2 Motor Control and Actuation Module

Module Due: 01FEB2018

Pre-Lab Due: 23JAN2018, beginning of class.

2.1 Module Overview

The primary output of this module is a set of completed and functional prototype circuits for the actuators that support meeting the requirements in the Pinball Machine Project Description document. We will integrate these components into a full system with Arduino control in later modules. For this module, we will build standalone circuits and demonstrate the capability for open loop control using an Arduino microcontroller.

In addition to design, fabrication, and characterization of the functional flipper system, this module has other activities (see section 0) that build relevant skills both inside and outside of the lab sessions. These other activities include introductions to electronic measurement tools and techniques, electronic circuit components and analysis, using transistors as switches, and Arduino microcontroller-based interfacing with electromechanical systems.

Save your circuits and optimize your breadboard space! You can keep most of your electronics that you design here for your final project, which will be **run completely from your breadboard.**

Prelab activity sections: *2.3.1.a*

2.1.1 Learning goals

Purposes ('What this module's activities seek to develop')

- Prepare students/teams for independent electronics design project work
    - Electronics measurement skills
    - Electronics prototyping skills
    - Basic circuit analysis and design skills
    - Knowledge of available tools, materials, and methods
- Introduce electromechanical components and circuits for controlling them
    - DC motors, RC motors, solenoids
    - Microcontrollers

Knowledge/Functional goals ("Students/teams will (be able to)…")

- Create component-level electronic hardware designs based on system requirements
- Use discrete electronics prototyping tools to build working circuits on a breadboard
- Use electronics lab instruments, including oscilloscopes, probes, multimeters, function generators, and power supplies, to characterize discrete circuits
- Set up and program an Arduino microcontroller
- Control electronic circuits using pulse width modulation (PWM) with an Arduino
- Know how to safely switch high current loads with electronic circuits
- Identify potential challenges/questions about an electronics design and test them through prototyping and experiments
- Document and evaluate electronic test results appropriately
- Create plans for next-step circuit prototype(s) based on existing prototypes and experiments

2.2 Deliverables

The list of deliverables for this module is uploaded in a document entitled Module 2 Deliverables on TritonED. Please create a **single .doc(x) file** with all of the deliverables for each section into that document and upload to TritonED. If an additional file is needed, upload it to TritonED (appropriately named) and take a screenshot to document the work in your single .doc(x) file. For each of the sections, what you need to include materials described in the green text, and follow the separate "**Module Deliverable Instructions**" on TritonED as per the last Module.

2.3 Activities

2.3.1 Basic electronic circuit analysis and measurements

This set of skills-building activities reviews fundamental concepts and laboratory techniques in electrical and electronic engineering that you will use throughout the course. The following textbook sections are directly relevant to this material and to understanding the basics of electronics more generally.

Text: *Practical Electronics for Inventors*, Scherz

Optional Reading

- 2.2: Electric Current; 2.3: Voltage; 2.5: Resistance, Resistivity, Conductivity; 2.10: Grounds; 2.11: Electric Circuits; 2.12: Ohm's Law and Resistors
- 3.1-3.1.3: Wires, Cables, and Connectors
- 13.2-13.2.3 (14.2-14.2.3 in Ed. 2): Constructing Circuits; 13.2.11 (14.2.11 in Ed. 2): Troubleshooting circuits; 13.3-13.3.1 (14.3-14.3.1 in Ed. 2) Multimeters
- 2.17: Kirchhoff's Laws; 2.14: Measuring Voltage, Current, and Resistance; 2.7: Heat and Power; 2.13: Voltage and Current Sources; 2.15: Combining Batteries; 2.16: Open and Short Circuits
- 14.3.2-4 (13.3.2-4 in Ed. 2): Multimeter theory and measurement errors

2.3.1.a Theory and Analysis of Simple Circuits (Pre-lab Activity)

In advance of your first Module 2 lab session, please complete the following exercises 2.3.1.a.1-3 that practice the circuit analysis and design concepts from the lectures. **Complete these in your lab notebook, including any relevant diagrams, and clearly label outputs/results** so that you can use them in lab and so that the TAs can check your work. You will use the results from these exercises during the lab. Please refer to Figure 1.

