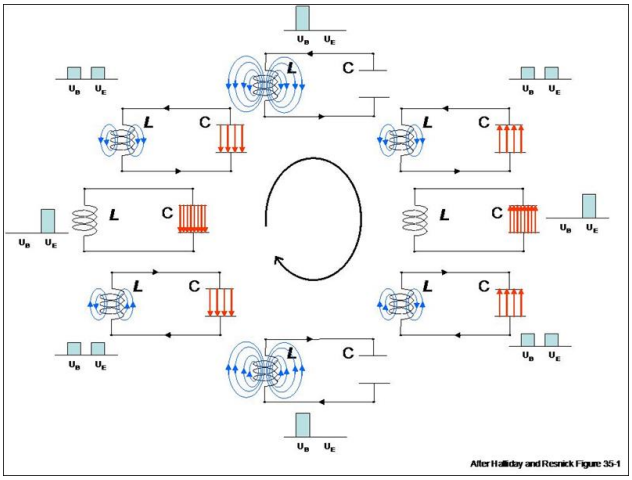
or

$$f = \frac{1}{2\pi\sqrt{LC}} \tag{8.2}$$

Why do we care? This is a tuning circuit used in radios! We can include a variable capacitor or a variable inductor in the circuit, and make it resonate with a desired frequency. Usually a variable capacitor is used. So when you turn the dial on your radio to adjust the frequency, you are changing the capacitance of a variable capacitor! But more importantly for us physicists (and in this class

we are briefly all physicists) this provides and easy way to test our inductance theory. We can send into our circuit a varying emf and if we get just the right frequency, the measured emf across some element in the circuit should get very large. So even if we put in a tiny varying emf, we get a measurable result. If it works, this is a verification of our inductance model.

If we had a superconductor with no resistance, the oscillation would go on forever. Energy would be stored in the capacitor, say, to start out, but as the current flows a magnetic field is produced. Energy is stored in this magnetic field. The energy for a resistance-less system would travel from the electric field to the magnetic field and back. The cycle would go on forever. But we will need



to be careful. We have real resistance in our lab wires, and the resistance acts like, well, resistance. Resistance is a non-conservative force, so the resistance will dissipate energy and our oscillation will eventually die out. But since we are adding in energy from the signal generator, the effect of resistance will be to make the maximum voltage of our sine wave less. So in designing your circuit, you will want to make sure your R is not too big.

I'm going to suggest that we take the resonance part of our inductance model to perform our model test. Consider the resonant frequency of a LRC circuit. If we could find the frequency at which our LRC circuit goes into resonance, then we could solve for L

$$L = \frac{1}{4\pi^2 f^2 C}$$

Since the capacitance is marked on the capacitor, we can calculate L knowing f. We could compare to our calculated value using

$$L = \mu_o n^2 V$$

to see if our solenoid inductance model worked. All we have to do is to measure f.

Really we know enough to make our experiment work. We find f such that the system is in resonance and then use

$$L = \frac{1}{4\pi^2 f^2 C}$$

to find the inductance of the coil. But we can envision this better if we do a little bit of higher math and draw another graph. Suppose we hook up our series LRC circuit as described above, and we acknowledge that we do have resistance. We would find that we could describe our physical situation with a differential equation. Not everyone is taking differential equations (M316) concurrently with this class, but most of us will take this class at some time. We would find that the differential equation for the amount of charge on the capacitor for our circuit would look like this

$$L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{Q}{C} = \mathcal{E}\sin(\omega' t)$$

where $\mathcal{E}$ is the maximum emf of our signal generator setting and ω' is the frequency setting of our frequency generator. Now think, resonance means that the amount of charge on the capacitor gets big. After taking a differential equations class, you will be able to solve for the charge on the capacitor as a function of frequency. And then, using

$$Q = C\Delta V$$

you will be able to find the voltage across the capacitor as a function of signal generator driving frequency, $f' = \frac{\omega'}{2\pi}$. Your solution will look something like this

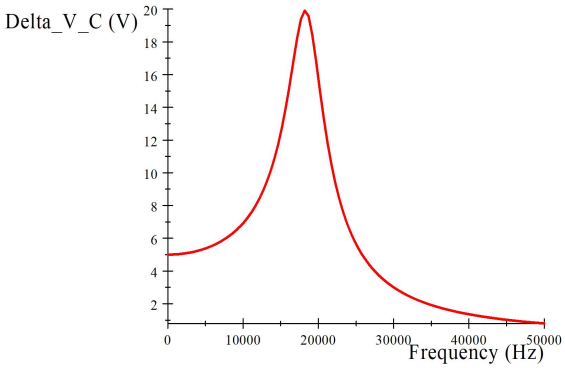
$$\Delta V_C = \frac{\frac{E}{CL}}{\sqrt{\left((2\pi(f))^2 - (2\pi f')^2\right)^2 + 2\left(\frac{R}{L}\right)(2\pi f')^2}}\sin(\omega' t - \phi)$$

