



## **Open-source Hardware – Microcontrollers and Physics Education - Integrating DIY Sensors and Data Acquisition with Arduino**

**Mr. Brian Huang, SparkFun Electronics**

Brian Huang is an Education Engineer for SparkFun Electronics, a cutting edge open-source hardware and electronics education company. Brian started his career in engineering with wireless transport technologies for ADC Telecommunications in Minneapolis, MN. While working at ADC, Brian volunteered at the Science Museum of Minnesota and quickly discovered a passion for teaching and working with students - especially in an environment that fostered and supported the "wow" factor associated with inquiry and discovery. In 2007, Brian left the world of engineering to pursue a career in education. For the past 5 years, Brian has taught various levels of high school physics, mathematics, applied technology, and robotics.

Brian joined Sparkfun Electronics to help integrate "tinkering," electronics, and computational thinking into the classroom. One of his goals is to help teachers to de-mystify how household consumer electronics work. With a few simple tools, classrooms can excite and encourage students to explore the possibilities of microcontrollers, electronics, and physical computing.

Brian Huang has a Bachelor's of Science in Electrical Engineering from the University of Illinois, Urbana-Champaign and a Masters in Education from the University of Colorado, Boulder.

# Open-source Hardware -- Microcontrollers and Physics Education - Integrating DIY Sensors and Data Acquisition with Arduino

## Introduction

Leveraging the simplicity and power of the open-source community, we have developed a series of experiments, activities, and labs to investigate, measure, and analyze physics phenomena in the classroom using low-cost microcontrollers and open-source electronics. In the early days, much of the foundations of physics were discovered and analyzed using fairly crude do-it-yourself (DIY) measurement tools like one's own heartbeat or sticks wedged into holes in the ground. We propose a series of activities that recapture this spirit of scientific exploration using low-cost and readily available electronics and equipment available today.

While equipment used in today's research labs are generally fully automated and digitally controlled, many physics labs and classrooms have implements that resemble the tools and techniques of the past more than of the present. In the past two decades, companies like Pasco and Vernier have helped provide classrooms with data acquisition tools and probes to automate measurements and focus on the analysis of data. However, even these tools are sometimes cost prohibitive and sometimes mask key teachable moments for students.

In this paper, we present a series of experiments, activities, and tools that can be integrated into any physics class that leverage an Arduino Uno, a low-cost, 8-bit microcontroller. The Arduino is built upon an open-source framework that encourages people from all communities to share their designs, drawings, and code examples. One of the key advantages of Arduino is that it is simple. Arduino was originally conceived by a team at the Design Institute in Ivrea, Italy for use by a largely non-technical audience of “artists, designers and architects.”<sup>i</sup> The Arduino is a single-board microcontroller that has 20 general purpose input / output (GPIO) pins. With these 20 GPIO pins, the Arduino can be programmed to sense or drive up to 20 unique devices or sensors. While originally intended for artists and designers, we have found that it is an incredible tool for use in the physics classroom. The Arduino microcontroller runs nominally at 16 MHz (16 million cycles/instructions per second or 62.5 ns per instruction) – which means that it can be used for fairly precise timing applications.

In each of the activities outlined in this paper, the software / code is completely open and exposed. Students are provided with the source code and shown how to read and interpret each line of code. This helps to demystify many of the steps in taking measurements, and teaches students how they can process raw data and automate the entire data collection process. And, because there exists a large community around Arduino, students can easily find resources,

support, and accessories that they can utilize to start building their own citizen science experiments beyond the walls of the classroom.

With the ease of use of Arduino, instructors, lecturers, and lab managers can take labs and experiments to the next level. Students will learn to build their own data acquisition devices to model how science is conducted and build a deeper understanding around data measurement, uncertainty, and calibration. This work is inspired by my background in Electrical and Computer Engineering, modeling physics instruction, and work done at Winona State University by Nathan Moore<sup>ii</sup>.

In this paper, all reference code examples are provided as URL links on [codebender.cc](http://codebender.cc).<sup>iii</sup> These examples can be compiled and run directly from a standard web browser using codebender's built-in browser extension or downloaded as original source and compiled locally using the standard Arduino Integrated Development Environment (IDE).

