ME 210 Winter 2026: Laboratory Assignment 1

Pre-Lab due at 11:59 pm on January 21, 2026.
Lab due at 11:59 pm on January 29, 2026.

Purpose:

This lab is intended to acquaint you with the behavior of signals from sensors, operational amplifiers and comparators. In addition, you will gain experience using the proto-board to try out circuits and to build physical circuits from a schematic. Finally, you will explore techniques to capture the responses from these circuits using a microcontroller.

Minimum Parts Required:

1 each: Arduino Uno, visible LED, LTE-5208A IR LED, LTR-3208E photo-transistor, LM339 comparator & MCP6294 op-amp. Assorted resistors and capacitors. See the TAs for any parts you need that are not in the parts kit.

References:

Our book, Introduction to Mechatronic Design, is an excellent reference for the materials, devices and methods needed for this lab. **We HIGHLY encourage you to acquire a copy of this book and read it!!**

The library has copies of *The Art of Electronics* by Horowitz & Hill—one of the best books on electronics. They also have several of the sources listed in the appendices, and most are also available on the Web.

Semiconductor manufacturers' websites are excellent sources of information about electronic components (e.g., selector guides, data sheets) and how to use them (e.g., application notes). The articles, and often the advertisements, are very good information sources. Take a few minutes and scan the materials provided by Freescale, National Semiconductor, Maxim Integrated Products, Texas Instruments, Allegro, and any other familiar company. The articles and advertisements will begin to make more sense to you as the quarter progresses. This will be a subject of many lectures this quarter!

Part 0: Pre-Lab

Reading:

Textbook Chapter 9, *Basic Circuit Analysis and Passive Components*

Online Digi-Key Scheme-it, Autodesk EAGLE, KiCad, Fritzing and other online circuit drawing and simulation tools.

Online The KiCad tutorial here: spdldaemon.github.io/docs/tools/kicad/

Background:

Complete the following exercises *after* you have read through the lab assignment and readings, and *before* coming into the lab to complete it.

Assignment:

Visit https://www.kicad.org/download/ to download KiCad for installation to your own computer. KiCad is free (both as in freedom and as in cost) and open source, and is available for Windows, Mac, and many flavors of Linux.

Read and work through the KiCad tutorial (link above). There are also several tutorials made by KiCad and by third parties linked at https://docs.kicad.org/#_getting_started that you may find to be helpful when learning KiCad. You will be using the PlatformIO IDE again, this time choosing "Arduino Uno" as the target board.

□ **0.1)** What are the values of resistors with the following color codes:

a)	red	red	red
b)	brown	black	red
c)	yellow	violet	orange
d)	brown	black	green

□ **0.2)** What are the values of the capacitors with the following designations?

a) 223

b) 476

□ **0.3)** Draw a schematic of the configuration you will use for Part 4 using KiCad.

□ **0.4)** Write, compile, and upload a program that outputs "Hello, World!" to the serial terminal once every second on the Arduino.

In the Report:

Include your answers to the exercises in the Pre-Lab.

Part 1: The Arduino as a Controllable Signal Source

Reading:

Online Arduino Uno R3 documentation: https://docs.arduino.cc/hardware/uno-rev3

Online `TimerInterrupt` documentation at https://github.com/khoih-prog/TimerInterrupt

Background:

In this lab, we will be using a more precise timing mechanism than the `Metro` library. The`TimerInterrupt` calls a provided function handle based on timer interrupts, rather than by polling the Arduino `millis()` function for timer expiration. What this means is that your function will execute at the defined interval, (without being slowed down by other functions), but you will need to consider the implications of interrupting your code and accessing/changing variables at any point.

Read through the `TimerInterrupt` README and pay particular attention to the sample code in section 1.3 *Set Hardware Timer Frequency and attach Timer Interrupt Handler function*.

We are now using the Arduino Uno microcontroller (and will be for the rest of the quarter). Read through the documentation at the above links to familiarize yourself with the specifications. Arduino Uno is a 5V microcontroller. Applying more than the rated voltage can damage your Arduino!