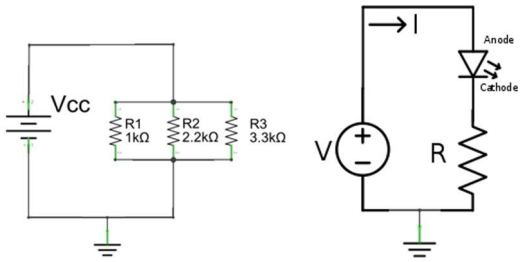


Figure 1: Circuit diagrams for 2.3.1.a (pre-lab theory) and 2.3.1.b (lab exercises).

2.3.1.a.1 Problem 1: For circuit 'a' from the diagram, calculate the following quantities:

- The currents in R1, R2, R3, the equivalent resistance of the combined Rs and the total current from the voltage source. Assume Vcc = 5V.

2.3.1.a.2 Problem 2: For circuit 'b' from the diagram

- Determine the values of R that will limit the current through the diode/LED. Assume a 5V supply, and the following LEDs:
  - Red LED: LTL2R3KRD
  - IR LED: IR204

  Find R for each LED and record this in a table.
- Does it matter if you switch the order of the resistor and diode/LED?

- Resistors are typically spec'd at 1/4 Watt, 5% tolerance, carbon film resistors (for special applications you can get lower tolerance, higher power dissipation resistors). Assume you have these resistors at 100Ohm increments (starting at 100Ohms). What resistor would be selected for each LED to achieve a safe maximum brightness?

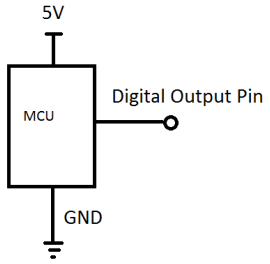
2.3.1.a.3 Problem 3: Implementing circuits in real life

Two concepts: the common ground, and pull-up/pull-down resistors, are important to understand. Feel free to ask the teaching staff more question if these concepts are not clear to you.

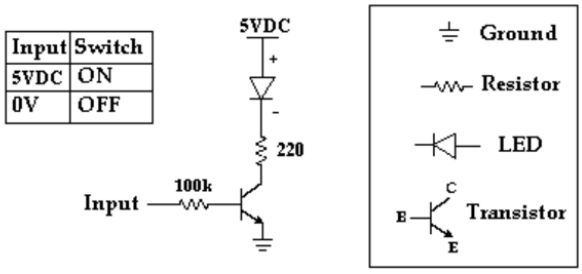
Common Ground: when dealing with logic circuits (including micro-processing units like Arduino), a common ground is used to make sure voltage potentials are read relative to the same "0V" level.

Pull up / Pull down resistors: Most ICs with input pins (expecting a voltage) are exactly doing that: expecting a voltage potential. If you do not give it one, and, lets say, you turned off your input or you don't set your input, then it will pick up whatever potential it can find, from the stray electromagnetic radiation in the room, from static, from the air, etc. You will get very strange results. Pull up and pull down resistors are large resistors such that it biases the circuit weakly towards the logic rail voltage (e.g. 3.3V or 5V) or ground, respectively, so that this "floating" behavior does not happen even when an input is cut off (e.g. no current running through it).

Consider the following:



(a) MCU black box



(b) a circuit with a digital input

Draw the above circuit in your notebooks and a block for an MCU with its own voltage rails and an output pin. Connect the Arduino block to drive this circuit considering the common ground and pull-up/pull-down resistors.

2.3.2 Oscilloscopes and Waveform Generators

This section will review oscilloscopes and waveform generators. You will use these frequently in this and following modules. There are no activities in this part of the module (2.3.2), and so this is for familiarizing yourself with using the equipment only. You can skip this if you are completely familiar.

2.3.2.a Oscilloscopes

In this section we will review the use of Oscilloscopes. Oscilloscopes measure the voltage between the center pin and the outer contact of its input port, producing a real-time plot of input voltage as a function of time. The most basic and frequently used controls adjust the time scale (x-axis) and the amplitude scale (y-axis) of the measurement on the screen.

The scope also has both vertical and horizontal measurement lines that allow for voltage and time measurements. You can use the controls near the top of front panel marked "Cursor", "Select" and "Display" to adjust the measurement lines. Using the controls along with the buttons at the bottom and to the right of the display, you can make voltage and time measurements. Note: when making voltage measurements be sure you know where the zero reference is—it is adjustable.