### Learning about measurements and uncertainties

Many physics classes start of the year with a simple data collection activity to encourage students to organize their data, create graphs, draw conclusions, and do some level of statistical analysis around errors and uncertainty. In this first project, students build their own reaction timer. We provide a cursory overview of what an Arduino is, how to connect the circuit, and how to upload the code. While the opportunity exists to discuss concepts around circuitry, voltage, and ohms law -- these are topics that we reserve for future activities. This activity provides an introduction and overview to using Arduino as a tool for scientific investigation.

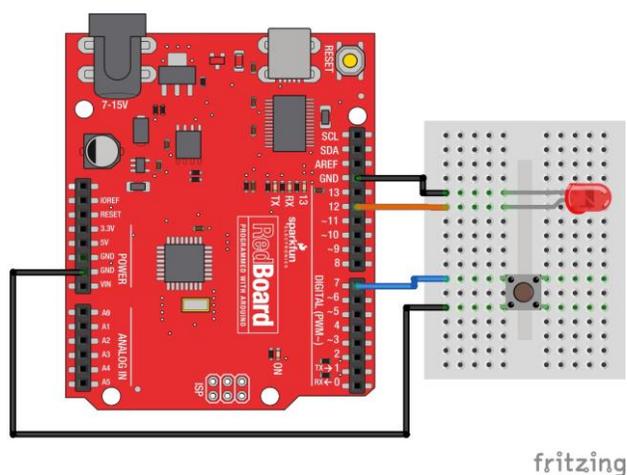


Figure 1 - Wiring Diagram for Arduino Reaction Timer

It should be noted that for simplicity, this circuit does not use a current limiting resistor for the LED nor a pull-up resistor for the button. On the LED, the pins of the Arduino are current

limited to 40 mA. While this is higher than the stated rating on most LEDs, we find that keeping the wiring simple at first is preferred. For the push button, we have the ability in the Arduino programming environment to enable an internal pull-up resistor in code. Again, for simplicity we opt to keep this first project down to two components and four wires. It does not require students to have any prior knowledge of electronics, electricity, or programming.

This activity is inspired by the classic ruler drop / catch activity. In practice, the ruler drop activity often has many undesired errors and uncertainties due to signaling, anticipation, and distractions – leading to large variances in data and difficulty to draw firm conclusions. Using Arduino, we can address many of these sources of error within code

The program logic is simple. To address the signaling and anticipation, the program pauses for a random wait time – in our example this is between 2.000 to 3.500 seconds – before turning on an LED. In response to the LED, the human subject reacts by pressing a button and the Arduino tracks the amount of time, in milliseconds, between the LED turning on and the button being pressed. Finally, this number is reported back to the computer through a serial terminal window (also called a Serial Monitor in Arduino). Students can quickly gather several data points on their own reaction time and look at patterns, measures of central tendency, and develop a sense of comparing and analyzing data.

Code Example:

DIY Reaction Timer -- <https://codebender.cc/sketch:62303>

In this activity, students are not focused on the design of the circuit or of the code. The Arduino serves as a simple tool, and introduces students to the idea that they can construct and program their own tools and equipment. With this activity, students can immediately investigate the nature of their own data and interpret parameters such as: outliers, standard deviation, measures of central tendency, and significant figures.

As a tool, this device allows students to quickly start asking their own questions. Questions posed by students have included: “Does the color of the LED make a difference in reaction time?”, “Is there a difference between gender and reaction time?”, and “Does right-hand / left-hand dominance have an effect on reaction time?” These are opportunities for instructors to introduce and discuss concepts such as sample size, data sets, and other areas of data analysis that lead directly into labs throughout the year.

## **Mechanics**

A traditional mechanics unit starts with an investigation of qualities that describe motion. Often, these labs start with simple things like the motion of a battery powered buggy or a simple wind-

up toy. Many instructors use stop watches and simple timers to have students characterize displacement vs. time and calculate characteristics like average velocity.

When looking at objects that move much faster such as the oscillation of a pendulum, the launch speed of a projectile, or the acceleration of a dropped picket fence, a photogate timer system is needed. Many commercial photogate timer systems exist. These systems cost between \$300 - \$400 per setup. Equipping a standard physics classroom with 8 – 10 stations might cost thousands of dollars. The cost of this setup is less than \$30 per device.