Assignment:

Complete the following exercises:

- ☐ **1.1)** You can use either the supplied prototyping shield for Arduino, or the independent proto-board (breadboard). We will be powering this entire lab using the Arduino's onboard 5V supply, which in turn draws from the 5V on the USB input. Use an oscilloscope to verify that the rails are at the correct voltages. Read the Arduino Uno documentation to verify that you know which pins correspond to +5V and GND supply rails.
- ☐ **1.2)** Write code to toggle an output at 1.25kHz. Examine your output with an oscilloscope and verify that you have the correct frequency.
- ☐ **1.3)** We will now modify the signal generation code to allow for variable frequency control. Connect the two outer leads from the 10kΩ potentiometer to the 5V and GND terminals. Verify that the voltage at the wiper (the center terminal on the potentiometer) swings between 0V and 5V as you rotate the knob. On your proto-board, connect the wiper to any of the analog pins on the Arduino. You could have built a unity gain buffer for this signal, using the MCP6294 from your lab kit. Why would you not want to do this? What limitation would this impose?
- ☐ **1.4)** Update your code to modulate the frequency of your output from 50Hz to 12.5kHz as a function of the potentiometer output. You may find the `map()` and `setFrequency` functions to be helpful for this.
- ☐**1.5)** We'll now use our interrupt timer to drive an IRLED. Connect the IR LED in series with an appropriately sized load resistor in direct drive from the Arduino. Set this up at one end of your protoboard. What resistor sizing did you choose, and why is this important? **Double check this setup before you turn it on, so that you don't damage your Arduino!**

- □ **1.6)** Examine the signal between the IR LED and the load resistor. Capture the waveform on the oscilloscope either using the scope's built-in capture tools, or by photographing the screen at both ends of the previously specified frequency range. Make sure the scope settings and measurements are visible on your captures.
- □ **1.7)** Set the frequency back to 1.25kHz for the rest of the lab; modulate the frequency at each step and use the oscilloscope to observe the effects on your circuit response.

In the Report:

Include the answers to the questions in parts 1.3 and 1.5, as well as the captured waveform from part 1.6.

Part 2: The Phototransistor

Reading:

Textbook Chapter 13, §13.5.1.2, *Phototransistors*

Assignment:

Complete the following exercises:

- ☐ **2.1)** Hook up the phototransistor in a sourcing configuration (collector to 5V) with a 1kΩ load resistor to ground. The phototransistor should be placed on the opposite end of the proto-board from the IR LED (along long axis). Aim the phototransistor at the IR LED (the sensitive region, and the emitter of the LED are opposite the leads).
- ☐ **2.2)** Draw a schematic in KiCad of the configuration in part 2.1.
- ☐ **2.3)** Using an oscilloscope, look at the output of the photo-transistor. Capture the waveform as you did in the previous part of the lab.
- ☐ **2.4)** Hook up the phototransistor in a sinking configuration (emitter to ground) with a 1 kΩ load resistor to power. The phototransistor should again be placed on the opposite end of the proto-board from the IR LED (along the long axis). Aim the phototransistor at the IR LED (the sensitive region and the emitter of the LED are opposite the leads and have narrow emission/detection angles!).
- ☐ **2.5)** Draw a schematic in KiCad of the configuration in part 2.4.
- ☐ **2.6)** Using an oscilloscope, look at the output of the phototransistor. Capture the waveform.
- ☐ **2.7)** Explain the differences between the two captured waveforms in parts 2.3 and 2.6.
- ☐ **2.8)** Replace the 1kΩ load resistor with a 10kΩ load resistor and return the circuit to the sourcing configuration.
- ☐ **2.9)** Capture the oscilloscope waveform of the output of the phototransistor. What is the impact of changing the load resistor to 10kΩ?
- ☐ **2.10)** Replace the 10kΩ load resistor with a 100Ω load resistor.
- ☐ **2.11)** Capture the oscilloscope waveform of the output of the phototransistor. What is the impact of changing the load resistor to 100Ω?

