Chapter 3

First DAQ Measurements: Voltage

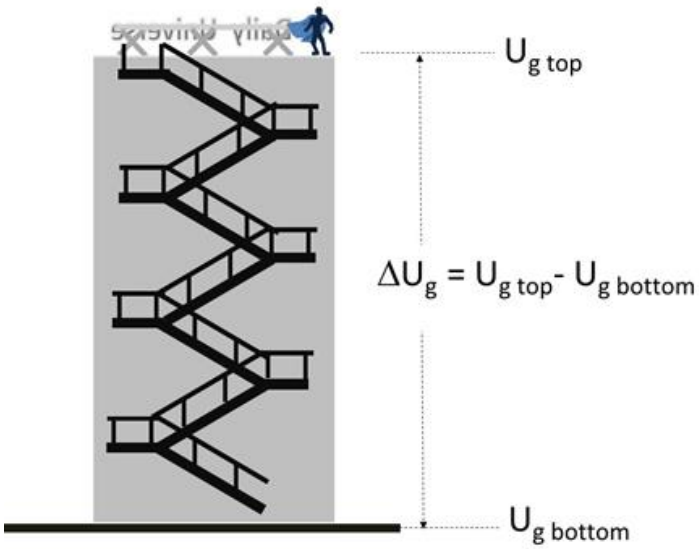
Let's review what we learned last week. In physics equipment, we try to measure voltages. If your data is not a voltage, we try to convert it into a voltage. Already we converted current into a voltage (using a shunt resistor). But what is a voltage? We said last week that it is a measure of electrical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

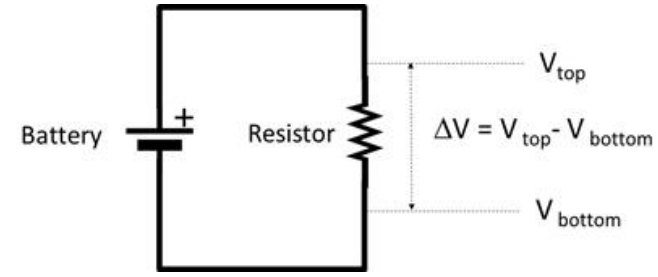
Soon your PH220 class will explain that "voltage" is proportional to the electrical potential energy difference. But for us, now, we just need to know that we are measuring something proportional to energy and we need to learn how to measure it.

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

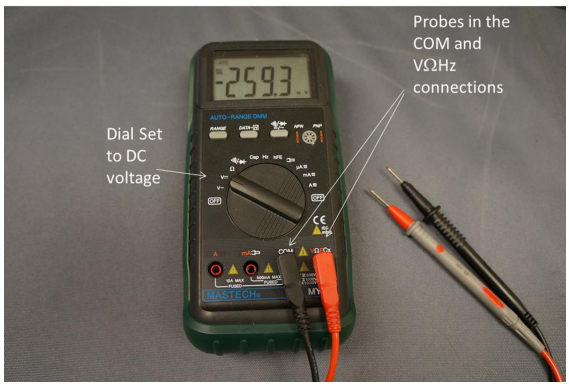
$$\Delta U_g = U_{g_{top}} - U_{g_{bottom}}$$

We will do something very similar in measuring voltages. We will measure the potential energy at two places. For example, suppose we have an electric circuit as shown in the next figure. The circuit is very simple, just a battery and a resistor. You have experience with batteries, and resistors now. A resistor is just a piece of material that has lots of electrical friction, or "resistance" that makes it hard for electrons to go through it. If we want to measure the voltage across the resistor, we have to measure on the top and bottom of the resistor. That will give us a measurement proportional to the potential energy difference from one side to the other of the resistor.





Most meters that measure voltage have two "probes" and do the difference calculation internally. These meters are called *voltmeters* and we used them last week. In today's world, voltmeters are usually just one part of a device



that can measure many things. We call these devices multimeters. To measure voltage we will set the multimeter on the DC Voltage setting. Notice the two probe wires in the figure. We need two probes to make the two measurements and the device does the subtraction for us.

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

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

few of these in this lab.

3.1 Building a voltmeter

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

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

$$4.32768543255325734 57\text{V}$$

and this would be perfectly valid for an analog signal.

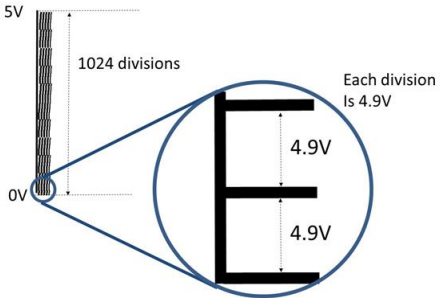
A battery, on the other hand, is not this way. It has a fixed voltage, say, 1.5V like the D-Cell batteries that we used in our last lab. Two D-Cell batteries could be used together to make 3V. But you can’t use D-Cell batteries to get 2.25V. The batteries come in discrete units.