2.3.2.a.1 Observing the calibration signal

Each oscilloscope has a calibration signal available on the front panel.

- Turn on the oscilloscope and attach a scope probe to the calibration signal
- Adjust the various controls until you see a signal

2.3.2.a.2 Scope probes and calibration

Any time you make a measurement of a circuit, your measurement equipment disturbs the circuit, and interferes with the measurement. Under most circumstances in this class, the effects will be negligible. However, for small signals, or at high frequencies, simply making the measurement can change the whole circuit and the measurement (!).

Before starting any measurement, it is a good idea to calibrate the scope probe, so that it is disturbing the circuit the least—or is at least displaying the measurement most accurately. There is an adjustment screw on the probe body for this purpose.

- With the probe connected to the calibration signal (still), turn the screw adjustment and observe the effect on the measurement output.
- Now adjust the screw on the probe until calibration signal is a 'perfect' square wave—this is the calibration procedure for the scope probe.
- Try touching your finger tip to the scope probe and adjust the display to view the signal. Observe what the frequency the signal is; where is the signal coming from? You may also want to do the same test by attaching a ~20cm length of wire between the ground of the scope and the input. This electromagnetic noise is prevalent in all signals and often, in sensors, must be filtered out in order to produce a robust sensor for noisy environments (Module 3).

2.3.2.b Function Generator

In this section we will review the use of Function Generators. The function generator generates sine waves, square waves, ramps, and a few other simple shapes. The waveform parameters such as frequency, and amplitude, and duty cycle are controllable via the front panel buttons/knobs. A perfectly symmetrical square wave has a 50% duty cycle. A waveform which is on for 1 ms and off for 9ms has a

10% duty cycle. Though this waveform is not "square" it is still commonly referred to as a square wave. **In this lab, the waveform generator is integrated inside our (fancy!) oscilloscope.**

2.3.2.b.1 Waveforms

- Turn on the function generator and measure the waveform on a scope channel.
- Ensure you can display the following waveforms as exercise:
    - 10 khz sinewave, 3 volt peak to peak, with 0 offset.
    - 1 khz square wave, 5 volt peak to peak, with duty cycle at 20%, offset so that the output is 0-5V

2.3.3 Using transistors as switches to drive high-current loads

When you want to turn on and control an electromechanical or high power load, the classic technique is to use low power signals (the "brains") to control high current switching elements (the "brawn"). This is because electronic digital controllers generally have limited current-sourcing and sinking capability. For example, the Arduino can drive a maximum of 40mA, which is just enough to run a few LEDs—not to turn a motor.

In this activity, you will use metal–oxide–semiconductor field-effect transistors (MOSFETs) and "logic level" signals from a function generator (0-5V @ µAs of current) to control the speed and actuation of high power loads (~1V @ Amps of current). MOSFETs in this application are basically highly reliable and fast electronic on/off switches. Using pulse width modulation from a function generator as the gate drive signal to a MOSFET allows for quasi-analog adjustment of speed or drive power simply by changing the duty cycle of the control signal.

In addition to the lecture material, the following textbook sections will provide useful background on MOSFETs and how to control electromechanical loads with low-power signals.

- 4.3.4 Metal oxide semiconductor field-effect transistors (Scherz)
- 14.1-14.2 (13.1-13.2 in Ed. 2): DC continuous motors, and speed control of DC motors (Scherz)

2.3.3.a Design and analyze a MOSFET motor speed controller circuit for a DC motor

You want to control the speed of a DC motor with PWM using a function generator and a MOSFET circuit. Note: you will only drive the motor in one direction.

- Draw a basic circuit that will allow you to control the speed of the motor with PWM using a function generator.
    - Assume you have a N-type power MOSFET, a dc motor, any passive components you might need (i.e. resistors, capacitors, diodes etc.) and a working function generator with variable duty cycle.
- How would you change the motor speed continuously?
- If the motor is a 12V dc motor with an armature resistance of 6Ω, what is the current drawn by the motor at stall?
- Look up the IRF540 N-channel MOSFET and the 2N3904 NPN BJT data sheets. Can you safely use both transistors to control the motor?