Notice the term $\left((2\pi(f))^2 - (2\pi f')^2\right)^2$ in the denominator. When the resonance frequency f and the driving frequency f' are the same, this term will be zero, and the denominator will be small. That makes the size of our ΔV_C sine wave big. That is resonance. Let's plot just the amplitude term (the part in front of the sine function) so we can see what it looks like. For a circuit with

$$\begin{aligned} \mathcal{E} &= 5\text{V} \\ L &= 2.3\times10^{-3}\text{H} \\ R &= 1\text{M}\Omega \\ f_{resonance} &= 18.552\text{kHz} \\ C &= 32\times10^{-9}\text{F} \end{aligned}$$

$$R > \sqrt{\frac{42.3\times10^{-3}\text{H}}{32\times10^{-9}\text{F}}} = 1149.7\Omega$$

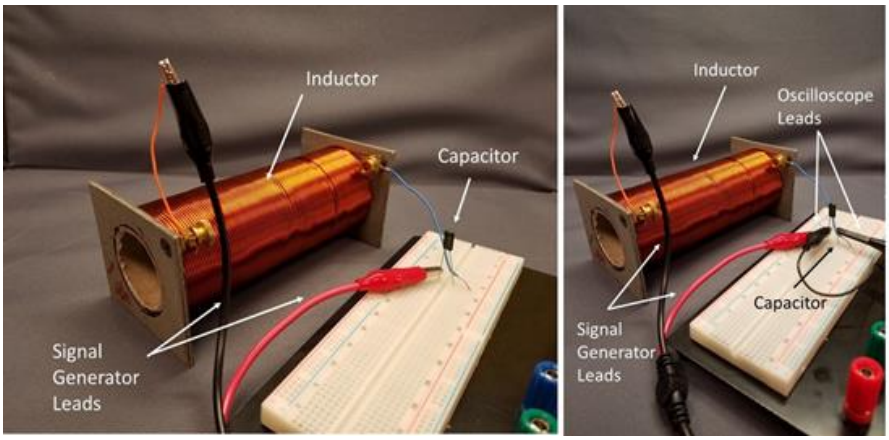
we get the following graph

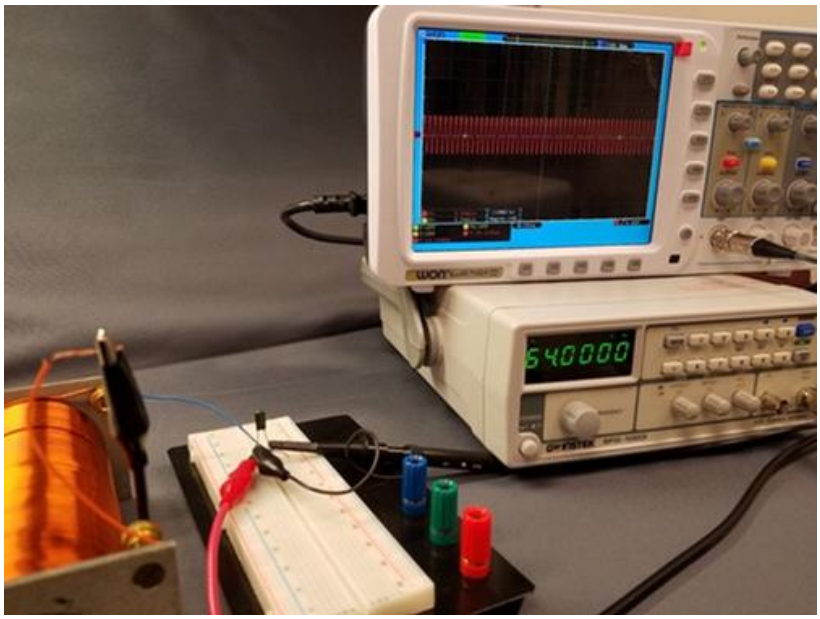


Let's interpret this graph. Suppose we start with our signal generator with a low frequency, say, 100Hz, We should see a sine wave on the oscilloscope at the same frequency as the signal generator frequency, but with an amplitude of about 5V (from the midpoint to the peak). Then we change the signal generator driving frequency until the measured voltage across the capacitor becomes very large. We want the maximum amplitude. That will be when $f = f'$ and we can read the resonance frequency off of the indicator on our signal generator because the signal generator frequency is equal to the resonance frequency. If we increase the frequency more, the amplitude will go down, making our sine wave smaller. You should check this to make sure you have found the maximum amplitude.

8.3 Lab Assignment

1. Estimate the inductance of your coil using equation 8.1.
2. Find The inductance of the coil using an Oscilloscope
   (a) Predict the resonant frequency of your *LRC* circuit using the large wire coils and something close to a non-electrolitic capacitor. Note that we are not putting in a resistor, the resistance of the wire in our indictor is enough to create damping. Also notice that some of the multimeters have a capacitor tester on them. You might want to check your capacitance for the capacitor you choose.
   (b) Set up an *LRC* circuit . Use one of our signal generators to produce as the source of variable emf.
   (c) Test the circuit using the oscilloscope. Make sure you see a nice sine wave. Either side of the capacitor is a nice place to hook the





oscilloscope probe, if your oscilloscope has a ground lead, hook it to the other side of the capacitor.

(d) Adjust your signal generator near your natural frequency of oscillation. Note what happens to the amplitude as you tune the dial.

(e) Determine your actual resonant frequency, f_A.

(f) Calculate your inductance based on f_A. Compare to your calculated value. If there is a difference, try to explain it.

