MECHATRONICS II
LABORATORY LECTURE & EXERCISES

ITEMS LIST (COMBINED)
- small: Arduino, cable, PC, Bb, wire, tact switch?, LED (2), R220 (2), TIP 121 transistor, R1.8k, high-power LED, solder, perf boards, accelerometer, DC gear motor with encoder, Cytron controller, M/F dupont, stepper motor and controller

OTHER - op amps

CONTENTS

PROJECT - balancer, PID control DC motor, DIY fan, Fanuc pick-&-place, PLC traffic light

LABORATORY INTRODUCTION

This document contains laboratory lecture & exercises for the mechatronics course. The exercises in this handout represent much of your homework and laboratory grades. Students are responsible for reading and understanding all of the material presented in each lab, including the introductory content, background information, and cross-referenced information (i.e., information in other documents). While the programming code for each exercise is usually provided, students are expected to go thru the code line by line and understand how the code works. Students may be required to create code for other exercises.

The main idea of our laboratory exercises will be to get mechatronics systems to work properly, or to program and operate an industrial mechatronics system. Generally submission of the labs involves showing the instructor that your hardware is working properly.

Mechatronics is a blending of **mechanical** and **electronics** systems. Mechatronics systems are essentially mechanical systems that are guided or controlled with a computer or electronics. If controlled with a computer they can be programmed and re-programmed to perform different tasks. Many modern manufacturing systems like robotics, 3D printing, CNC, and laser cutting are actually mechatronics systems.

We will use the ARDUINO micro-controller to control most of our systems. It will be programmed to receive sensor data and to respond to those signals by sending out signals that control lights, motors, and other systems. Arduino has become very popular in the hobbyist and academic communities. The hardware and software are both open-source, and the controller itself is fairly easy to learn. In addition, because of its popularity, you can find a lot of sample programs online. The Arduino software is free to download and use.

HARDWARE NEEDED FOR CLASS

The instructor will go over items that you will need to purchase for the laboratory activities. Some of these items are listed below:

1. Arduino kit (including an UNO and USB cable) (instructor will specify which one)
2. Digital multi-meter (DMM)
3. (Possible) Power supply or batteries (9V battery and 4 AA batteries)
4. Laptop (you can also borrow one of SAC's when on campus)
5. USB flash drive – to store files and also transfer them to and from some of our machines.

These items can be purchased online (robotshop, adafruit, sparkfun, amazon, etc.) or at local electronics stores. "Arduino-compatible" with off-brand names also work fine (e.g., Elegoo, Inland, etc.).

ARDUINO HARDWARE

Refer to the posted Arduino and C programming guides (link provided in class). The guide will provide information about the Arduino hardware and software. It will also help with how to set up Arduino.

USING SAC LAPTOPS

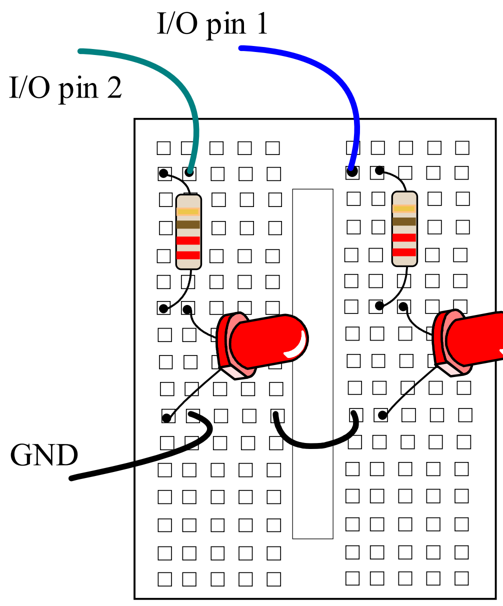
Students (while in lab) may use a lab laptop. Please read the posted guide on how to properly borrow and return lab laptops.

SAFETY

It is essential that the laboratory is a SAFE environment. The laboratory exercises in this course involve the use of a variety of fabrication equipment. The equipment is generally safe and easy to use, but they can still pose a danger to students if not used properly. Students must pass a safety exam before using any of the lab equipment.

Refer to the SAFETY chapter in the CT GUIDE handout.

LAB 1 - NO DELAY BLINK (Home)



ITEM LIST

1.) Arduino, cable, PC, Arduino IDE
2.) LED (2), 220-ohm R (2)
3.) Wire, Bb (or Bb shield)

INTRO

REFER to ARDUINO GUIDE (Ch 7 - No Delay)

One of the first exercises we did with Arduino was to blink an LED. This was done by toggling the digital output pins between LOW and HIGH. The duration was controlled with the delay ( ) function. Using the delay ( ) function is easy and works fine for some programs, but it becomes problematic if you are writing a MULTI-TASKING code (executing several tasks at the same time). The delay ( ) function is problematic because it essentially tells the CPU to STOP DOING ANYTHING for that period of time. A simple example of multi-tasking would be to blink 2 different LED's at the same time but at different frequencies. Using the delay ( ) function will not work in this case.

Alternatives to delay ( ) are to run the process in an "if" statement. The "if" statement would test whether a time variable has exceeded some threshold. You can also use Arduino's "sine ( )" function.

Frequency of a repeating signal is often quantified as "Hertz" (Hz). One Hz is one cycle per second. A cycle for a blinking LED would be the period with the LED ON and then OFF.

1.1 TIME THRESHOLD

EXERCISE

Program the Arduino and build a circuit to blink 2 LEDs at different rates. Do not use the delay ( ) function. Instead use "if" statement and time thresholds. If you cannot figure out the programming yourself, you can probably look that up online ("Arduino blink no delay"). This approach works well if you wish to control the time the LED is on and the time it is off.

1.2 SINE FADING

EXERCISE

Program the blink using the `sin()` function in Arduino. Here the LED will fade on and off (use analogWrite). Sine waves oscillate between -1 and +1, but we know Arduino analog out ranges from 0 to 255. You can either add an ampitude and offset to ensure the sine wave osciallates between 0 and 255, or you can use the MAP function to change the range. With MAP you should still use an ampltiude (eg, 200) on the sine wave to ensure int truncation has small effect. Note analogWrite only works with PWM pins (marked with "~"). This program makes it easy to control the FREQUENCY of the blinking LED (in Hz, or blinks/second in this case).

```
float tnow = millis()/1000.0;
float amp = 100.0;
sig = amp * sin (2 * PI * freq * tnow) + amp;     // oscill betw 0 & 2*amp
```

1.3 SINE BLINKING

EXERCISE

Program the blink using the `sin()` function in Arduino. Here the LED will BLINK on and off but based on a sine wave. The sine wave makes it easy to adjust the blinking frequency. However this method does not work well if you wish to precisely control time on and off time periods. Do NOT offset the sine wave. When you do this the sine will be >0 half the time and <0 half the time. Then insert a line of code (if statement) that tests if the sine wave is (+) (then turn the LED on), or (-) (turn the LED off)

SUBMISSION

Show the instructor in person that your hardware is functioning properly.

OPTION (if instructor allows it)
Video upload to Canvas showing your functioning hardware.