PH 250

Intermediate Physics Laboratory for Physics and Physical Science Teaching Majors

R Todd Lines
David Oliphant
Kevin Kelley

February 3, 2022

Contents

List of Figures

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.

Keeping a Lab Notebook

Keeping a detailed and accurate record is a critical laboratory skill. An informal version of this record is often referred to as a "lab notebook", because traditionally they were kept in physical paper notebooks. Lab notebooks would include written notes and observations, equations, and sketches of the experimental apparatus. With the advent of modern technology, we can make these lab notebook even better. If kept as an electronic document, you can easily copy and paste code into the lab notebook, or include photographs of your experiment.

One requirement of this course is that you submit a lab notebook for each lab. The criteria on which they will be graded can be found on your course

syllabus or on the learning management system.

Using This Lab Manual

You should plan on reading the appropriate chapter of the lab manual before you come to each lab activity. This will save you a lot of time once you get into the lab, as you'll already have the "big picture" of what we are doing, and also kind of know how you should proceed. Trying to read along as you go in lab will simply result in you still being in lab after everyone else has finished and left.

Each chapter has a defined list of objectives that tell you what you should be able to do after each lab activity. These objectives are found in a callout box that appears as follows:

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Objective no. 1....
- Objective no. 2....
- etc....

As you begin reading the chapter, pay attention to these objectives. When you have completed the lab, ask yourself whether you have fulfilled those objectives (and whether you would be able to do so again).

Below the objectives, you will find another callout box identifying the concepts that you will need to know from your electricity and magnetism course. Rather than rehash all of those topics in this text, you should simply turn to your E&M textbook (or another resource) and review any items that are unfamiliar. The list of review items will appear as follows:

ITEMS TO REVIEW

Please review the following concepts as part of your preparation for this lab.

- First topic to review....
- Second topic to review....
- etc....

As you proceed through the chapter, you will encounter places where you would now complete an activity in the lab. To make these activities easy to find, they also can be found in callout boxes that look like the following:

LAB ACTIVITY

This is an example of an activity callout box. It will tell you what you need to do at this point in the lab.

Part I

Computer Interfacing

Chapter 1

Introduction to the Arduino

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Explain how to use a prototyping board, including the identification of which pins are electrically connected.
- Demostrate proper soldering technique.
- Describe the two primary functions included in every Arduino sketch, when they are called, and what they do.
- Compile and upload sketches to an Arduino.
- Demonstrate the use of digital output on the Arduino by constructing several circuits involving blinking lights.

ITEMS TO REVIEW

Please review the following concepts as part of your preparation for this lab.

- Voltage (electric potential).
- Current.
- Resistance.
- Diodes and LEDs.

There are two ways in which we want our computers to communicate with our physics experiments:

1. Have the computer control the experiment. For example, we could have the computer turn on our apparatus, or turn it off.

2. Have the experimental device provide data to the computer. The computer can then record this data for later analysis.

In both cases, we need something that goes between the experimental measuring devices and the computer. These go-between pieces are often called Data Acquisition (DAQ) boards. There are many different forms of DAQs. They can cost anywhere from a few dollars to many thousands of dollars.

In this lab, we will use a low cost DAQ, the plans for which the developers placed in the open-source world so that no one would get royalties (including themselves) for the design (this is why the boards are inexpensive). It is low cost, but not low quality. The developers named their DAQ the "Arduino". There are many different variations on the Arduino. We will specifically be using the Arduino UNO.

Like all DAQs, the Arduino is fragile, so we will need to exercise some caution. Arduinos can be destroyed by putting too large a voltage or electrical current into the input pins. They also can be destroyed by mechanical means, such as being dropped or crushed.

By the end of this chapter, you will have successfully used an Arduino to communicate with an electric circuit. However, before we can interface with a circuit, we have to build one.

1.1 Building circuits

When you are first designing and constructing a circuit, you want to be able to easily and quickly make and revise electrical connections. Once the circuit is designed, you often want to make it more permanent with solid connections that cannot be jostled loose. We'll now consider how we go about building these temporary and longer lasting circuits.

1.1.1 Temporary circuits: the prototyping board

Temporary circuits are usually built on a prototyping board, which is a plastic board full of holes, as seen in Figure 1.1. Each of the holes in the prototyping board (sometimes a called a "protoboard" or "breadboard") is a place where you can insert the lead from an electrical component, such as a resistor or capacitor, or a wire. (A "lead" is one of the wires protruding from the component.)

The advantage of a prototyping board is that each of the holes is electrically connected to at least four other holes. Thus, by plugging leads wires into two connected holes, the leads or wires are now electrically connected to each other. The connections in a prototyping board are typically as follows:

- All of the holes adjacent to each of the blue lines are connected. These are typically used for connections to an electrical ground or the negative terminal of a power source.

- All of the holes adjacent to each of the red lines are connected. These are typically used for connections to an electrical power source (generally the positive terminal for DC power sources).
- Each vertically oriented set of five holes in the center of the board are connected.

These connections are illustrated in Figure 1.2. Please note that not all prototyping boards are alike, and the connections may vary from board to board.

As an example, let's consider how we might wire together a simple circuit involving a 9 volt battery and an LED using a protyping board. Note that nine volts is usually a little high for powering a single LED, so we are also going to put a resistor in our circuit to keep the current down.

First, let's connect our battery to the red an blue rows of the prototyping board. You don't necessarily have to connect the power this way (in fact, once you know how the connections in the prototyping board work, you can connect things in any fashion consistent with the circuit you are trying to build), but doing so is generally considered good practice. We can also connect each set of row together, so that we are providing a power source and ground at both the top and bottom of the board. As you know, the color of the wires I use doesn't make any difference in how the circuit operates, but the colors *can* help me keep

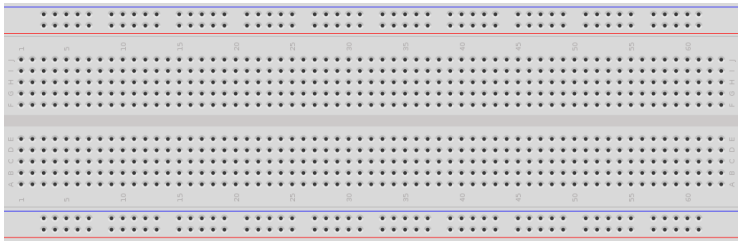


Figure 1.1: An empty prototyping board.

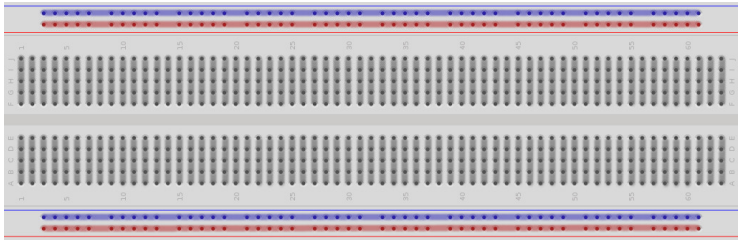


Figure 1.2: Typical connections on a prototyping board. The holes adjacent to each of the red and blue lines are connected to each other, as indicated by the red and blue highlights on the figure. The remaining holes are connected in groups of five, as indicated by the grey highlights.

track of how I've built my circuit. Red is typically used for power sources, and dark blue or black is typically used for ground (or the negative terminal for DC sources). Once I've made these connections, my prototyping board looks like Figure 1.3.

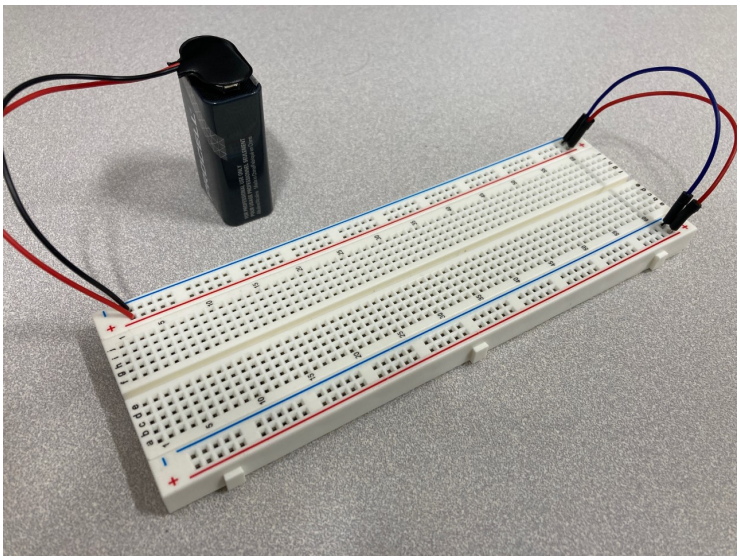


Figure 1.3: Connecting power to a prototyping board. Note that the positive lead from the battery holder connects to the "red" horizontal line on the prototyping board, and the negative lead (the black wire) connects to the "blue" line. The red and blue wires on the right side connect the two red and blue lines together, respectively.

Our next step is to connect the LED to the red power row. LEDs are picky about which lead is connected to power. One of the leads will be longer than the other. This is the lead that should connect to the power. For our circuit, I will plug this long lead directly into one of the holes I've provided power to, and the other lead into one of the center strips, as seen in Figure 1.4.

At this point, our LED has not lit, because it needs to form a complete circuit, which we could do by connecting a wire from the unconnected lead to ground (one of the blue rows). We don't want to do this, though, as the full nine volts from the battery is probably too much for our LED. So we're going to put a resistor in first. I'll use a 330 Ohm resistor for this exercise. I connect one of the resistor leads into the same five hole group that the LED is connected to, and the other lead into a hole from a different group (putting both leads into holes from the same group would be like connecting the two ends of the resistor together, effectively removing it from the circuit). These connections can be seen in Figure 1.5.

Our LED is still not lit. We will now complete the circuit by adding a ground connection. I could have done this by putting the second lead of the resistor

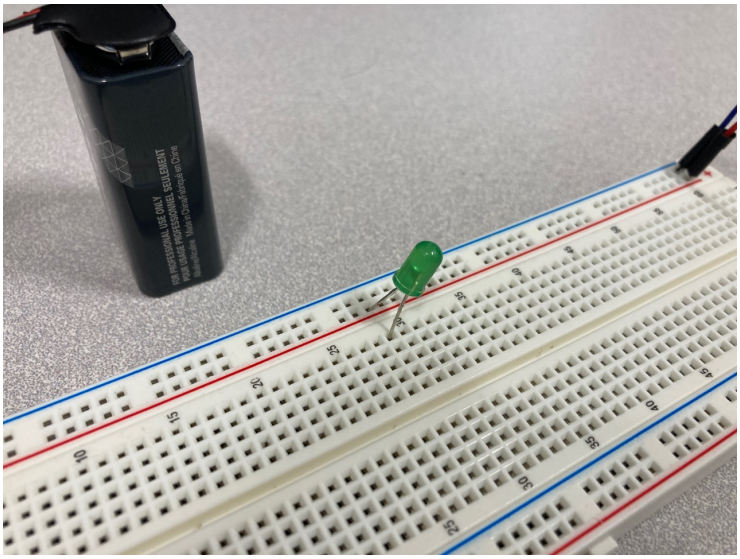


Figure 1.4: Adding an LED to the prototyping board. The long lead is placed in one of the holes connected to power, and the other lead is connected to a center strip.

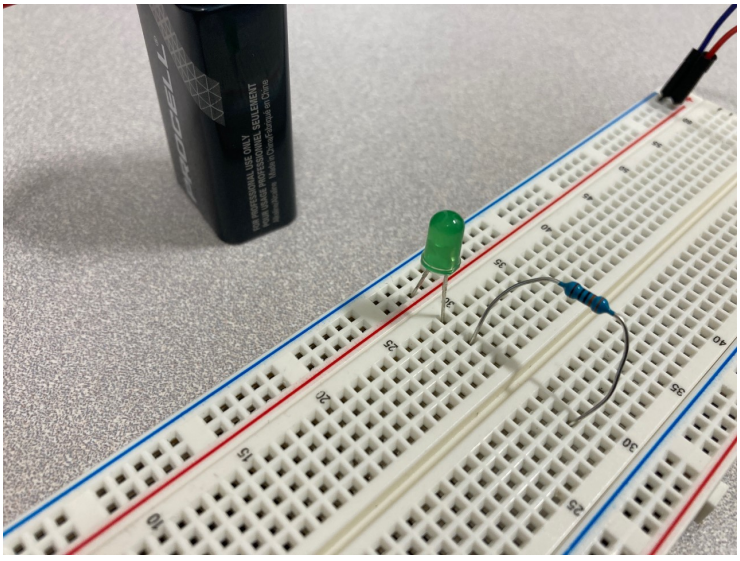


Figure 1.5: Adding a resistor to the prototyping board. One lead is placed in one of the holes in the same group as the LED lead, and the other lead is connected to a different group.

into one of the ground holes, but I'll instead use a wire to finish the connection. The LED is now lit (Figure 1.6)!

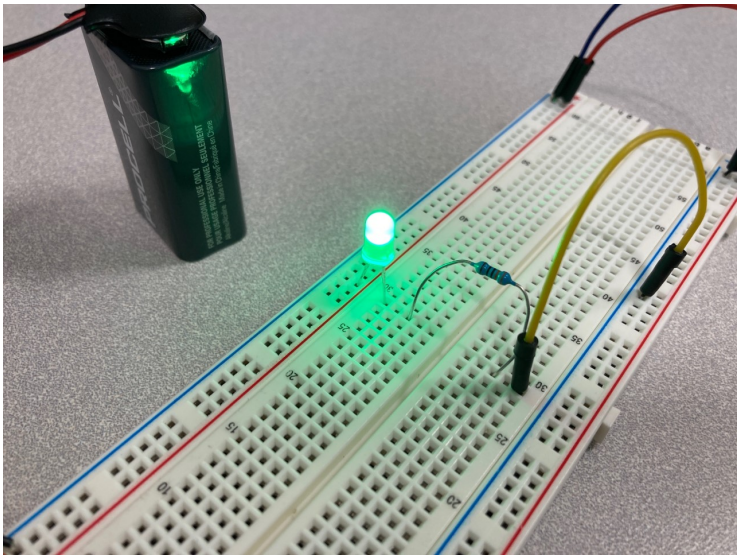


Figure 1.6: Completing the simple LED circuit. One end of the yellow wire is connected to the same group as the end of the resistor, and the other end of the wire is connected to the ground (blue) row. Having completed the circuit, the LED is lit.

LAB ACTIVITY
With the exception of the 9 V battery, build the circuit from this example on a prototyping board.

This concludes our example for using a prototyping board to construct a circuit. Before this lab concludes, we are going to use our Arduino as the power source for this circuit, and also have a little fun using the Arduino to control the behavior of the LED. Before we do that, though, we need to talk about soldering.

1.1.2 Durable circuits: soldering

If you followed the last example carefully, you may have noted a few things:

- Using the prototyping board makes this circuit a lot larger than it needs to be.
- We made electrical connections by easily sliding a lead or wire into a hole. The wire or lead can just as easily be removed from the hole.

- The entire thing looks like it will fall apart if I try to pick it up and move it.

It is fairly apparent that prototyping boards are good for designing circuits and building a prototype for testing (hence the name "prototyping board"). But if I am going to put this circuit to any sort of practical use, I need to build it in a more permanent fashion. The way we do this is by electrically "gluing" the connections together with solder.

Solder consists of a ductile metal alloy (usually tin, silver, and/or lead) mixed with a small amount of material called flux. Solder has a relatively low melting point, and is also electrically conductive. By melting a little bit of solder between the wires or leads we want to connect, and then letting the solder reset, we can make a mechanically strong connection.

In order to make a mechanically strong and electrically sound connection, soldering must be done correctly. The process begins by making sure you have the right equipment and that it has been properly prepared. Figure 1.7 shows the elements of a good soldering station, including (from left to right)

- A soldering iron. This provides the heat to melt the solder.
- Solder. In this case, it is provided on a spool.
- Flux paste. Flux is what helps solder flow. Sometimes solder doesn't have enough flux, and having a little extra paste handy can make a huge difference.
- Tip cleaner. The tip of a soldering iron, after a little use, will start get gummed up with oxides. When this happens, it doesn't conduct heat very well. Frequent tip cleaning makes your life a lot easier!
- Helping hands. This device includes a few clamps that can hold your work in place, and usually also includes a magnifying glass to help you see what you are doing. This piece of equipment is critical! Normally, when soldering a circuit, you need to hold the circuit, the element you are soldering to it, the soldering iron, and the solder all at the same time. Unless you have four hands, you'll need one of these devices!
- Wire wick. Sometimes you make mistakes while soldering. Wire wick, when heated, will sop up solder so you can remove it from your work.

In addition to this equipment, it's also convenient to have a surface to solder our components on, rather than trying to solder them together directly. A common surface is pictured in Figure 1.8, and is called a perfboard. It's basically just a fiberglass board with a bunch of holes (perforations) drilled in it, and the holes are lined with some sort of conductive material.

Now, let's get started. First we'll plug the soldering iron in and give it a few minutes to heat up. Once heated, we then want to clean and prepare the tip. This is done by plunging the tip of the soldering iron in and out of the tip cleaner several times. At this point, we'll also dip the tip very briefly into the

Figure 1.7: Soldering equipment. Included here are a soldering iron, solder, flux paste, a tip cleaner, a set of "helping hands", and wire wick. See the text for a more full description of these items.

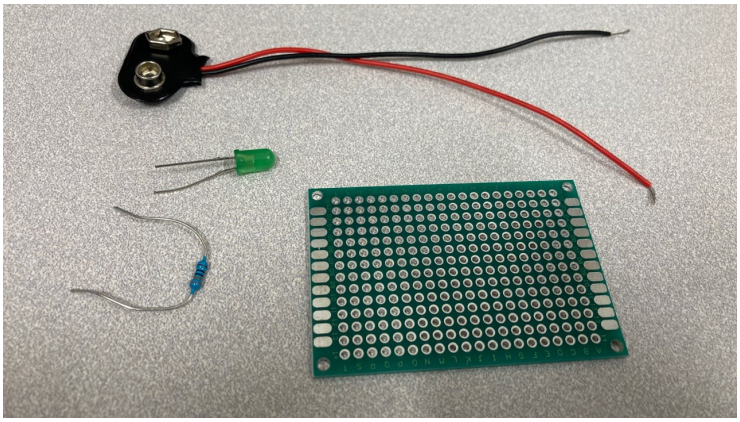


Figure 1.8: A perfboard, along with some items we will solder together on it.

flux paste, and then clean the tip again. Finally, we'll melt a small amount of solder onto the tip, and then clean it a third time. At this point, the tip should be clean and shining with solder, similar to Figure 1.9.

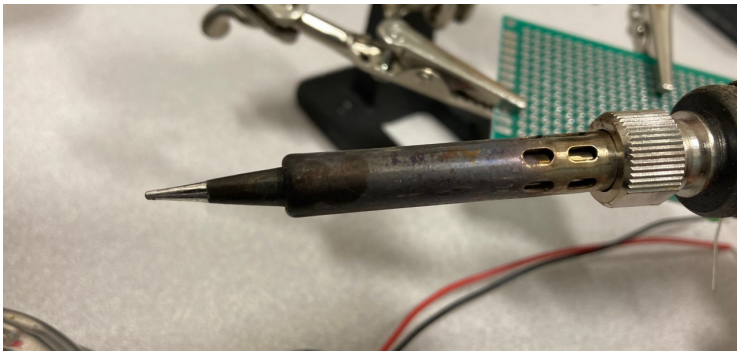


Figure 1.9: A clean soldering iron tip, ready for use.

The next step is to start putting elements on the perfboard. Components with leads can usually be held in place prior to soldering by bending the leads on the backside of the board. These bent leads can also conveniently be used as wires between components as well. In Figure 1.10, all of the components for the circuit have been placed on the boards, and the leads bent to make electrical connections. Normally you would only add a few components at a time, solder those into place, then add more.

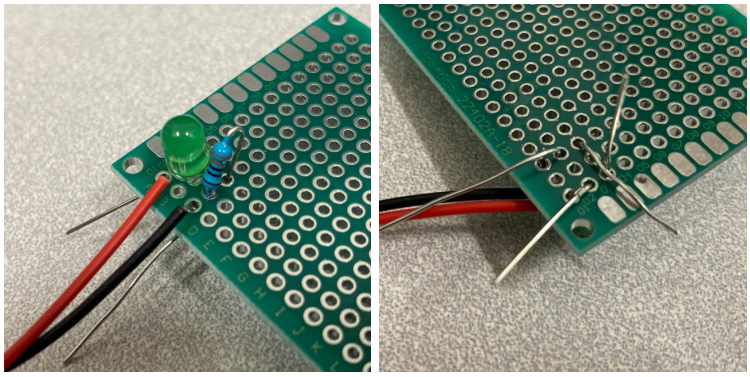


Figure 1.10: Components added to a perfboard, ready for soldering. Note how, on the back side of the perfboard, the leads have been bent to hold the components in place as well as act as wires.

Now to apply the solder. Rather than trying to hold the entire spool of solder, it's easiest to use the soldering iron to burn off a three or so inch segment instead. A key thing to remember is that you should ***heat the work, not the solder***. If you just touch the solder with the soldering iron, it will melt, and it

might even transfer onto the parts you are trying to solder. However, the flux tends to move to the surface of hot metal, and you'll basically end up with a ball of solder that is insulated from the work by a thin layer of flux. This is referred to as a *cold solder* joint, and will likely not provide a lasting electrical connection. By heating the work and then melting the solder onto the work, you'll end up with good connections.

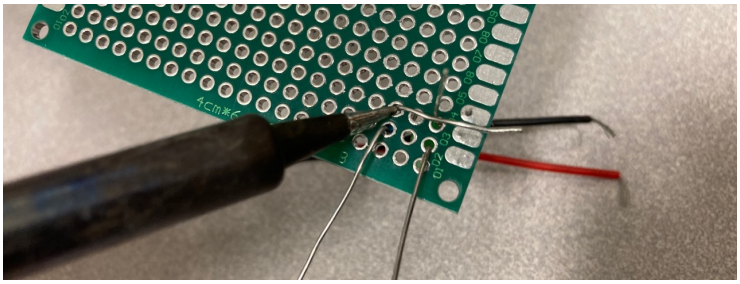


Figure 1.11: Heat the work, not the solder!

Once the iron is removed from the working area, the solder will cool and set very quickly. This process is repeated for each of the connections. Figure 1.12 shows the results with just a few connections left to solder. In this figure, you'll note that one of the clamps on the helping hands is holding the wire in place.

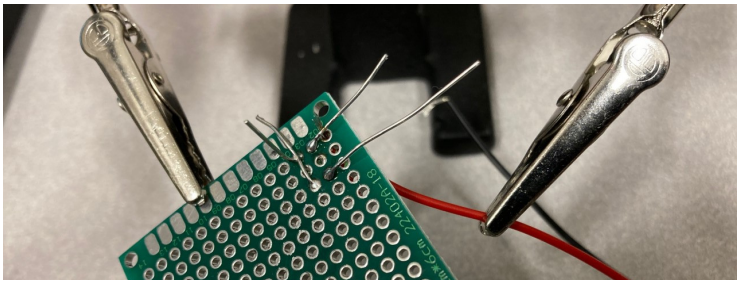


Figure 1.12: The soldering project near completion. The connections to the 9 V battery connector still need to be soldered. Note that one of the clamps of the helping hands is holding the red wire in place.

Once all of the soldering is complete, you'll have some tails of wire and component lead sticking out. Use a set of close clipping wire cutters to remove these extra and unnecessary pieces of metal. Now our project is complete (see Figure 1.13. Note that the finished product is significantly more durable than what we had on the prototyping board, and also significantly smaller. (The perfboard used here is 4 cm by 6 cm, and clearly a much smaller board could have been used.

When a particular circuit needs to be produced in large amounts, someone will usually design and produce a printed circuit board, or PCB. A PCB consists

of several layers of conductors, insulating substrates, and inks. The conducting layers make the connections between circuit elements, so one only needs to solder the circuit elements in place, and not worry about wiring the connections between them. An example of a PCB can be seen in Figure 1.14. The conducting layers are most obviously visible in the lower left portion of the PCB.