Part III

Student Designed Experiments

Chapter 9

Student Designed Experiments

The rest of the course (up to the final) consists of student designed experiments. The process for the student designed experiments is as follows:

1. You and your lab group fill out a brainstorming sheet to come up with possible experiments. You and your lab group prioritize the list and hand it in.

2. From that list I will approve an experiment to try.

3. You will have to write a proposal that describes what you plan to do for your experiment.

4. Upon approval, you will perform the experiment, and report on the results in a written paper and a (formal) oral presentation.

We have talked about each of these parts along the way. In this section of the manual, I will describe what I am expecting for each of these assignments. You will have seen some of this before.

9.1 Proposal

You already have produced a proposal. This document was what you used to convince me that your experiment could work and that you should be given the resources and support to perform the experiment. Your proposal has the following parts:

1. Statement of the experimental problem

2. Procedures and anticipated difficulties

3. Proposed analysis and expected results

4. Preliminary List of equipment needed

We will now reuse these parts to perform the experiment and produce your final paper and presentation.

9.2 Performing the experiment

I will provide you with the equipment we have agreed upon from your proposal. You will have three lab days to perform your experimentation. I will be available for advice and to watch for problems or safety issues. But you and your team will perform the experiment. You will want to keep good notes in your lab notebook. You will likely have to change your procedure after you start because of problems. Take careful note of what was actually done, and what your measurements were. Note any unusual things that happen. Carefully record what you do.

9.3 Written report

The written report is designed to match a normal format for an applied physics article in a journal like*Applied Optics* or the *IEEE Transactions* journals.. There should be an introduction, description of the procedure, description of the data and results, a description of the analysis, and a conclusion. These sections are described in detail in the following table.

9.4 Oral report

Your group will have ten to fifteen minutes to explain your experiment and present your results and conclusions. I will grade your presentation on the following areas:

Professionalism in the delivery	25
Clarity of the delivery	25
Quality of Visual Aids	25
Team support and Participation	25
Total	100

The format of the presentation should follow the format of the written report. Don't forget to give proper credit for pictures, or ideas and quotations in your presentation just as you will in your written report.

9.5 Lab Notebook

Hopefully you noticed that a lab notebook is required for this class. The lab notebook is designed to be a record of what you did. If you had to repeat today's experiment five years from now, could you do it based on what you write today?

At most professional labs and major engineering companies your lab notebook is considered the property of the company or organization. It is the proof that you did the experiment that you say you did, and that you got the results you say you got. It has to be readable and understandable to someone who did not participate in the lab with you. This is a pretty tall order.

Of course the evidence that you participated in the group project will all be found in your lab notebook. You have had experience in PH150 and throughout PH250. So you are an expert in keeping lab notebooks. But as usual with a group project not all of what happens will be written in your notebook. Some will be in your coworker's notebooks. That is fine, because you know to refer to that work in your notebook with a reference to the notebook of the person that did the work. Note that the grade for the lab notebook is a large part of your semester grade, this represents the fact that your lab notebook is a large part of what a scientist does. Remember that the lab notebook must be kept *as you go*. It is not OK to try to recreate it after the experiment is over. This takes time away from fiddling with equipment and thinking about procedure, but it IS PART OF PERFORMING THE EXPERIMENT. So recreating something at the end is the same as not doing the assignment. When you are practicing in your field you will find that courts of law feel the same way about lab notebooks. To prove you own the intellectual property you have developed, the lab notebook has to be kept *as you go*.

Here are reminders from PH150 on how to keep a lab notebook:

9.5.1 Designing the Experiment

In PH150 we learned that to design an experiment we needed the following steps. Some evidence of these steps should be found in your lab notebook:

1. Identify the system to be examined. Identify the inputs and outputs. Describe your system in your lab notebook.

2. Identify the model to be tested. Express the model in terms of an equation representing a prediction of the measurement you will make. Record this in your lab notebook. (If you have not solved this problem in your PH121 class yet, call me over and we will go through it together).

3. Plan how you will know if you are successful in your experiment. Plan graphs or other reporting devices. Record this in your lab notebook.

4. Rectify your equation if needed. Record this in your lab notebook.

5. Choose ranges of the variables. Record this in your lab notebook.

6. Plan the experimental procedure. Record this in your lab notebook.
7. Perform the experiment . Record this in your lab notebook (see next section). You will need your uncertainty equations from the proposal.

9.5.2 Performing the Experiment

Step 8 is really many individual steps recording the actual performance of the experiment. You learned this in PH150, but here is a review of the criteria I will use to grade your lab book:

- Describing the goal for the work
  - Usually this takes the form of a physical law we will test.
- Give predictive equations and uncertainties for the predictions based on the physical law.
  - This usually involves forming a mathematical model. You should record any assumptions that went into the model (e.g. no air resistance, point sources, massless ropes, etc.).
- Give your procedure
  - Recording what you really did (not the lab instructions), tell what changes you make in your procedure as you make them.
  - Record as you do the work.
  - Record the equipment used and settings, values, etc. for that equipment (see next item).
  - Did you learn how to use any new equipment? What did you learn that you want to recall later (say, when taking the final, or when you are a professional and need to use a similar piece of equipment five years from now).
- Record the data you used. . The data are all the measurements you took plus your best estimate of the uncertainties in the measurements. Record any values you got from tables or published sources (or from your professor) and state where you got these values. You don't always want to write down all the data you use. If you have a large set of values, you can place them in a file, and then record the file name and location in your lab notebook. Make sure this is a file location that does not change (emailing the data to yourself is not a good plan).
- Give a record of the analysis you performed. You should have given some idea of how you got your predictive equation. Now, what did you do to get the data through the equation? Were there any extra calculations? Did you obtain a set of "truth data" (data from tables or published sources, or from an alternate experiment) for your experiment? If so, did you do any calculations, have any uncertainty, etc. associated with the truth values?

