Overview: In this lab, you will build a motor testing rig and design its control software. Your rig will be able to perform a variety of tests using several different kinds of motors and motor control hardware.

This lab will introduce you to some of the CMPE118 motor driver boards and the CMPE118 Uno Stack, hardware designed to facilitate mechatronic prototyping.

You will develop your testing software in the ES framework.

WARNINGS:

- **Do NOT put more than 5V into the servo motor.** You will kill the servo and need to buy a new one. Be extremely careful with this one.

- **Do NOT hook power and ground up with the power supply turned on.** It is the easiest way to fry microcontrollers, followed by reversing power and ground. Always hook up wires (power or otherwise) with power supply TURNED OFF!

- **Triple-check your power wires before turning on the power supply.** In particular, make sure you haven't reversed the wires.
  - Use consistent color coding (Red → + voltage, Black → ground). You should also include reverse-polarity protection in any circuit you make (a diode in series with the power supply usually suffices). While we have protected some of the boards, we have not protected them all.

- **The ATX power supplies we are using will fry chips.** They are robust, clean, and supply a great deal of current (~16Amps). While they will save themselves from a short circuit, they will NOT save your board.

- **All** of your daughter boards (Driver, H-Bridge, and Stepper) should be powered by the power distribution board (the one on the bottom of the Uno32 stack).
  - The power distribution board is protected with fuses and has indicator lights to tell you if you blew a fuse. You will use this throughout the remainder of the course.
  - You will need to solder the power connectors that connect the power distribution board to the motor driver boards. Ensure you’ve properly soldered the BLACK ground and RED power cords, and make sure that heat shrink snugly covers the leads.

- **DO NOT** stick hookup wires or probes into the headers on the Uno32 Stack.
The only thing you should stick into the headers are the dual pins or push-on connectors. This is **NOT** negotiable. Probes, wires, etc. often break off inside the connectors, forever blocking that pin.

- **IF you fry something, TAKE A PICTURE OF YOUR SETUP AND POST IT TO PIAZZA IMMEDIATELY.** This way the staff can diagnose your error. We understand that you are students and that you will make mistakes. We will not be angry, but we need to know what those mistakes are so we can reverse them and prevent them in the future.

**DELIVERABLES**

- Lab report
- An ES Framework service (FSM or HSM) which allows you to run ALL of the tests described in this lab.
  - Also include any substate machines or modules you develop.
- Stepper.c and Stepper.h
  - A modified version of the stepper software module as required in part 4
Pre-Lab:

Complete the following exercises AFTER you have read through the lab assignment and BEFORE coming in to complete the lab. This pre-lab will help you get through the lab quickly and efficiently.

0.1 **Create a Block Diagram for each part of the lab.** Your block diagram should communicate the following information:

- Names of the pins on the boards you are using (both on the Uno and on the daughter boards). For the Uno, these should be of the form (Port X Pin 3 or X3).
- An arrow indicating whether an Uno pin is an input or an output (or both).
- Your power source (12V rail, 5V rail, power distribution board, etc.).

On the same page as each diagram, also include:

- Why you chose this port for that part of the lab.
- What, if any, initialization code you will need for the port. We want the literal code that you will execute: functions(), PIN_NAMES, and semicolons. You shouldn’t need more than 4-6 lines per lab part. Look at the test code at the end of the libraries if you need inspiration.
- The type of signal each wire is carrying (analog, digital, etc.)

Below is an example block diagram for part 1 (which you still need to modify) to show you how these should look:

While you are picking out Uno pins and working out your breadboard layout, keep in mind that you will probably come back to many parts in this lab. You will find it
advantageous to work out a system that makes it easy to switch between connection
setups (the ribbon cable connectors are particularly useful for this).

0.2 What one thing and one thing only will you stick into the 14-pin connectors on the
boards?

0.3 Draw a detailed state diagram for a software PWM driver. Include an example plot
of the expected output signal for a given frequency and duty cycle.

HINT: The transitions between states should be triggered by timer events.

In the pre-lab: 1 PWM State Machine and an answer to 0.2.;
6 Block diagrams;
2 schematics;
3 stepper state machines.

**Hints on working this assignment**

- Come into the lab with a plan. A plan for each part. A plan for the day. A plan for the week.

- This lab is about preparation. If you try to do your prep work at the same time as you do the lab,
you will take MUCH longer and break more things. Prepare first, then do the labwork.

- Test your hardware and software separately.

- If you have spent more than one hour on any single task, you are probably not on the right path.
STOP and ask a neighbor, tutor, or TA to take a look at what you are doing.

- Make a pair of 14-wire ribbon cables. This is not required but it will save much time in the long
run.