The type of soldering presented here is called "through hole" soldering, so named because leads from the components are placed through holes and then soldered into place. Smaller circuit elements are often soldered directly to pads

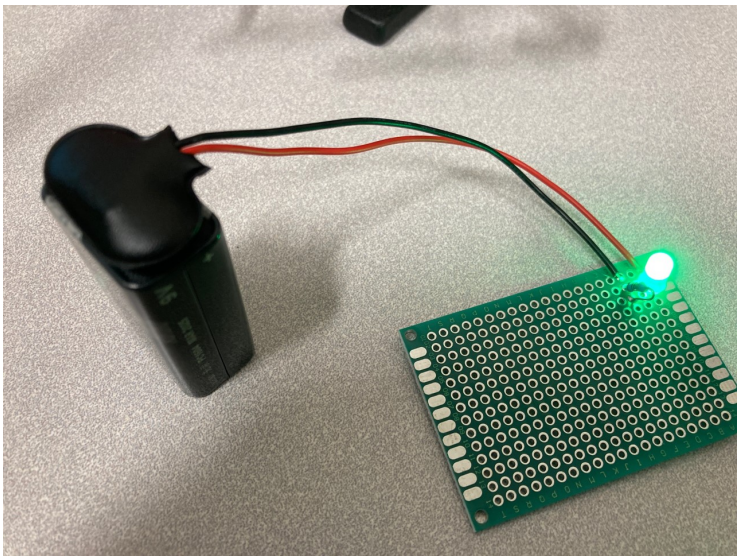


Figure 1.13: The completed soldering project.

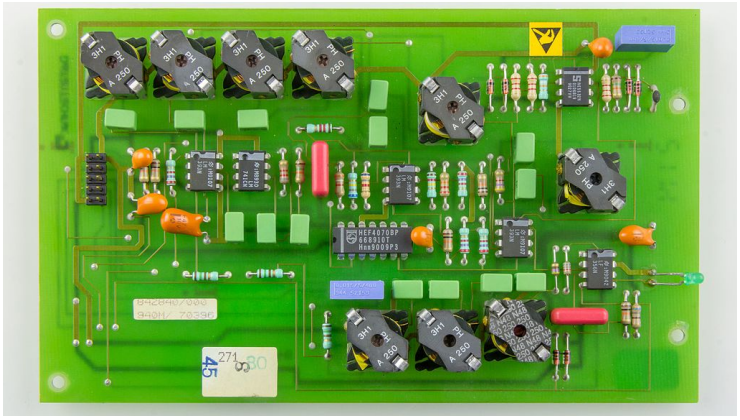


Figure 1.14: An example of a printed circuit board, populated with components. The conducting layers are highly visible in the lower left portion of the board.

on the surface of a PCB. This type of soldering is called "surface mount", and is somewhat more difficult to perform. We will not be doing any surface mount soldering in this lab. Both through hole and surface mount connections can be seen in Figure 1.14. The diodes, resistors, and inductors (all cylindrically shaped components with one or more bands around their circumferences) have been connected with through hole soldering. The ICs (the black boxes) are connected via surface mount.

LAB ACTIVITY
Successfully complete three or more soldering connections.

Now that we've seen how to design, build, and solder circuits, let's look at controlling our circuit using an Arduino.

1.2 Arduino Circuit Control

Let's start our study of computer control by controlling something simple with an Arduino DAQ: the LED circuit that we constructed earlier on the prototyping board. Our objective is to use the Arduino to turn the LED on and off.

1.2.1 Attaching the Arduino to the circuit

Connecting the Arduino to our LED circuit built earlier is quite simple. We will be using the digital I/O ports found on one side of the Arduino, as seen in Figure 1.15. In this figure, you will see that these ports are numbered, starting from the left, 1-13, GND, AREF, SDA, and SCL. Of those ports, the ones of greatest interest to us today are the numbered ports and the GND port (GND stands for "ground"). The numbered ports will be our voltage source, and the GND port will be our ground connection (akin to the negative terminal of a battery). [1]

To connect the Arduino to our circuit, we place a wire into port 13 (or any of the other numbered ports, but for the example here we're going to use port 13), and the other end of the wire into the red row of our prototyping board. This is the same way we connected the positive terminal of the battery earlier. A second wire connects the GND port of the Arduino to the blue row, where we had previously connected the negative terminal of the battery. These connections can be seen in Figure 1.16.

A discussion about "ground" is warranted here. The name "ground" is quite literal, in a way. The Earth is big, and can act as a giant sink for electric potential. More importantly for us, the ground provides a reference that we can measure electric potentials with respect to. The numbered ports on the Arduino

[1] You will note that the Arduino is assembled on a printed circuit board. Can you identify which components were attached by through hole v. surface mount soldering?

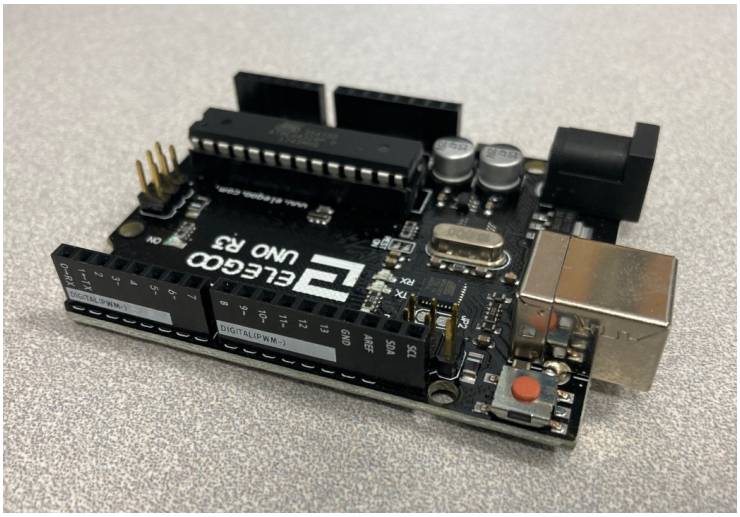


Figure 1.15: The digital ports on an Arduino.

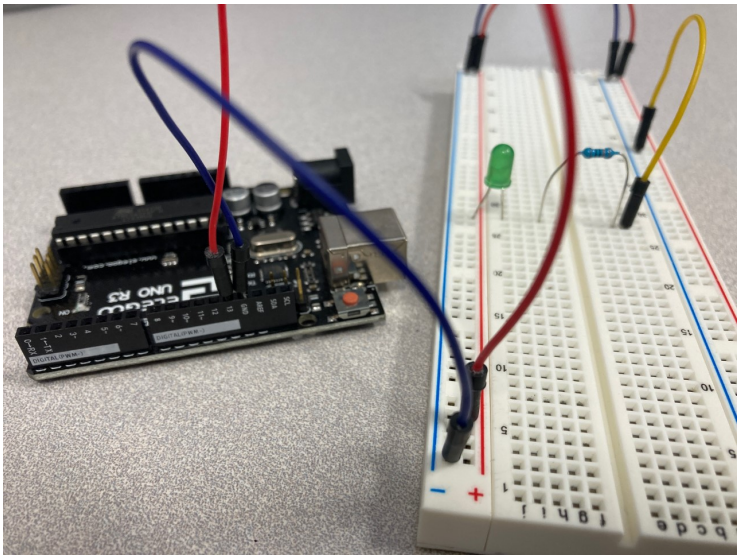


Figure 1.16: The Arduino connected to the LED circuit. The connections are described in the text.

are going to provide five Volts of potential relative to ground. How does the Arduino know what the "ground" is? Because it will also be connected to our computer. The plug on our computer has a "ground" prong (the round prong), which (if the building you are in is wired to applicable codes) is electrically connected to a copper rod that has been pounded down into the ground.

From this point on, we will no longer provide pictures of the circuits that we build (usually). Circuits are typically represented using schematic diagrams instead of pictures. A schematic diagram is a simplified representation of a circuit. Each type of component has a symbol associated with it. Connections between components are represented with lines. Connections to things outside of the circuit are represented by pins. As an example, the circuit we have just built is represented in Figure 1.17. The rectangle labeled with "R1" is our resistor, and you can see the resistance value next to the label. The arrowhead terminating on a line, labeled "D1", represents a diode. The smaller arrows leaving the diode indicate that this is a light emitting diode, or LED (also indicated by the label). The pinouts to the Arduino are labeled "J1" and "J2", and their intended connections are also included on their labels. Part of the fun of building electric circuits is constructing a physical circuit from the information given in the schematic!

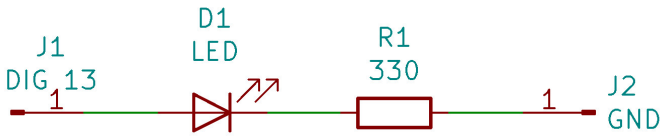


Figure 1.17: Schematic diagram for the blink circuit. The rectangle symbol represents the resistor, and the other symbol represents the LED. The pinouts to the Arduino are labeled at the ends. See the text for a more full description.

At this point, the LED is still out. There are two reasons for this. First, we have not yet provided power to the Arduino. Second, we have not yet told the Arduino to turn on the LED. Providing power to the Arduino can be done in several ways. For today, we will simply connect the Arduino to our computer using a USB cable. (We need this USB connection in order to program the Arduino, also.)

1.2.2 The Arduino sketch: an introduction

We now have our "hardware" built for a blinked LED. We still need to give instructions to our Arduino so it knows how to use pin 13 and GND to make the LED blink. The Arduino has a small computer on board, and we need to upload a computer program to the Arduino in order to tell it what to do.

You have likely already written programs at some point in your education. Common programming languages include Python, Java, and C++. Arduino

programs are similar to other programming languages[2]. They have loops and conditional statements, but additionally have special code for controlling the Arduino.

Most Arduino codes are very short. For today's code, we just need to tell the Arduino what to turn on and when, and when to turn it back off again.

Every Arduino program has two required parts: the `setup()` function and the `loop()` function. When the Arduino is powered on, the `setup()` function is executed once. As the name of the function implies, it is typically used to set up variables or environments that the rest of the program will use. Once the `setup()` function has been executed, the `loop()` function is executed. This is where most of the action happens in the program. This function is repeated over and over again until the Arduino is powered down.

Let's introduce the commands we will use for this lab. Each digital output on the Arduino has a number. The code will need to know which pin number we are working with. There are two ways we can do this. The first is to define a variable of type integer (int) and set it equal to the pin number. The code to do this would typically be written before the `setup()` function, and would look something like this:

```
int ledPin=13;
```

The other way we can do this is with a `#define` statement at the beginning of the code, before the `setup()` function. A `#define` statement simply tells the compiler (the thing that turns our human readable code into binary code that the processor understands) to replace the defined name with whatever follows in the definition. Such a definition would look something like this:

```
#define ledPin 13
```

As stated before, most of the action of the program happens in the `loop()` function. In our case, our loop function needs to turn on the digital pin our LED is connected to, wait for a short while, turn it off again, and then wait some more. A digital pin on the Arduino can be set to a LOW voltage (zero Volts relative to ground) or a HIGH voltage (+5 V relative to ground). Each digital pin that we use also needs to be set up for output or input. This is done with a "pinMode" statement. For example, to set pin 13 to output, I would use the following in my `setup()` function, recalling that we had previously set `ledPin` to a value of 13:

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

If I had wanted to set up the pin for input, I would simply replace the `OUTPUT` with `INPUT`.

To set the pin to LOW or HIGH voltage, use a digitalWrite command:

```
    digitalWrite(ledPin,LOW);
```

[2]The Arduino programming language is essentially C++ with some modification. If you are already familiar with C or C++, Arduino programming will feel quite comfortable.

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

And, finally, to leave the light on (or off) for a while, we can use a delay command, where the argument to the command gives the delay time in milliseconds:

```
    delay(100);
```

Altogether, our Arduino code for blinking lights would read as follows:

```
#define ledPin 13

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin,HIGH);
    delay(100);
    digitalWrite(ledPin,LOW);
    delay(100);
}
```

There are a few things to note about the syntax. First, every line of command within the code is terminated with a semicolon. This tells the compiler that the command is complete, and also allows us to continue a command onto a second or third line when needed. The only exception is the `#define` statement. Second, any grouped set of command, be that in a function, a loop, or a conditional statement (the latter two do not appear in this code) are encapsulated in a set of curly braces. Once again, this tells the compiler where the set of commands begins and ends.

The code above, while it will make the Arduino behave the way we want, is not ideal. It is missing a ***very critical*** part: comments. You should always include abundant comments in your code, including a comment block at the top describing what the code does overall, comments describing each variable you declare or define, and comments that describe each part of the algorithm in the code. Why do you want so many comments? Some people would say that you include comments to help other people understand what your code is doing. The reality is, though, that you put the comments in your code so that ***you*** can remember what your code is doing when you come back to it weeks or months later. For this lab course, comments are required in all of your code.

In the Arduino language, single line comments can be started with a double forward slash "//". Block comments can also be started with a "/*" and ended with a "*/". Once I've included adequate comments, my code will look something like this

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

```
// Arduino sketch to blink one LED
//   Written by (your name would go here)
//   Feb. 6, 2017
////////////////////////////////////////////////////////

// Define the pin we are using for output

#define ledPin 13

////////////////////////////////////////////////////////
// The arduino setup function comes next. We need to set
// up pin 13 for output.
////////////////////////////////////////////////////////

void setup()
{
    pinMode(ledPin, OUTPUT);
}

////////////////////////////////////////////////////////
// Now comes the loop function. We simply turn on the
// LED, wait for a while, turn off the LED, and wait
// some more.
////////////////////////////////////////////////////////

void loop()
{
    digitalWrite(ledPin,HIGH);  // Turn on the LED
    delay(100);                 // Wait 100 ms
    digitalWrite(ledPin,LOW);   // Turn off the LED
    delay(100);                 // Wait 100 ms
}
```

1.2.3 Compiling and uploading the sketch

So that's how you write Arduino code. How do you get it to your Arduino? There is a special integrated development environment (IDE) for programming our Arduino. The IDE can be seen in Figure 1.18.

We will need to install this IDE before we can communicate with our Arduino. To do so we use a web browser to go to

https://www.arduino.cc/en/Guide/HomePage

and choose to install the Arduino IDE for your computer. In the figure, you will see two important buttons on the left side of the toolbar. One is the "compile"

button. The Arduino cannot interpret the human readable code that we write in the IDE. It needs to have the code translated into binary (ones and zeroes) that digital logic circuits understand. This translation process is called "comiling". As the code is compiled, the Arduino software looks for errors in the code. If there are errors, it will tell you at this point. This way we can correct the errors before we send the compiled instructions to the Arduino.

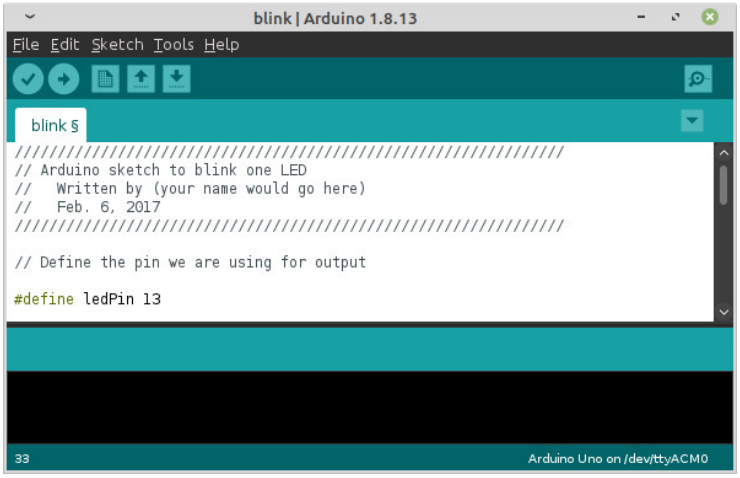


Figure 1.18: The Arduino IDE. The compile and upload buttons are found at the left side of the tool bar.

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. Once the code is uploaded to the Arduino, we should see some blinking lights!

One last thing to note: Arduino programs are called "sketches". As you keep your lab notebook, you should always include a copy of your sketch, along with notes on how you got it to work. It's also a good idea to include pictures and schematics of any circuits you have constructed.

LAB ACTIVITY

Build your own clone of the blinking light circuit described in this section by doing each of the following:

- Construct the LED circuit on a prototyping board.
- Connect the Arduino to the LED circuit and to your computer.
- Write, compile, and upload the "blink" sketch to the Arduino.
- Verify that the circuit is working correctly.

1.3 Adding More Complexity

Now it's your turn to practice these skills by designing a building a new circuit. Our objective is to build an Arduino controlled circuit that does the following:

- Increments a counter every half second.
- Turns on a red LED only when the counter is an even number.
- Turns on a yellow LED only when the counter is a multiple of three.
- Turns on a green LED only when the counter is a multiple of five.
- Turns on a blue LED only when the counter is a multiple of seven.

Most of hardware setup will be the same, but we will be using multiple digital pins on the Arduino (one for each LED). We will have to abandon our nice +5V top row on our prototyping board as our connection to pin 13, because we need multiple pins and we need them to operate independently. Instead, we can wire pin 13 of our Arduino directly to the top lead of one LED, wire pin 12 to the top lead of another LED, and so on. It's OK for each branch of this circuit to share the same ground connection.

To do this, we will need a counter variable. The variable `i` is a common choice. You'll want to declare this variable as an integer at the beginning of your sketch, and set its value equal to zero. At the end of the loop statement, you can increment the counter using

```
    i=i+1;
```

or

```
    i++;
```

You will also need to make the Arduino to some math. Specifically, you'll need to test whether the counter is currently a multiple of the desired number. This can be done using a conditional statement. In the Arduino language, conditional statements will take on the following form:

```
    if(a<b)
    {
        firstFunction();
  delay(100);
  secondFunction();
    }
    else if(a<c)
    {
        anotherFirstFunction();
  delay(200);
    }
    else
```

```
    {
        finalFunction();
    }
```

Each part of the condition includes a test statement found in the parentheses. If the statement is true, the code included in the curly braces that follow will be executed. In this example, if `a>b` evaluates true, then the `firstFunction`, `delay`, and `secondFunction` will be executed in order. It will then skip over the rest of the code in the example. If `a>b` evaluates false, the code will procede to the `else if` portion of the conditional, executing the `anotherFirstFunction` and `delay` functions provided that `a<c` evaluates true. If neither of those test statements evaluate true, then the `finalFunction` function will be evaluated. You will need to use conditional statements in this exercise.

You will also want to use a mathematical operator called the modulus, represented by a "`%`" symbol. The operator divides the first operand by the second and returns the remainder. For example, `23%5` would return `3`. This operator can be used in a conditional statement to see if one number is a multiple of another as in

```
    if(i%3==0)
    {
        // The variable i is a multiple of 3.
    }
    else
    {
        // The variable i is not a multiple of 3.
    }
```

Again, save your sketch and take a photo of your hardware. Place a copy of 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 circuit to work also.

LAB ACTIVITY

Using the Arduino and prototyping board, construct a circuit with all of the following characteristics:

- Increments a counter every half second.
- Turns on a red LED only when the counter is an even number.
- Turns on a yellow LED only when the counter is a multiple of three.
- Turns on a green LED only when the counter is a multiple of five.
- Turns on a blue LED only when the counter is a multiple of seven.

Chapter 2

Electrical Measuring Devices

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Demonstrate the use of a DC power supply and a signal generator.
- Use a multimeter to measure DC voltage and current.
- Demonstrate the basic functionality of an oscilloscope, including channel selection, setting the volts per division, the time per division, and the triggering level.
- Use an oscilloscope to measure DC voltage and AC frequency and voltage.
- Demonstrate how current measurements are derived from voltage measurements by building a simple ammeter device, utilizing a stand alone volt meter.
- In each of these activities, identify and quantify the sources of uncertainty in your measurements.

ITEMS TO REVIEW

Please review the following concepts as part of your preparation for this lab.

- Electric potential and potential difference.
- Electric current.
- Resistance and Ohm's law.
- AC and DC potentials.

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 - building new instruments using instruments we already know and some electrical equipment.

This lab consists of a large pre-reading section that will give you background information. It will describe the various stand-alone devices, along with some electromagnetic theory. Scattered throughout, you will be asked to practice making these measurements with our equipment.

2.1 What We Measure: Voltage

Ultimately, all of our electronic sensors measure one thing: voltage. You may want to measure something else, say relative humidity. To record relative humidity on our computer we need to first (somehow) convert relative humidity into a voltage.

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, as in Figure 2.1.

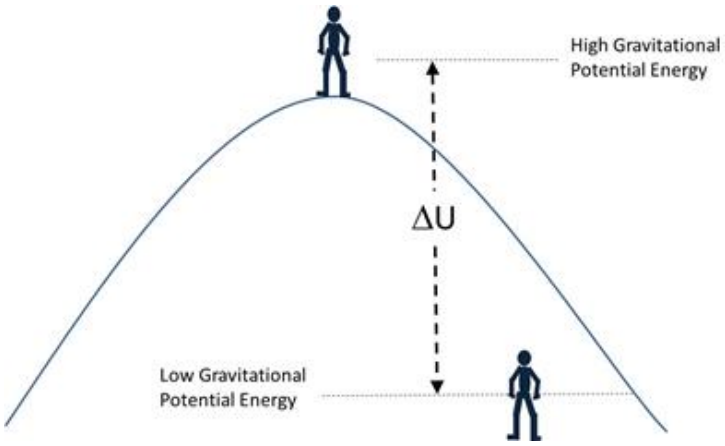


Figure 2.1: Gravitational energy difference. In order to measure the difference in gravitational potential energy from the bottom of the hill to the top, we would need to know the altitude at each position. Note that the difference in the two potential energies is independent of where zero point of the y coordinate is.

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 Figure 2.2. The positive end

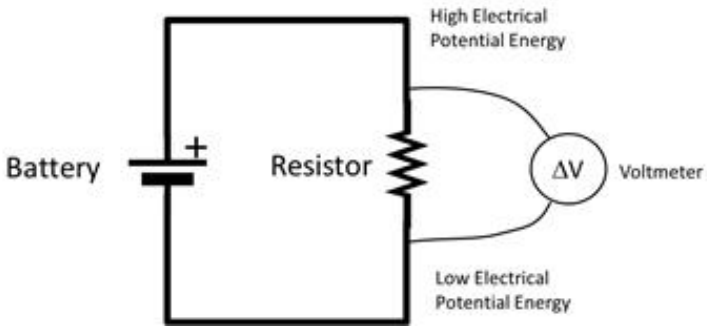


Figure 2.2: Electric potential difference. As with gravitational energy, we measure the potential at two different points and compare the two measurements.

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?" The potential difference is the same, but the probe we thought was at

the high potential is at the low potential, and *vice-versa*. The result is that our reading will change by a negative sign.

In Figure 2.3 you can see how to actually perform this voltage measurement with one of our meters.

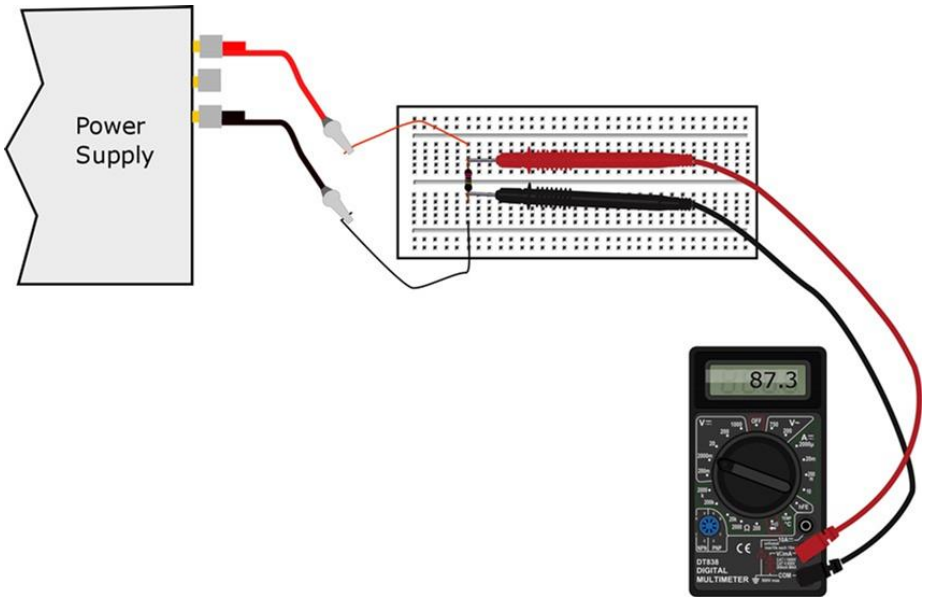


Figure 2.3: Measuring voltage with a multimeter. The red probe is placed on the lead of the resistor connected to the high potential of the power source, and the black probe is placed on the lead connected to the low potential of the power source.