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

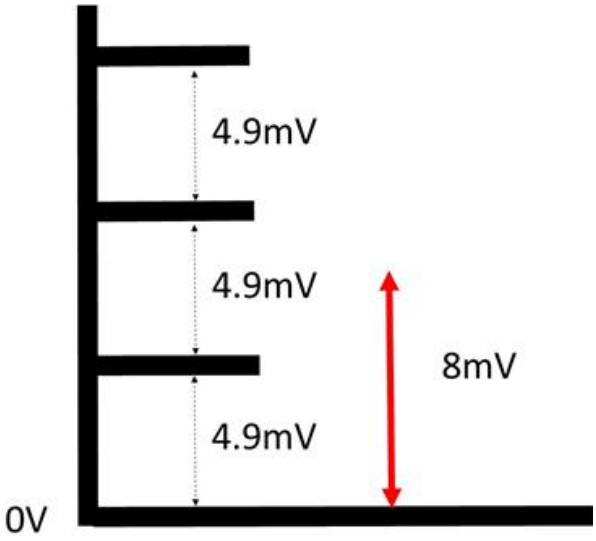
ADC than just a voltage range. The Arduino chops the voltage range into 1024 discrete voltage divisions. Each division is then

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

This means that changes in voltage that are less than 4.9mV won't be seen,



since it takes a whole 4.9mV to get a different division. So if we give our Arduino 8mV this is not enough to fill the second 4.9mV division, so our Arduino would still read only 4.9mV. If we gave it 11mV it would then read 9.8mV because $9.8\text{mV} = 2 \times 4.9\text{mV}$ and 9.8 is the closest whole unit of 4.9mV.This is called



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

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

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

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

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

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

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

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

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

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

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

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

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

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

and in the loop function we use the command

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

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

```
float delta_v_min=0.0049;
```

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

```
int value = 0;
```

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

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

analogRead()

will do this.

The whole Arduino sketch might look like this: Download here

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

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

////////////////////////////////////////////////////////
void loop() {
  // Read in the voltage in A2D units form the serial port
  //   remember that AI0 is the pin number to read from
  ADC_value = analogRead(AI0);
  // Let's print out our A2D version of our signal
  Serial.print("␣A2D␣");
  Serial.print(ADC_value);
  // Now convert to voltage units using delta_v_min
  voltage = ADC_value * delta_v_min;
```