**Electronic Hygiene**

This lab involves high currents. High currents are dangerous, both for you and for your hardware. 10
amps on the wrong wire can blow the traces off your board, fry any chip, and melt plastic. Silly mistakes
and accidents can easily destroy days of hard work or cause days of delays. This not only endangers your
schedule, it also produces pain, stress, sleeplessness, frustration, and dismay. Therefore, maintaining good
“electronic hygiene” is CRITICAL to your success and sanity.

Good Electronic hygiene practices include:

- **NEVER, NEVER, NEVER plug things in and out of your board with the power supply
turned on.**

- Use banana plugs to connect power. They are stable and secure—they're a reason all power
supplies use banana connectors! Do not use alligator clips.

- Check your polarity AT LEAST THREE times before hooking up power to things. Color-code
everything to facilitate this process. You'll have four rails in this lab (GND, 3.3v, 5v, and 12v). Give each one a different color.

- Be very careful about wires dangling from power supplies or O-scope leads shorting things out. Every bit of bare metal is a risk, so take the time to tuck away your idle connectors.
- Always inspect your boards for solder bridges and the like. Little flecks of solder or wire can have big consequences.
- Take the time to make your connectors well.
  - Make sure they're a comfortable length.
  - Make sure you have heat shrink covering every joint—bare metal is a recipe for sparks and shorts.
  - Test your wires and connections with a multimeter on continuity test mode (it is usually depicted with a diode and a sound wave)
  - Don't forget to cross test—The black lead on one end should connect to the other black lead, AND should not connect to the red lead.
- Diagnose errors before continuing.
- Avoid frustration. Frustration can impair your decision-making. If you are having difficulty, ask for help. Breathe. Take a walk outside. Communicate thoroughly and kindly with your lab partner.
- Anticipate user error. You and your partner will be tired, frustrated, or rushing at some point during this lab.

It is often tempting to skip some of these practices. If you find yourself feeling such temptation, imagine what you will feel when something important explodes, fries, or melts. Frustration? Anger? Dismay? Stress and sleep deprivation? It is worth taking an extra 15 minutes now to lower the odds of feeling emotions like that later.

### Part 1: Driving an R/C Servo Motor

**Prelab:**

Draw a schematic for your potentiometer circuit. Remember, you should NOT put more than 3.3v on an Uno32 pin. Make a block diagram of your setup in this part.

Include your initialization code.

**References:**

CKO Ch. 27 (especially section 27.3), RC_SERVO module documentation. RC Servo tutorial at: [http://www.societyofrobots.com/actuators_servos.shtml](http://www.societyofrobots.com/actuators_servos.shtml)

**Requirements:**

You are going to drive an R/C servo with position feedback. The goal is a position control system in which the position of the servo is controlled by the potentiometer.
Set up an ES Framework project and service to let you select tests. You can use KEYBOARD_INPUT mode if you like, or come up with some other approach. As long as you use ES Framework and can easily select the test you want to run, that’s fine.

Use a potentiometer to create a voltage divider that ranges from 0 to 3.3v.

Use the AD (analog-to-digital) software module to measure the output of the voltage divider.

The LEDs on the Uno32 board reflect the divided voltage.

Use the RC_Servo software module to control the position of the servo.

0v should set the servo at one end of the range, 3.3v at the other. Note that the range of the RC servo is going to be +/- 40-60 degrees. Don’t change the MIN_PULSE and MAX_PULSE ranges.

Uno connects to your breadboard via a ribbon cable with headers.

**General Instructions:**

You will need no daughter boards for this section.

You may connect the ATX power supply's 5v output directly to your breadboard. Alternatively, you may use a 5v regulator from the 12v line and make your own 5v rail.

The servo has three wires. One of these should connect to ground, one should connect to a MAXIMUM of 5v, and the third wire carries a control signal. **DO NOT CONNECT THE SERVO TO A HIGHER VOLTAGE, YOU WILL DESTROY THE SERVO.**

**THE UNO PINS CAN BE DAMAGED BY MORE THAN 3.3V.** Be sure that your voltage divider (and anything else you ever connect to the Uno32) doesn't exceed 3.3v in any situation (Anticipate user error). From the 5v rail, you may use a resistor in series with your potentiometer, or use a regulator to set up a 3.3v rail.

**DO A SCOPE TRACE OF THE OUTPUT FROM THE CONTROL PIN ON THE UNO BEFORE YOU CONNECT THE SERVO.** You should see a square pulse of no more than 2ms width. If you do not see this, DO NOT connect the servo to that signal!

Your Uno32 stack and the servo need to have a common ground connection back to the power supply. This can be directly to the ATX, or through the power distribution board.