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 difference. Knowing a little about voltage, let's now look at our devices that produce voltages and then the devices that measure voltages.

2.2 Sources of DC and AC potentials

In today's lab we will study four hardware devices. A power supply, a signal generator, a multimeter,and 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 DC Power Supply

When most people think about sources of DC potentials, they will first think about batteries. Of course, we use batteries in many applications in today's world, and they power circuits designed to work on a single, specific DC voltage.

A DC power supply is like an adjustable battery. An example can be seen in Figure 2.4. Usually a power supply takes electrical energy from a wall outlet and converts that energy into the specific voltage range 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. Other devices are typically connected to a power supply using wires with "banana" connectors at the end.

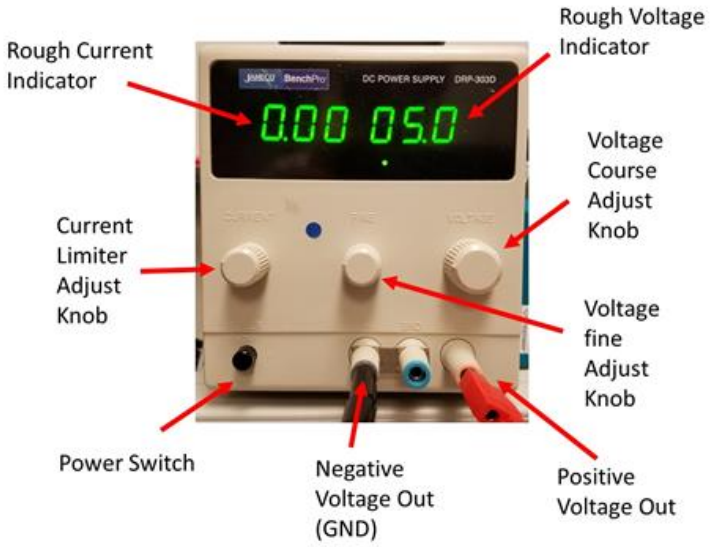


Figure 2.4: A DC power supply.

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 basically a fancy power supply. An example can be seen in Figure 2.5. It makes voltages that change over time 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.

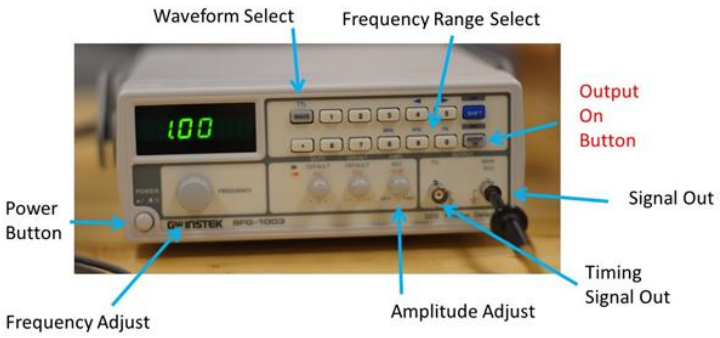


Figure 2.5: A signal generator.

Different brands and types of signal generators are operated differently. We will focus on the one shown in Figure 2.5. 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. Figure 2.6 shows a signal from the signal generator displayed on one of our measuring devices, the oscilloscope.

A household power outlet is also a source of sinusoidally varying voltage, but

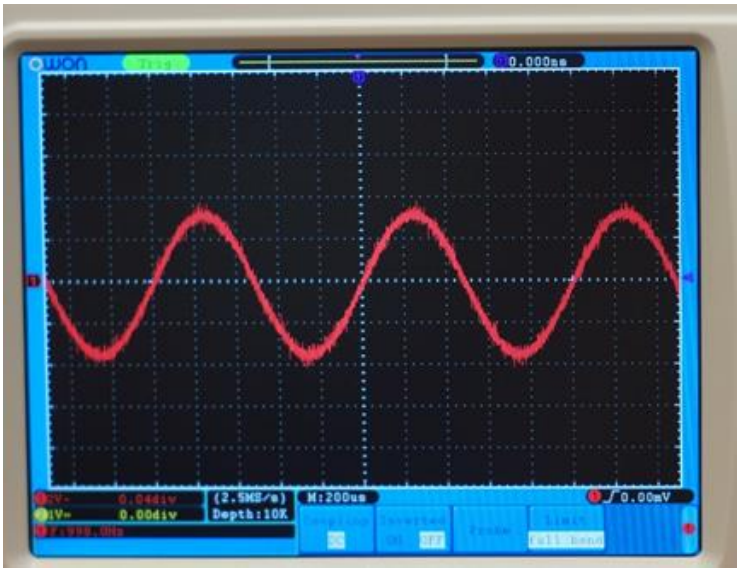


Figure 2.6: Sine wave output from a signal generator, displayed on an oscilloscope.

with two limitations. First, the amplitude of the signal is (hopefully) always at the same set value (170 Volts in the United States), and the sine wave always has the same frequency (60 Hz in the United States). A signal generator provides far more flexibility in both the amplitude and frequency of the signal, as well as the wave form.

2.3 Instruments for Measuring Voltage

Now that we've met a few sources of voltages, lets consider some stand alone devices that can measure those voltages.

2.3.1 Voltmeters

Our first measurement device is voltmeter. It measures the electric potential (voltage) between its two leads (sometimes called "probes"). Figure 2.7 shows an example of a multimeter 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 current (AC) voltage. The one that has a straight line with three dots under it is the direct current (DC) voltage. For now, we will just use the DC 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, causing the meter to stop working altogether

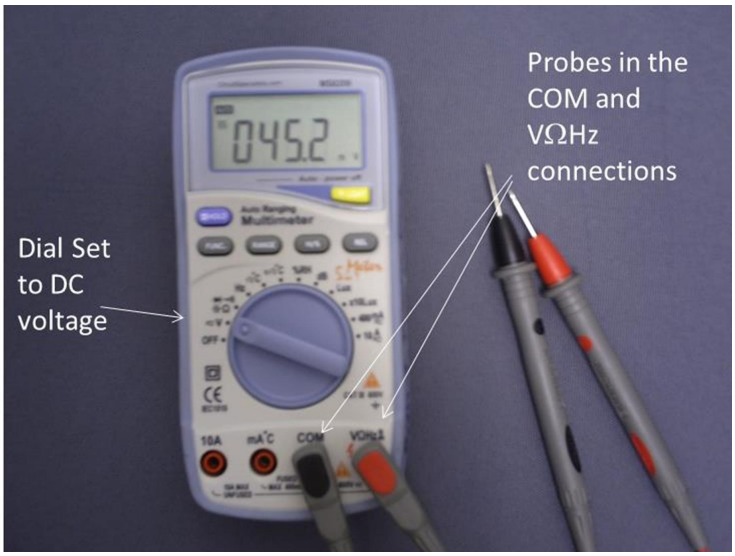


Figure 2.7: A multimeter set to measure DC voltage.

or display incorrect measurements. 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 to be behaving oddly, it may be an indication someone who used it before you connected the probes incorrectly!

For reference, Figure 2.8 shows a picture of a different model of multimeter.

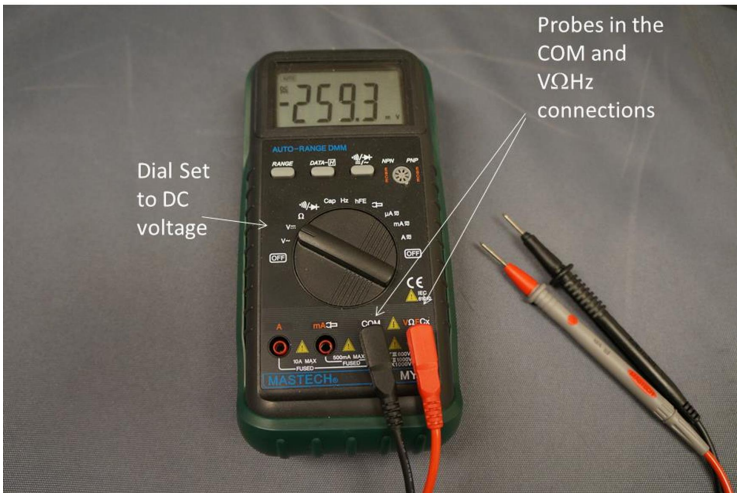


Figure 2.8: A different model of multimeter, also set to measure DC voltage.

You will notice that multimeters have other settings besides volts. They are

designed to be able to measure more than one thing. Ultimately, however, each of these other measurements are in actuality derived from voltage measurements. We will likely use some of the other settings on the multimeter during the semester.

2.3.2 Oscilloscopes

Our next device, the oscilloscope, is just a fancy voltmeter. A voltmeter only displays a single number, giving us no additional details other that what the currently measured voltage is. The oscilloscope measures voltage *as a function of time*, providing us with significantly more information. The downside is that, where a multimeter is pretty simple to set up and use, an oscilloscope is a far more complex peice of equipment.

The oscilloscope can accurately and precisely measure changing voltages and usually has a way to graph the changing voltage as a function of time. A sinusoidally varying voltage should look something like Figure 2.9 when plotted. This plot represents what our oscilloscope does. We should see something like Figure 2.9 on the oscilloscope screen. In figure 2.6, found in our discussion of the signal generator, you know that this is just what we see.

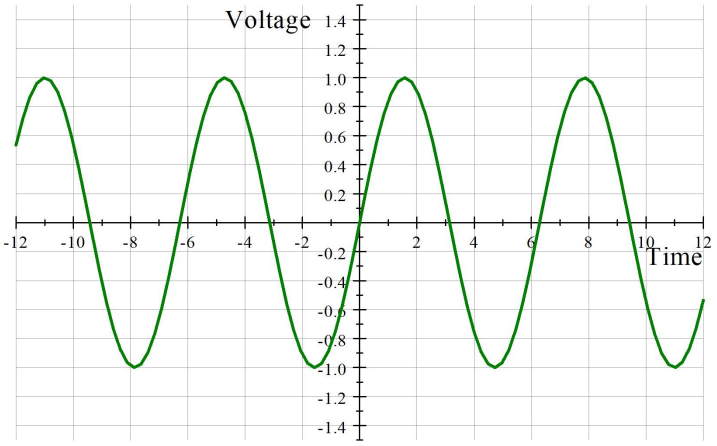


Figure 2.9: Sinusoidally varying voltage. This is the type of information an oscilloscope is designed to display.

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 found in Figure **??** was taken by stabilizing a sine wave from a 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 built-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. This knob can be seen in Figure 2.10.

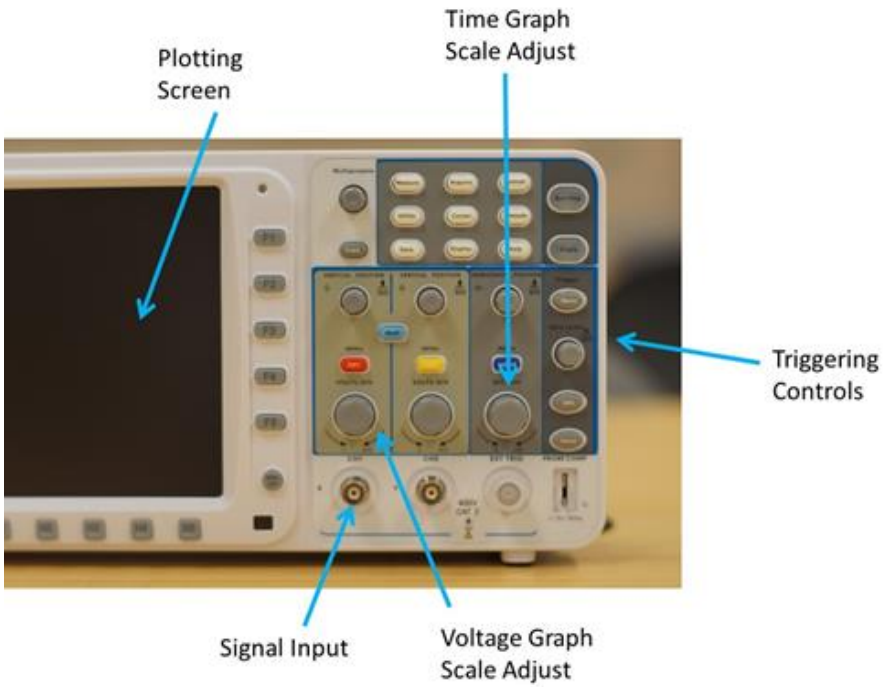


Figure 2.10: Some of the controls on an oscilloscope, including the scale adjustments and triggering controls.

The other labeled controls in Figure 2.10 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 addition to displaying the measured voltage signal, the oscilloscope screen shows us information about the settings of the oscilloscope. In Figure 2.11, at the lower left hand corner, there is a red dot next to a number. This number tells us the voltage represented by each vertical division (square) on the display. When the picture in Figure 2.11 was taken, the voltage knob was set to 2 V per division, so the amplitude of this signal is somewhere around 3 V.

Most oscilloscopes have two signal inputs, meaning we 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

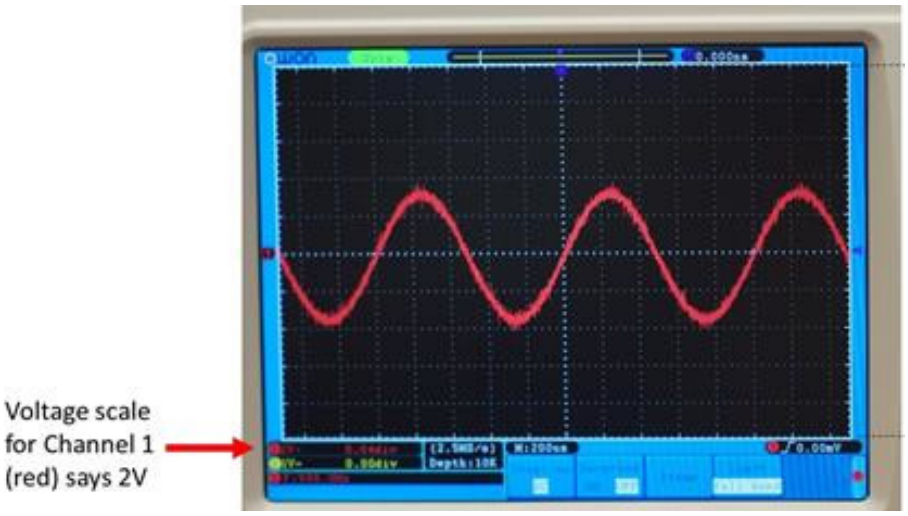


Figure 2.11: The display of an oscilloscope, with the volts per division set to 2 V. The signal seen on the generator has an amplitude of approximately 1.5 divisions, or 3 V.

each have a BNC connector. We use oscilloscope probes connected to these connectors. If we want to measure two voltages at once, we need two sets of probes!

To check that our oscilloscope is working correctly we can measure a known voltage, say, the voltage of a regular battery for example. What would we expect to see? Since the battery provides a constant voltage, we should simply see a straight line on the oscilloscope, with the position of the line corresponding to the voltage of the battery. For example, if I was using a 1.5 V battery, and had the oscilloscope set to 0.5 Volts per division, I would expect to see a straight line at the third division on the display.

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 location of the calibration signal source can be seen in Figure 2.12. The calibration source makes a 5V square wave. By connecting the channel probes to the calibration source, a square wave should be seen on the display (you may have to change the time per division to see the wave form). If the top of the wave crosses at 5V on the display, and the bottom of the wave crosses at the voltage axis, then the oscilloscope is properly calibrated. If the oscilloscope is not properly calibrated, you can find the procedures for doing so in the user manual (search online for the manual of the model you are using).

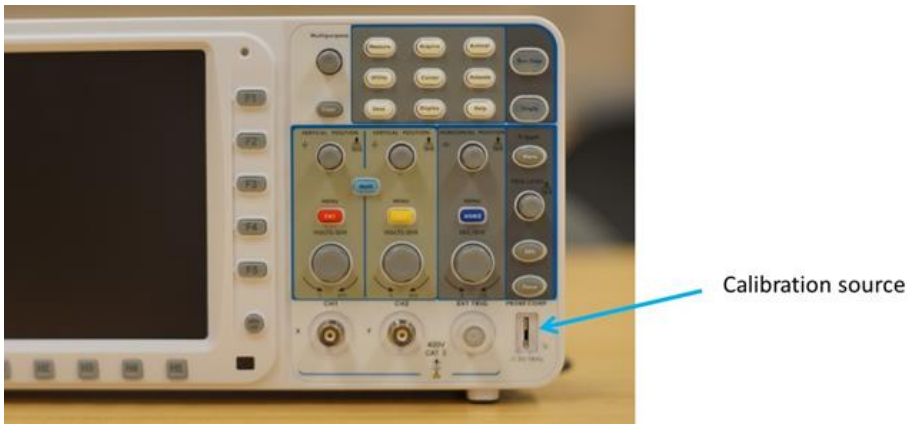


Figure 2.12: Oscilloscope calibration source, built in to the oscilloscope. The source produces a 5V square wave, which can be tested using the channel probes.

2.4 Measurement Uncertainty: A Review

As you are aware, every measurement comes with some uncertainty, and voltage measurements are no exception. The multimeters only display voltages to a specific decimal point, leaving us uncertain of what the next digit would be. Thus, the precision of the last digit of the display is the minimum amount of uncertainty we would have in that measurement.

There are potentially other sources of uncertainty as well. If I was using an analog voltmeter (something we won't do in this lab), I would be making a judgment about which mark on the scale the needle was the closest to. This is akin to using a ruler to measure a distance.

Digital multimeters, such as ours, operate using an analog-to-digital converter (ADC). We'll be learning more about ADCs later in the semester. For now, it is sufficient to note that an ADC introduces some additional uncertainty into measurements obtained using digital voltmeters. (However, the relative size of these ADC uncertainties in a modern digital voltmeter is probably far less than the uncertainty from the limited number of digits on the display).

Note that at this point we haven't even considered the question of whether the meters are reading accurately or not. It is always a good idea to check the calibration of your equipment!

The oscilloscope presents some additional challenges when it comes to uncertainty. It doesn't provide a nice digital readout like the multimeter. When we "read" voltages or times from the oscilloscope, we are making a judgment as to which division the signal is the closest to. Even if I'm pretty good at making judgments, I probably can't depend on my measurement being any more precise than the smallest division mark on the screen.

Keep these uncertainties in mind as you complete the following activities.

LAB ACTIVITY

Practice measuring voltage using a multimeter and an oscilloscope.

- Measure the voltage of a D cell or 9V battery using the multimeter. How certain is this measurement? What measurement do you get if you reverse the probes of the multimeter? (Remember that by "reversing" the probes, it means you touch the ends of the probes to the "wrong" sides of the battery. Do not reverse the ports that the probes are plugged into on the multimeter!)
- Build the circuit from Figure 2.3, and provide a modest amount of voltage to the resistor using the power supply. (If you put too much current through a resistor, it will literally burn up. Keep the current down to a few hundred milliamps at most.) Use the multimeter to measure the voltage across the resistor. Compare this reading to the voltage reading on the power supply. Do they agree? Do they agree within the uncertainties? If they did not agree, which reading would you trust more? Why?
- Measure the voltage of a battery using an oscilloscope. How certain is this measurement? Can you improve the relative precision of the measurement by changing the settings on the oscilloscope? What happens to the measurement if you reverse the probes? Which device is typically better for measuring battery voltage: the multimeter or the oscilloscope? Why?
- Generate a sinusoidal signal using a signal generator. Display the signal on the oscilloscope. Report the amplitude and frequency of the signal as measured by the oscilloscope, along with their uncertainties. Compare these readings to those indicated by the signal generator.

2.5 Measuring Something Else: 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 charge flowing instead of water. We can write the flow of charge as

$$I = \frac{\Delta Q}{\Delta t} \tag{2.1}$$

where for us ΔQ is the amount of charge that has gone by in the time Δt. Physicists normally use the letter I to represent 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 almost instantly through the hose, and 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 behaves in the same way. 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.

Pictured another way, turning on the switch connects the wires to an electric potential. This potential results in an electric field propagating very quickly (like at speeds close to the speed of light) through the wires. As soon as that electric field hits any point of the wire, the free electrons there begin to move.

2.5.1 Current is a Flow

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. 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 break open the circuit and install a meter through which all of the current must flow, as in Figure 2.13. 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 Figure 2.14. 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". 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*. Ammeters measure electrical current. Figure 2.15 shows the probe connections for measuring current with two kinds of multimeters.

Many multimeters have two ports for measuring current, one for milliamps (mA), and one for Amps (A). If you don't know what the current is that you are trying measure, it's always a good idea to start with the probe connected to the port designed to measure Amps. Too high a current will burn a fuse in the multimeter, or worse yet, destroy the multimeter.

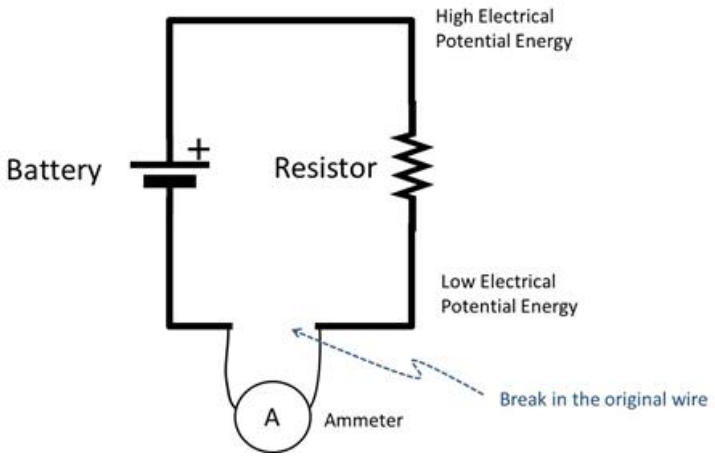


Figure 2.13: Measuring current. The current meter must be positioned such that all of the current flows through the meter.

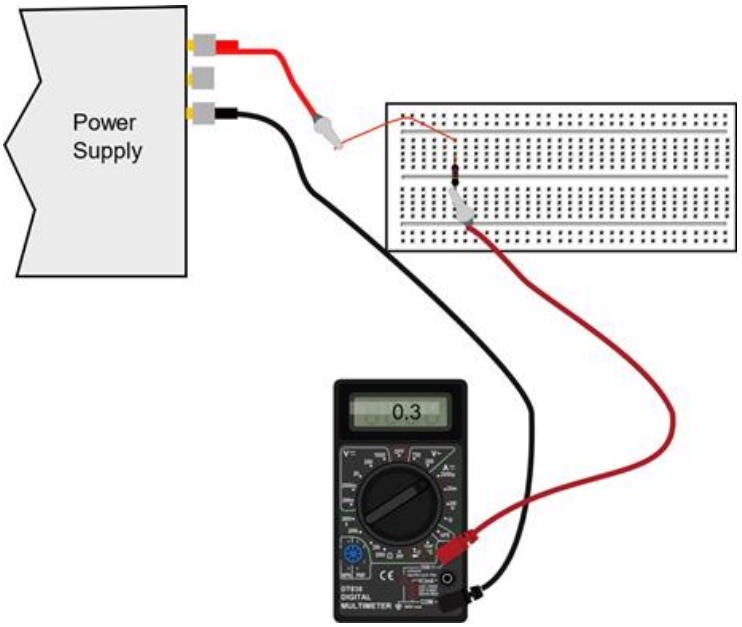


Figure 2.14: Measuring current with a multimeter.