In the Report:

Include the captured waveforms from parts 2.3, 2.6, 2.9 and 2.11, the schematic diagrams from parts 2.2 and 2.5, and your answers to questions in parts 2.7, 2.9 and 2.11.

Part 3: Op-Amps

Reading:

Textbook Chapter 11, *Operational Amplifiers*

Assignment:

Complete the following exercises:

- ☐ **3.1)** Draw a schematic in Kicad of a non-inverting op-amp configuration with a gain of about 10. The schematic should be complete with component values, based on the parts you have in your kits.
- ☐ **3.2)** Construct the circuit you designed in part 3.1 on your proto-board.
- ☐ **3.3)** Connect the output of the phototransistor from part 2.8 to the op-amp input. Capture the waveform at the op-amp output.
- ☐ **3.4)** Reverse the polarity of the phototransistor (switch the direction it plugs into your proto-board). What happens to the amplitude of the signal at the output of the op-amp?
- ☐ **3.5)** Return the phototransistor to its previous orientation. Using an oscilloscope, measure the output of the op-amp. What is the measured gain? Does it agree with the calculated gain for your design? If not, explain the deviation(s).[1]
- ☐ **3.6)** Describe the oscilloscope settings you used to make the measurements in part 3.5. Why is the configuration you chose the best configuration for making this measurement?

In the Report:

Include the schematic from part 3.1, the waveform from part 3.3, and the answers to the questions in parts 3.4 to 3.6.

[1] Hint: think about tolerance and how they affect your circuit.

Part 4: Trans-resistive Amplifiers

Reading:

Textbook Chapter 11, §11.4.7, *Operational Amplifiers*

Background:

In the previous section, we interpreted the phototransistor's output by amplifying the voltage drop caused by the current across the phototransistor. However, we can also use a transresistive (also called transimpedance) amplifying configuration to more effectively measure the output of current-producing components (e.g. phototransistors). The transresistive configuration allows the current to be delivered to a constant voltage independent of the amount of current, which gives the most linear response from the phototransistor. Make sure you understand the circuit design before attempting this section.

Assignment:

Complete the following exercises:

- ☐ **4.1)** Using the example in the textbook from Chapter 11.4.7, draw a schematic in KiCad of a transresistive amplifier circuit with component values. In the provided example, both positive and negative supply voltages are used. In our case, we do not have negative voltages, so connect your op-amp supply voltages to 5V and GND, and instead provide a 2.5V reference voltage to the noninverting input. Use the LTR-3208E datasheet to size a gain resistor (you can test and change this later on). Based on the datasheet, why might a V_{CE} of 2.5V not be ideal for the LTR-3208E?

- ☐ **4.2)** What would you expect to be the output of your transresistive amplifier to be at 1mW/cm^2? of irradiance on the phototransistor? What about at 5mW/cm^2?

- ☐ **4.3)** Construct the circuit you designed in part 4.1 on your proto-board.

- ☐ **4.4)** Capture the waveform at the op-amp output, and mark on the captured image which parts of the waveform represent incident light on the photodiode, and the lack of light, respectively. How does this differ from the output in Part 3? How could we make the signal amplitude (the difference between the minimum and maximum output voltages) larger?

In the Report:

Include the schematic from part 4.1, the waveform from part 4.4, and the answers to the questions in parts 4.2 and 4.4.

Part 5: Comparators

Reading:

Textbook Chapter 11, §11.5, *The Comparator*

Canvas *Comparators* handout, provided with Lab 1 materials

Assignment:

Complete the following exercises:

- ☐ **5.1)** Design a comparator in an inverting configuration with hysteresis. Shoot for trip points at approximately 2.4V and 2.6V. Use the comparators handout for reference, and approximate component values with ones you can build from your lab kit
- ☐ **5.2)** Draw a schematic in KiCad of the desired configuration, complete with component values.
- ☐ **5.3)** Connect the wiper of the potentiometer you used in Part 1 to the input of the comparator. Observe both nodes with an oscilloscope as you modulate the voltage between 0V and 5V. Empirically, what are the trip points for your circuit?
- ☐ **5.4)** Connect the output of your op-amp circuit from Part 4 to the input of the comparator.
- ☐ **5.5)** Using an oscilloscope in dual-trace mode, observe the output of the op-amp and the output of the comparator. Capture the dual waveform.

