Lab 1 – Reading and Recording Data

ME 451 - Introduction to Instrumentation and Measurement Systems, Spring 2019

### Lab Objectives

* Learn about the Arduino ADC.
* Learn how to establish a communication channel with your Arduino using the serial monitor.
* Learn how to read simple sensors (temperature and light sensors) with an Arduino.
* Gain experience writing code for the Arduino UNO.

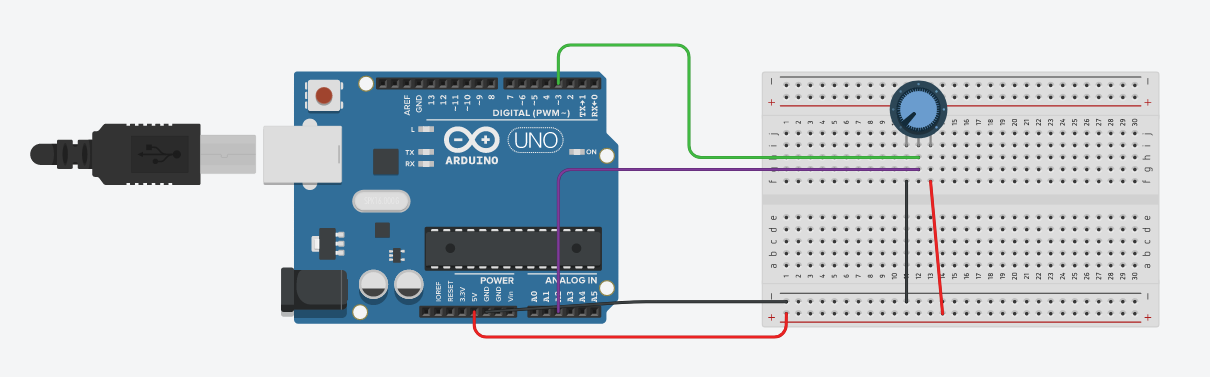
### General Arduino Resources

* Arduino Language Online Reference: [link](https://www.arduino.cc/reference/en/)
* Arduino Code Foundations homepage: [link](https://www.arduino.cc/en/Tutorial/Foundations)
* Arduino Language Book Reference: [link](https://drive.google.com/file/d/1KPhmzgGUUam43ORjgPdoNnn6JnzkZK7U/view?usp=sharing)

**Lab Sensors for report:** Light Sensor, Temperature Sensor. This Lab is a *2 day lab*.

*Note:* We will use MATLAB for this lab. If having the right version of MATLAB is an issue, use [matlab.mathworks.com](http://matlab.mathworks.com)

# Section 1: Reading Voltage

1. Build the potentiometer sensor setup (circuit provided above).
   1. We will connect the potentiometer to both a digital and analog pin.
   2. Build a program that will read both the digital and analog signal on the potentiometer.
   3. In order to communicate with the Arduino about what it sees, we will need to setup serial communications.
      1. Refer to the Serial Monitor section in the *Arduino Code Reference* resource, the function descriptions in the *Arduino Serial Reference*, or the Arduino serial examples ([File] -> [Examples] -> [Any example with serial in name]).
2. **Signoff 1**: Show TA a correctly functioning system. You will need to demonstrate:
   1. A working serial monitor.
   2. That your circuit is correct.
   3. You know how to use a Digital Multimeter (DMM).
3. We will now rotate the potentiometer and record the analog and digital values that the Arduino sees. Use the DMM as a ground truth.
   1. Go in steps of 0.2V up to 5V. Record the analog and digital values that the Arduino sees at each step.
   2. Then go in steps of 0.2V downwards to 0V.
4. When you have collected your data, please move on to Section 2. You can answer the discussion questions on your own time.