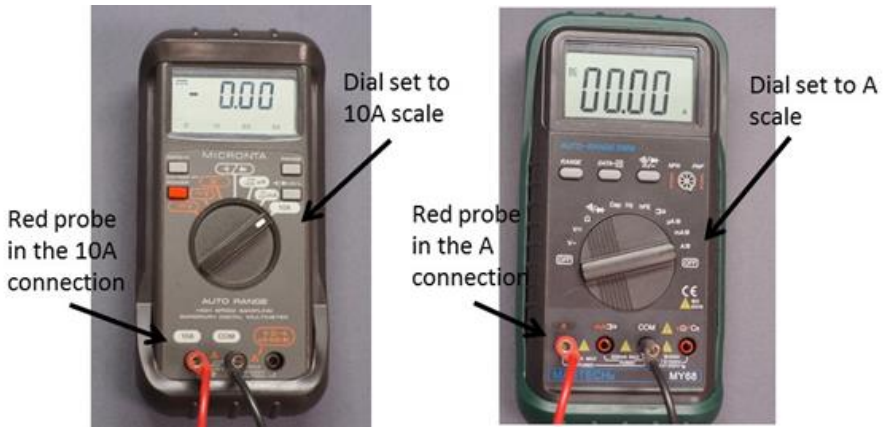


Figure 2.15: Two multimeters prepared for measuring current.

2.5.2 The Physics of Measuring Electric Current

We said before that physicists like to change any measurement they can into a voltage measurement, because, ultimately, that is the only thing that we can measure with electronics. So, if we want to measure current, the first thing we need to do is transform a current into a voltage.

This is true not only for current, but for any other quantity that we would like to measure electroncially. In order to build a new instrument, we need to understand the physics of the thing that we really want to measure.

Ohm's Law

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 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 current interacts with the hose. The more the current interacts with the hose, the more friction it will experience. 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 current a given voltage would produce. 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

$$\Delta V = IR \tag{2.2}$$

an equation referred to as *Ohm's law*. The ΔV is our old friend, voltage, and the I represents current. The R in the equation represents the "friction" provided by the ciruit. Experiments show that this constant R depends 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

$$R = \frac{\Delta V}{I} \tag{2.3}$$

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.

Ohm's law holds well for metals and many materials, but, like Hooke's law, this "law" does not always hold. Devices that do provide a constant resistance coefficient R are called *resistors.* In schematic diagrams, resistors are usually represented by zig-zag lines or by rectangles. In reality, they look more like Figure 2.16.

Figure 2.16: A resistor. The colored bands indicate the resistance.

Resistor Code

Many commercially produced resistors come conveniently marked with a color code that helps you identify their resistance. The basics of the color code are seen in Figure 2.17.

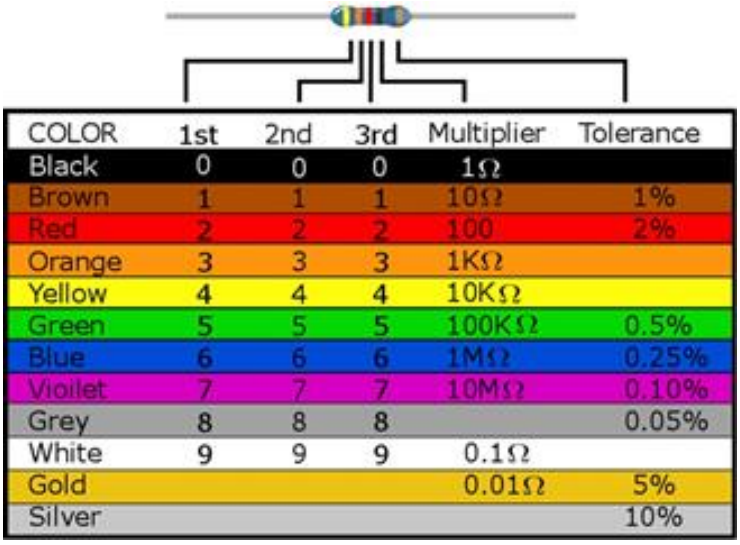
To use the code, do the following:

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



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%

Figure 2.17: Resistor code. The use of this code is described in the text.

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 ±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, as seen in Figure 2.18. 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.

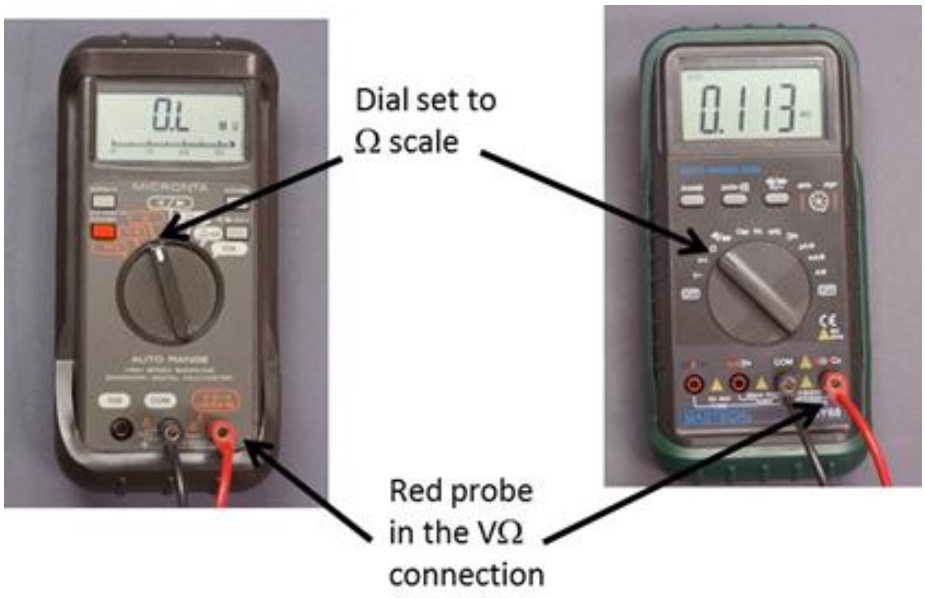


Figure 2.18: Multimeters prepared to measure resistance.

A Note on the Direction of Current Flow

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.

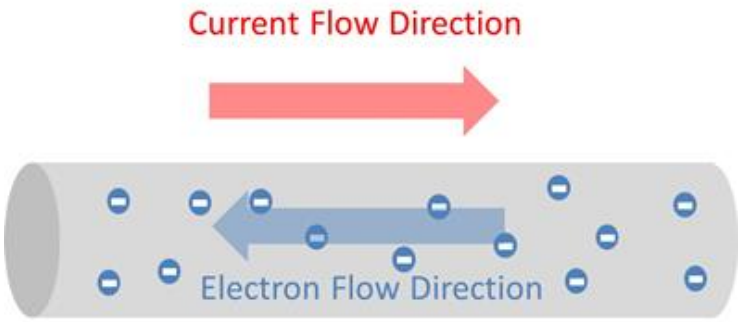


Figure 2.19: Electron flow v. current flow. The usual charge carriers in circuits, the electrons, are negatively charged, and move opposite the direction of the conventional current.

The truth is that it is very hard to tell the difference between positive charge

flow and negative charge flow the other direction. As it turns out, mathematically, the flow of electrons one direction is equivalent to the flow of positive charges the other direction, as seen in Figure 2.20. Why don't we then just redefine current to be the flow of negative charge. For one thing, there are cases where it is positive charges that the charge carriers in a current. In biological systems, it is positive ions that flow, thus the resulting current arises from positive charge carriers. Moreover, in semiconductors (special electronic devices in all computers and in our Arduinos) it is positive charge that flows. In many electrochemical reactions *both* positive and negative charges flow.

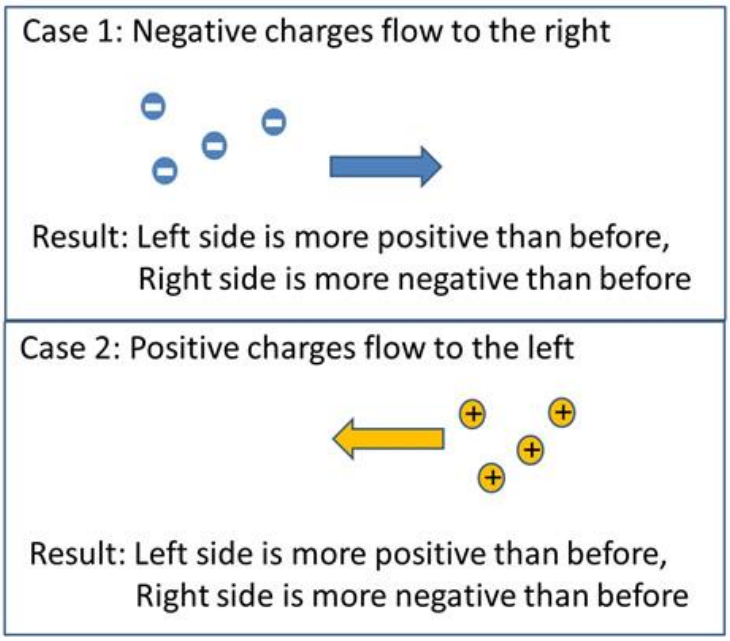


Figure 2.20: Mathematical equivalence of positive and negative charge carriers flowing in opposite directions

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 electricity and magnetism.

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.** This may seem a little backwards at first, but we all get used to it.

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 causes the free charges to experience a force, which subsequently causes their motion. 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, which is just Ohm's law once again.

2.5.3 How to Build an Ammeter

Ohm's law gives us a way to relate current to voltage, provided that we have a resistor of known resistance. Knowing how to measure voltage, we can therefore also measure current.

Consider adding an additional small resistor to our previous circuit (Figure 2.3), as seen in Figure 2.21. If we take a small resistance (one that is small compared to all the other resistances in the circuit) and put it in the circuit, it will slow the current more, 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. Let's give this additional small resistor a name. Let's call it the "shunt resistor."

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.

$$I = \frac{\Delta V_{meter}}{R_{shunt}} \tag{2.4}$$

Today we will do the calculation by hand. In future labs, we will use code to carry out calculations, so it happens automatically.

Remember that when we try to make any sort of measurement using electronics, the first step is to convert the thing we are trying to measure into a voltage, as we have done here.

2.5.4 Testing the New Instrument

Any time we build a new instrument, we need to verify that it is working correctly. In this case, we are fortunate that our multimeters can also measure current. By comparing the current we measure with our new instrument to the current indicated by the multimeter, we can validate our new instrument.

Remember that in order for a multimeter to measure current, the current must flow through it. We have to break the circuit somewhere and place the multimeter into the flow, as done in Figures 2.13, 2.14, and 2.21.

Also remember that when you are using a multimeter to measure current, you want to start by assuming that the current will be high. Turn the dial of the multimeter to the highest current setting, and make sure that the red probe is connected to the right port on the multimeter. If the multimeter reads zero

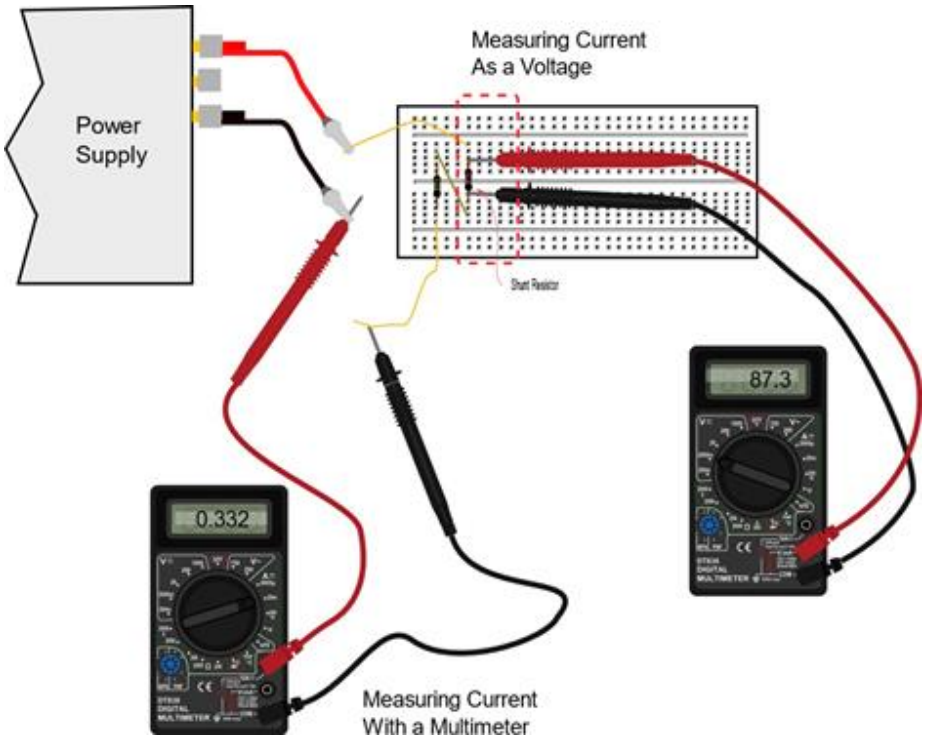


Figure 2.21: Building an ammeter. The process includes adding a shunt resistor to the ciruit and measuring the voltage across the shunt resistor. When this measured voltage is known along with the shunt resistance, current can be calculated. The portion of this circuit constituting the ammeter is enclosed in the dotted red line. In this circuit, an additional multimeter, measuring current directly, is included.

current, it could mean (a) that the fuse in the multimeter is blown (you can check this by trying to measure a known current, say from a DC power source that displays the current as well), or (b) the current is smaller than what the multimeter can read on the current setting. In the latter case, you will need to change the dial on the multimeter to a lower current setting and reconnect the red probe to a different port.

As it turns out, when a multimeter measures current, it uses the same method we have been discussing. The multimeter includes a few networks of small, known resistances. The current is sent through the appropriate resistor network, and the voltage across that network is measured. The multimeter then calculates the current and displays it.

After we calculate the current from our known shunt resistance and measured voltage across the shunt resistor, we can compare it to the current displayed by the multimeter. It the two numbers agree, we can be confident that our newly built ammeter is working properly.

However, there is a complication. Recall that every measurement carries an amount of uncertainty, and current measurements are no different. It's pretty easy for us to judge the amount of uncertainty that the multimeter reading has, but our calculated value for current has uncertainty coming from two sources: the resistance of the shunt resistor and the measured voltage across the shunt resistor. Now is probably a good time to review how measurement uncertainty propagates into our calculations.

2.6 Propagation of Uncertainty: A Review

Every measurement has an uncertainty, so any quantity calculated from a measurement has an uncertainty also. If there are multiple measurements going into a calculation, we have to account for multiple uncertainties. This is most easily done using derivatives.

The derivative of a function at a particular point tells us how quickly the function is changing at that point. Imagine that we have some function $y(x)$. The derivative of the function with respect to time, dy/dx, at a particular position x, can be represented by the tangent line that goes through the point $y(x)$, as seen in Figure 2.22.

Now suppose there is an uncertainty δx in the position x. Because of that uncertainty in x, I would also have some uncertainty in y. I have two ways I could determine the range of possible values that $y(x)$ could take on:

1. I could simply calculate the value of $y(x)$ at the extreme possible values of x, i.e. calculate $y(x-\delta x)$ and $y(x+\delta x)$. These two values are represented by the red dots in Figure 2.22. As long as $y(x)$ is monotonically increasing or decreasing in the range $x \pm \delta x$, then these two values of will represent the highest and lowest possible values of $y(x)$ for the range of possible values in x.

2. As long as δx is small, I could approximate the function $y(x)$ as a line. The

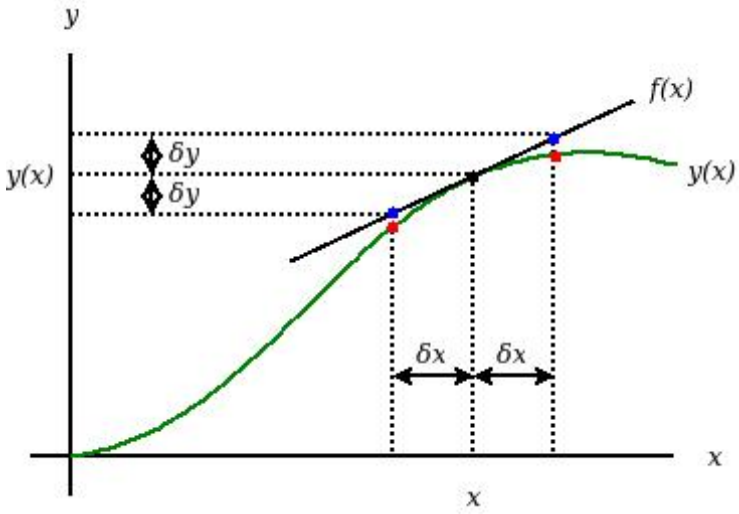


Figure 2.22: Using derivatives to approximate uncertainties. The black dot represents the value of the function at a specific point. The red dots represent the values of the function at two other points bracketing the uncertainty in the independent variable. The blue points show the same two values on the tangent line.

values of the function at the extreme ends of our range of x are represented by the blue dots in Figure 2.22. The equation of this tangent line is

$$f(x) = mx + b \tag{2.5}$$

where the slope m is the derivative of $y(x)$ at x, i.e.

$$f(x) = \frac{dy}{dx}x + b \tag{2.6}$$

It is understood, in this equation and those that follow, that the derivative dy/dx is evaluated at our particular x. The intercept b is unknown at this point, but it turns out we won't need it anyway. If I now wanted to estimate how much $y(x)$ changes as I go from x to $x + \delta x$, I could make use of this line:

$$\begin{aligned} y(x+\delta x) - y(x) & \\ &\approx f(x+\delta x) - f(x) \\ &= \left[\frac{dy}{dx}(x+\delta x) + b\right] - \left[\frac{dy}{dx}x + b\right] \\ &= \frac{dy}{dx}x + \frac{dy}{dx}\delta x + b - \frac{dy}{dx}x - b \end{aligned}$$

In the last line of this equation, the two $\frac{dy}{dx}x$ cancel, as do the two unknown intercepts b, leaving us with

$$y(x+\delta x) - y(x) \approx \frac{dy}{dx}\delta x$$

or simply

$$y(x+\delta x) \approx y(x) + \frac{dy}{dx}\delta x \tag{2.7}$$

Similarly, if we considered the value of $y(x)$ at $x - \delta x$, we would obtain

$$y(x-\delta x) \approx y(x) - \frac{dy}{dx}\delta x \tag{2.8}$$

We can see that in this approximation, an uncertainty δx in the variable x yields a consistent variation in $y(x)$). Consequently, we write the uncertainty in $y(x)$ as

$$\delta y \approx \frac{dy}{dx}\delta x \tag{2.9}$$

Since our measurement uncertainties will almost always be relatively small, we will be making use of the second method almost exclusively.

What if our calculated value depends on more than one measurement? In other words, what if instead of an $y(x)$ I have an $y(x,z)$, where x and z both represent measured quantities with uncertainties δx and δz? We can easily find the uncertainty that arises in $y(x,z)$ from each of the two measurements independently by using partial derivatives. We would obtain

$$\delta y_x \approx \frac{\partial y}{\partial x}\delta x \tag{2.10}$$

$$\delta y_z \approx \frac{\partial y}{\partial z}\delta z \tag{2.11}$$

The question now remains, how to we "add" these two uncertainties together?

If x and z are independent variables, then we would end up adding the two uncertainties in quadrature (which is a fancy way of saying we'll use the Pythagorean theorem). The reason for this can be seen in Figure 2.23. If I had a variation in both x and z, the total variation would be represented by a diagonal line, and the three lines together would form a right triangle. The result is that our total uncertainty in $y(x,z)$ becomes

$$\delta y = \sqrt{\delta y_x^2 + \delta y_z^2} = \sqrt{\left(\frac{\partial y}{\partial x}\delta x\right)^2 + \left(\frac{\partial y}{\partial z}\delta z\right)^2} \tag{2.12}$$

If our calculated value had more than two measurements associated with it, we would just iterate this process a few more times. the result would be

$$\delta y = \sqrt{\sum_i \left(\frac{\partial y}{\partial x_i}\delta x_i\right)^2} \tag{2.13}$$

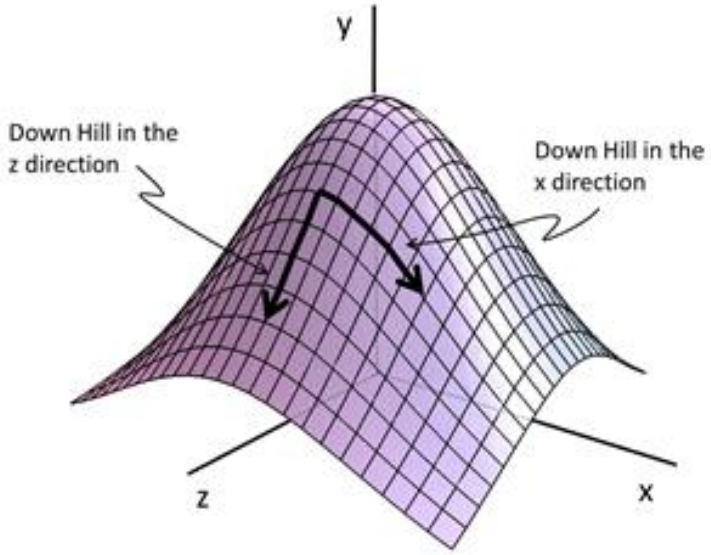


Figure 2.23: Variation in a function of two variables

where the x_i and δx_i would represent the measurements and their respective uncertainties.

Tie to statistics

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 $y(x, z)$ the error is given by

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

If instead of uncertainties in x and z, what we had were standard deviations σ_x and σ_z, we would find the standard deviation in y as

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

When measured values are subject to fluctuation, 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.

2.7 Calculating the Uncertainty in Current

Getting back to our home brewed ammeter now, we were calculating the current from a known resistance and measured voltage, via the equation

$$I = \frac{V_m}{R_s}$$

(the m subscript indicates "measured" and the s subscript indicates "shunt"). The resistors we are using typically have a tolerance of about 5%, meaning that the relative uncertainty in the resistance (as read from the colored bands on the resistor) is 5%, or $\delta R_s = 0.05 R_s$. The uncertainty in our voltage reading is (probably) just the precision of the multimeter display for that measurement.

To find the uncertainty in the current, I would need to know the partial derivatives

$$\begin{aligned} \frac{\partial I}{\partial V_m} &= \frac{1}{R_s} \\ \frac{\partial I}{\partial R_s} &= -\frac{V_m}{R_s^2} \end{aligned}$$

and the uncertainty in the current would therefore be

$$\delta I = \sqrt{\left(\frac{1}{R_s}\delta V_m\right)^2 + \left(\frac{V_m}{R_s^2}\delta R_s\right)^2} \tag{2.14}$$

where we've dropped the negative sign for the derivative with respect to R_s, since it will go away when squaring that term anyway.

LAB ACTIVITY

Build an ammeter from a shunt resistor and voltmeter.

- Choose a resistor in the 10-50 kΩ range. Use this as the resistor for the primary part of the circuit.
- Choose a shunt resistor with maybe a few hundred Ohms of resistance. Use the colored bands to determine the resistance and tolerance of this resistor.
- Set up the circuit in Figure 2.21.
- Turn on the DC power supply to provide voltage. Record the voltage drop across the shunt resistor along with its uncertainty, as well as the current read by the second multimeter and its uncertainty.
- Calculate the current, and its associated uncertainty, from your shunt resistance and measured voltage.
- Compare your calculated current to the reading on the second multimeter. How well did your ammeter work?

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Describe the function of an analog to digital converter, and identify the limitations of analog to digital conversion.
- Calculate the uncertainty in a voltage measurement arising from analog to digital conversion.
- Identify the voltage limits of the Arduino, and how one can avoid exceeding those limits.
- Build a voltmeter using the ADC on an Arduino that can measure 0-5 Volts.
- Build a voltmeter using the ADC on an Arduino that can measure 0-30 Volts.

ITEMS TO REVIEW

Please review the following concepts as part of your preparation for this lab.

- Electric potential and voltage.

Chapter 3

First DAQ Measurements: Voltage

Let's review what we learned last week. When we are making measurements electronically, we are ultimately measureing voltages. If the data is not a voltage, we must convert it into a voltage. We have already converted current into a voltage (using a shunt resistor).

What is a voltage? Let's review quickly. In the last lab we said that voltage is a measure of electrical potential energy. It is also likely that you are familiar with 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.

For today, the key things for us to remember are (1) that voltage is proportional to an electrical potential energy difference, and (2) that when we measure voltage we are measuring something proportional to energy.

