MAE106 Laboratory Exercises
Lab # 1 - Laboratory tools

University of California, Irvine
Department of Mechanical and Aerospace Engineering

Goals

To learn how to use the oscilloscope, function generator, breadboard, and potentiometer.
To learn how to use the Arduino microcontroller board.

Parts & equipment

Qty	**Part/Equipment**
1	Breadboard
1	Potentiometer
Various	Wire
1	Function generator
1	Oscilloscope
1	Multimeter
1	Seeduino board (Arduino)

Introduction

The oscilloscope and function generator are useful tools for making measurements and debugging machines. The solderless breadboard is useful for building circuits. Potentiometers are common circuit elements to control voltages and measure rotations.

The Arduino board is a very useful microcontroller that provides an easy setup to control electromechanical systems.

Note: When making electrical circuits in lab, a mistake in wiring may result in a component getting "fried". If you smell something burning, immediately turn off your power supply and debug your circuit.

Part I: The oscilloscope and function generator (45 min.)

When you a building device, sometimes you want to be able to apply certain input voltages to them. A function generator is a device that produces voltage waveforms such as sine, square, and triangle waves, all with variable amplitude, frequency, and offset. For example, the function generator can produce a voltage with the form:

$$v(t) = v_{offset} + a * sin(\omega t) \qquad \text{Equation 1}$$

where the amplitude (a), the frequency (ω), and the offset (v_{offset}) are all adjustable.

When you build an electromechanical machine, such as a robot, you need to be able to measure the voltages the device sends to different places so that you can debug your design. A very useful tool for this purpose is an oscilloscope, which allows you to measure and view voltage as a function of time.

Reading a Signal Using an Oscilloscope

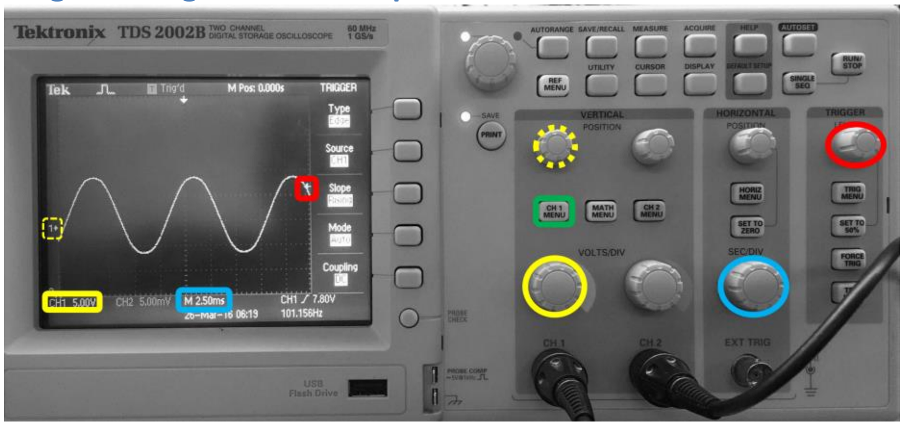

Fig. 1: Oscilloscope. Important control knobs and on-screen information

1. Make sure that the oscilloscope CH 1 probe is connected to the output you want to read and that the alligator clip is connected to ground. The multiplier on the oscilloscope probe should be at the 1x setting.

2. Press the CH 1 MENU button (Fig. 1, green) and make sure that Coupling is set to DC.

3. Adjust sec/div (Fig. 1 blue) – each horizontal line on the displayed screen represents the sec/div you selected, channel 1 volts/div (Fig. 1 yellow) – each vertical line on the displayed screen represents the volts/div you selected, and channel 1 vertical position (Fig. 1 yellow), until you can see the yellow line (channel 1 signal) on screen.

4. Adjust the trigger level until it crosses the signal on screen. The trigger level is shown by the little arrow on the right of the screen. Refine the scale and positioning settings until the signal is shown as desired. Do not use the run/stop button when your signal is scrolling vertically on the screen. Setting the appropriate trigger level will fix that issue and give you extra information about the signal.