In our first activity, students were introduced to the use of the Arduino as a reaction timer. Using a similar setup and example code, we will setup and build a simple photogate timer in the classroom. A photogate is simply an infrared (IR) transmit / receive pair. The pair creates an IR beam between the two elements, and instead of using a button press, Arduino will trigger the starting and stopping of the timer based on the breaking of this beam.

Again, the focus of this activity is not circuit building nor electronics. The configuration is very simple and involves two components and four wires.

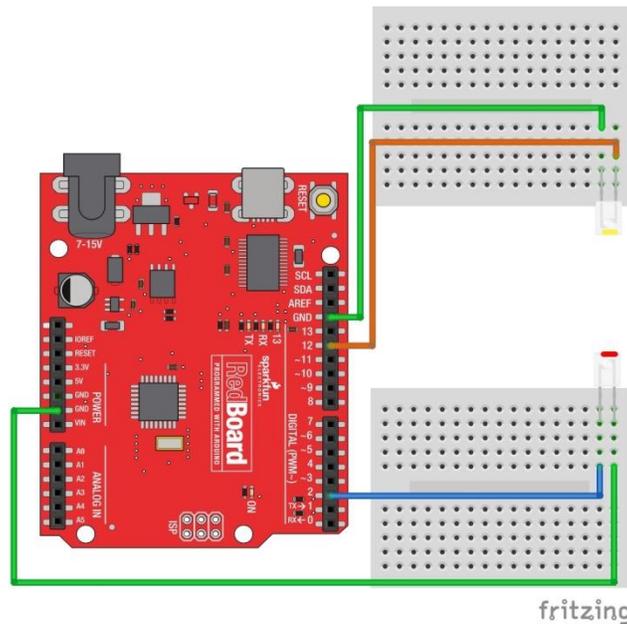


Figure 2 - Wiring Diagram for Arduino Photogate

Code Example:

DIY Photogate Timer -- <https://codebender.cc/sketch:62316>

In this example, the code turns on the IR LED on pin 12. It monitors the input value of the infrared receiver on pin 2. When the input changes, indicating that the IR beam is either blocked or if it was blocked, it is now unblocked, it records the time, in milliseconds. This example code also turns on the debug LED on pin 13 when the beam is unblocked. This gives the user a visual indication that the transmit / receive pair are lined up.

Using this very simple setup, a piece of cardboard to mount the two breadboards, we have a setup that very closely resembles several commercially available photogate elements.

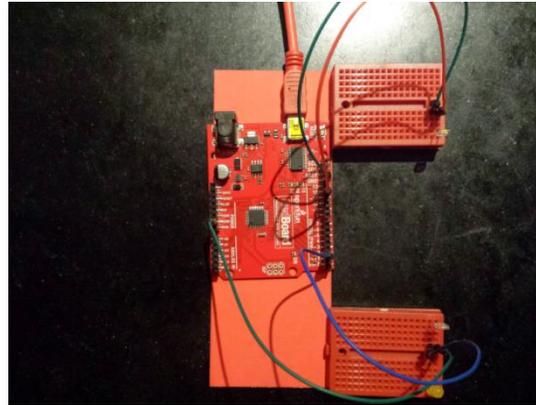


Figure 3 - Completed Cardboard Arduino Photogate Timer



Figure 4 - Commercially available photogate (w/o timer)

In practice, these photogates may be susceptible to stray IR signals from windows and sunlight. If proximity to windows is a problem, it is possible to build a small baffle or cover around the IR receive element to shield it from stray sunlight.

This simple tool can be used for a variety of experiments in the mechanics unit of physics from pendulum timing, dynamics carts, air tracks, free-fall experiments (picket-fence), CO2 cars, or any other timing needs to remove human error / reaction time.

## Heat / Energy / Temperature

In the first two activities, the Arduino is simply used as a tool. A wiring diagram and example code are both provided, and there is no focus on modifying either. These are simply introductions to using Arduino. In this next activity, we look at ways of measuring the average kinetic energy of an object. There are a couple devices that we use to accomplish this. The first is a simple integrated temperature sensor by Analog Devices called the TMP36.