**Checkoff:**

Demonstrate that the potentiometer controls the servos and the LEDs.
Final Report: Include:

- Annotated scope traces of the output of the servo control pin
- What is the minimum change of the pulse high time that results in motion of the servo head? (The module allows you to control the pulse width to 1uS resolution).
- In degrees, what is the range of the servo? What is its maximum angular velocity? What is its minimum angular velocity for smooth motion? (You may not want to have the servo under potentiometer control while performing this test).

Part 2: Interfacing to a DC Motor and Potentiometer

Prelab: Draw a block diagram of the setup you will use in this part. Include your initialization code.

Reference: CKO Ch. 8, CMPE118_AD, CMPE118_LED, and CMPE118_PWM reference documentation, the CMPE118 Uno32 IO Board documentation, the CMPE118 DS3658 Module board documentation, the MPLAB-X Quickstart manual.

Overview: You are to design the software necessary for the PIC32 to drive the supplied DC motor using the DC3658.

Requirements: Design Requirements:

- Motor Speed is regulated by a potentiometer.
- 12 LEDs on the UNO light up to show the speed from 0 to 12. 0 is a stopped motor, 12 is a 100% duty cycle.
- The PIC32 outputs to the motor using the provided PWM software module.
- The tests are integrated into your ES Framework testing suite.
- Uno connects to your daughterboard via a ribbon cable with headers.

Perform the following experiments:

- Take an O-scope trace of the voltage across the motor at an 80% duty cycle, and a 20% duty cycle. Identify the important features of the waveform: The active period, the inductive kickback peak, the decay period, and the effect of back-EMF.

General Instructions: Set up your breadboard rails first. The 12v, 5v, and GND should all be connected directly to the ATX. Don’t forget to connect power using the
**banana plug sockets on the breadboard!**

The DC3658 only sinks current, so it needs to be connected to the low voltage side of the DC motor. The high voltage side of the motor should be sourced directly from the UNO power distribution board, using a single-wire power connector you will need to make for this purpose. (Connect these together either via the screw terminal of the breadboard, or using the breadboard itself).

You will also need to make a power connector to power the DC3658. **Color-code the wires, use heat shrink on both ends of the connector, and triple-check the polarity.**

Check your potentiometer output with a multimeter to ensure it never exceeds 3.3v BEFORE connecting it to the UNO IO board. **Always check the voltage range on a wire before you connect it to the UNO IO board!** Students have destroyed many input pins in this class.

Before you drive the motor, **be sure your DC3658 board is operating in clamped mode!** If you fail to do so, kickback could permanently damage the board.

If you don’t remember how to make a new MPLABX project for the PIC32, go back to the handout from Lab 0.

Keep your block diagram at hand!

**Checkoff:**
Demonstrate that the potentiometer controls the motor speed and LEDs. Show a scope trace of an inductive kickback response.

**In the report:**
Block Diagram and your final program. A photo or capture of your oscilloscope traces, with descriptive labels.

---

**Part 3 Understanding DC Motors**

**Prelab:**
Block diagram of the setup you will use in this part. Schematic of the breadboard circuit you will use. (You may find it advantageous to use a jumper wire to switch between circuit setups).

**References:**
Also answer: Which of the circuits in the requirements section corresponds to the DS3658 when operating in clamped mode?

CKO Ch. 22, 23, 24, 26 and 27. Documentation for the DS3658.

**NOTE:** DS3658 Can only sink 600mA per output, if you have a non-gearhead motor, connect two stages in parallel to up the current capability.

**Overview:**
Become familiar with the waveforms that DC motors exhibit, and learn to identify their features. Understand the differences between different kickback protection circuits.

**Requirements:**
Create a setup for the following experiments, and then perform them:
Waveforms: For each of the following kickback protection circuits (note the Transistor represents the one on Darlington Driver Board):

1) Take an O-scope trace of the motor’s waveform at an intermediate duty cycle, a very high duty cycle, and a very low duty cycle. What differences do you observe between these circuits? Explain these differences. For each circuit, identify and quantify the kickback peak and the decay time.

2) Find limits on the frequency and duty cycle for each circuit. In what circumstances is the motor’s speed a linear function of duty cycle?

3) Add resistive torque to the motor (grip the shaft with your fingers, then with a pair of pliers). What is the effect on the waveform? Why?

4) Uno connects to your daughterboard via a ribbon cable with headers.

Checkoff: Show illustrative scope traces from your experiments to staff, and be prepared to answer questions.

General Instructions: To measure the waveform across the motor, don’t connect the probe’s ground lead to V_collector—you will cause a short across the DS3658 through the oscope. Instead, connect the oscpe’s ground lead to ground, and the high lead to V_collector. This is an indirect measurement of the motor’s waveform—you’ll have to do a little math to recover the actual waveform.