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 Guy jumps from the bottom of a building to the top we need two measurements, one at the bottom and one at the top (Figure 3.1). The potential energy difference is then found by taking the difference in energies at the top and bottom of the building:

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

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 Figure 3.2. The circuit is very simple, just a battery and a resistor. A battery is just a source of electric potential energy, and a resistor is just a piece of material that has lots of electrical friction, or "resistance" that

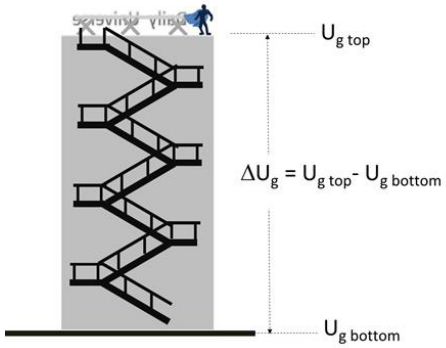


Figure 3.1: Gravitation potential energy difference is found by taking the difference of potential energies between the initial and final position.

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.

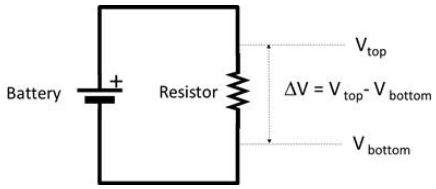


Figure 3.2: Measuring electric potential energy difference, or voltage. Two measurements are required, one where the potential is high, and one where the potential is low. The voltage is the difference between these two measurements.

As we saw in the last lab, most meters that measure voltage ("voltmeters") have two "probes". The meter reads a potential from each probe and completes the difference calculation internally. These meters are called *voltmeters* and we used them last week. In today's world, voltmeters are usually just one function provided by multimeters.

We learned to use a stand-alone voltmeter in the last lab. Now we also need to read in voltages in a way that the data can be transferred automatically to a computer. To do this, we will use the *analog* pins on our Arduino board.

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 (seen in Figure 3.3) will appear in the text, just to remind you to be careful! You may see quite a few of these in this lab.

Figure 3.3: An example warning sign. Wiring an Arduino incorrectly can destroy parts of the Arduino. When this is a possibility, you will see this warning sign.

3.1 Building a Voltmeter

To act like a voltmeter that communicates with a computer, the Arduino needs to do three things:

1. Read the voltage signal.
2. Convert the voltage signal to a number.
3. Send the data to the computer.

Let's look at these in greater detail.

3.1.1 Analog Pins on the Arduino

Analog signals are read at the analog pins on the Arduino. These can be seen in Figure 3.4, and are labeled A0 through A5. You will also note that there are two ground (GND) connections on this side of the Arduino. When measuring voltages with the Arduino, the wires that we use as our probes will go into one of the A0 to A5 pins and into one of the GND pins, or into two of the A0 to A5 pins.

3.1.2 Analog to Digital Conversion

Our Arduino has a device on board called an 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 that range. We represent an analog value with a real number. We might have a voltage of exactly

$$4.3276854325532573457\text{V}$$

and this would be perfectly valid for an analog signal.

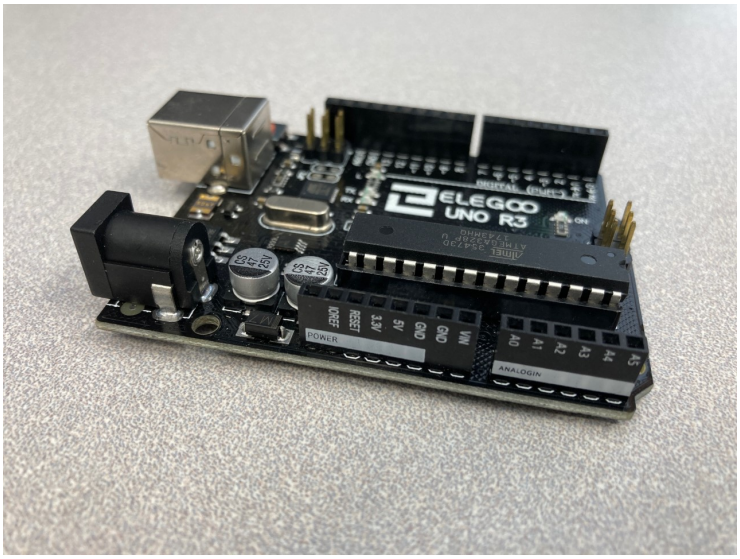


Figure 3.4: The analog pins on the Arduino, seen on the right side and labeled A0 through A5.

A battery, on the other hand, does not work this way. It has a fixed voltage. The common AA and AAA batteries have a fixed voltage of 1.5V. Two AA batteries could be used together to make 3V. But you can't use AA 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. Anything more than 5V will more than likely destroy part (or all) of your Arduino, so we have to be careful! But there is more to the ADC than just a voltage range. This is necessary if we are to store the voltage value digitally.

Digital devices, like computers, work with numbers in binary. As you may know, binary is a number system where each digit of a number is either a one or a zero. Binary numbers are easy for computers to work with, since computers only store information in "bits" that are either on or off. The more bits you have, the larger the number you can represent.

The ADC on the Arduino has a 10 bit register for storing data. The largest number that can be represented by ten binary digits is 1023 (all ten digits are ones), and the lowest number that can be represented by ten binary digits is zero (all ten digits are zeroes). Thus, the Arduino must take the 0-5 Volt range as chop it into 1024 voltage divisions. Each division is then

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

These divisions are represented in Figure 3.5.

Changes in voltage that are less than 4.9mV cannot be measured by an Arduino, since it takes a whole 4.9mV to get a different division. So if we give

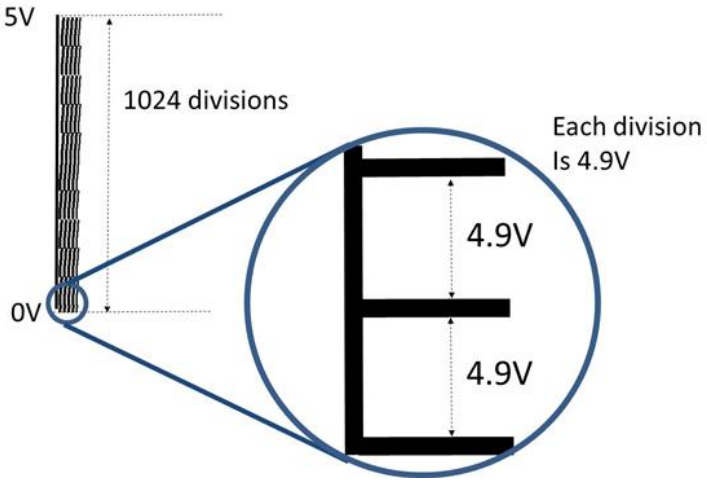


Figure 3.5: Discretization of voltage measurements on an Arduino. The Arduino can measure a range of 0-5 Volts, but the digital representation of that measurement is stored in a limited number of bits. As a result, the continuous 5 Volt range is chopped up into smaller divisions.

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 $n \times 4.9\text{mV}$.

A helpful analogy is to think about a ramp next to a set of stairs. The ramp is like the analog signal. You can be at any elevation between the bottom and the top of the ramp. The stairs are like the digital representation of that signal. There are only discrete elevations that you can obtain. When we convert an analog signal to a digital signal, we basically just report whatever "step" is closest to the analog value.

As a second example, consider using our Arduino to measure a voltage of 3.793V. We can only measure in steps of 4.9 mV, and if we divide our voltage by that minimum step size, we get

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

The Arduino has to record this ratio as an integer value, so the 0.08 would be dropped, and the ADC would simply report that the measurement is on step 774, which corresponds to $774 \times 4.9\text{mV} = 3.7926\text{V}$. Similarly, let's consider the 4.327685432553257 3457V from our hypothetical power supply. Following the same method, the Arduino finds that this is closest to "step" 883 in the ADC, which corresponds to 4.3267 Volts. (Make sure you can see how this result is

obtained!)

All of this means that, when we measure voltages with our Arduino, 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

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

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

One way to reduce the amount of quantization error in a measurement like this is to use a data register with more bits. A ten bit register has 1024 (2^{10}) discrete values. A twelve bit register would have $2^{12} = 4096$ discrete values. Modern computer chips work with 64 bit registers, which have $2^{64} = 1.845 \times 10^{19}$ discrete values!

Of course, the only way we could increase the number of bits available on the Arduino's ADC would be to start replacing physical elements on the Arduino board. We're not going to attempt that, so we'll just have to live with our 4.9 mV quantization uncertainty.

It's important to note at this point that the ADC does not record the voltage. Rather, it records which "step" the voltage corresponds to. This integer value is sometimes referred to as an ADC unit. If we have a signal voltage of 9.8mV, the ADC doesn't provide us with a "9.8", but instead it provides a "2" because $9.8\text{mV} = 2\Delta V_{\min}$. The ADC units are the number of $\Delta V_{\min}$ sized units that are in our signal voltage. To get back to voltage units, we need to multiply the ADC units by $\Delta V_{\min}$. In our code we will do this before sending the value to the computer.

3.1.3 Sending the Data to the Computer

The Arduino reads voltage in discrete steps, but can also do further processing with the ADC values.

Of course, we would like to record the voltage that we measure rather than the ADC value. There is a simple way to do this in the Arduino sketch, and that is simply multiplying the ADC value by the $\Delta V_{\min}$. 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's introduce the new commands that we will need. To get the Arduino to communicate with the computer we use the command `Serial.begin();` in the setup portion, 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 data type for real numbers is the word "float." For example, `float delta_v_min=0.0049;` defines a real number variable named "delta_v_min" and sets it to the value 0.0049. If we want an integer number we use the data type "int." For example `int value = 0;` defines an integer variable named "value" and sets it equal to 0.

If you are familiar with Python, you know that creating a list of variables near the beginning of your code is a good idea, not only so you can be sure that all of your variables are properly initialized, but because it also makes it a lot easier to decipher your code later on. With the Arduino (as with C or C++), if you don't declare your variables up front, the code will not compile, and the Arduino software will spit out an error message.

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:

```
////////////////////////////////////////////////////////
// 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. Remember that 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. It's a good habit to include 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.

3.1.4 Wiring the simple voltmeter

Now we're ready to build our 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 will be damaged and will never work the same way again (probably won't work at all!). So wire first, then before you connect the Arduino have a group member check your wiring, then check the wiring with a stand-alone meter (that is why we learned to use them last time). Also remember, signals of more than 5V (or less than 0V) 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!

3.1.5 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 (Figure 3.6).

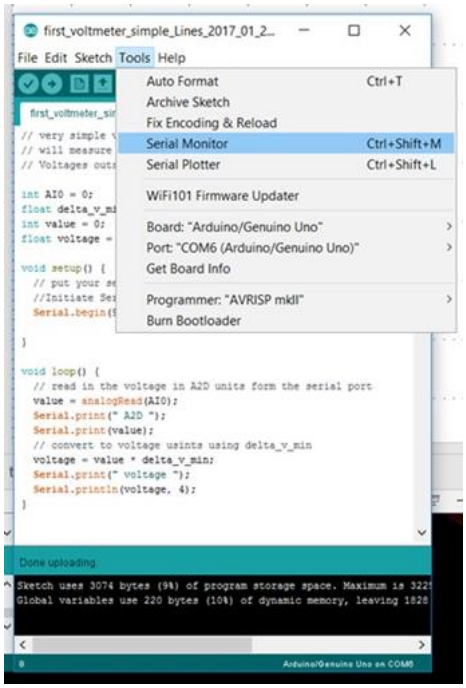


Figure 3.6: Serial Monitor location in the Arduino IDE.

When activated, the serial monitor will look something like Figure 3.7

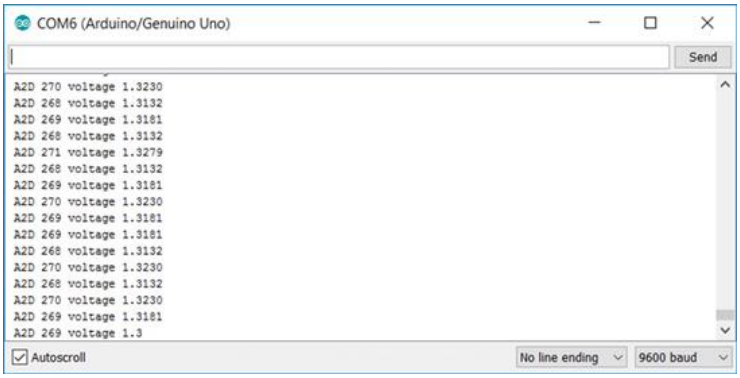


Figure 3.7: The Serial Monitor window.

The Arduino Software can also plot the data from the serial port. This is accessed just below the serial monitor in the Arduino IDE menu. Figure 3.8 shows a plot of the same data that we saw in Figure 3.7. In this image, we are plotting both our voltage values and our ADC values. This makes the voltage values hard to see (since they are very small, numerically, in comparison with the ADC values). We could fix this by commenting out the lines that print the ADC values (putting "//" at the beginning of the line), so that those lines won't be executed by the Arduino. Then we get just the voltage (Figure 3.9. Notice that the horizontal axis does not represent the exact time. It is just a data point number. We could convert this to time with some calculation if we know how often the Arduino sends us a data point. This will be left as an exercise.

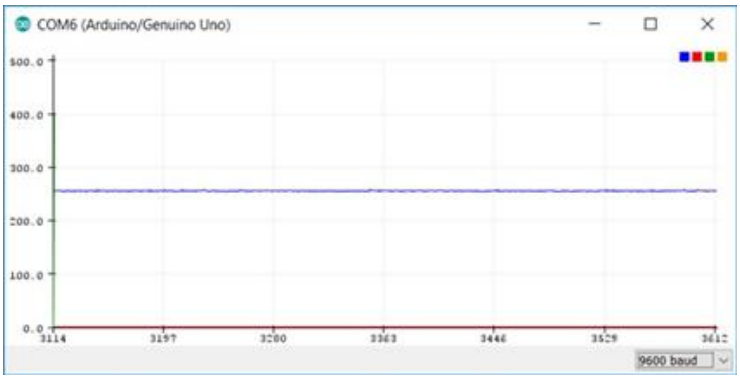


Figure 3.8: The Serial Plotter window.

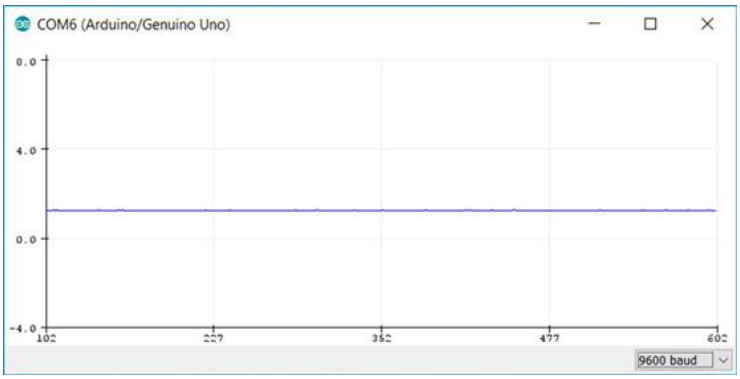


Figure 3.9: The serial plotter with voltage only.

The Arduino is now measuring the voltage (or rather the voltage ADC value), converting it to usable data, and sending it to our computer! Eventually we will want to be able to save this data to a file, so it can be used later. That will be a topic for the next lab period.

LAB ACTIVITY

Build and use a simple Arduino voltmeter.

1. Build a circuit with a power supply and a resistor as in Figure 3.2. You can choose any resistor.
2. Write the simple voltmeter sketch.

3. Wire the Arduino across the resistor, and measure the voltage across the resistor using the Arduino and the serial monitor. Be careful to stay in the 0 to 5V range!
4. Calculate the uncertainty due to quantization error for your Arduino simple voltmeter
5. 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?)

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 the circuit in Figure 3.2, which consisted of a battery and a single resistor. We have a battery, which will make the current flow much like a pump makes water move through pipes.

The water in a pipe system (Figure 3.10) 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 Figure 3.10). 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

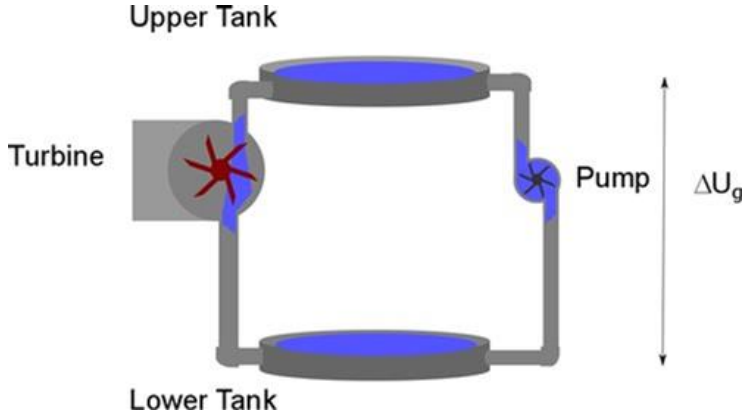


Figure 3.10: Water pump analogy for a simple resistor circuit. The pump and the battery play the same role.

to the battery.

Now suppose we have two resistors in a circuit, as in Figure 3.11.

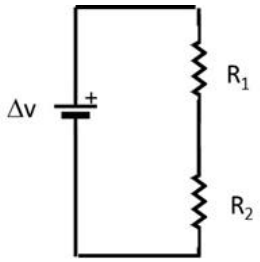


Figure 3.11: A two-resistor circuit. The two resistors in series are also called a voltage divider.

Our water analogy can still help us understand what will happen. Suppose that we have two turbines in our pipe system, as in Figure 3.12 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 lose 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

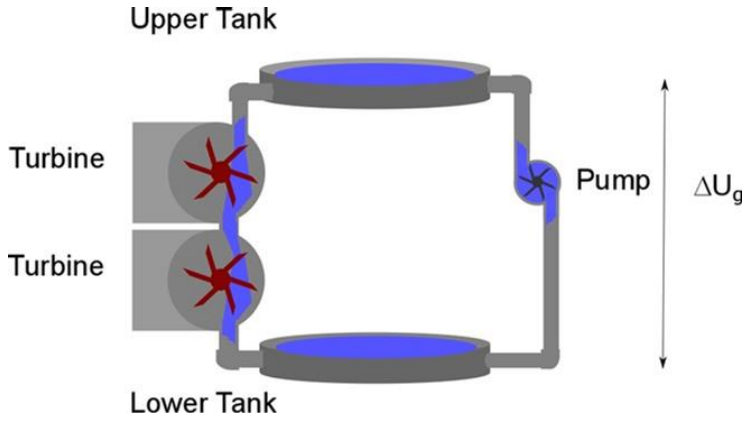


Figure 3.12: A water pump system with two turbines, analogous to a two resistor circuit.

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

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

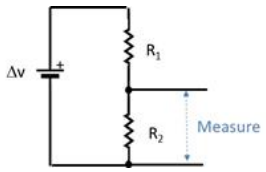


Figure 3.13: Measuring voltage in a voltage divider.

But suppose we measure the potential change across just resistor R_2, as in Figure 3.13. 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 lose all the ΔV_{total} given to the current by the battery. And

$$\Delta V_2 = IR_2$$

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. 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 $\Delta V_{total} = 20$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 $\Delta V_{total} = 0$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 too close to 5V we are safer. So this set of resistors 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

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

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

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

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.

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

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 $\left(\frac{1}{R_2}(R_1 + R_2)\right)$ 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

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

The first term in parenthesis is essentially 1 digitizer unit multiplied by ΔV_2 and the second term in parenthesis converts the ΔV_2 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. 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

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

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**.

```
////////////////////////////////////////////////////////
// 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?

LAB ACTIVITY

Build an extended voltmeter using a voltage divider.

1. 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.

2. 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 (or whatever the maximum of the power supply is) 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!

3. 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.

4. What is the quantization error for this voltmeter? Check to see if this matches your values on the serial monitor. (The values on the serial monitor should "jump" in steps with a size equal to the quantization error).

Chapter 4

Getting Data to the Computer

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Have a working Python installation on your computer.
- Use Python to retrieve Arduino data from the serial port of a computer.

ITEMS TO REVIEW

Please review the following concepts as part of your preparation for this lab.

- Basic Python syntax.

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 SPSS or Python?

There are lots of ways to do this, but an easy way is to use Python to 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

Python is a programming language. The code that we write in Python 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.

There are many different distributions of Python, and any distribution will work just fine. In the instructions that follow, we are going to focus on the Anaconda distribution.

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].

When you click on a link, it will likely bring up a dialog box will come up telling you that you are downloading something. In windows, it looks like Figure 4.1

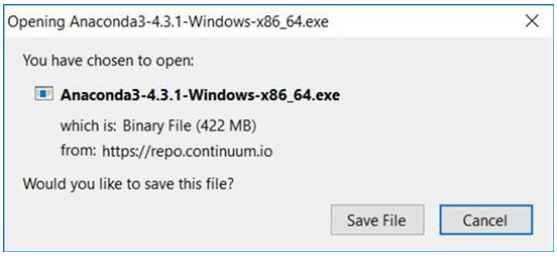


Figure 4.1: Anaconda download confirmation.

Choose "Save File" and when the file is downloaded open it to start the installation. You should see something like Figure 4.2.

Choose "Next" and follow the installation instructions. If all goes well, you should have a new set of apps. Figure 4.3 shows what these apps look like in the start menu of Windows 10.

The list of the apps has an app called Spyder. This is the integrated development environment (IDE) that comes packaged with Anaconda, and is where we will do most of our work. The interface for Spyder can be seen in Figure 4.4

[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, you have no idea what that means and can ignore it.

Figure 4.2: The Anaconda installer welcome screen.

Figure 4.3: Anaconda apps on the Windows 10 start menu.

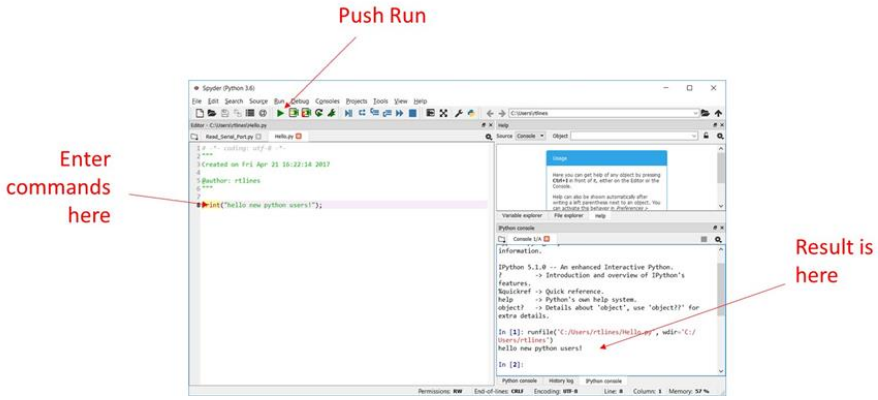


Figure 4.4: The Spyder IDE. Python code is written in the frame on the left. The code is executed by clicking on the "Run" button, and the resulting output appears on the frame at the lower right.

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 Figure 4.4, 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.

4.1.2 Getting the PySerial library for Anaconda

Now that we have Python, we need to update 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 are using a different Python distribution, ask your instructor for help.

We will use the Anaconda prompt to get the PySerial library. The Anaconda prompt is an app that lets 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, as can be seen in Figure 4.3.

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`, as seen in Figure 4.5.

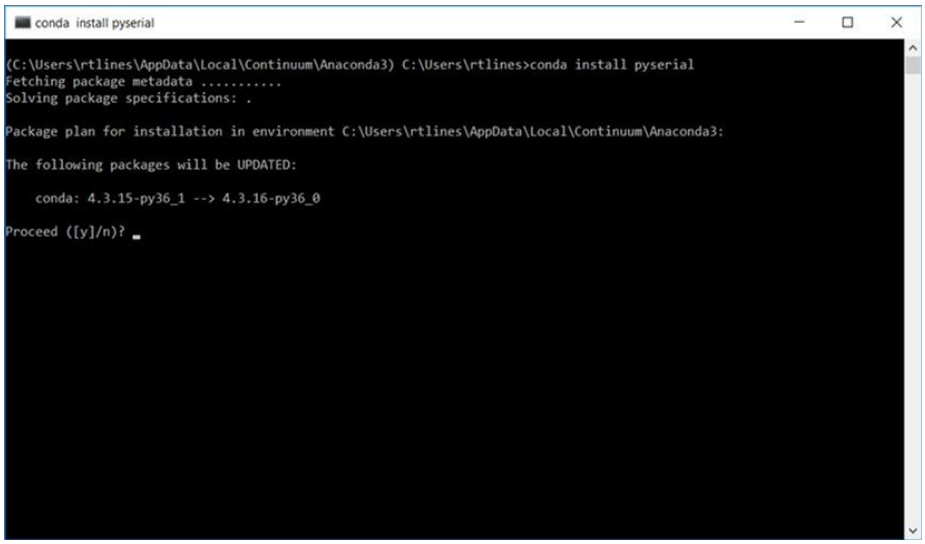


Figure 4.5: Installing Pyserial in the Anaconda Prompt window.