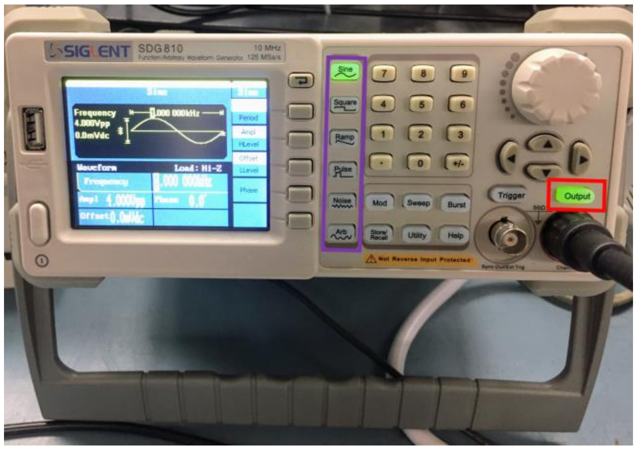
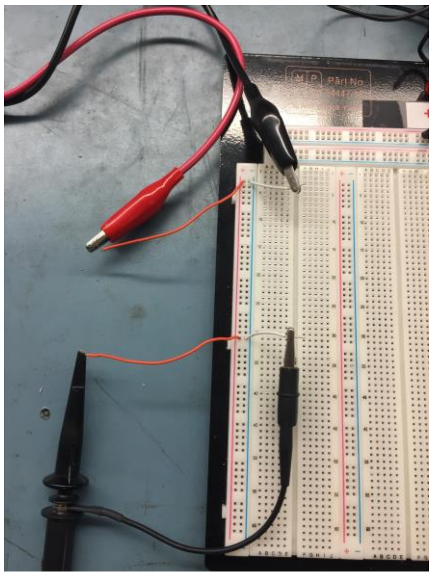

Fig. 2: Function generator on the left and connection between function generator output and oscilloscope CH1 probe (alligator clip goes to GND)

5. You can now read the signal by using the information on the screen. The yellow arrow on the left of the screen corresponds to the ground level. The reading for 'CH1' at the bottom of the screen corresponds to the number of Volts between horizontal lines. 'M' corresponds to the amount of time between vertical lines. Using this information, you should be able to extract the main features of the signal (peak to peak voltage, frequency, DC offset, etc).

Practical Exam # 1

Use the function generator to create a 100Hz sine wave with $a = 1$-volt amplitude and 2 volts DC offset. Using Equation 1, write down the mathematical formula for this wave and label its components (amplitude, offset, and frequency). Finally, display this wave in the oscilloscope and show it to your TA along with the mathematical formula (with labels) describing the wave.

A Word about Breadboards

You will use an electronic breadboard (solderless breadboard) to wire up temporary circuits as you build different devices you design. Electronic components and wires are inserted into the numerous sockets (holes) on the board. The sockets (dots) are connected internally (lines) as shown in Figure 2. A good method for wiring complicated circuits is to connect the source voltage (+5V, ±15 V) and ground terminals from the trainer kit to the long narrow horizontal strips (Figure 2). The electronic chips you place on the board now have ready access to power through short wires to sockets along the long strips. After wiring a circuit to the solderless board, the oscilloscope is useful for measuring voltages at various points on the circuit, by using one of the scope probes.

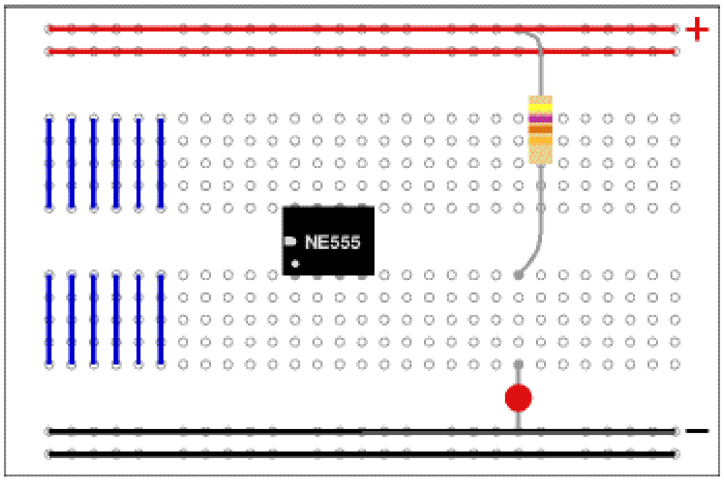

Fig.3 Solderless Breadboard

Part II: Potentiometers (45 min.)

A potentiometer (also called *pot*) is a device that you can use in your machine designs to measure a mechanical rotation, because a pot outputs a voltage proportional to the rotation of its shaft. This is useful for different design activities, such as for sensing the angle of a robotic arm joint or a wheel attached to a motor shaft. You can also use a pot to set the voltage input into a circuit (for example, to allow you to adjust the loudness of a radio or the control gains for a robotic controller.