```
  // And print out our voltage version of our signal
  Serial.print("␣voltage␣");
  // Print the voltage with 4 significant figures)
  Serial.println(voltage, 4);
}
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
```

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

3.1.1 Wiring the simple voltmeter

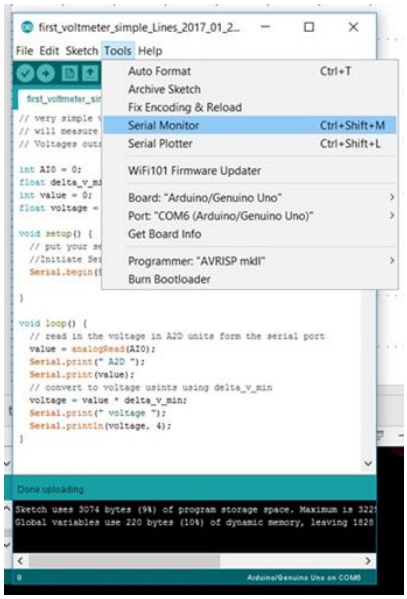
You knew that was coming, didn't you! We must be very careful to wire our Arduino correctly. Our Arduino can measure 0 to 5V. But if we switch the 5V and the 0V by plugging them into the wrong pin, our Arduino will be damaged and will never work the same way again (probably won't work at all!). So wire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

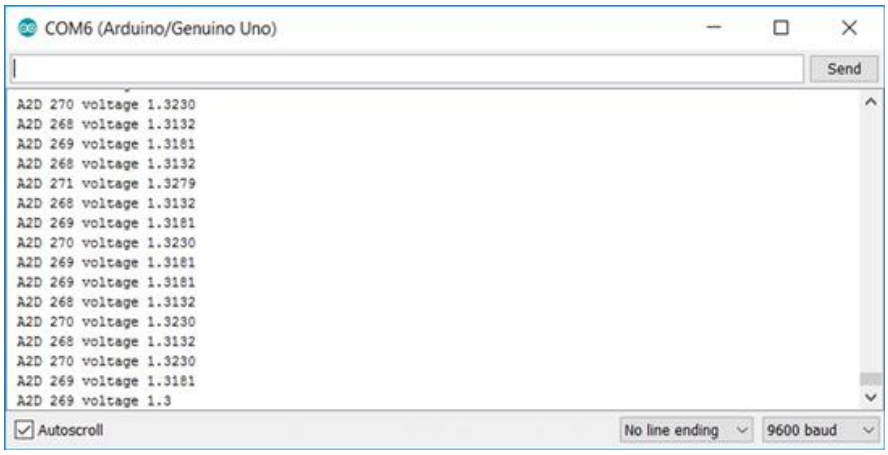
We need one wire attached to the pin marked A0. We need another wire attached to one of our Arduino ground pins marked GND. And we connect the first wire to the positive output of our signal source (say, our power supply) and the GND wire to our negative output of our signal source (say the negative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

3.1.2 Seeing the data

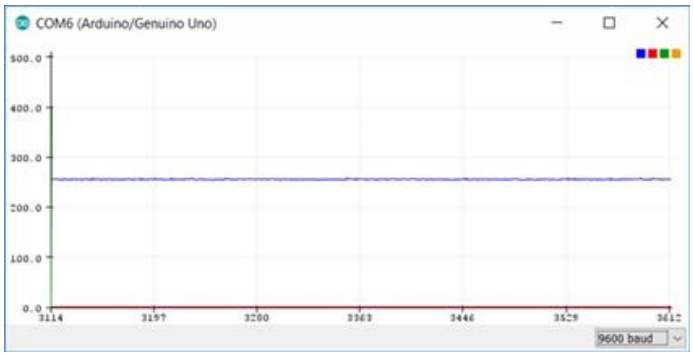
Once the code is compiled and uploaded, the Arduino will send data to the serial port. The serial monitor can display the data. The serial monitor is found under the Arduino Software Tools menu.

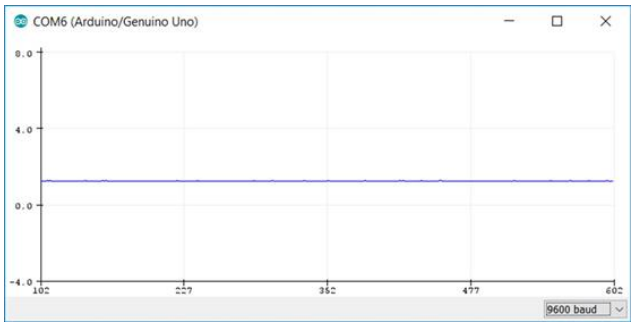
You should see something like this:





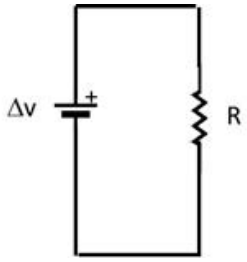
The Arduino Software can also plot the data from the serial port. Here is a plot of the same data that we saw on the serial monitor. Notice that it plotted our voltage values and it also plotted our ADC values. This makes the voltage values hard to see. We could fix this by commenting out the lines that print the ADC values (putting "//" at the beginning of the line). Then those lines won't be executed by the Arduino. Then we get just the voltage. Notice that the horizontal axis is not exactly time. It is just the data point number. We could convert this to time with some calculation if we know how often the Arduino sends us a data point. I will leave this as an exercise.





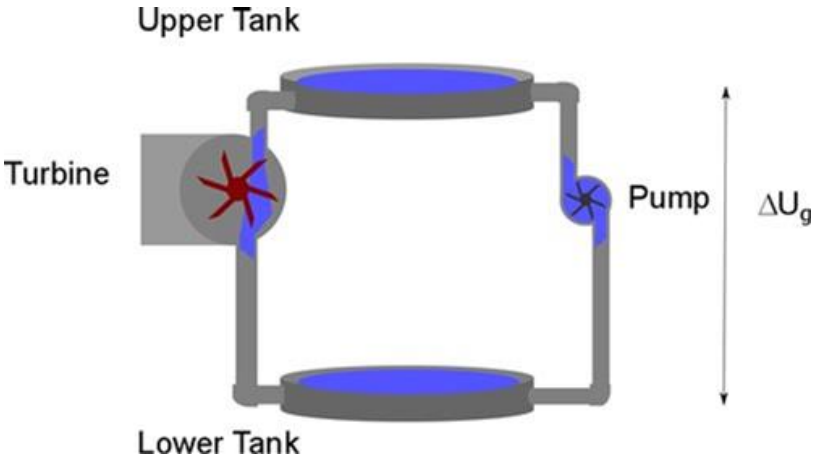
3.2 Extending our voltmeter with a voltage divider

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



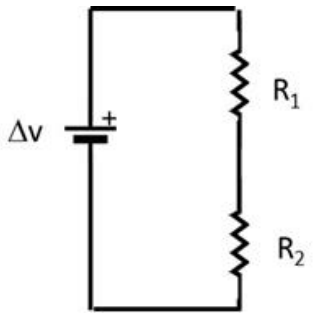
battery, That will make the current flow much like a pump makes water move through pipes.

The water in a pipe system gains potential energy as it moves up. In our circuit we will find that electric charge gains potential energy as we move it

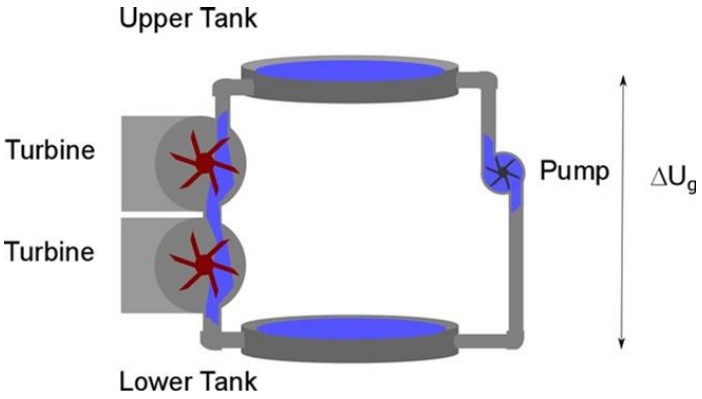


across a battery. Then the charge will move down the wire like water moves down a pipe until it is out of potential energy. Notice that the water in a pipe system will lose all the potential energy that it gained when the pump raised it to the upper tank (see previous figure). That is true of electric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

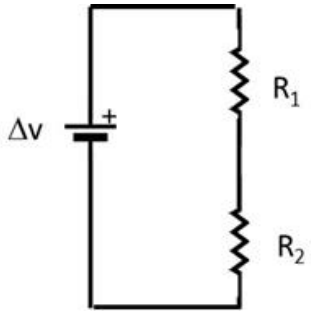
Now suppose we have two resistors in a circuit.



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



second resistor, we can guess that we must be back at the low electric potential we started with.



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

$$\Delta V = RI$$

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

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

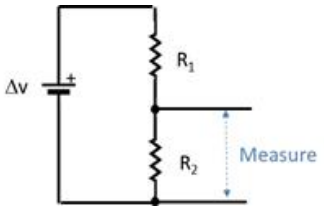
Note that the same current must flow through each of the resistors, since the current leaving R_1 is the current flowing into R_2. Then

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

Our current will be

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