.

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 in Figure 4.6, and you will be all ready to start writing code to get data from the serial port.

4.2 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. This process is illustrated in Figure 4.7.

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, i.e. we only want some of the data that the Arduino has sent. In Figure 4.7, this data is indicated by the red arrows. 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. 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

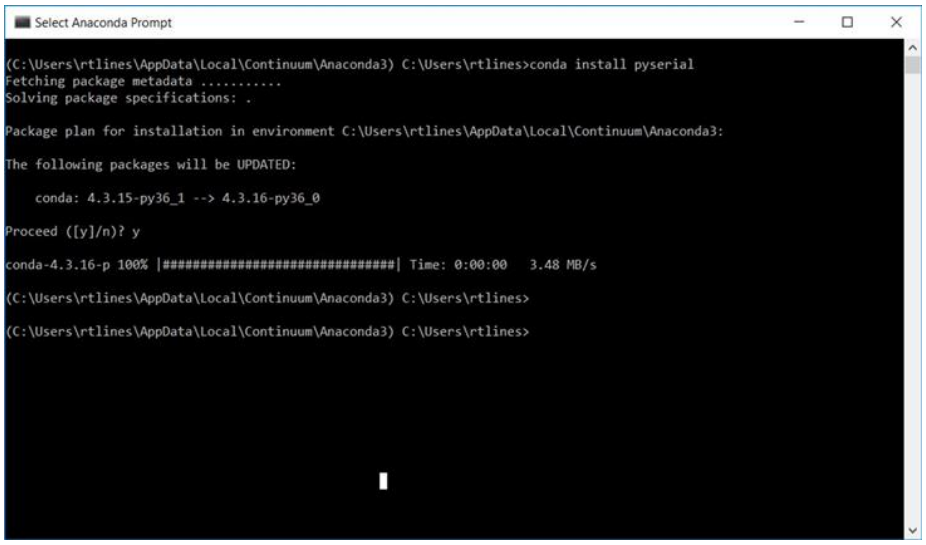


Figure 4.6: Pyserial installation complete.

Figure 4.7: A representation of data at the serial port. The Arduino may send more data to the serial port than we need. In the examples in the text, we only want to retrieve the data indicated by the red arrows.

continuously read in data, but only store it every few seconds. We will use this technique in the code that follows.

Just for fun, let's think of an actual data collection scenario. 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. Suppose we also want to retrieve a total of 10 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's 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 "\\." Normally, the Windows operating system delineates directories with a single backslash. However, in Python, the backslash character is used to instruct the computer that the character that follows is special. The double backslash simply tells Python that the special character it needs is a backslash. This is a quirk in most programming languages when you need to include backslashes in a string of characters, and you need to write the path this way.

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 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 connected to the port COM6, but that might not be true for you. Use the Arduino app to find out where your Arduino is connected, as in Figure **??**.

The port in our Python code must match the port indicated in the Arduino IDE. Ports are named differently on a Mac or on Linux, 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 code writes the data point to a file. I have it calculating the time since the beginning of our data collection (we might just need that in a future lab) and outputting that 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`.

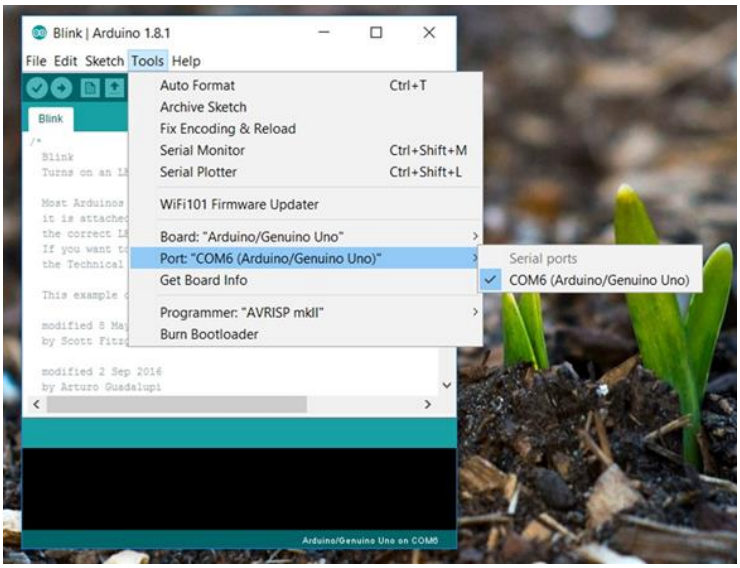


Figure 4.8: Finding which serial port the Arduino is connected to.

At the end of the program we close the file

```
fileObject.close()
```

We also absolutely must close the serial port with

```
ser.close()
```

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

```
#-----------------------------------------------------------------------
# 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
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('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 writes combines the time since we started and the
                                      Arduino
    # value from the serial port into one string
    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 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.

We can modify the simple voltmeter sketch from the last lab to give the time that the data was taken and to send both the time and the voltage to the serial port. Here is the updated sketch (remember since it is a simple voltmeter it can only handle 0V to +5V.)

```
////////////////////////////////////////////////////////
// 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);
  }
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
```

Only one process can connect to a serial port at any given time. Consequently, you can't simultaneously use the Arduino serial monitor and get data from the serial port using Python. 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.

4.3 Getting Pyserial if you have a Mac

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

4.3.1 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.

4.3.2 Manually install Pyserial using the Terminal app

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.

4.3.3 Mac Paths and Port Notation

Mac computers list paths to files differently than PC computers. 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 in 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.

```
#-----------------------------------------------------------------------
# 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 writes combines the time since we started and the
                                          Arduino
    # value from the serial port into one string
    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. Most of the code is comments, so don't be discouraged by the length.

LAB ACTIVITY

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 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.
3. Together with your group, brainstorm some ideas for your design project later in the semester.

Chapter 5

DataLogging

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Explain what an Arduino shield is.
- Use a datalogging shield to record Arduino data to an SD card.
- Use an external power source with the Arduino.

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 (no surprise here, since that's what all of our electronic measuring devices must do).

Our focus is to build an independent system to measure temperature, one that doesn't have to be connected to our computer. Up to this point, our Arduino has been powered via the USB connection to our computers, and the data has been recorded directly on our computers. 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 needed to send a sensor into the atmosphere via a high altitude balloon, you would not want (or even be able) to send your laptop up with it.

5.1 Arduino Shields

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.

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. Figure 5.1 shows an example of a simple prototyping shield (basically like our prototyping boards, but with the intention that circuit elements will be soldered to the board). Notice the long wire pins on the bottom. These are designed to be plugged directly into the Arduino. (When the shield is not in use, these pins should be pushed into the conductive foam block. This will not only protect them from getting bent, but will also protect the electrical circuits on the shield from static shocks.)

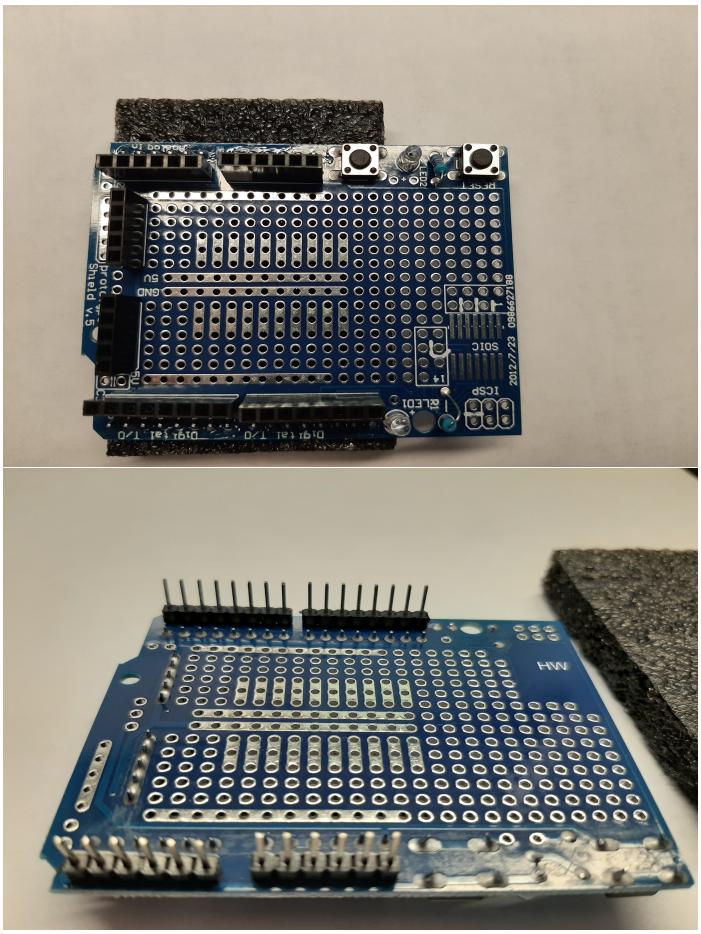


Figure 5.1: An arduino prototyping shield. The upper image shows the top of the shield, and the lower image shows the bottom of the shield.

Many shields are compatible with other shields such that they can be stacked on top of each other. Whether or not two shields are compatible depends on which 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, you must be aware of which pins the shield is using. Otherwise, you might interfere with the operation of the shield.

5.2 Data Logger Shield

Data logging is one of the most important parts of any experiment. In addition to collecting the information from our sensors, we also often want to record the time at which that data was collected. 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 because it has a built in real time clock (RTC).

Take a close look at the data logging shield shown in Figure 5.2. You will notice that the SD card slot (right next to the words "Data logger Shield") is close to an IC. That IC (a black rectangle) controls the SD card.

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. 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 tunning fork that keeps oscillating. These oscillations are counted by the IC and are used to keep time.

The pins on the shield are labeled, just like on your Arduino. Notice, though, that some of the pins are labeled with a white square and a black 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.

5.3 Setting up the data logging shield

The data logger shield uses a set of pre-written code, known as a "library". 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.

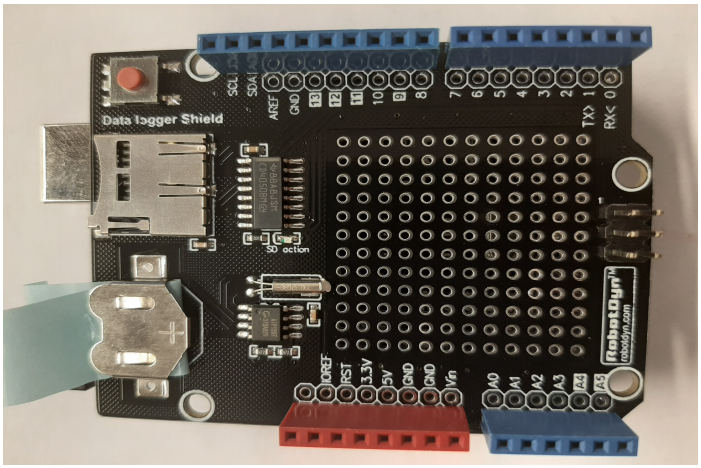


Figure 5.2: A data logging shield, with the top side shown.

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 it is still there. Then upload the code below. 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));
```

The arguments to the `DateTime` function are the year, month, day of the month, hour (24 hour time), minute, and second. Change the numbers to match the current date and time, then upload the code again. This should 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.

```
// ================================================
// 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"

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. You will need to modify the code such that it saves the temperature.

5.4 Powering the Arduino

Up to this point, we have powered our Arduino by means of the USB cable attached to our computer. Clearly, if our objective is remote data logging, we'll need a different way to provide power.

You undoubtedly have noticed that the Arduino has a cylindrical port on the same edge as the USB connection. This can be seen in Figure 5.3, on the right side of the Arduino. You'll also note that your Arduino kit included a 9 V battery and a connector, also both shown in this figure. The external power solution is fairly obvious at this point. (However, be advised that many of the batteries that come in the Arduino kit will already be "dead". If things aren't working as expected, check your battery voltage using a multimeter.)

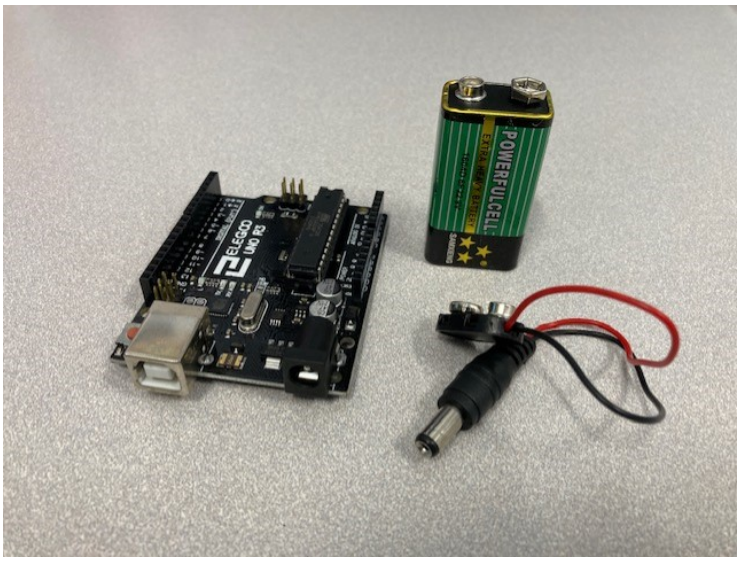


Figure 5.3: Powering an Arduino with a 9 V battery.

LAB ACTIVITY

Work in groups of three to five for this set of activities. 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 team member has completed the previous step.

1. Get the Datalogger sheild up and running and set the correct date and time.

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. (This is a great exercise in reading the documentation for circuit elements!) 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. Power your sensor system with a battery to make sure it can operate independent of the computer.

LAB ACTIVITY

It is also time to start thinking about your final project. We're now going to ask you to read ahead, and look through the chapter on writing a proposal. Brainstorm some project ideas with your group, settle on one option (it's probably a good idea to run that option by your instructor), and start writing a proposal.

Part II

Testing Models

Chapter 6

Validation of Ohm's Law

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Use an Arduino to make simultaneous measurements of voltage and current.
- Explain the considerations associated with choosing a good shunt resistance.
- Conduct a linear fit to data, accounting for uncertainties in the fit parameters.

What physicists do is to try to understand how the universe works. To do this we use the scientific method, and what differentiates the scientific method 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.

6.1 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 distil 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 can plot our equation for Ohm's law to show the ΔV vs. I relationship, as in Figure 6.1.

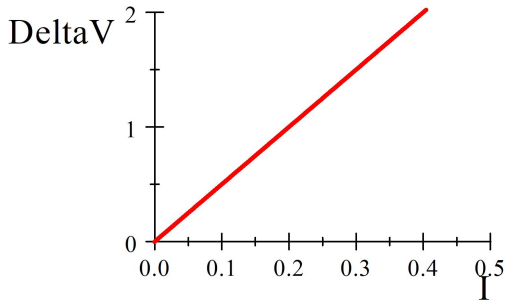


Figure 6.1: Ohm's law, showing the linear relationship between current and potential difference for an Ohmic resistance.

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 Figure 6.2 instead? Such a device would *not* follow Ohm's law. We would say that such a device is *nonohmic*.

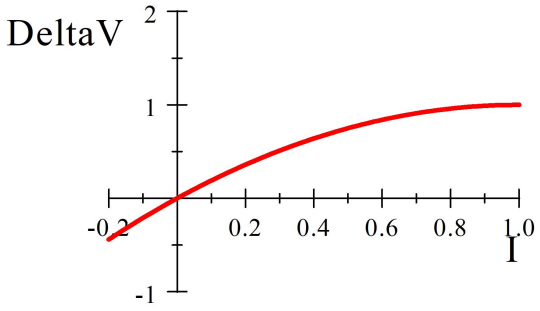


Figure 6.2: Voltage v. current for a non-Ohmic resistance.

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 (both ΔV and I). This isn't a problem, though, because our Arduninos have five analog inputs. 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?

6.2 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.

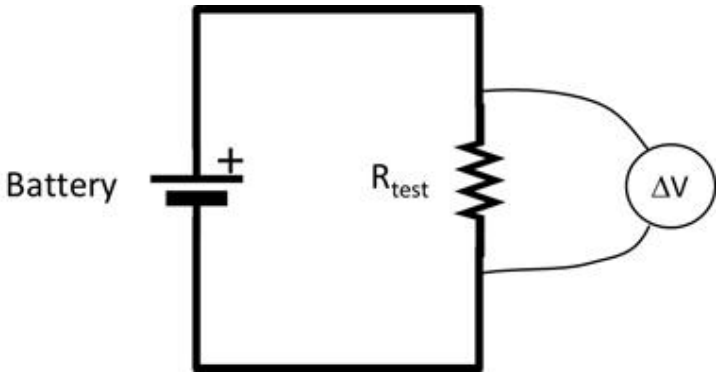


Figure 6.3: Measuring the voltage drop across a resistor.

As seen in Figure 6.3, ee 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.

To measure a current with our Arduino we have to somehow turn that current into a voltage. This is true of all measurements we make with an Arduino or any other DAQ. 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

$$\Delta V_s = IR_s$$

which we can solve for I

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

If we add a second resistor, R_s to the circuit, as seen in Figure **??**, and measure the voltage across that circuit, we will be able to calculate the current.

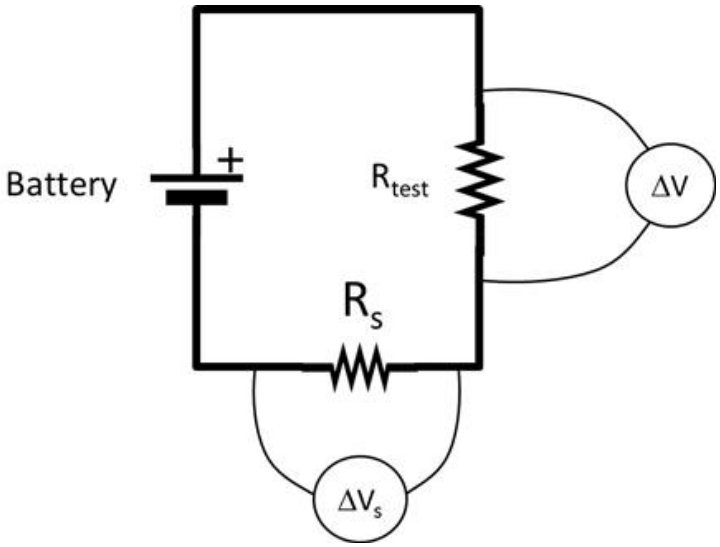


Figure 6.4: Adding a shunt resistor to measure current.

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}.

$$R_s \ll R_{test}$$

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!

6.3 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 (Figure 6.5).

By drawing the diagram in Figure 6.5, 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 as in Figure 6.6. Of course, we need to wire the negative pole of the battery or power supply to the GND pin. Then our two voltage

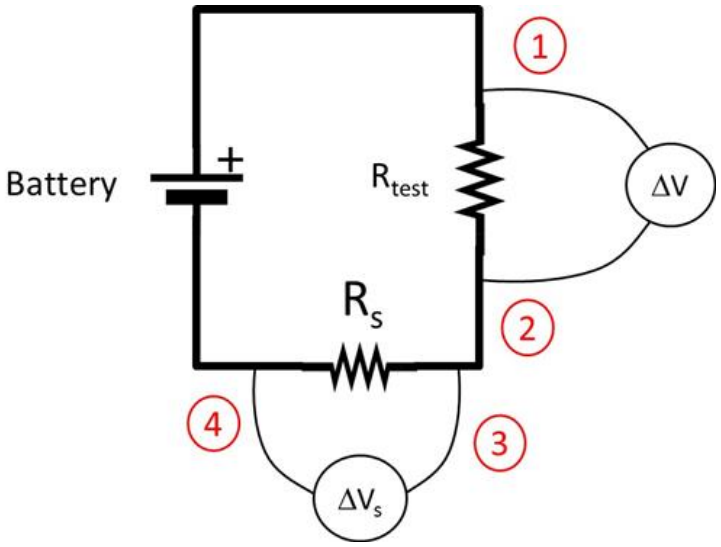


Figure 6.5: Voltage locations needed to test Ohm's law.

difference measurements will be formed from

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

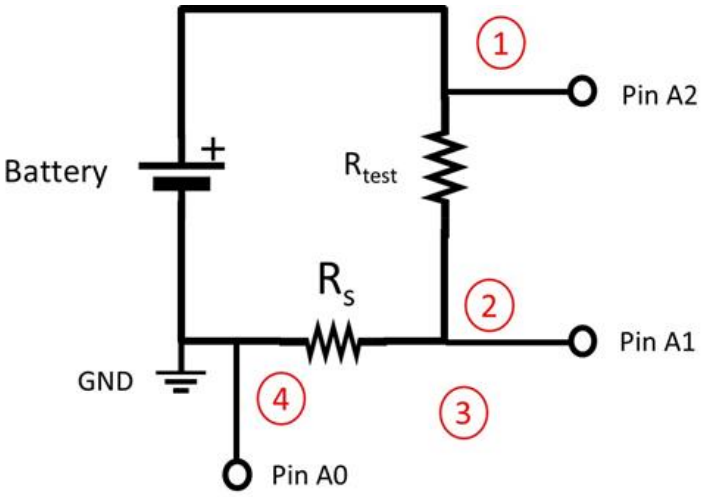


Figure 6.6: Wiring the Arduino to test Ohm's law.

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.

```
 ////////////////////////////////////////////////////////
// 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= 220;
    //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);
  Serial.print( "␣resistance␣" );
  Serial.println(voltage/amperage, 4);
}
```

6.3.1 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, nor should it be so large that it effects 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. We recognize this from our previous labs. 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,

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

and this is larger than ΔV_s so once we use our Arduino analog to digital converter (ADC), 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

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

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.

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

This 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.

6.4 Finding Uncertainty in a Calculated Value

In the last section we calculated 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 previous lab work. 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 $\Delta V_s = 9.98\text{mV}$ and $R_s = 170\Omega$. We know that $\delta \Delta V_s = 5.0588\text{mV}$ and our resistors are only good to 1% so that would be $\delta R_S = 1.7\Omega$

$$\begin{aligned} \delta I &= \sqrt{\left(\left(\frac{1}{170\Omega}\right)(5.0588\text{mV})\right)^2 + \left(\left(-\frac{65.764\text{mV}}{(170\Omega)^2}\right)(1.7\Omega)\right)^2} \\ &= 3.0008 \times 10^{-5}\text{A} \end{aligned}$$

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

$$\begin{aligned} I &= \frac{\Delta V_{ps}}{R} \\ &= \frac{2\text{V}}{5000\Omega} \\ &= 0.0004\text{A} \\ &= 4 \times 10^4\text{A} \end{aligned}$$

so the fractional uncertainty in the current will be

$$100\frac{3.0008 \times 10^{-5}\text{A}}{4 \times 10^4\text{A}} = 7.502\%$$

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

$$\left(\left(\frac{1}{R_s}\right)(\delta\Delta V_s)\right)^2 = \left(\left(\frac{1}{170\Omega}\right)(5.0588\text{mV})\right)^2 = 8.8552\times10^{-10}\text{A}^2$$

$$\left(\left(-\frac{\Delta V_s}{R_s^2}\right)(\delta R_s)\right)^2 = \left(\left(-\frac{65.764\text{mV}}{(170\Omega)^2}\right)(1.7\Omega.)\right)^2 = 1.4965\times10^{-11}\text{A}^2$$

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.

6.4.1 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 spreadsheet or symbolic math processor 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

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

As you can see, it is probably best to put all this in a symbolic package (like Mathmatica or Maple, or Sage, 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 (Figure 6.7). 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.