2.3.3.b Build a MOSFET motor speed controller circuit

- *Note the difference between the logic level supply voltage (5V) and the typical motor supply voltage for the motor in your kit (12V). We will use the MOSFET to separate the circuits and supply the motor with its full range supply. Remember to connect the grounds of these two supplies.*

- Construct the motor speed circuit you designed in 2.3.3.a using your breadboard and jumper wires. Connect the motor leads to the breadboard using either simple solid core wire or alligator clips.
    - **Note: The motor is an inductive load, and thus generates a very large voltage when it experiences rapid current changes (i.e. V = L*dI/dt). When using a motor experiences a switching signal (e.g. motor on/off or a PWM signal) the sharp edges of the square wave cause large voltage spikes that can burn out the MOSFET.** Consider Figure 2, where a motor (represented by an inductor) is driven by a load (the switch is on). A kickback diode (e.g. 1N4001) in this configuration prevents this from happening by offering a return path for the current.

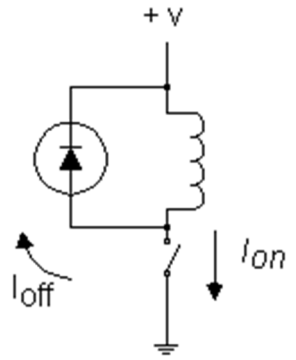


Figure 2: Kickback diode configuration

Ensure the kickback diode is in place whenever driving a motor.

    - *Note: Do all circuit construction without the power supply connected or on.*
- Generate a PWM signal with the function generator. Use the oscilloscope to monitor the output of the generator, being sure to connect the grounds of the two probes to each other. Generate a square wave at 1kHz with a 50% duty cycle ranging from 0 to 5V. Use the cursors and/or quick measurements to verify the values in your input signal. Check to see that you can vary the duty cycle with the “duty” knob on the function generator or by changing the waveform shape.

2.3.3.c Design and build a MOSFET solenoid driver circuit

In this activity, you will control a pull-type solenoid using a function generator and a MOSFET circuit.

2.3.3.c.1 Circuit analysis: For Circuit in Figure 3

- Assume that V_{DS} (the voltage between node D and node S) is 0V when the MOSFET is on and that I_D (the current from node D to node S) is 0A when the MOSFET is off
- Assume further that a capacitor acts as an open circuit at steady state and as a short circuit immediately after a sudden application of voltage
- (A) Calculate the voltage across R2 immediately before the SPST switch is closed (assuming the MOSFET has been off for a long time)
- (B) Calculate the voltage across R2 immediately after the SPST switch is closed (part 1)
- (C) Calculate the voltage across R2 after the SPST switch has been closed for a long time
- (D) Draw the voltage of R2 given a step input; what is the time constant of the R1C1 circuit and why would this type of voltage curve be useful?

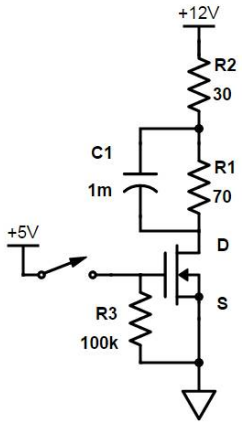


Figure 3: Circuit diagram for 2.3.3.c.1

2.3.3.c.2 Design

You want to control a solenoid using a function generator and a MOSFET circuit. The solenoid should respond quickly when provided an input and should be able to hold its position as long as necessary without burning out.

- Using your knowledge from 2.3.3.c, draw a basic circuit that will allow you to control the solenoid action such that you have a large impact when you turn it on, but that holding the solenoid closed requires much less current. Consider, how quickly do you want the system to die down and to what level?
    - Assume you have a N-type power MOSFET, a solenoid, any passive components you might need (i.e. resistors, capacitors, diodes etc.) and a working function generator with variable duty cycle.

2.3.3.c.3 Build

- Construct the solenoid driver circuit you designed in 2.3.3.c.2 using your breadboard and jumper wires. Connect the solenoid leads to the breadboard using jumper wires. Remember to use a kickback diode. How much power are you dissipating through R1? Note, resistors have a 1/4W rating – did exceed this?
- Use the function generator to generate 0.5Hz **square** wave ranging from 0 to 5V.
- Connect the Function Generator output to the gate of the power MOSFET in your circuit; you can change the duty cycle (0 to 50%) and observe what happens. Verify your solenoid fires properly.