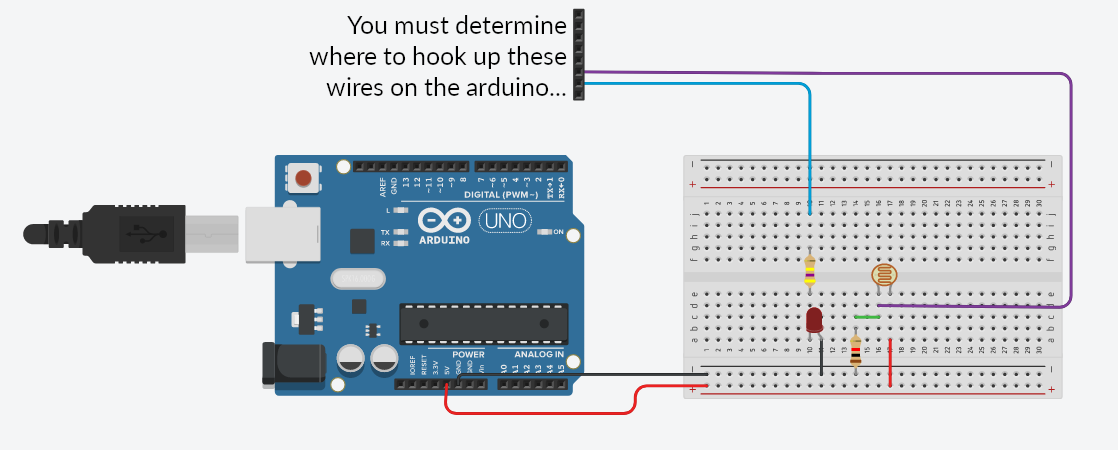
## Section 1 Discussion Questions

**Discussion Question 1:** What does the serial monitor do? What are the two pins called that the serial monitor needs for communication? What are the roles of these two pins? What digital pins do they correspond to on the arduino?

**DQ 2:** Put your results into a table (use this [latex table generator](http://www.tablesgenerator.com/) for the report). What is the voltage level that first registers a digital value of 1 on the way up? Does the voltage switch from 1 to 0 at the same voltage going down? Why? (*Helpful keyword*: logic level)

**DQ 3:** What does the Arduino ADC do? How many bits does the Arduino UNO ADC have? Calculate the voltage values from the ADC values that you collected and put them in a new column on your table.

# Section 2: Automated Data Collection

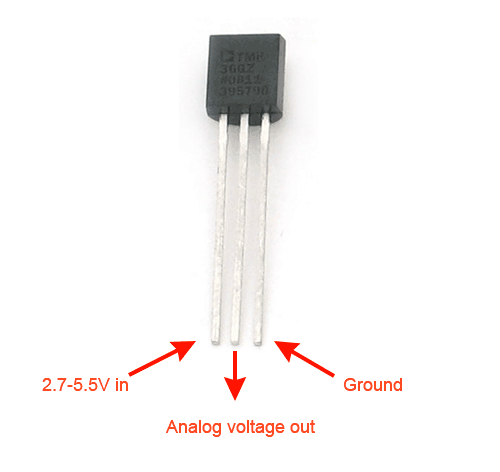


1. Take apart your previous circuit and rebuild the light sensor setup from Lab 0 (circuit provided above).
   1. Make sure to connect the light sensor and LED to functioning pins like in Lab 0.
   2. Modify your code to send the light level sensed by the Arduino to the serial monitor.
   3. Change the delay at the end of loop to 10 milliseconds.
2. We need an external program to record the data that the Arduino sends to the serial monitor. For Windows/Linux computers we will use [Putty](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html). For Mac computers we will use [Coolterm](https://freeware.the-meiers.org/) (it also works on Windows).
   1. Follow these tutorials ([Putty](https://www.youtube.com/watch?v=Cb21y910CKE) (captioned video) or [Coolterm](https://www.youtube.com/watch?v=xqdC__jZILk) (captioned version of video available on Canvas under Media)) or find your own. Provided below are some helpful notes:
      1. You can’t have the serial monitor and Putty/Coolterm recording at the same time. Make sure the Arduino serial monitor is closed when you are using putty/coolterm.
      2. {Putty} In [Session - logging], you can change the name of the file AND the type of file generated. Change the file to the .csv file extension.
      3. {Putty} In [Session], you can save your settings in the [Saved Sessions] section of the page.
      4. {Coolterm} If you are using Coolterm, it will automatically add a timestamp to each line in your csv file. To keep your file simple to use with MATLAB, we suggest disabling that function.
      5. Learn about csv files [here](http://frictionlessdata.io/guides/csv/). There are some changes that you will have to make to the serial.print lines in the code to correctly make a csv file and plot it in MATLAB.
3. When we take data, we will need to make timestamps. This will make us able to accurately relate our data to time.
   1. The simplest way to tell time on an Arduino is with the [millis()](https://www.arduino.cc/reference/en/language/functions/time/millis/) function.
      1. *Note:* you should not use an int variable to collect the millis function’s output. Refer to the code example on the millis function reference sheet for what variable type you should make it.
      2. *Note:* do not name your variable *time* like in the example. Generally any text that is automatically colored orange is something built into arduino, so be wary when using it as a variable name (it can cause problems in later labs). Suggested alternatives are timer, t, or times (although you can do whatever you want!)
   2. Modify your code to output the time elapsed next to your data reading in the serial monitor. Popular convention is to put the timestamp first in your serial monitor. Please follow popular convention.
      1. *Note:* remember to separate your data into separate columns!
4. **Signoff 2:** Show your TA that you have a working system. You will need to demonstrate:
   1. A correct circuit.
   2. You have correct code: reading the light sensor on an analog pin, correct usage of the millis function, good variable type definitions, delay of 10 milliseconds.
   3. You have Putty/Coolterm installed. You must demonstrate that you can record data to your TA.
5. Now it is time to record data and time together. We will have you record light sensor data from a strobe light setup found at the TA desk.
   1. Collect data for one minute with the strobe light on, on the low setting.