In the Report:

Include the schematic from part 5.2, the waveform from part 5.5, and your answer to the question in part 5.3.

Part 6: Capturing the Output

Reading:

Online `AttachInterrupt()` documentation at:

https://docs.arduino.cc/language-reference/en/functions/external-interrupts/attachInterrupt/

Online Nick Gammon's interrupt notes at gammon.com.au/interrupts

Background:

We will now use pin change interrupts on the Arduino to calculate the exact frequency of our detected signal. Read the linked documentation above for `attachInterrupt()` to understand the considerations necessary around interrupt programming. Of particular note is the necessity for fast-executing code (no `Serial.print()`!) and careful management of variable access/modification.

It is also important to recognize that the Arduino environment abstracts away several key details of interrupt handling (i.e. clearing flags, setting priorities) that are still important to understand, and may require more detailed examination for debugging later on.

Assignment:

Complete the following exercises:

- ☐ **6.1)** Connect your signal to any of the digital pins on the Arduino, and define that pin as the constant `PIN_SIGNAL_IN`.
- ☐ **6.2)** Write a function called `CountFallingEdges()` that increments a counter. This will be part of an interrupt service routine (ISR) to be triggered whenever a falling edge is detected on a pin.
- ☐ **6.3)** Attach your ISR by calling `attachInterrupt(digitalPinToInterrupt(PIN_SIGNAL_IN), CountFallingEdges, FALLING)` within your setup function. This Arduino-defined function does much of the interrupt vector handling for you.
- ☐ **6.4)** Now we want to calculate our detected frequency. Write a function that calculates the frequency in Hz based given the number of edge counts and a time interval (i.e. number of edges in 0.5s).
- ☐ **6.5)** Periodically print out your calculated frequency to the Serial Monitor, and compare this value to the one detected by the oscilloscope—these should be very close to each other! Modulate your frequency and verify that both continue to display correctly. Check this off with a TA, and you're done with Lab 1!

In the Report:

Include a listing of your code for parts 6.2 to 6.4.

Debugging Tips for Lab 1

This lab can be much harder to debug than Lab 0—start early!

Power-Related Issues:

- The power and ground rails on either side of many breadboards are not internally connected. You have to add jumper wires to connect them externally. Be sure to check.

Noise-Related Issues:

- Check that everything you're using is connected on a common ground line—scope, Arduino, ICs.
- Check that all of your connections are properly made, and NOT loose—this can result in 'wavy' signals rather than square ones.
- Wavy signals can also be caused by stray IR radiation—you can try isolating your phototransistor from other light sources to test this.
- Bypass capacitors are essential for your power and ground rails to help isolate noise.

General Debugging:

- When you approach a TA (or another student) with electronics questions, make sure you have your schematic in hand—it will make it much easier to quickly understand your circuit.
- Make sure all the component values that you think you have are the ones that you actually have—it can be very easy to misread orange for red, and then your circuits may not work as intended or at all.
- Trace your steps back from the power source in your circuit, then each pin on your ICs, etc. Once you have verified all components in one section of your circuit, move onto the next. Use a scope and probe each of the pins/locations in your circuit, and make sure they are exhibiting their expected behaviors.
- The IR LED and phototransistors can be a little finicky to get working—make sure they are pointing at each other to get a strong signal!
- Make sure you are using the LM339, not the MCP6294, for the comparator portion of the lab!
- Please ask other students for help in the lab! Especially when debugging electronics, it can be incredibly useful to have a fresh set of eyes on a circuit you've been staring at for hours.