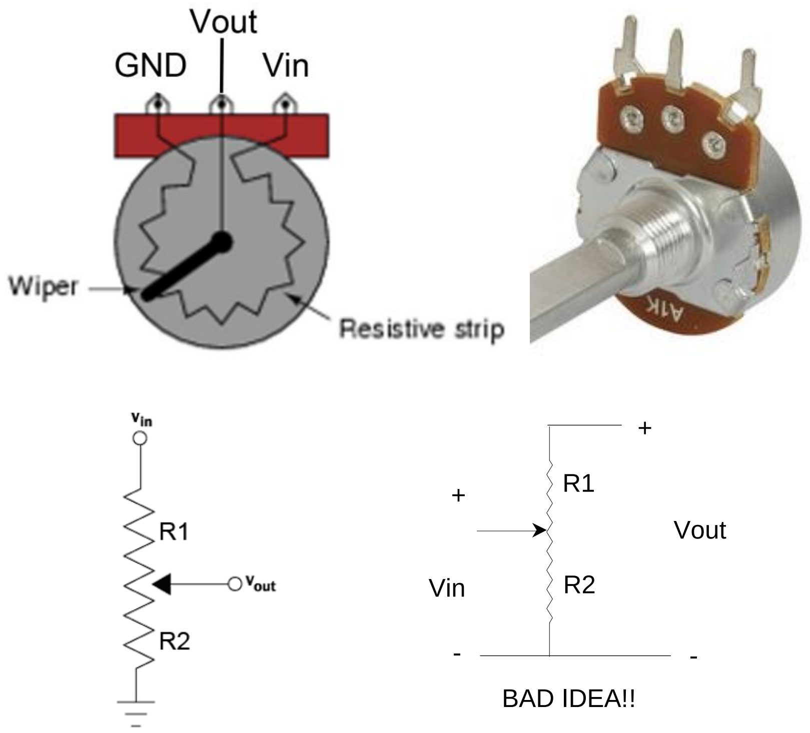

Fig. 5 Potentiometer circuits, and actual potentiometer. In the circuit diagrams, the wiper is the wire with an arrow on it. Unless otherwise labeled, the wiper is the middle connector.

Figure 4 shows a pot being used that produces an output voltage proportional to the shaft rotation. The pot works as follows. As you turn the shaft of the pot, the wiper (which is wired to the middle terminal on the pot) moves along a resistive element. So, in the figure above, R1 and R2 change as the wiper slides along the resistance, while their sum, R1+R2 = constant (which is the "value" of the pot, so a 50K pot as R1+R2 = 50 Kilo ohms).

Derive V_{out} as a function of V_{in} ,R_1, and R_2 for Figure 3A. Assume R2 changes proportionally to the shaft angle of the pot. How then does the output voltage change as a function of the shaft angle of the pot. Using this information, brainstorm two possible uses for potentiometers on your final project.

$$Answer: Vout = \frac{R2}{R1 + R2} Vin$$

Note: Vout is linearly proportional to the angle of the potentiometer shaft!

Proof: If R2 = kθ, where k is constant and θ is the angle of the pot, and we set Vin = constant, and we know R1 + R2 = Rtot = constant, then Vout = aθ/Rtot*Vin = Cθ, that is, Vout is proportional to the shaft angle, with some proportionality constant C.

Why should you never use the circuit in the bottom/right of Figure 4?

Answer: If the pot is turned to the extreme end, there is no resistance to current flow and the end part of the resistive element will be burned out.

Practical Exam 2: Show your TA you can control the voltage output of the pot by turning the shaft. Show the TA the voltage output using the oscilloscope.

Part III: The Seeeduino board and Arduino IDE (45 min.)

In this section of the lab you will work with a well-known microcontroller board that can be used to read many types of sensors as inputs and control a variety of actuators (e.g. dc motors, electronic valves). These microcontrollers can be programmed and monitored using the Arduino IDE.

Additional resources:

- Microcontroller board:
    - http://www.seeedstudio.com/depot/Seeeduino-V42-p-2517.html?cPath=6_7
- Arduino IDE:
    - http://arduino.cc/en/main/software

For this portion of the lab you will use the Seeeduino board to make an LED light blink and display this behavior on the oscilloscope. Note that these boards are equipped with an on-board LED (connected to the digital pin #13) that we will use in this portion of the lab (this LED can be very useful when debugging programs).

Software setup

First, ***connect the Seeeduino to the computer and open the Arduino Integrated Development Environment (IDE).*** *You can use either the lab's computer or download it into your own computer. Depending on whether the drivers for the microcontroller have been previously installed you may need to wait for your computer to install the required drivers (this should not be the case if you are using the lab's computer).*

Once the computer is done installing the required drivers, you now need to setup the IDE for the appropriate board and communication port (COM port in Windows machines). You can do this going to ***Tools > Board and selecting the "Arduino Duemilenove"*** *and then going to* ***Tools > Serial Port and selecting the appropriate port*** *(this value may vary but it will most likely be the highest number port on the list). If* ***"Arduino Duemilenove"*** *does not work, you can try* ***"Arduino Uno"*** *and* ***"Arduino Nano"****. Also, plugging the USB cable on and off can solve some of the connection problems you may face.*

Example: Blink

You can now either copy the code below or open it directly from the Arduino IDE by going to: File > Sketchbook > Examples > Digital > Blink.