## Section 2 Discussion Questions

**DQ 4:** Plot your strobe light data. Using MATLAB, calculate the frequency of the strobe light by looking at the peaks and time. Describe your frequency calculation method, and describe any special functions you made/used.

# Section 3: Using Documentation to Read Data Better



1. Now we will switch sensors to the temperature sensor (TMP36 by Analog Devices).
   1. We will not give you a circuit diagram for this sensor, but we will provide the diagram above that will tell you what each pin does on the sensor. You will need to hook it up on your own. You will not need any extra components, like resistors.
2. You are also on your own to write code to read from the temperature sensor. Use your code from Section 2 as a base!
3. Getting the sensor voltage isn’t entirely useful for sensors. For this temperature sensor, we will need to add a conversion from voltage to temperature to your code. We will use the sensor’s datasheet to get the information necessary for this. You will need to search Google to find the correct datasheet yourself.
   1. Find the voltage-temperature reference point and the resolution for our sensor in the datasheet.
   2. Using this information, make a y = mx + b equation (with x as voltage, y as temperature) that will go from voltage to degrees celsius.
4. **Signoff 3:** A correctly configured and working system.
   1. Correct y = mx + b code, outputs data in units of celcius to the serial monitor.
   2. Temperature sensor circuit is correct.
   3. The rest of your code is correct.
5. It’s time to record some data! We will collect data to see how the sensor warms up and cools down. Don’t forget to use Putty/Coolterm to collect the data!
   1. Measure the ambient temperature at your desk for 30 seconds.
   2. *While still recording your data*, put the temp sensor into the heater’s path (the TA will set the temperature) for 3 minutes.
   3. Still keep your data running! Put the sensor back in the ambient temperature at your desk until your sensor goes back to reading the ambient temperature.
6. You’re done with Lab 1.
   1. Make sure to get your Signoff 4 (the Lab 0 signoff)!
   2. Make sure to clean up your space.

## Section 3 Discussion Questions

**DQ 5:** What is the rise time? How do you calculate it (use specifics)?

**DQ 6:** Plot your temperature sensor data. Qualitatively (meaning you don’t need to use equations, just look at the plot), what is the rise and fall time of your sensor? Is it the same for rising/falling? Why would this be so?

**DQ 7:** What is the time constant? Why is it important for sensors? What does it tell us?

# Post-Lab Questions

**Post-Lab Question 1:** Describe the process that you had to go through to record data using your Arduino and Putty/Coolterm?

**PLQ 2:** What is the baud rate in the serial port? What does 9600 baud (for Serial.begin(9600) in your code) mean? What is the highest baud rate that you can set with the Serial.begin command? In theory, what is the fastest baud rate the Arduino could put out?

### 

### Group Names:

(in pen)

# Lab 1 Signoffs

1. \_\_\_\_\_\_\_ A working potentiometer reading circuit with serial monitor implementation.
2. \_\_\_\_\_\_\_ A working light sensor circuit with working putty/coolterm installation.
3. \_\_\_\_\_\_\_ A working temperature sensor circuit and code that outputs data in degrees celsius, not voltage.
4. \_\_\_\_\_\_\_ Completed Lab 0.

**TA Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Date**:\_\_\_\_\_\_\_\_\_\_\_