- Give a brief statement of your results and their associated uncertainties.
- Draw conclusions
  - Do your results support the theory? Why or why not? What else did you learn along the way that you want to record.
  - This is where we may compare the percent error to our relative uncertainty.

This may seem like a lot of work (that is because it IS a lot of work). It takes practice to be able to do this professionally, which is why we do it here.

Section/Value	50-40pts	40-30 pts	30-20 pts	20-0 pts
Introduction: **Answers the question "what is this lab about?"**	• Answers the question "what is this lab about?" sufficiently that a person who did not perform the lab would understand • Gives enough background so that the lab report makes sense as a stand-alone document • Tells the reader what your expected outcome is based on theory.	• Answers the question "what is this lab about?" sufficiently that a person who was part of your lab group would understand • Gives enough background so that the lab report makes sense to someone who knows the lab topic well	• Mentions what the lab is about • Gives some background	• It is difficult to tell from the introduction what the lab is about • Little or no background provided
Procedure: **Answers the question "what did you do?"**	• This section answers the question "what did you do?" sufficiently so a non-expert can understand what was done. • Describe the entire procedure, especially indicate any deviations from your plan and explain why those deviations were necessary.	• This section answers the question "what did you do?" sufficiently so your lab partner could understand what was done. • Tells where you deviated from the plan	• Major points of the procedure are listed	• It is difficult to tell what you did from your description
Data: **Answers the question "what did you measure?"**	• Each measured value is given with units • Each value is given with a good estimate of uncertainty • Only measured values that are needed are given • The data is presented in a way that is easy for the reader to find and read. (e.g. label graphs and table columns)	• Each measured value is given with units • Each value is given with an estimate of uncertainty • Extra values that were not needed are given	• measured values are given	• It is not clear what you measured
Analysis: **Answers the question "how did I get from my data to my results?"**	• It is clear how you got from your measured values to your results • Major equations are given and discussed. • The method of determining uncertainties is discussed	• It is possible to tell how you got from your measured values to your results • Major equations are given • The method of determining uncertainties is discussed	• It is possible to tell how you got from your measured values to your results • Major equations are given • Method of determining uncertainty is not discussed	• It is not possible to tell how you got from your measured values to your results • Major equations are missing • Method of determining uncertainty is not discussed
Results: **Gives the results of your analysis**	• There is a clear, understandable answer to the question the lab asks. For example, if I ask you how fast a car is going, the result would be a calculated speed, with its calculated uncertainty and units. • Report percent error or percent difference • Report fractional uncertainty	• There is a an answer to the question the lab asks with uncertainty and units • Report percent error • Report fractional uncertainty	• There is a an answer to the question the lab asks uncertainty and units are missing • Percent error or fractional uncertainty is missing	• There is no clear answer to the question the lab asks • Percent error or fractional uncertainty is missing
Conclusion: **Answers the question "did the experiment show what was intended?"**	• There is a clear discussion of whether the experiment was supported or falsified the theory. • This discussion includes a comparison of the percent error and fractional uncertainty • If there were difficulties, they are discussed here • There is a statement of what you learned from this experiment. Note any problems and how you would resolve them if you were to redo this experiment.	• There is a general discussion of accuracy (often with percent errors quoted) • There is some mention of whether the predictive theory is supported • Problems are noted and how you would resolve them if you were to redo this experiment is discussed.	• There is no comparison of the percent error and fractional uncertainty • There is a statement of what you learned from this experiment.	• There is no outcome of the accuracy of the experiment • There is no comparison of fractional uncertainty and percent error • There is no clear conclusion about the predictive theory • There is little mention of what was learned

Part IV

Testing Models II

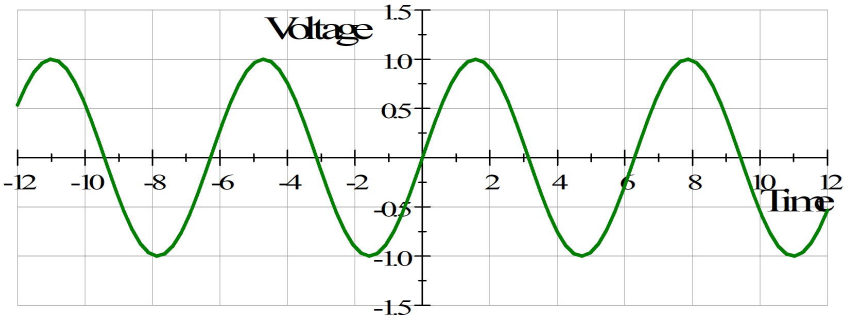
Chapter 10

Inductance and Series RLC circuits Part 2

In our last lab, we used an oscilloscope to measure voltage and to plot it as a function of time. Of course our Arduino can do this. You have seen the serial plotter already as you have used the Arduino software. But you may see a problem with building an actual oscilloscope. For one thing, our signal generator voltage goes negative. Let's consider how we could design a circuit for our Arduino that will allow us to make this kind of measurement.