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



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

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

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

$$\Delta V_2 = IR_2$$

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. And we will choose our resistor so that when the total voltage is 20V but the voltage across our resistor is 5V (or less). Since we will know the resistances, we can use a little math to calculate what the total voltage was using the voltage measurement from just one of the resistors.

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

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

Let's try it. We would get

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

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

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

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

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

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

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

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

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. And this is the cost of using a voltage divider to extend our voltage range. For the bigger voltage range we get a bigger uncertainty.

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

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

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

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

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

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

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**. Download here

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

int ADC_value = 0;     // Place to put the A2D values
float voltage = 0.0;  // calculated signal voltage
//mV Arduino's minimum detectable voltage
float delta_v_min = 0.0049

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

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

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

3.3 Practice Problems

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

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

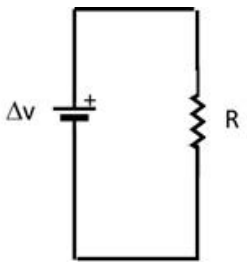
$$\begin{aligned} R_1 &= 43.2\text{k}\Omega \\ R_2 &= 15.2\text{k}\Omega \end{aligned}$$

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

3.4 Lab Assignment

1. Simple Arduino Voltmeter

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



   time write the simple voltmeter sketch, wire it up, and measure the

voltage across the resistor using our Arduino and the serial monitor. Be careful to stay in the 0 to 5V range!

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

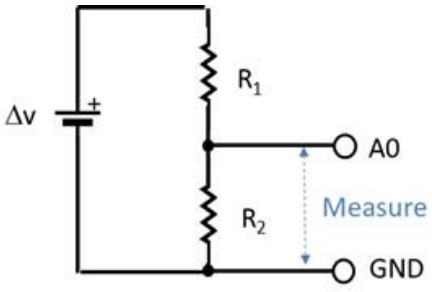
(c) Compare your calculated uncertainty to the measured uncertainty that you see in your device output. (This is tricky, does the power supply give a truly constant voltage?)

2. Extended Voltmeter Using a Voltage Divider

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

(b) Use a multimeter to verify that the output of the voltage divider is never more than 5V and never less than 0V. Take your power supply all the way from 0V to 20V and watch the multimeter to ensure it stays in the 0 to 5V. Range. **Do this with a multimeter before you hook up your Arduino.** You are making sure everything works so you won't destroy your Arduino!

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



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

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

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