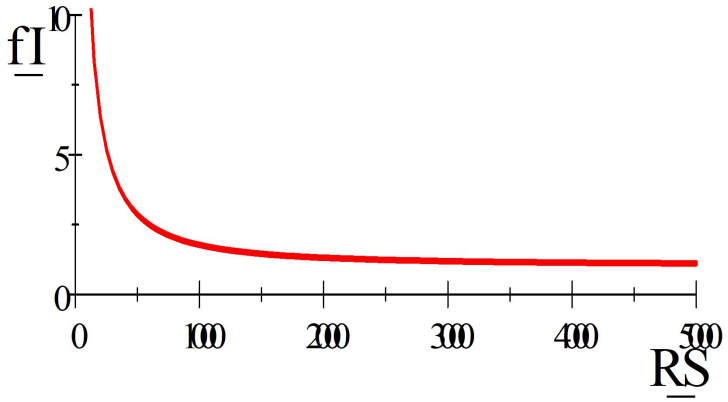


Figure 6.7: Fractional uncertainty in current v. shunt resistance.

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} uncertainty δR_{test} to see if Ohm's law really works. In finding the uncertainty in our measuring devices we haven't found δR_{test}. We will review a different way to do that analysis.

6.5 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 Figure 6.8.

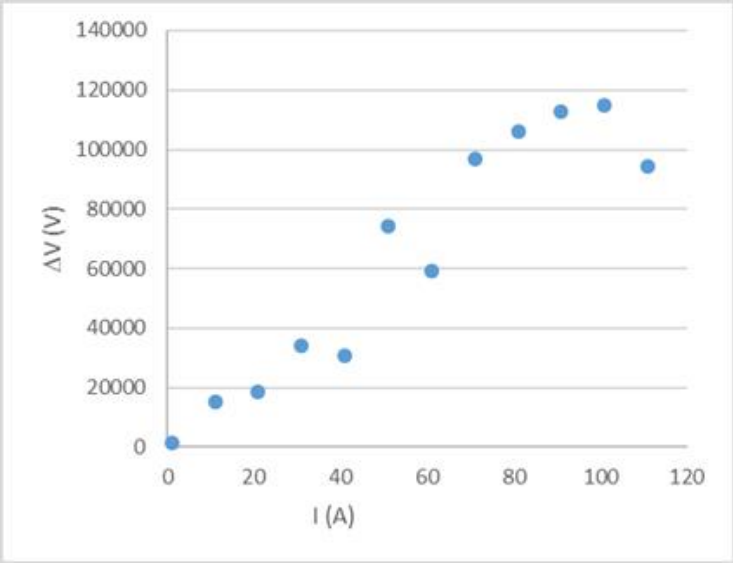


Figure 6.8: Sample data for testing Ohm's law.

This looks pretty good. It seems to be sort of linear. We might guess from this that Ohm's law is begin obeyed. But we want R_{test}, and R_{test} is the slope of this line. 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}. In the past, you may have done such a thing using a *curve fit*. In Figure 6.9, a curve fit is shown for the data from the last figure.

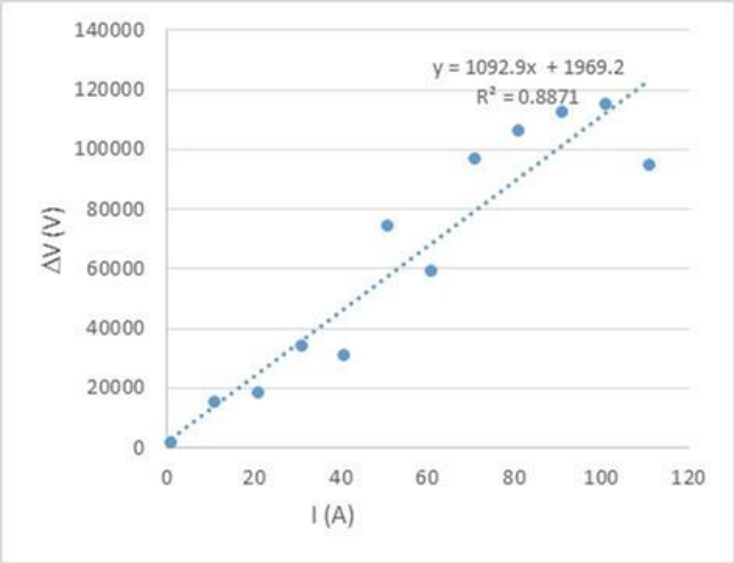


Figure 6.9: A linear fit to the Ohm's law data.

Notice that this fit was performed using Microsoft Excel, but you could also do this directly in Python. You could also do this in LoggerPro or many other data analysis programs. From the curve fit equation we can see that we have a slope of 1092.9. 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. In the data above the we can see that the resistance is a little more than 1092Ω because that is the slope of our fit line. But we know we need an uncertainty along with this nominal value. Can we get the computer to do this as well?

The answer is, of course, Yes! And that will save us a bunch of math! In some data analysis programs this is easy. LoggerPro, for example, just gives you the uncertainty in m and b. Excel does not. If you want to use LoggerPro, that is fine. If you know how to do this in Python, go ahead. These uncertainties can also be found using Microsoft Excel, using the `linst` function. That process, however, is relatively clunky, and you will be far better off using a different tool to complete your fit. (You can find instructions for using LoggerPro to fit data in the next lab.)

6.6 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.

LAB ACTIVITY

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. If you still have time, repeat part 2 for a 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 7

Resistors and Capacitors

OBJECTIVES

At the conclusion of these laboratory activities, you should be able to do each of the following:

- Build an instrument using the Arduino that is able to measure time varying data and record it to a computer.
- Use LoggerPro to fit a model to your data.
- Verify that the equations describing current in an RC circuit are valid.

ITEMS TO REVIEW

Please review the following concepts as part of your preparation for this lab.

- *RC* circuits.

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.

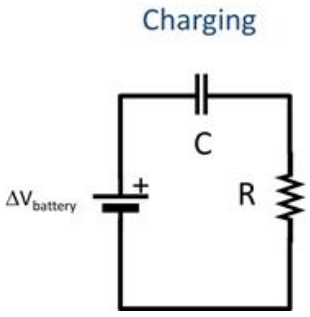


Figure 7.1: Charging and RC circuit.

7.1 The Model to Test

Let's start by thinking of hooking up a capacitor and a resistor in series with a battery, as seen in Figure 7.2. The capacitor will become charged. The voltage across the capacitor as a function of time will be given by

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

where

$$\tau = RC \tag{7.1}$$

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

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

while we charge up the capacitor.

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 the voltage as a function of time, as seen in Figure **??**. Notice, that by about $t = 70$s we essentially have $\Delta V_C = \Delta V_{battery}$. But up to that point,

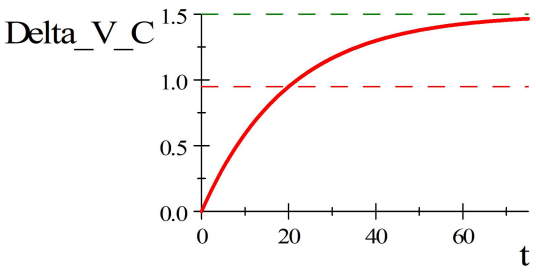


Figure 7.2: The voltage in a charging RC circuit as a function of time.

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. You should recall that $e^0 = 1$, so at $t = 0$ we do have $\Delta V_C = 0$ on the capacitor. 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.

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 $\Delta V_{battery}$.

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, we have to get an idea of how to make the measurement from the model we are testing.

7.2 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, and either one of these is sufficient to test the model. For what follows, our focus will be on measuring the voltage. You know from a previous lab how we might add current as a function of time.)

We've already seen how an Arduino can measure voltage and record it as a function of time. 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. For our capacitor, the voltage difference can be found by

$$\Delta V_C = V_2 - V_1$$

where V_2 and V_1 are the potentials on either side of the capacitor (see Figure 7.3), neither of which will be ground. We really have to measure both with our Arduino, and we also need a ground connection. The wiring diagram to do this is seen in Figure 7.3. The sketch will be similar to one from a previous lab.

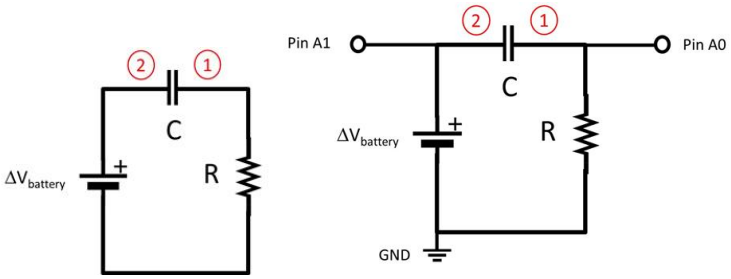


Figure 7.3: Wiring the RC circuit to the Arduino. The left panel indicates the two voltages that we need to measure, and the right panel shows where we would connect the circuit to the Arduino pins.

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
// very simple voltmeter
// will measure 0 to 5V only!
// Voltages outside 0 to 5V will destroy your Arduino!!!
// Delta_V_shunt must therefore be much less than  5V
////////////////////////////////////////////////////////////
// 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 sketch is similar to our simple voltmeter, except that it uses two ADC

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. This code is similar to our previous version, with a few modifications.

```
#-------------------------------------------------------------------
#-------------------------------------------------------------------
# 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 combines the time since we started and the
                                        Arduino
    # value from the serial port into one string
    writeString=str(arduinoData) #+ " \n"
    # The file write command adds a new line character The Arduino
                                        added a
    #   new line character, so we have double spacing. Let's remove one
                                          of
    #   them before we write to the file.
    writeString = writeString.replace("\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()
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
```

7.3 Fitting the Data

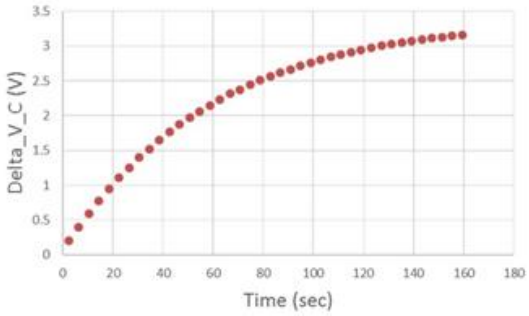


Figure 7.4: Sample data for the RC circuit capacitor voltage.

The data we collect with our instrument, when plotted, might look something like Figure 7.4. You can make this plot in Excel, but Excel doesn't have the right function built into it for a curve fit. We will use a different analysis program called LoggerPro. It is not hard to use, and you can copy your data from a file (or from Excel) into LoggerPro easily. The next section shows how to make this work. If you are a fantastic Python programmer, you could use Python for this part. If you are a die-hard Excel user, it is possible to use Excel and get the same result (but not very easy).

7.3.1 LoggerPro Curve Fitting

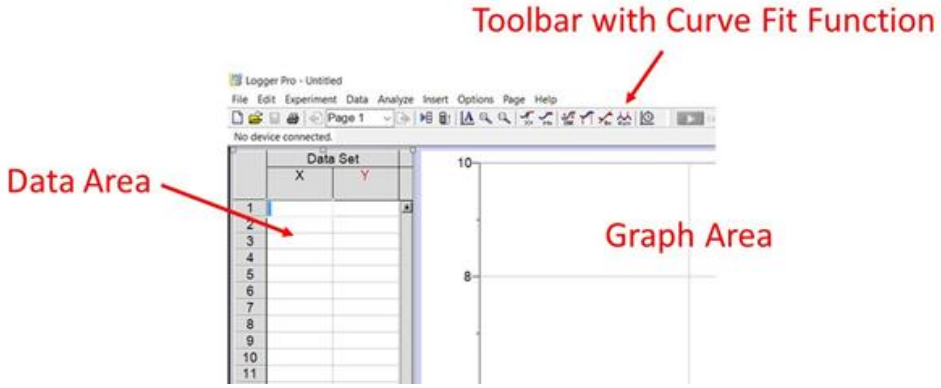


Figure 7.5: The LoggerPro interface.

Let's start by noting that you can download LoggerPro to your own computer if you would like. You should have received instructions for doing so from your

instructor. When you open LoggerPro, the window will appear similar to Figure 7.5. You will note that on the left side of the screen there is a spreadsheet-like area that holds your data, and on the right is a graph of the data. The toolbar at the top includes the functions we will need to fit our data.

If you have already imported your data into Excel or some other program, it can be copied and pasted into the data area in LoggerPro. Place the time data in the "X" column, and the voltage data in the "Y" column. When you have done so, LoggerPro will graph the data, and you will see something like Figure 7.6.

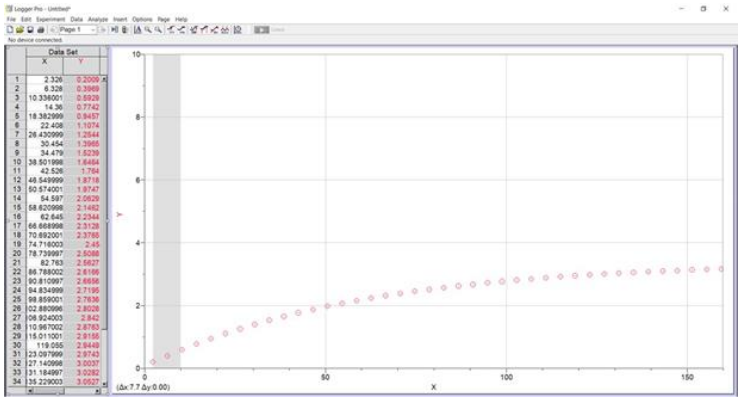


Figure 7.6: Capacitor voltage graphed in LoggerPro.

We now want to fit a curve to this data. The curve fit function can be accessed from the tool bar. The button is shown in the left panel of Figure 7.7. When you click this button, a dialog box will appear. This dialog box can be seen in the right panel of Figure 7.7.

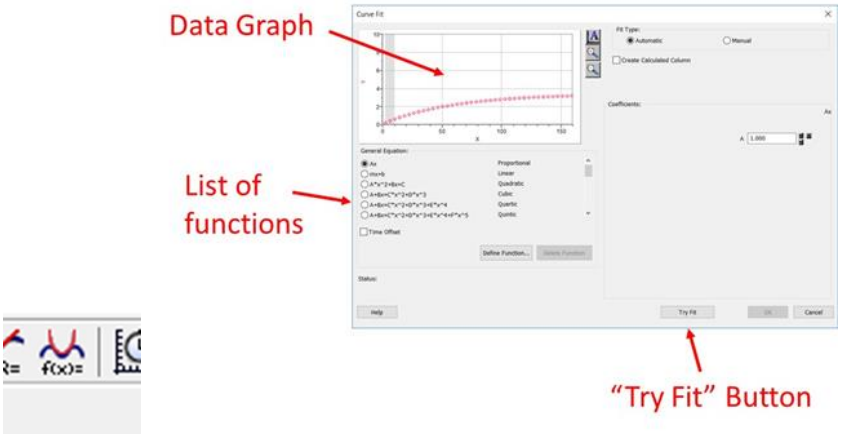


Figure 7.7: The curve fit button and dialog in LoggerPro.

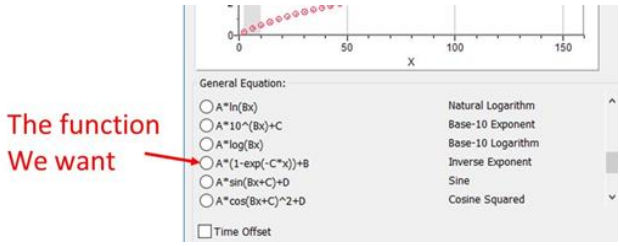


Figure 7.8: The LoggerPro fit function needed for our model.

It may be tempting to try just any function to see if it fits. Remember, though, that our goal is not to have a great fit. The goal is to see if our data fit the equation from our capacitor model.

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

We need to find this particular function (which is not one of the default functions in Excel). Of course, we won't find a match with our exact notation. The one we want is given as

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

in LoggerPro (Figure 7.8), and now we have to match our variables with theirs. Let's compare the equations.

$$\begin{aligned} y &= A * (1 - \exp(-C * x)) + B \\ \Delta V_C\left(t\right) &= \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) + 0 \end{aligned}$$

We can see that our $\Delta V_{\max}$ corresponds to their A, and our τ corresponds to their $1/C$. Our equation doesn't have a variable corresponding to their B.

Knowing what the variables will represent for our equation, we can now see how well the function fits our data. Do this by clicking the "Try Fit" button in the lower right portion of the dialog.

If the fit looks good, choose the "OK" button. You will return to the main LoggerPro window, and will see your graph, but now with the line of best fit and a new little box (Figure 7.9). The curve fit looks nice and that is comforting. For the data shown here, it looks like our capacitor model might be correct, but we can't be sure until we add in error bars. Before we do that, though, let's look at the new little box (Figure 7.9, right panel). The box has our fit equation that we chose, and it has values for the fit parameters and their uncertainties. We will need those later!

Let's add on the error bars now. Right click on the graph if you have a PC or do the Mac equivalent if you have a Mac. A new dialog appears and in this case choose "Graph Options" (Figure 7.10, left panel). On the "Graph Options" dialog make sure both x and y-error bars are checked. Chose "Done".

Right click on the graph again. This time choose "Column Options" (Figure 7.11, left panel). In the dialog, choose the y-data set. Another dialog appears

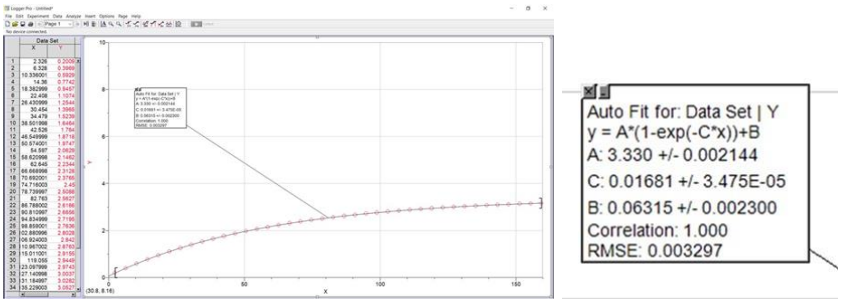


Figure 7.9: The LoggerPro window after completing the fit. The left panel shows the main window, and the right panel shows the detail of the box containing the fit parameters.

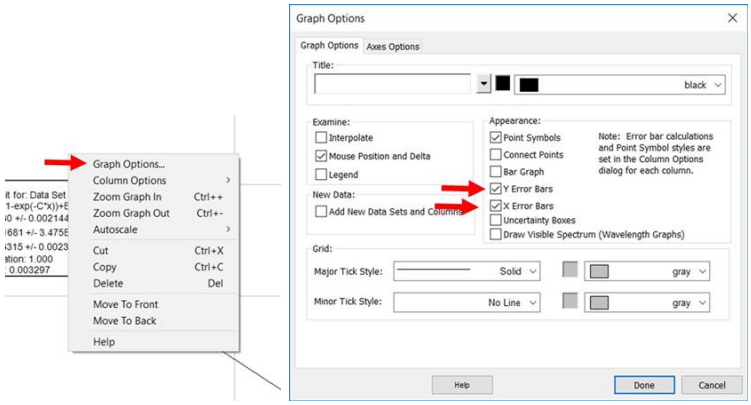


Figure 7.10: LoggerPro graph options, used to add error bars to the graph. The left panel shows the pop up menu that appears upon right clicking the graph. The right panel shows the graph options dialog.

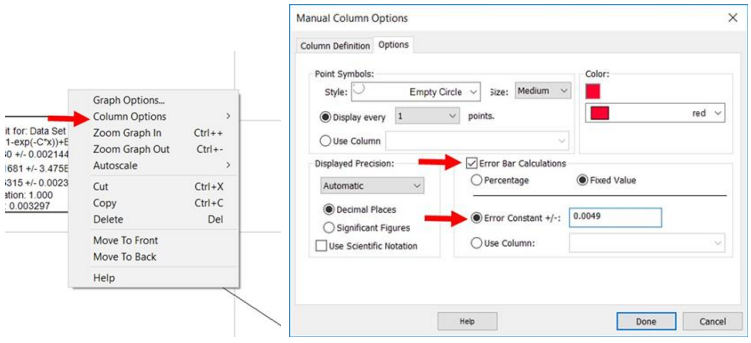


Figure 7.11: LoggerPro column options. The left panel shows the pop up menu that appears upon right clicking the graph. The right panel shows the dialog for the y data set.

with two tabs (Figure 7.11, right panel). Choose the "Options" tab. You will see a place to choose how error bars are calculated. If you used the simple voltmeter sketch as your basis, you know the quantization error is about 4.9mV. That will be true for every voltage measurement so we can input this as a constant value. If you used a voltage divider, you will have to use your calculated error value here. When you have your error value in place, choose "Done". The final product will look like Figure 7.12.

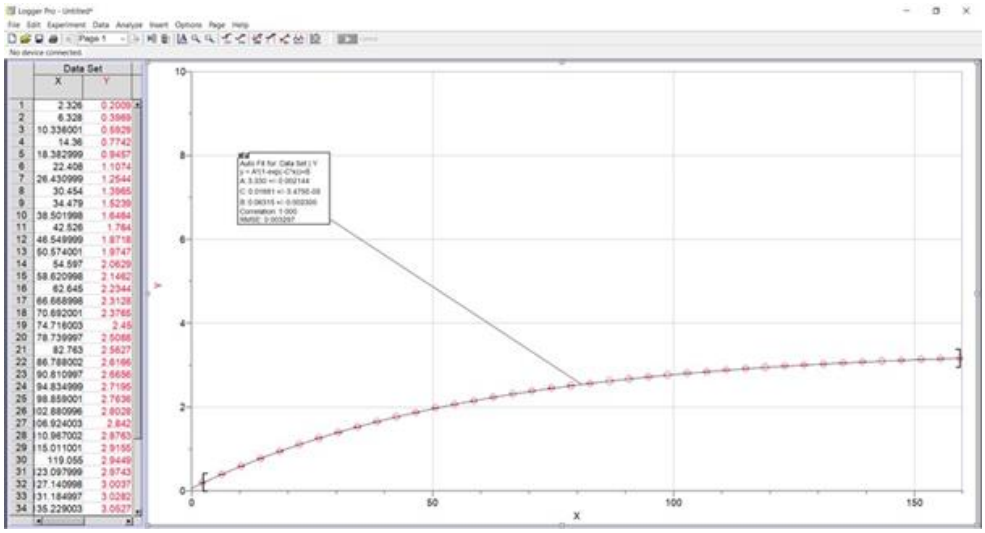


Figure 7.12: The final graph, now with error bars.

The voltage error was so small that it is hard to see the error bars! That is fine. What this tells us is that our fit line must go right through the center of each data point, so we are really doing fine. The data supports the model.

But now let's go back to our little box of curve fit parameters. We identified

$$\tau = \frac{1}{\mathtt{C}}$$

and for my data I have

$$\mathtt{C} = 0.01681 \pm 3.475 \times 10^{-5}$$

We need units, and looking at the equation we know τ has units of seconds, so C must have units of inverse seconds.

$$\mathtt{C} = \left(0.01681 \pm 3.475 \times 10^{-5}\right) \frac{1}{\mathrm{s}}$$

so we can find a value for τ. For my data, I have

$$\begin{aligned} \tau_{measured} &= \frac{1}{0.01681 \frac{1}{\mathrm{s}}} \\ &= 59.488\mathrm{s} \end{aligned}$$

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.

LAB ACTIVITY

We have two equations that describe the voltage across the capacitor and the current in an RC circuit as a function of time:

$$\begin{aligned} \Delta V_C(t) &= \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) \\ I(t) &= I_{\max} e^{-\frac{t}{\tau}} \end{aligned}$$

where $\tau = RC$ is the time constant. As you complete these activities, follow good lab notebook procedures by recording everything!

1. Using a capacitor with a capacitance of about 20 μF and a resistor of about 1 MΩ, create a series RC circuit. You can power your circuit with a power supply, but be careful to stay in the 0V to +5V range (or to use a voltage divider to achieve this range at the Arduino input). Note that we will be using electrolytic capacitors today, which only work in one direction. If the circuit doesn't work, try turning your capacitor around.

2. Predict the time constant for your circuit, using the resistance and capacitance values.

3. Build your instrument, write the sketch, and write the Python collection code. Test every part of the instrument before you start collecting capacitor data. Don't forget to find your uncertainties.

4. Work with a lab partner from your group and collect the voltage data for your charging capacitor. Compare your data among your group to make sure things went well. Record your data in your lab notebook, or give a location of where the data is stored.