Note: anything that has a lot of current (e.g. > 200mA) flowing through it is going to heat up. We have heatsinks to put on your MOSFETS if you are pulling a few hundred mAs to keep things from getting burned.

2.3.3.d Control an RC servomotor

In this activity, you will use control the angular position of an RC servomotor shaft using a pulse signal, repeated every 20 msec, from the function generator. An RC servomotor is a nice package that provides the ability to control the position of a motor shaft without extra electronics. It has been used very often in RC toys and low-end robotic systems. The RC servomotor has 3 pins to control the motor: GND, Vs, and Signal. The duration of the pulse on the signal line determines the angle at which the motor shaft is at. The pulse duration ranges from 1 msec to 2msec, which corresponds to the min and max angles of the servo. *Do not let the pulse stay for more than 2 msec, which can cause the internal electronics of the servo to behave badly and can fry the motor.* Note that there is a "pulse" setting on your function generator --- and this is not the same as PWM.

Limitations of an RC servo is that it has a fixed range of angles, and usually is not able to do continuous rotations. Ranges are typically within 0-180 degrees, and some from 0 – 90 degrees. Furthermore, an RC servo is always "on" and cannot passively hold an angle: it must continue to see a PWM signal to stay at that angle. An RC servo relies on potentiometers to measure the angle, and overtime will wear out. Finally, speed of motion is predetermined by an onboard controller, and thus while you can control angle, you cannot control speed.

Benefits of an RC servo is the simplicity in its use and its low cost in comparison to high end servomotors with optical encoder feedback.

Additional Background Reading on RC Servos and varieties (Optional):
http://www.rchelicopterfun.com/rc-servos.html

2.3.3.d.1 Design

You will build a (very) simple circuit to control the angle of your RC servo.

- Using a function generator as a signal input, draw the circuit that you will use in your design.
- What is the frequency of the input waveform required to drive the RC servo?
- Is the voltage rails to the RC servo part of the logic supply or a high-load supply?

2.3.3.d.2 Build

- Build the circuit and verify that you can control the angle.

2.3.4 Putting it all together: Arduino basics—reading inputs and controlling outputs

In this activity, you will get an Arduino microcontroller running on a PC. You will then program the Arduino microcontroller to control an RC servomotor and a DC motor speed according to the real-time setting of two potentiometers.

- Learn how to read analog inputs and set outputs of an Arduino microcontroller
- Learn how to create a program in an Arduino to adjust outputs based on inputs

2.3.4.a Arduino programming

View the Arduino main website (arduino.cc) for relevant and useful background reading. You should explore these pages and explore further on your own. Below are some useful commands for Arduino; see the Arduino Language Reference for a complete list.

- pinMode (pin, 'input') or pinMode (pin, 'output') to designate the pin mode
  - Unnecessary for analog input pins
- analogRead(pin) reads the value of an analog pin. This is a 10 bit read so it varies between 0 and 1023.
- digitalRead(pin) reads the state of a digital pin, will be either 0 or 1
- digitalWrite(pin, 0) or digitalWrite(pin, 1) to set a digital pin high or low
- analogWrite(pin, value), to set the value of an analog output pin. Value is an 8 bit number ranging from 0 to 255 that maps to a voltage of 0-5V. If the pin is one of the PWM designated pins, the output will be a PWM signal ranging from 0-5V with a duty cycle calculated by: 100*value/255, i.e. a 50% duty cycle is roughly 127

In this section, we will drive the three actuators (solenoid, RC servo, and DC motor) on the Arduino. Produce the following setup in the Arduino code:

- Set A0 to read the voltage across a potentiometer and write a PWM to your servo motor that allows it to sweep the full range of motion. Demonstrate the functionality to a TA.
  - *be careful about this --- consider if the potentiometer is turn to 0 Ohms, and how much current will be generated in the circuit if there was no resistance? You will want to put a small resistor in series with the potentiometer such that nearly all of 0-5V is achieved, while calculating the power dissipation to ensure you do not exceed the 1/4W limit of the resistor*.
- Set A1 to read the voltage across another potentiometer and write a PWM to your DC motor to set the speed. Demonstrate the functionality to a TA.
- Do Arduino pins "float" in value or are they pulled up/down by an internal resistor? Set D3 to be an input so that when it is LOW, it will turn D4 HIGH and drive the solenoid circuit; when it the input is HIGH, it will turn D4 LOW to turn off the solenoid circuit. Demonstrate the functionality to a TA.