10.1 The Instrument

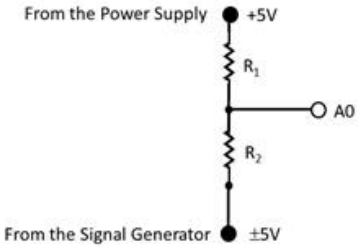
We have a new challenge. Our signal from our LRC circuit is sometimes negative. We will have truly negative voltages compared to our ground. Can we send a negative voltage into our Arduino? The answer is an emphatic NO! Negative voltages will also destroy our Arduino. But we really have a sinusoidal signal.



The easiest way to fix this problem is to use, once again, use our voltage divider. but this time we will fix each end at a different voltage. We will need

one end at a positive voltage. The other can then go as negative as the first end is positive. Let's take a concrete example to show how this works.

Suppose we want to measure a voltage that could be as negative as -5V or as positive as $+5$V. We still need to map this to our 0 to 5V Arduino range. Consider the following voltage divider.



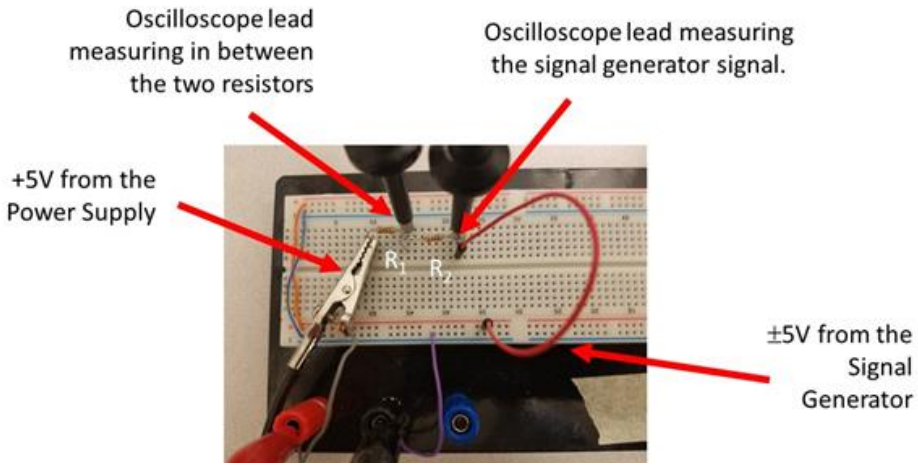
Let's start by setting $R_1 = R_2$. Then any voltage at the junction between R_1 and R_2 should be midway between the voltages input on the top of R_1 and the Bottom of R_2. We put in $+5$V from our power supply on the top. And suppose we hook -5V to the bottom (say, from our signal generator). We expect the voltage divider to divide the total voltage difference in half. We have 10V range from -5V to $+5$V. We expect the junction between R_1 and R_2 to be right in the middle of that range. So we expect to see 0V on pin A0 when the signal generator outputs -5V. So far so good! Now suppose the signal generator gives us $+5$V. Half way between $+5$V and $+5$V is still $+5$V. This is just what we want! Any voltage from the signal generator between $+5$V and -5V will end up between 0V and $+5$V at input A0. For example, say we have -2V from the signal generator. The at $A0$ we will have a voltage half way between $+5$V and -2V. Pin A0 would have 1.5V.

We must be very careful to get this one wired right before we hook it to our Arduino. We also need to make sure our signal generator and power supply are plugged into grounded outlets so they have a common ground.

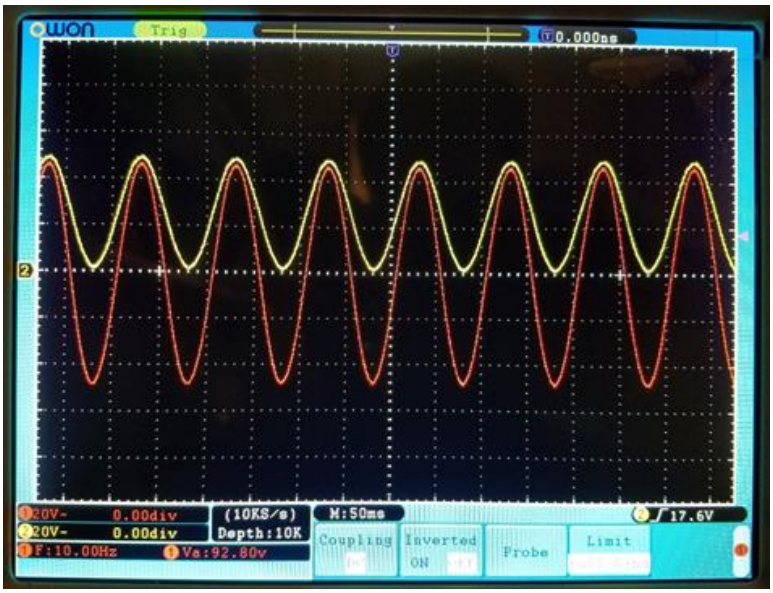
The oscilloscope, power supply, and signal generator are all grounded through their grounded plugs. But our Arduino is not. We need to tie all the grounds of all our equipment together. So put a wire on the power supply negative output and wire it with an alligator clip to the grounded exterior of the signal generator TTL BNC connector. Then take another wire and connect it to the power supply negative output and wire this to a GND pin on the Arduino. This should ensure that all three devices have the same ground point (so we won't get a spark from one to another).