In this activity, students are introduced in more depth to the coding environment of Arduino. Students are told that they have been contracted by a company to integrate this device, TMP36, and characterize the temperature of the lab. Student read the datasheet on the TMP36 which states that it has a voltage output that is relative to the ambient temperature. The sensor has a scale factor equal to 10 mV / degree C and is 750 mV at 25 degrees C. As a homework assignment, students are asked to derive an equation / relationship for converting a voltage to a temperature. Students are provided with one hint:  $y = mx + b$ . This linear relationship is central to a number of key movements in physics education research such as Modeling from David Hestenes at Arizona State University<sup>iv</sup>.

As an introduction to the lab, we work through the derivation in class. After converting units and solving for the y-intercept, students should arrive at an equation which reads:

$$temperature [^{\circ}C] = 100 \frac{^{\circ}C}{V} \cdot (sensor\ voltage [V]) - 50\ ^{\circ}C$$

We now have a short discussion on how the Arduino samples sensor input. The Arduino has a built-in 10 bit Analog to Digital converter (A/D). With 10 bits, it converts a 5V signal to a value of 1023 and a 0V signal to 0. Using this, we show students how to write their own function which converts an input value from the Arduino to a voltage. There is no official name for the units of the A/D converter, but the agreed upon name in our lab has been LSB or least significant bit. Therefore, in following in the modeling methodology, a conversion factor exists: 5.0 V = 1023 LSB.

```
rawAnalogReading = analogRead(TempPin);  
sensorVoltage = rawAnalogReading * 5.0 / 1023;
```

Figure 5 - Code Snippet illustrating the use of converting between LSB (analogRead value) and voltage

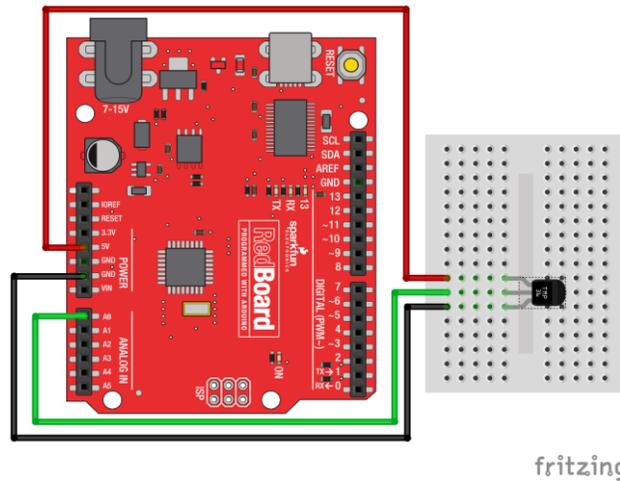


Figure 6 - Wiring Diagram TMP36 Temperature Sensor

Code Example:

DIY TMP36 Sensor -- <https://codebender.cc/sketch:79440>

In their lab groups, the need to use this, combined with the derivation of sensor voltage to temperature to report back the temperature of their sensor.

This activity emphasizes the use of linear relationships, scaling and offset, and slope-intercept models to convert between sensor values to real-world metrics. It also introduces students to basics of programming, variables, and mathematical manipulations in code.

Once the code is working, students often ask how do we “know” the sensor is right. This provides a smooth continuation in a discussion on calibration, errors, and uncertainty. Leveraging on lectures and material from the text, we identify the temperature standards from which the Celsius scale is derived: boiling water and freezing water (ice-water). A zip-lock bag and a few extension wires provide us with a simple, low-cost water proof temperature sensor. Because boiling water will melt most plastic bags, we generally opt for a single calibration point of freezing water and double check it with a standard alcohol or mercury thermometer.

As an extension to this activity, students are asked to use a second method of characterizing temperature using a thermistor. A thermistor is more accurate, but requires the use of a more complicated (non-linear) Steinhart-Hart normalization model. Vernier Software and Technology published a guide on this calibration technique with their own sensor in their *Engineering Projects with NI LabVIEW and Vernier*<sup>v</sup>. An evaluation copy of this chapter can be attained here: [http://www2.vernier.com/sample\\_labs/EPV-01-build\\_temperature\\_sensor.pdf](http://www2.vernier.com/sample_labs/EPV-01-build_temperature_sensor.pdf)