Once the code is ready, go ahead and compile/verify the code (either by pressing CTRL + R or clicking on the 'Verify' icon). Once the IDE reads 'Done compiling.' go ahead and upload the code to the microcontroller by either pressing CTRL + U or clicking on the 'Upload' icon. If the IDE reads 'Done uploading.' then you have successfully uploaded the code to the microcontroller. If your code was successfully uploaded you should see the board's LED turn on and off in 1-second cycles.

```
/*
 Blink
 Turns on an LED on for one second, then off for one second, repeatedly.

 Most Arduinos have an on-board LED you can control. On the Uno and Leonardo,
it is attached to digital pin 13. If you're unsure what   pin the on-board
LED is connected to on your Arduino model, check the documentation at
http://www.arduino.cc
This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup function runs once when you press reset or power the board
void setup() {
 // initialize the led pin as an output.
 pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
 digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
 delay(1000);               // wait for a second
 digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW
 delay(1000);               // wait for a second
}
```

More info: https://www.arduino.cc/en/Tutorial/Blink

Arduino programs must always include at least two functions: `setup()` and `loop()` (although they may also include more). As soon as the microcontroller is powered it will first run the commands in the `setup()` function and then move on to executing the commands in the `loop()` function until the power is removed.

Serial Monitoring and Millis Function

Many times you will be interested in monitoring values in your Arduino program. To do this, you can use the Serial Monitor. The Serial Monitor is a feature of the Arduino IDE that allows you to communicate with the board via serial commands. To explore this feature we will print to the serial port the value being set to the LED pin. First, you need to modify the 'Blink' example to allow for serial communication between the computer and the microcontroller; do this by adding the following command to the `setup()` function: `Serial.begin(9600)`. You can now "print" to the serial command the value being set to the LED pin.

Note: Serial.print(…) differs from Serial.print**ln**(…) in that 'println' includes a 'carriage return' at the end of the line, which ends printing on the line. In other words, the next print statement will start on a new line.

For the final portion of this lab we will add a measure of the time that has elapsed since the program began executing. This will be a brief introduction into using measurements of time with Arduino programs. To do this, we will use the `millis()` function and print the current time of the program each time we print the state of the LED pin (see below).

Once the code is ready, go ahead and "Verify/Compile" and then "Upload" it to the board. Finally, to monitor the values being printed to the serial port, open the Serial Monitor (click CTRL + SHFT + M or click on the Serial Monitor icon). The Serial Monitor should update every second with the state of the LED pin.

To read values from the Arduino, connect the oscilloscope CH 1 probe to pin 13 (use a wire to connect the probe to the Arduino pin). Make sure to connect the alligator clip from the oscilloscope to the Arduino GND pin!

Practical Exam #3: Modify the code provided in the 'Blink' example and make the LED blink at a frequency of 4Hz. Show your TA that the light is blinking at this frequency by connecting the board to the oscilloscope and displaying at least two periods of the signal.

Extra time, self-motivated, and curious?

Try to figure out how to read the pot voltage into the Arduino using one of the A/D ports, then use it to turn the LED on and off.

```
/*
 Blink and print
 Turns on an LED on for one second, then off for one second, repeatedly.
 Writes the state and the current time to the serial port.
*/

// Pin 13 has an LED connected on most Arduino boards. Give it a name:
int led = 13;

// Variable to store the LED state: it can only be HIGH or LOW
bool ledState;

// Variable to store the current time
unsigned long currentTime;

// the setup function runs once when you press reset or power the board
void setup() {
 // initialize digital pin 13 as an output.
 pinMode(led, OUTPUT);
 Serial.begin(9600); // Start serial communication
}

// the loop function runs over and over again forever
void loop() {
 ledState = HIGH;             // HIGH is the voltage level
 currentTime = millis();      // Get milliseconds since the arduino was
turned on
 digitalWrite(led, ledState); // turn the LED on

 // Write to serial the state of the LED and the current time
 Serial.print(ledState);
 Serial.print("\t");
 Serial.println(currentTime);

 delay(1000);              // wait for a second

 ledState = 0;
 currentTime = millis();      // Get milliseconds since the arduino was
turned on
 digitalWrite(led, ledState);    // turn the LED off

 // Write to serial the state of the LED and the current time
 Serial.print(ledState);
 Serial.print("\t");
 Serial.println(currentTime);

 delay(1000);              // wait for a second
}
```