Before we hook this bit of electronics to our Arduino, we want to check it on one of our meters. But our multimeter is not the best choice. For this test, let's use the oscilloscope.

Our oscilloscopes have two channels where we can hook probes. Let's use one to measure the signal as it comes directly from the signal generator. Let's use the other to measure at the junction in between R_1 and R_2 right were we will connect pin A0. That should be our output.



What we see on the oscilloscope should look like this: The red trace is the



signal from the signal generator. The yellow trace is from our voltage divider. Notice that our signal generator is giving us -5V to $+5$V and our output from the voltage divider is 0 to 5V just as we hoped.

Of course in our sketch, we have to do a little math to have our output at the serial monitor be values from -5V to $+5$V. We mapped this to 0 to 5V. That is a 10V range into a 5V range. So each volt measured pin A0 is really worth two volts that were input from the signal generator. But we are offset by

5V, so we need to multiply our 0 to 5V ADC units by 2 and subtract 5V

$$\Delta V_{measured} = 2\Delta V_{ACD} - 5\text{V}$$

Notice that our minimum detectable voltage difference of $\Delta V_{ADC\,\text{min}} = 4.9\text{mV}$ will map to

$$\begin{aligned} \delta V &= 2\,(4.9\text{mV}) \\ &= 9\,.8\text{mV} \end{aligned}$$

We have a much higher uncertainty using this set up! So there was a cost to using this method of measuring both positive and negative voltages.

We can use the Oscilloscope stand-alone device to measure our voltages before we hook up our delicate Arduino. The Oscilloscope is designed to make this type of measurement. It likes periodic signals and it likes positive and negative voltages. It can handle fairly large voltages. It is nice because it plots the voltage vs. time graph. It has adjustment knobs to change the scale of the graph axes.

We can set up our circuit and clip the oscilloscope probe to either side of the capacitor. As we change the frequency of the signal generator the frequency of the voltage across the capacitor will change. As we get near the resonant frequency, the amplitude of the capacitor voltage will grow. Right at the resonant frequency, ΔV_C will be largest. We will need to measure this before using our Arduino to be sure the voltage won't go over 5V. We need to keep it to less than $\pm$5V at resonance. If we keep changing the frequency the amplitude will go back down. So when we see the amplitude grow, max out, and then diminish we know we have just passed resonance. This is just what we saw in last week's lab.

Once we have this working on our oscilloscope, we can try it on our Arduino. Of course we need a sketch for this. Here is a simple example. Download here

```
////////////////////////////////////////////////////////
// Extended Voltmeter for plus and minus five volts
// This voltmeter with the values given below
// is designed to measure a-5V0 to +5V range with 1014
// discrete values of with an uncertainty of about 98mV.
// A voltage divider is use with equal resistances.
// +5V is given to one side and our +-5V signal to the
// other side. The output voltage is taken in between the
// two resistors. I think we need to wire all the voltage
// devices to the GND pin to keep common ground.
////////////////////////////////////////////////////////
//set up a variable to represent Analog Input 0
int AI0 = 0;
int value = 0;          // Place to put the A2D values
```

```
float voltage = 0.0;  // calculated signal voltage

//////////////////////////////////////////////////////////
 void setup() {
  //Initiate Serial Communication
  Serial.begin(9600);    //9600 baud rate
}

//////////////////////////////////////////////////////////
 void loop() {
  // read the serial data from AI0
  value = analogRead(AI0);
  // if you want to, print out the channel A2D values.
  // Uncomment if you want them.
  //Serial.print("analog channel value ");
  //Serial.print(value);
  // calculate the signal voltage
  voltage=(2*value*(5.0/1024.0))-5;
  // print out the signal voltage
  Serial.print("␣voltage␣");
  Serial.println(voltage, 4);
}
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
```

So our new instrument takes in an alternating voltage, and displays it. We will have to adjust the input voltage on the signal generator. When the signal generator frequency is just right, the voltage we measure should become large. By noting the resonant frequency we can check our model for inductance.

10.2 Sampling Theory, a complication

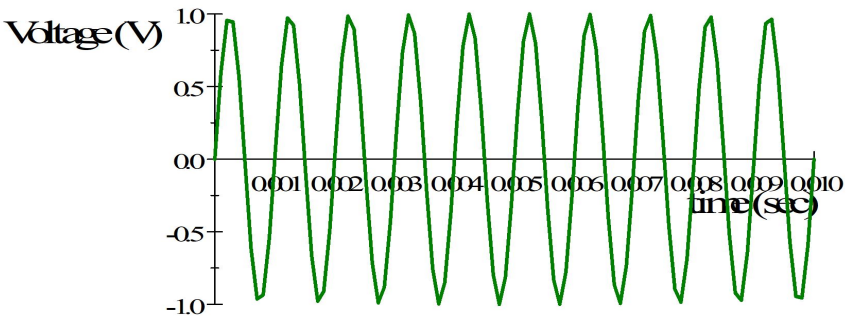
Before we finish designing this experiment, we need to think about another limitation of our Arduino devices. That is that they can only take up to 2000 measurements in a second. We say they have a maximum sampling rate of 2000Hz. To try to understand this, consider trying to measure a sine wave.