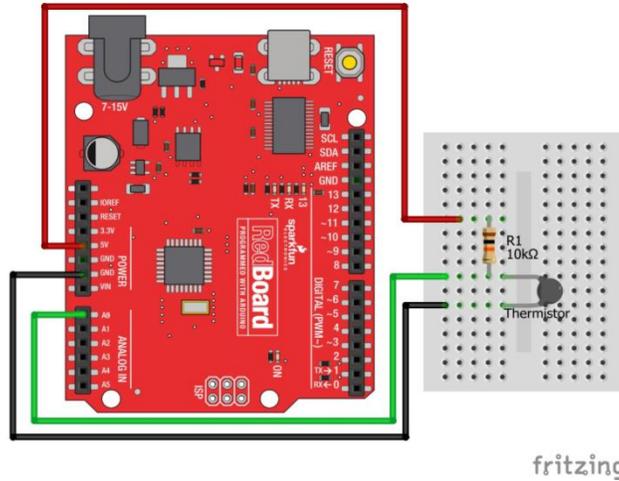


Figure 7 - Wiring Diagram for Arduino Thermistor Circuit

The Steinhart-Hart model matches the resistivity of the thermistor vs. temperature to a logarithmic response curve which fits this equation:

$$T(\text{Kelvin}) = [K_0 + K_1(\ln R_T) + K_2(\ln R_T)^2 + K_3(\ln R_T)^3]^{-1}$$

Figure 8 - Steinhart-Hart Model for thermistor response

Where  $K_0$ ,  $K_1$ , and  $K_2$  are coefficients specific to the thermistor. From the datasheet, the 10K Thermistor used in these labs,  $K_0 = 3.354016 \cdot 10^{-3}$ ;  $K_1 = 2.569850 \cdot 10^{-4}$ ;  $K_2 = 2.620131 \cdot 10^{-6}$ ;  $K_3 = 6.383091 \cdot 10^{-8}$ ;

PARAMETER FOR DETERMINING NOMINAL RESISTANCE VALUES											
NUMBER	$B_{25/05} (K)$	NAME	TOL. B (%)	A	B (K)	C (K <sup>2</sup> )	D (K <sup>3</sup> )	$A_1$	$B_1 (K^{-1})$	$C_1 (K^{-2})$	$D_1 (K^{-3})$
1	2880	Mat O. with Bn = 2880K	3	-9.094	2251.74	229098	-2.744820E+07	3.354016E-03	3.495020E-04	2.095959E-06	4.260615E-07
2	2990	Mat P. with Bn = 3990K	3	-10.2296	2887.62	132336	-2.502510E+07	3.354016E-03	3.415560E-04	4.955455E-06	4.364236E-07
3	3041	Mat Q. with Bn = 3041K	3	-11.1334	3658.73	-102895	5.166520E+05	3.354016E-03	3.349290E-04	3.683843E-06	7.050455E-07
4	3136	Mat R. with Bn = 3136K	3	-12.4493	4702.74	-402687	3.196830E+07	3.354016E-03	3.243880E-04	2.658012E-06	-2.701560E-07
5	3390	Mat S. with Bn = 3390K	3	-12.6814	4391.97	-232807	1.509643E+07	3.354016E-03	2.993410E-04	2.135133E-06	-5.672000E-09
6	3528 (1)	Mat I. with Bn = 3528K	0.5	-12.0596	3687.667	-7617.13	-5.914730E+06	3.354016E-03	2.909670E-04	1.632136E-06	7.192200E-08
	3528 (2)			-21.0704	11903.95	-2504699	2.470338E+08	3.354016E-03	2.933908E-04	3.494314E-06	-7.712690E-07
7	3560	Mat H. with Bn = 3560K	1.5	-13.0723	4190.574	-47158.4	-1.199256E+07	3.354016E-03	2.884193E-04	4.118032E-06	1.786790E-07
8	3740	Mat B. with Bn = 3740K	2	-13.8973	4557.725	-98275	-7.522357E+06	3.354016E-03	2.744032E-04	3.666944E-06	1.375492E-07
9	3977	Mat A. with Bn = 3977K	0.75	-14.6337	4791.842	-115334	-3.730535E+06	3.354016E-03	2.569850E-04	2.620131E-06	6.383091E-08

Figure 9 - Thermistor coefficients from manufacturer datasheet (Vishay NTCLE100E3)

The model is based upon  $R_T$ , the actual resistance across the thermistor.  $R_T$  can be calculated based on manipulating the voltage divider equation. Here is where we introduce our first Ohms