Tips

- Make sure to connect one of the ground pins on the Arduino to the ground on your breadboard!
- Reminder to keep your circuits clean, compact and intact – they will be used directly in your pinball machine.

2.3.5 Pinball RC Servomotor/DC Motor Application Design Cycle

This activity of Module 2 applies the skills and practice from 2.3.1-2.3.3 to the definition and creation of a motorized application (servomotor or DC motor) in the pinball machine. This could be a door, a carousel, etc. **KEEP THIS CIRCUIT ON YOUR BREADBOARD WHEN YOU ARE DONE** – you can use it directly in the pinball machine (also, plan to use your breadboard space efficiently), as you will continue to add more and more circuits throughout the course – additional breadboards are available but are limited).

You and your partner will create a motorized component of your choice for your pinball machine along with appropriate documentation of the following steps in this cycle, including: associated system requirements, block diagram(s), design specifications, prototypes, test measurements, and verification/demonstration of specifications and requirements.

2.3.5.a Pinball RC Servomotor/DC Motor Application Design Cycle: Plan

- Draw by hand a circuit design for the system that can meet all the requirements. You may, instead, refer to previous circuits you have drawn rather than duplicate drawings but be sure to note any differences in component values.
- Sketch your design by hand.
- Draw (using CAD tools) a mechanical design for the motor with any additional mechanical components
- Write electrical and/or mechanical design specifications for the motor subsystem.
    - E.g., 'the maximum voltage the motor receives is ~X volts'
- Use the specifications/metrics to create a verification testing plan that you will execute in section 2.3.4.c.
- Lead a design review with the teaching staff before beginning to build your flipper.

2.3.5.b Pinball RC Servomotor/DC Motor Application Design Cycle: Build

- Fabricate mechanical components and assemble
- Fabricate circuits on a breadboard according to the plan/layout

2.3.5.c Pinball RC Servomotor/DC Motor Application Design Cycle: Test

- Perform circuit subsystem verification tests, based on specifications/plans in 2.3.4.a.
- Record appropriate data to document test results

2.3.5.d Pinball RC Servomotor/DC Motor Application Design Cycle: Assess

- Analyze verification test results
- Create notes for potential redesign (in lab notebook/digital format)

2.3.6 Pinball Flipper Testbed Design Cycle

This activity of Module 2 applies the skills and practice from 2.3.1-2.3.3 to the creation of the pinball flipper testbed began in Module 1 while following a coherent system design cycle through at least one iteration of Planning, Building, Testing, and Assessing.

You and your partner will complete the pinball flipper testbed along with appropriate documentation of the following steps in this cycle, including: associated system requirements, block diagram(s), design specifications, prototypes, test measurements, and verification/demonstration of specifications and requirements. Keep your circuit on your breadboard when you are done since this will be you final circuit (unless you wish to change it in the future).

2.3.6.a Pinball Flipper Design Cycle: Plan

- Draw by hand a circuit design for the system that can meet all of the requirements. You may, instead, refer to previous circuits you have drawn rather than duplicate drawings but be sure to note any differences in component values.
- Sketch your design by hand.
- Draw (using CAD tools) a mechanical design for the flipper testbed
- Write electrical and/or mechanical design specifications for the flipper subsystem.
    - E.g., 'the maximum voltage the actuator receives is ~X volts'
- Use the specifications/metrics to create a verification testing plan that you will execute in section 2.3.5.c.
- Lead a design review with the teaching staff before beginning to build your flipper.

2.3.6.b Pinball Flipper Design Cycle: Build

- Fabricate mechanical components and assemble
- Fabricate circuits on a breadboard according to the plan/layout

2.3.6.c Pinball Flipper Design Cycle: Test

- Perform circuit subsystem verification tests, based on specifications/plans in 2.3.5.a.
- Record appropriate data to document test results

2.3.6.d Pinball Flipper Design Cycle: Assess

- Analyze verification test results
- Create notes for potential redesign (in lab notebook/digital format)

Free and have extra time? You may built extra circuits from this module you know you will use for your pinball machine and test/verify they work so that you don't need to do this later on when you are trying to complete the entire machine.