When comparing the different diode circuits, pay close attention to the decay time of each circuit. You may need to adjust your timescale to get a good measurement.

In the report: Annotated scope traces showing the differences between the kickback circuits. Discussion of the advantages and disadvantages of different approaches to kickback protection.

Part 4 Interfacing to a DC Motor and H-Bridge
Prelab: Draw a block diagram for this part.

References: The CMPE 118 DRV8814 Dual H-Bridge Module documentation.

Overview: The driver in the last two parts only provides uni-directional control, but an H-bridge can control a motor in either direction. In this part, you will learn to use the DRV8814 board to provide bi-directional control of a DC motor.

Requirements: Duplicate Part 1 with more functionality. This time:

1) Your motor can change directions. Now a 0v signal from the potentiometer should be the motor turning counterclockwise at max speed and 3.3V should be the motor turning clockwise at max speed. (1.65V is stopped)

2) Use the LEDS to show your speed and direction. Your code should use bit-shifting and bit-masking to perform this task in few lines of code (if you’re using a very long switch or nested if statements, you are doing it wrong).

3) Determine the frequency limits of operation for your setup. If they differ from those found in parts 1 or 2, explain why.

4) The tests are integrated into your ES Framework testing suite.

5) Uno connects to your daughterboard via a ribbon cable with headers.

General Instructions: For this part, you will use the CMPE118 DRV8814 Dual H-Bridge Driver Module board. You will need to supply enable and direction inputs as well a PWM signal.

Activate your software and use an oscilloscope to test your outputs (BEFORE you connect them to the boards). To facilitate O-scope reading of pins, each 14 pin connector has ground pins at 1, 2, 13, and 14.

Checkoff: Demonstrate that the potentiometer controls the motor speed and direction, and that the LEDS reflect this.

In the report: Block diagram, code.

Part 5 Interfacing to a Stepper Motor

Prelab: Block diagram for this part. State machines to drive a stepper motor in Half-step, Full-step, and Wave mode.

References: CKO Ch. 26, the CMPE118 DRV8814 H-Bridge Driver Module documentation, CMPE118 STEPPER software module documentation.

Overview: In this part, you will learn to write software that drives a stepper motor, and
become familiar with the properties of a stepper motor.

Requirements: You will build a setup that:

1) Uses a DRV8814 module to drive a stepper motor
2) Can drive a stepper motor in Half-step, full-step, and wave mode (you will need to make a copy of stepper.c and modify it).
3) Find the maximum step rate at which the stepper can rotate.
4) Find the maximum step rate at which the stepper can rotate without losing steps.
5) Are these results affected by adding load torque? (put your fingers on the shaft and squeeze).
6) Uno connects to your daughterboard via a ribbon cable with headers.
7) The tests are integrated into your ES Framework testing suite.

For bragging rights, demonstrate that you can successfully enter the pull-in region of stepper motor operation without losing steps (read the CKO chapter on steppers if you don’t know what that means).

General Instructions: Test your UNO outputs with an oscilloscope before you connect them to the daughter board!

The stepper motor has four wires, one for each end of two coils. Use a multimeter or find the datasheet to determine how they are paired.

To determine if a stepper motor is losing steps, a good trick is to put a piece of tape on the shaft and attempt to move it at some multiple of 360 degrees. See if the tape returns to its original location.

The course stepper software uses a Full-step mode state machine by default. This state machine is isolated in a single function. Find that function and write new versions to use wave- and half-step modes.

Checkoff: Show off your software and prowess to the staff for check off. Reproduce your experiment to find the maximum step rate without losing steps.

In the report: Block Diagram of your module, code for Wave Drive and Half-Step Drive, explanation of issues, results of your experiments. Describe the advantages and disadvantages of each mode.

**Part 6 Interfacing to a Stepper Board**

Prelab: Block diagram

Reading: CKO Ch. 26, Stepper software module documentation.
Overview: As is often the case in mechatronic design, specialized hardware can reduce the complexity of the software. In this part, you will drive a stepper motor using the DRV8811 board.

Requirements: You will build a setup that:

1) Uses the DRV8811 to drive a stepper
2) Uses yet another state machine of your own creation in the stepper software to control the DRV8811

You will also perform the following tests:

1) Drive the stepper at half step and full step. How does its performance compare to the software-driven version in the previous part?
2) Try running the stepper at 0.5 amps and 2 amps. How does the current limit affect its behavior?

General Instructions: Use the jumpers on the DRV8811 to select its mode of operation and current limits. Be careful when running it at high currents—the steppers can get very hot! Carefully test this with your fingers.

Checkoff: Show your working setup, show your code.

In the report: Your block diagram, code, and a discussion of the performance differences between hardware and software stepper drivers.