There are an infinite number of points in a sign wave. That is,

$$V(t) = \sin(\omega t)$$

for every t at all. So ideally we would have an infinite number of t values between 0 and 1s so that we would not miss any $V(t)$ values. But our Arduino can't take an infinite number of values. It an only take up to 2000 values a second. Suppose we have a signal frequency of 1000Hz. That means there should be

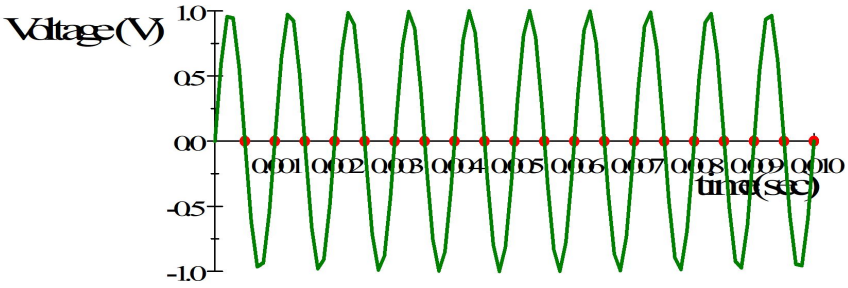
$$\frac{1}{1000\text{Hz}} = 0.001\,\text{s}$$



in between peaks of our sine wave. And suppose we wish to measure this. We can only measure at a rate of 2000Hz, so there will be

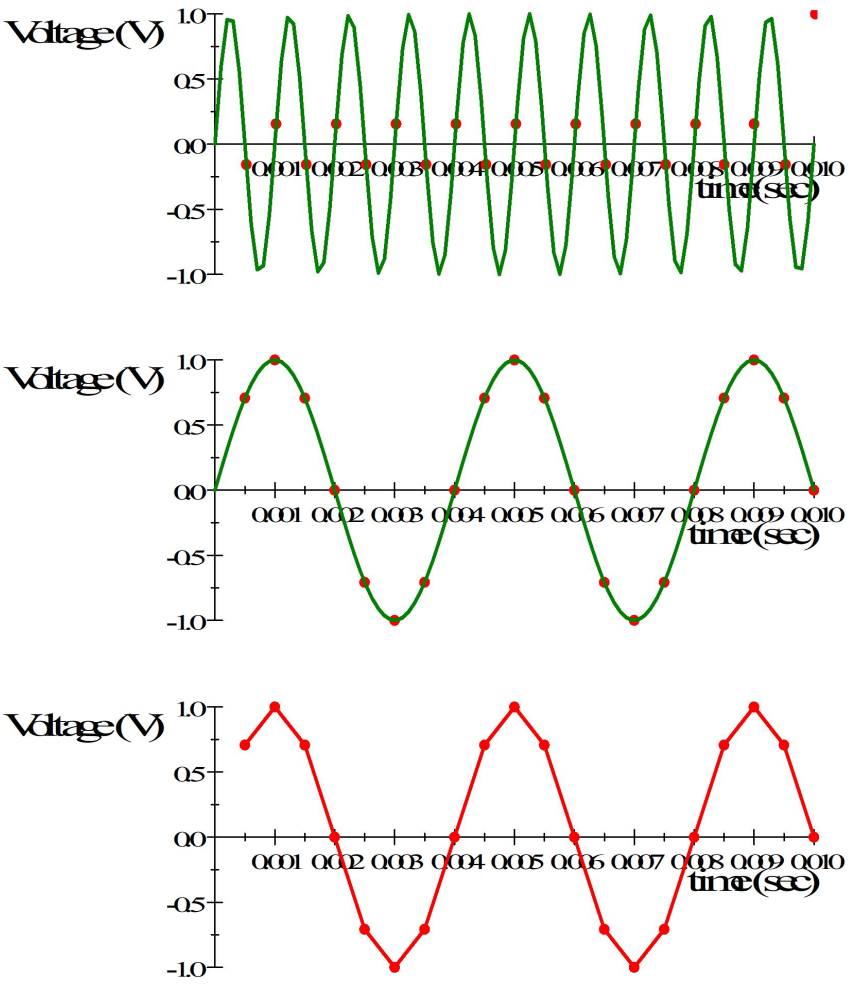
$$\frac{1}{2000\text{Hz}} = 0.000\,5\text{s}$$

between measurements. That would look like this



where the red dots are the measurements. And look, we seem to have had back luck and we got all zeros!. This is no good! Because we are taking measurements in sync with the sine wave, we get the false impression that we are measuring a constant voltage. Even if we offset the dots, it wouldn't help much. Now we do see that we have some change in the voltage, but the measurements aren't representing the actual change. The solution to this problem is to lower our signal frequency so that we have more points per signal period. Say, we have a sine wave with a frequency of 250Hz. Now our graph would look like this.