Law discussion of voltage, resistance, and voltage dividers. In this example, we use a 10K pull-up resistor. Therefore,

$$V_{R_T} = \frac{R_T}{R_T + 10K} \cdot V_{CC}$$

$$V_{R_T}(R_T + 10K) = R_T \cdot V_{CC}$$

$$V_{R_T}(10K) = R_T \cdot (V_{CC} - V_{R_T})$$

$$\frac{V_{R_T}(10K)}{(V_{CC} - V_{R_T})} = R_T$$

On the Arduino,  $V_{CC}$  is 5V, and we can manipulate all of these calculations easily in code. The code example is available at the link below.

Code Example:

DIY Thermistor Sensor -- <https://codebender.cc/sketch:62399>

While the code example for using the thermistor does use a more complex scaling model for the sensor, it does illustrate that not all sensors are linear. It also illustrates how we can utilize a computer (or microcontroller) to perform tedious and repetitive calculations and conversions from an analog input reading to a voltage to a resistance to a corresponding temperature.

## **RC Time Constant Measurement**

The final activity utilizing the Arduino is one for characterizing RC time constant. In a traditional lab, this is performed with a time varying square wave which charges and discharges an RC circuit. Students move cursors around on an oscilloscope, write down a few numbers corresponding to 36.8% of  $V_{max}$  or 63.2% of  $V_{max}$ . This is, of course, the same as the quantity  $(1 - e^{-1})$  and  $e^{-1}$ , respectively. Students often graph these relationships using an oscilloscope and see a constantly changing charging and discharging cycle resulting from the square wave.

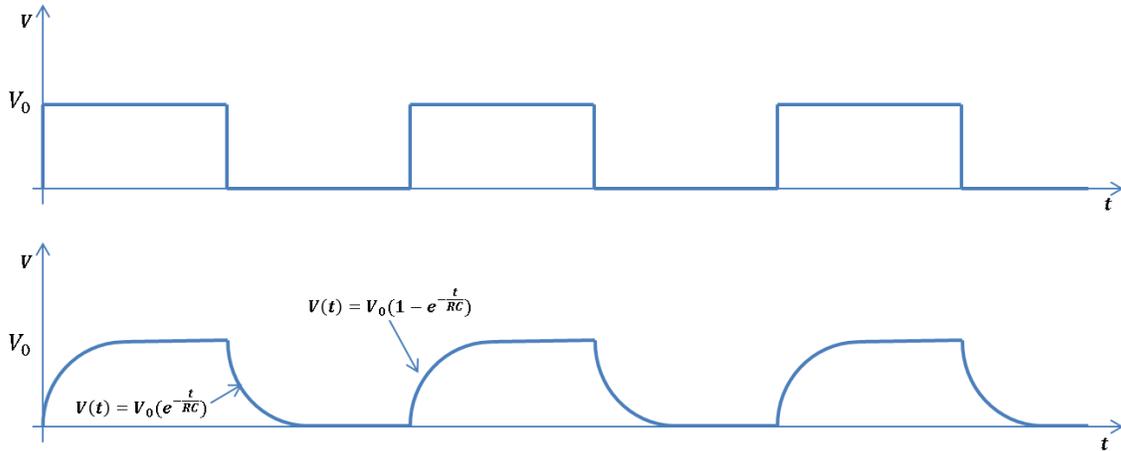


Figure 10 - Typical oscilloscope output plot showing continuous charging and discharging cycles.