5. Graph your data. LoggerPro is fine for both graphing and the curve fit (next item). You should include this graph in your lab notebook (but might also include the curve fit described in the next item on the same graph).

6. 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 thoretical curve fits the data, it makes sense that something about the model is right. 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).

7. Find the time constant from your fit, 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.

8. Draw a conclusion: is our capacitor model good?

Chapter 8

Inductance and Series RLC Circuits Part 1

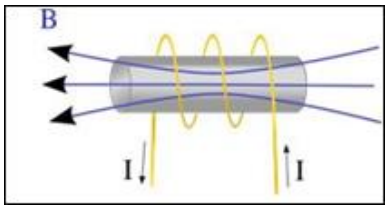
Hopefully by now you have learned in your PH 220 class that magnetism is related to current. Oersted discovered this by accident (so all those accidents we have experienced in our lab could be telling us something!). If you don't know yet, you will soon learn that changing magnetic fields can cause currents. 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*. I will give some review material here on how magnetic fields, currents, and inductances are related. Lenz's law is involved, and I will assume you know this rule.

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. So as the current starts to flow, Lenz's law tells us that there will be an induced emf that will oppose the flow. The potential drop across the resister in a simple battery-resister circuit is the potential drop due to the battery emf, *minus the induced emf*.

We can use this fact to control current in circuits. To see how, we can study a new caseLet'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 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 divide 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

$$\mathcal{E} = -N\frac{d\Phi_B}{dt} \equiv -L\frac{dI}{dt}$$

so for this case

$$-N\frac{d\Phi_B}{dt}\frac{dt}{dI} \equiv -L$$

$$L = N\frac{d\Phi_B}{dI}$$

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

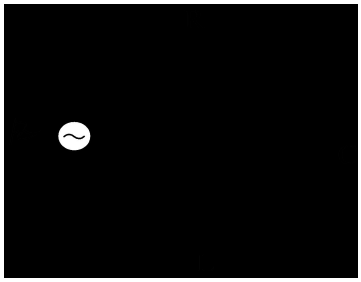
$$\begin{aligned} L &= N\frac{\Phi_B}{I} \\ &= N\frac{\left(\mu_o \frac{N}{\ell} IA\right)}{I} \\ &= \frac{\mu_o N^2 V}{\ell^2} \\ &= \mu_o n^2 V \end{aligned}$$

$$L = \mu_o n^2 V \tag{8.1}$$

where V here is the volume of the solenoid, $V = A\ell$.

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. We will use our signal generator to get a sinusoidally changing voltage. The circuit diagram should look like this.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

$$f^2 = \frac{1}{4\pi^2 LC}$$

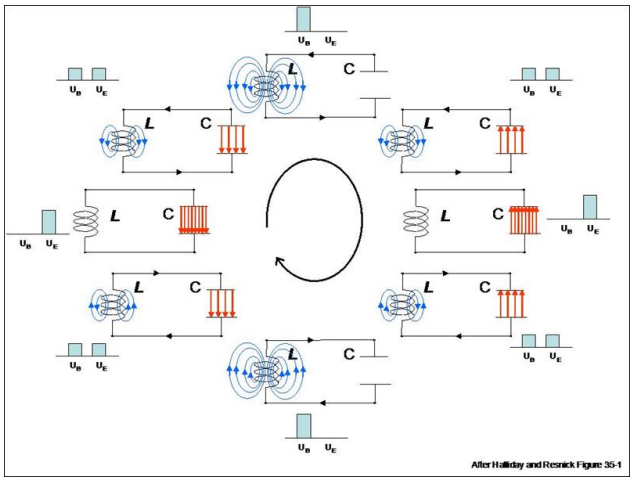
or

$$f = \frac{1}{2\pi\sqrt{LC}} \tag{8.2}$$

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!

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 enought 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 envison 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

$$\Delta V_C = \frac{\frac{E}{CL}}{\sqrt{\left(\left(2\pi\left(f\right)\right)^2 - \left(2\pi f'\right)^2\right)^2 + 2\left(\frac{R}{L}\right)\left(2\pi f'\right)^2}} \sin\left(\omega' t - \phi\right)$$

Notice the term $\left(\left(2\pi\left(f\right)\right)^2 - \left(2\pi f'\right)^2\right)^2$ 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

$$\begin{aligned} \mathcal{E} &= 5\text{V} \\ L &= 2.3 \times 10^{-3}\text{H} \\ R &= 1\text{M}\Omega \\ f_{resonance} &= 18.552\text{kHz} \\ C &= 32 \times 10^{-9}\text{F} \end{aligned}$$

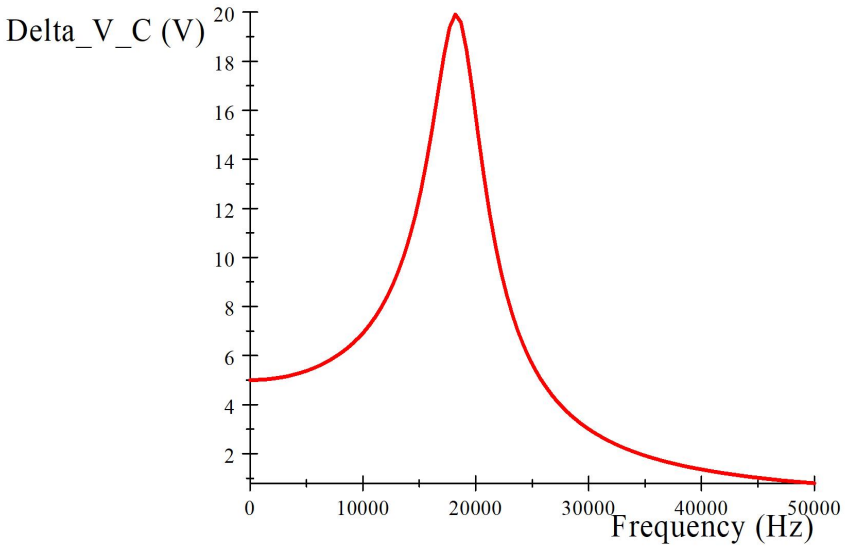
$$R > \sqrt{\frac{42.3 \times 10^{-3}\text{H}}{32 \times 10^{-9}\text{F}}} = 1149.7\Omega$$

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

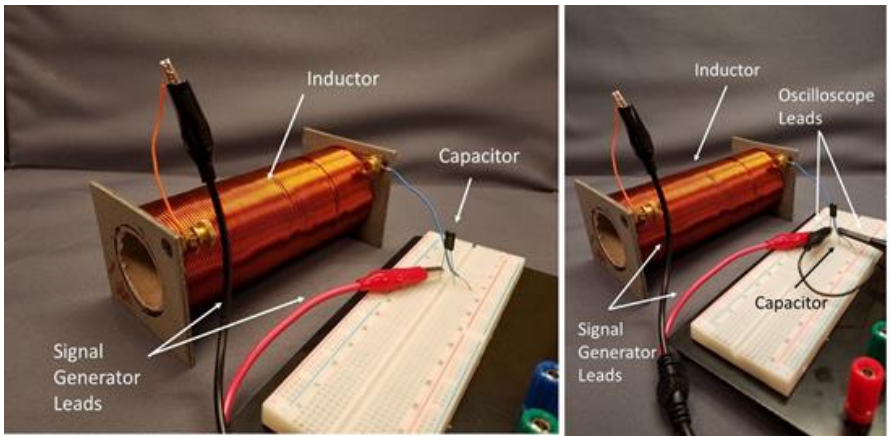
1. Estimate the inductance of your coil using equation 8.1.



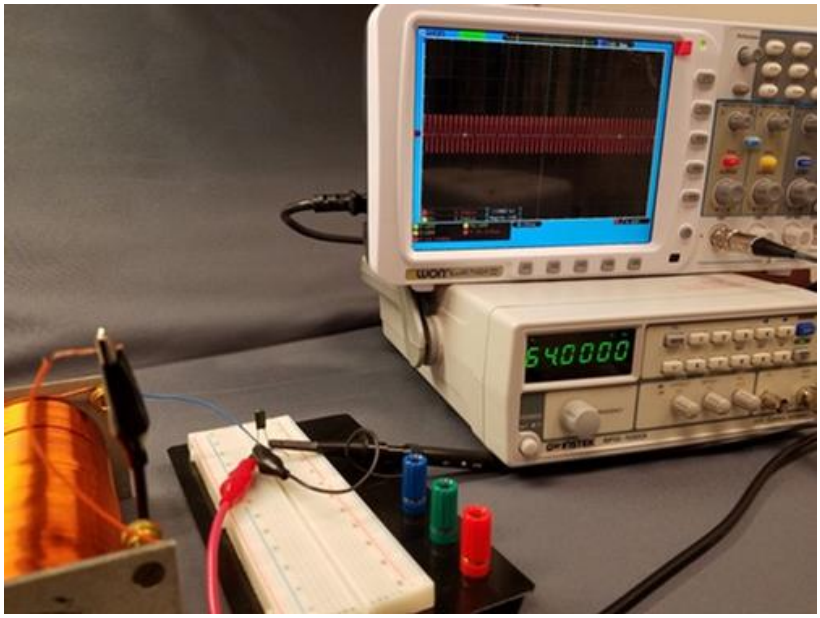
2. Find The inductance of the coil using an Oscilloscope

   (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 . Make sure your resistance is not too high using equation **??**. 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.

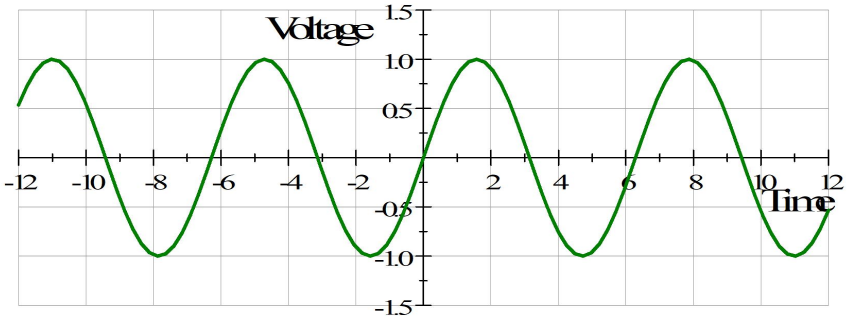
Chapter 9

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.

9.1 The Instrument

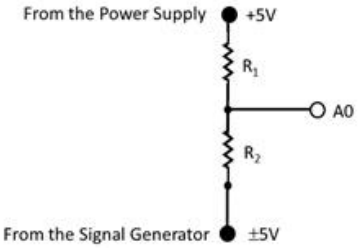
We have a new challenge. Our signal from our LRC circuit is sometimes negative. We will have truly negative voltages compared to our ground. Can we send a negative voltage into our Arduino? The answer is an emphatic NO! Negative voltages will also destroy our Arduino. But we really have a sinusoidal 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.



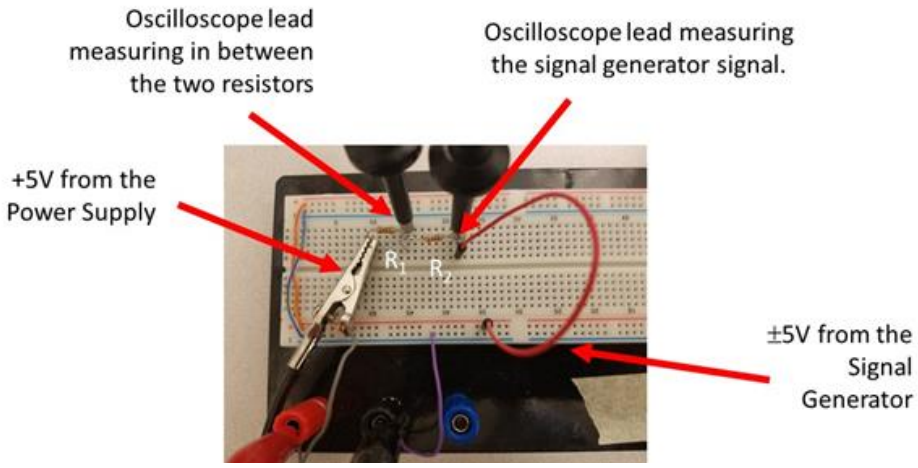
Let's start by setting $R_1 = R_2$. Then any voltage at the junction between R_1 and R_2 should be midway between the voltages input on the top of R_1 and the Bottom of R_2. We put in $+5$V from our power supply on the top. And suppose we hook -5V to the bottom (say, from our signal generator). We expect 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.

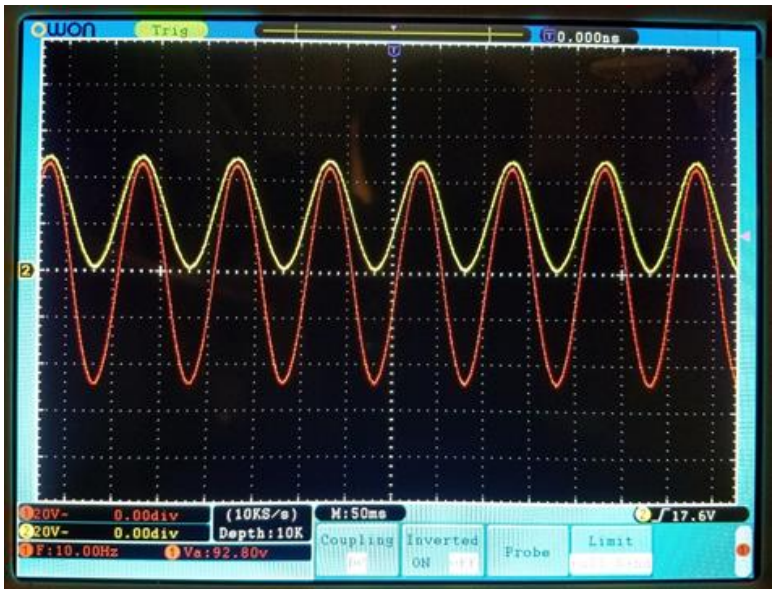
The oscilloscope, power supply, and signal generator are all grounded through their grounded plugs. But our Arduino is not. We need to tie all the grounds of all our equipment together. So put a wire on the power supply negative output and wire it with an alligator clip to the grounded exterior of the 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.

9.2 Sampling Theory, a complication

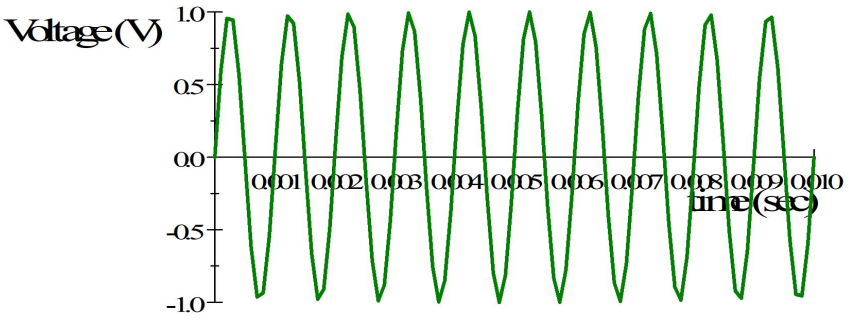
Before we finish designing this experiment, we need to think about another limitation of our Arduino devices. That is that they can only take up to 2000 measurements in a second. We say they have a maximum sampling rate of 2000Hz. To try to understand this, consider trying to measure a sine wave.

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

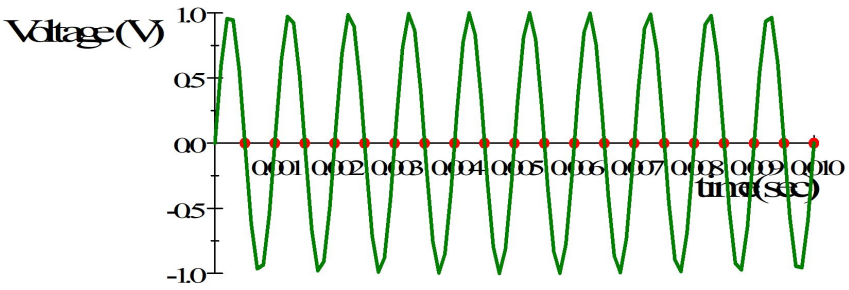
$$\frac{1}{1000\text{Hz}} = 0.001\,\text{s}$$



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

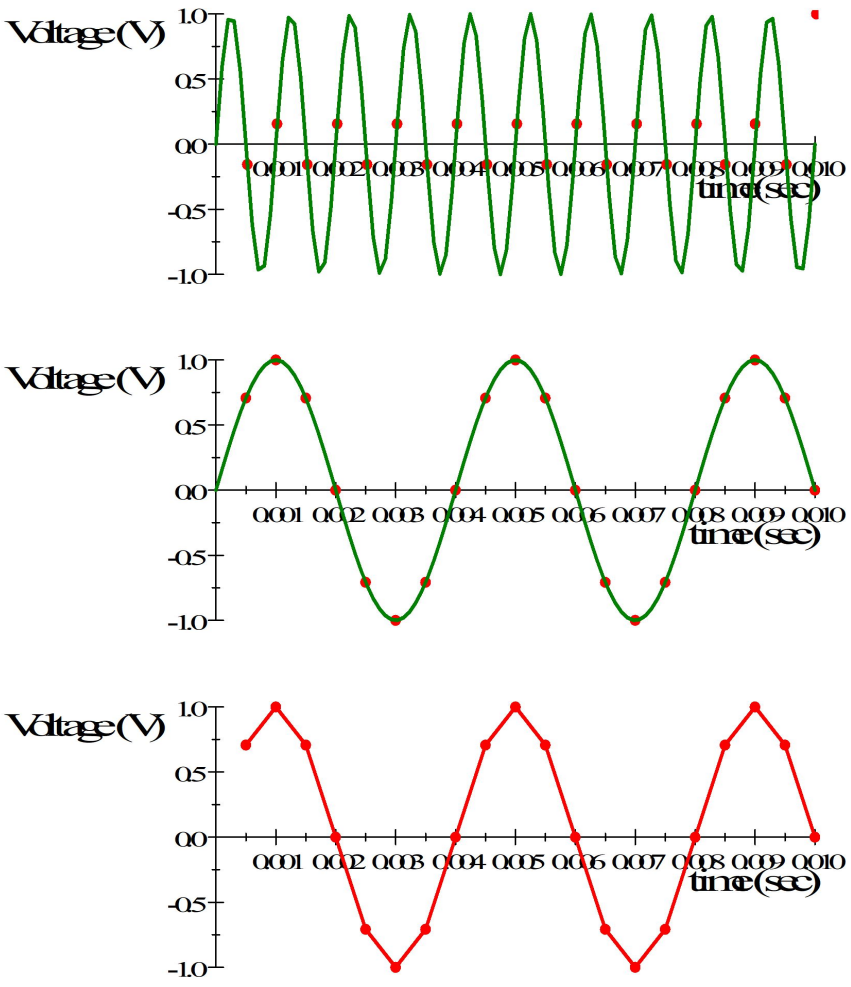
$$\frac{1}{2000\text{Hz}} = 0.000\,5\text{s}$$

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.

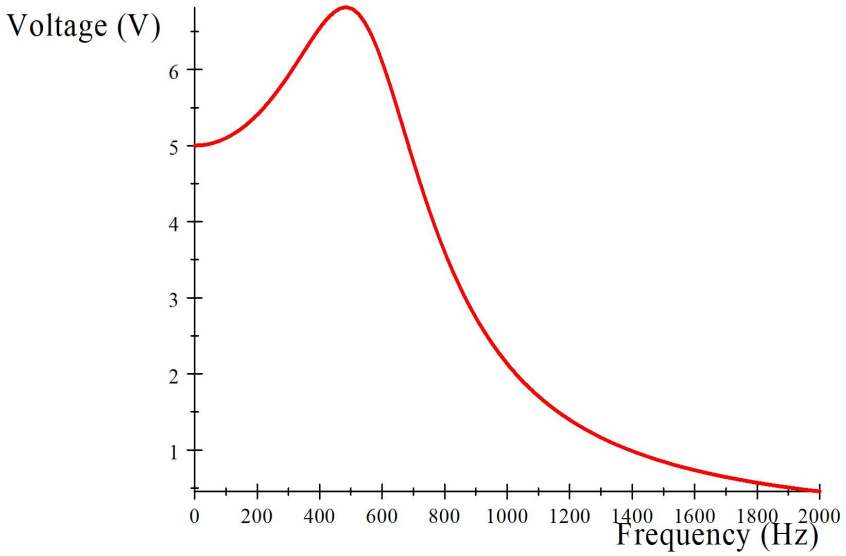
Now if we just plotted the measurements, we could still tell it was a sine wave.Granted, it doesn't look great, but we wouldn't mistake it for a straight line. This problem of having to take measurements faster than our signal changes is called aliasing (because if you get it wrong, it looks like the 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 some-



what ratty, but we should be able to see a sine wave and find when it is maximum.

9.3 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.

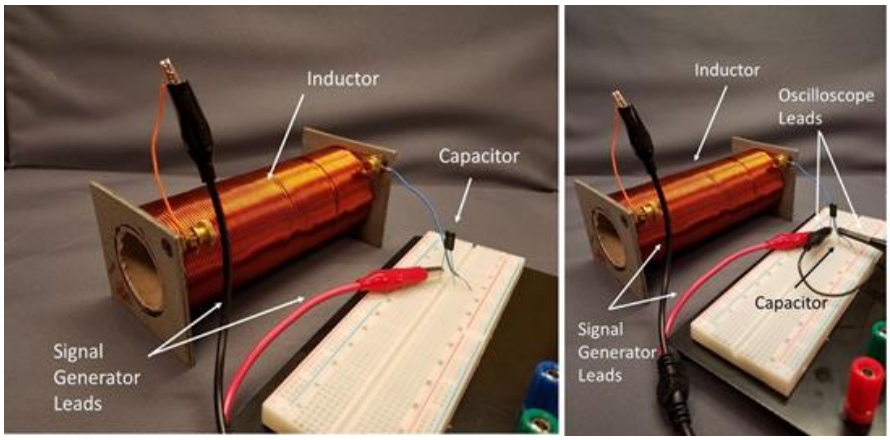
2. Measuring the inductance using resonance with an Arduino based oscilloscope

   (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.

Part III

Student Designed Experiments

Chapter 10

Writing a Proposal

It's time to start thinking of what experiment you and your group will design. 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

Each of these are discussed below.

10.1 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.

10.2 Procedures and anticipated difficulties

Hopefully, your reader will be so excited by the thought of you solving your experimental problem 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.

10.3 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 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, 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 text book 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 equaiton (like the resistance and the capacitance in today's lab) you will know how much uncertatinty 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 not using 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 performing an analysis like this, try for a symbolic equation for uncertainty. Program these equations into Scientific Workplace, or Maple, or SAGE, or MathCAD, or whatever symbolic math processor you have. Alternatevly, you could code it into Python. Then, as actual measurements change, you 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.

10.4 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.

10.5 Designing the experiment

Of course, as part of your proposal, you will have to design your experiment. To design an experiment we need to complete 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).

10.6 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

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

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

Suppose that our design is to have a copper wire with

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

This would give a resistance of

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

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

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

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

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

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 increase δA. We could try $1.0 \pm 0.1 \times 10^{-9}\text{m}^2$. In order to have the same resistance we will also have to change the length of the wire from 10m to 5m.

$$\begin{aligned}\rho &= 1.68 \pm 0.03 \times 10^{-8}\Omega\text{m} \\ \ell &= 10.0 \pm 0.1\text{m} \\ A &= 1.0 \pm 0.1 \times 10^{-9}\text{m}^2\end{aligned}$$

Checking we see we do get the same resistance

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

But now for the last term we would get

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

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.

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

$$\left(\frac{\rho}{A}\delta\ell\right)^2 = \left(\frac{1.68 \times 10^{-8}\Omega\mathrm{m}}{1.0 \times 10^{-9}\mathrm{m}^2}(0.1\mathrm{m})\right)^2 = 2.8224\Omega^2$$

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. 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 spreadsheet so that as the design changes your uncertainty estimate will change too without having to manually recalculate it.

Chapter 11

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.

11.1 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.

11.2 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.

11.3 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.

11.4 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.

11.5 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:

11.5.1 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.

11.5.2 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 with 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