Now if we just plotted the measurements, we could still tell it was a sine wave. Granted, it doesn't look great, but we wouldn't mistake it for a straight line. This problem of having to take measurements faster than our signal changes is called aliasing (because if you get it wrong, it looks like the wrong function came from the signal generator). We need to make sure our signal frequency is much lower (like ten times lower) than the frequency at which we can take measurements, or we will be fooled. Last lab, we had resonance frequencies that were around 20000Hz. That is way too high for our Arduinos. We need

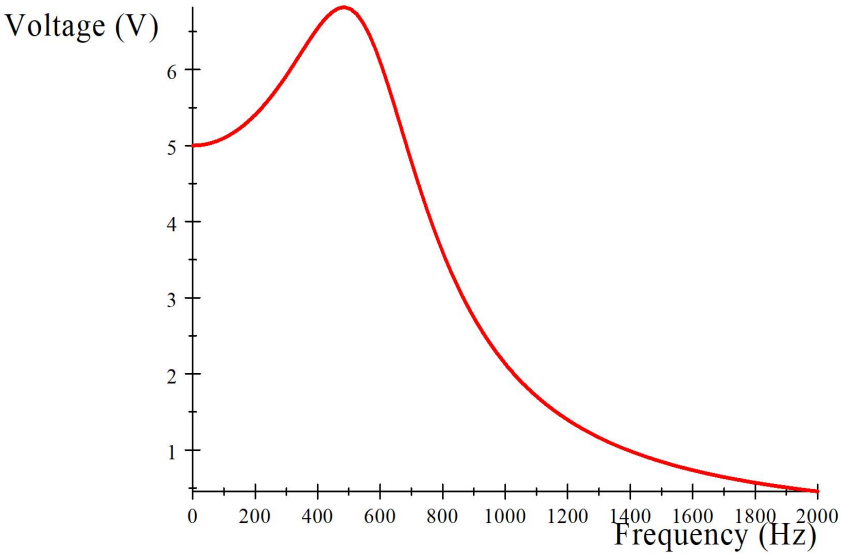


to change capacitors so that our resonance frequency is more like 200Hz or our measurements will be aliased. Suppose we use the following

$$\begin{aligned} \mathcal{E} &= 5\text{V} \\ L &= 2.3 \times 10^{-3}\text{H} \\ R &= 10000\Omega \\ f_{resonance} &= 18.552\text{kHz} \\ C &= 32 \times 10^{-6}\text{F} \end{aligned}$$

We should get something like this for our resonance plot.

And this we could reasonably detect. Our measurements might look somewhat ratty, but we should be able to see a sine wave and find when it is maxi-



mum.

10.3 Lab Assignment

1. Estimate the inductance of your coil using equation 8.1. You may use your estimate from last week and any insight into that estimate that last week's measurements might have given you.

2. Measuring the inductance using resonance with an Arduino based oscilloscope

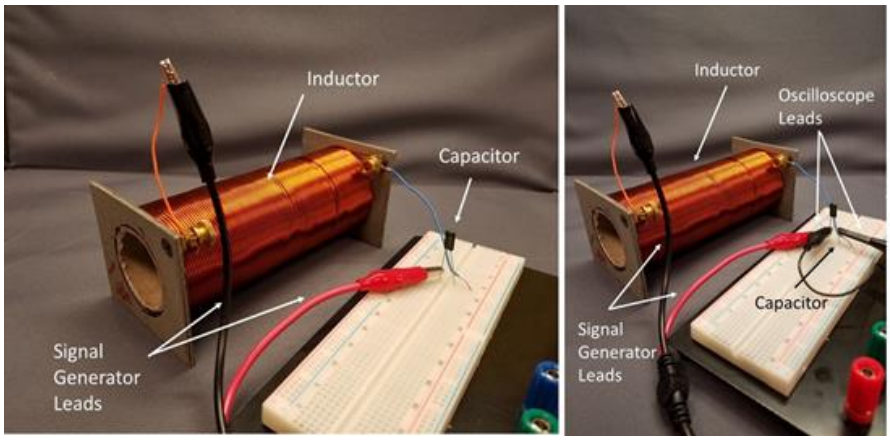
   (a) Set up an *LRC* circuit like we did last lab. Make sure your resistance is not too high using equation **??**. Use one of our signal generators to produce as the source of variable emf.

   (b) Build the circuit described above to measure positive and negative voltages with our Arduios and the serial plotter.

   (c) Predict the resonant frequency of your *LRC* circuit using the large wire coils and something close to a 330nF capacitor (Note, this is a different capacitance than last lab!). This is necessary because our Arduinos can only collect data 2000 times a second. That limits the frequencies we can see.

   (d) Set up an *LRC* circuit with the new capacitor in it (really just keep the same circuit and swap out capacitors

   (e) Test the circuit using the oscilloscope. Again you should see a nice sine wave, but now we need to be sure that the voltage is far less



than our 5V limit for our Arduino. We know the voltage is going to get bit at resonance. So we need to be careful!

(f) Adjust your signal generator near your new natural frequency of oscillation. Note what happens to the amplitude as you tune the dial. This should be the same as what you saw on the Oscilloscope (I might suggest just leaving the oscilloscope connected).

(g) Once again, determine your actual resonant frequency, f_A.

(h) Once again, calculate your inductance based on f_A. Compare to your calculated value. If there is a difference, try to explain it.

3. Now just for fun (time permitting), place a metal rod in your solenoid. Observe what happens to the amplitude. What happens to the resonant frequency? Discuss how metal detectors might work.

4. Check with your professor to make sure everything is set to start your student designed project next week.