$$\text{Charging: } V_{cap} = V_{max} \cdot \left(1 - e^{-\frac{t}{RC}}\right)$$

$$\text{Discharging: } V_{cap} = V_{max} \cdot e^{-\frac{t}{RC}}$$

Figure 11 - Standard voltage model for charging / discharging RC Circuit

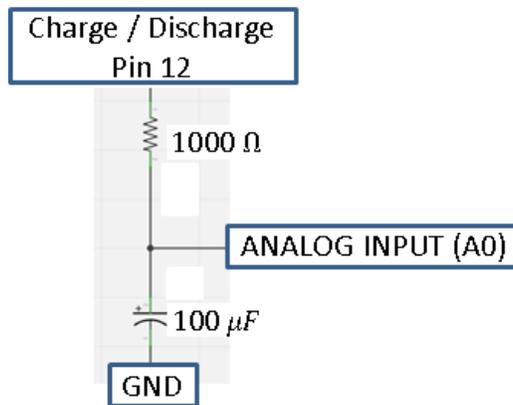


Figure 12 - Wiring Model for Charge / Discharge Circuit

The continuous charge / discharge cycling, the calibration and settings on an oscilloscope, and the multiple traces all add unnecessary confusion to a very simple phenomenon. Applying similar exercises involving the timer and the sequencing in code, we break down this problem into a single charge and a single dis-charge cycle of the circuit. In the code, we are able to charge the circuit and poll the voltage level continuously until it reaches  $V_{max}$  or close to it. Likewise, we can then dis-charge the circuit through the Arduino and read the voltage level continuous

until it discharges down to 0 V. With the speed of the Arduino, we can quickly capture enough data points and graph a single charge / discharge cycle for our circuit.

Students can leverage this tool to quickly investigate other combinations of R and C and apply their analysis using tools such as Microsoft Excel, MatLab, Mathematica, or other graphing and analysis tools. Here, we isolate the problem to a single charge cycle and a single discharge cycle.

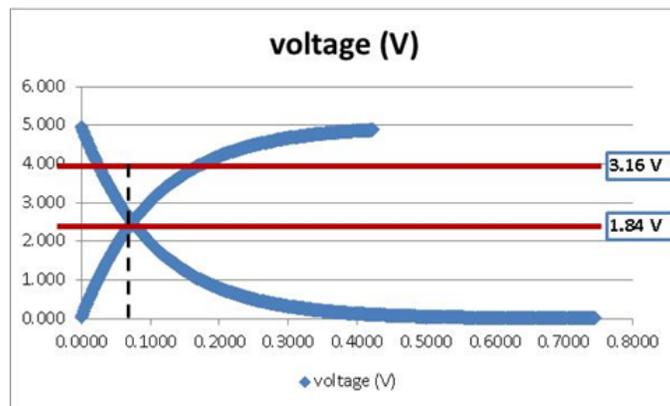


Figure 13 - Example data for Single Charge & Discharge Cycle w/ Annotations

#### Code Example

<https://codebender.cc/sketch:51144>

#### **Conclusion**

Providing students with exposure to Arduino and open-source hardware enhances the physics classroom experience beyond the traditional book problems and traditional demonstration labs. It is a low-cost and effective means to providing accurate and repeatable measurements in the classroom and lab.

Arduino provides a great platform for building high accuracy timing experiments and automating a large number of measurements. The models used in converting sensor voltages from the examples such as the TMP36 or the thermistor can easily be adapted and extended to a large variety of sensors for investigating the nature of the physical world. A majority of the sensors provided by companies such as Vernier are analog sensors. In fact, opening up the configuration files inside their software reveals the calibration scale factor (slope) and offset (intercept) for all of their standard sensors. The code examples provided in this paper can be manipulated and modified for any analog sensor available. This starts students on a path to understanding how to build their own sensor interfaces.

Unlike many pieces of hardware and apparatus in a physics classroom, Arduino is a tool which students can find, purchase, and take home to build their own experiments. It is a piece of

equipment that allows students to extend physics and scientific curiosity beyond the walls of the classroom.

---

<sup>i</sup> Alicia M. Gibb, “New Media Art, Design, And The Arduino Microcontroller: A Malleable Pool” (Master’s Thesis, Pratt Institute, 2010) p7.

<sup>ii</sup> N. Moore and A. Haugen. “Adapting Modeling Instruction to DIY Arduino (Microcontroller) Lab Equipment Development.” AAPT 2014 Summer Meeting: Minneapolis, MN.

<sup>iii</sup> codebender.cc is a free, on-line programming environment for Arduino microcontrollers. It allows for open collaboration and sharing of code examples.

<sup>iv</sup> D. Hestenes (1997), [Modeling Methodology for Physics Teachers](#). In E. Redish & J. Rigden (Eds.) The changing role of the physics department in modern universities, American Institute of Physics Part II. p. 935-957.

<sup>v</sup>Perrin, Michele, Steve Decker, Sam Swartley, and Dave Vernier. *Engineering Projects with NI LabVIEW™ and Vernier*. Beaverton: Vernier Software & Technology, n.d. Print.