ENGR 40M Project 2b: Making the Useless Box More Awesome

Prelab due 24 hours before your section, July 17–20
Lab due before your section, July 25–28

1 Objectives

You've built your first useless box, spent countless hours playing with it and impressed your friends with your fun toy. You might, justifiably, be asking yourself, "how can we make this even better?" In this lab, we'll do just that—by adding a computer!

Computers are everywhere in electronics. It's not just your computer or even your phone. Countless everyday objects have a microcontroller in them—fridges, door locks, pacemakers, dot-matrix displays and cars (where you'd find several dozen), to name a few. These microcontrollers interact with the physical world, vastly expanding what we can design as engineers.

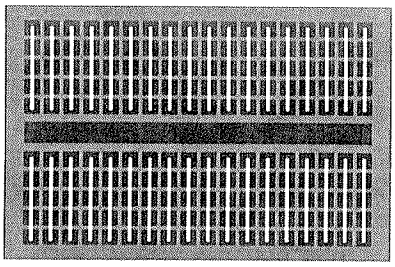
By the end of this lab, you should be able to

- Use MOS transistors to make a driver circuit, interfacing between an Arduino and a motor
- Program your Arduino to interact with physical inputs (switches) and outputs (motor) via digital pins
- Understand finite state machines and use them to organize your code effectively

2 Parts

2.1 Breadboard

A breadboard is a jig that makes it convenient to temporarily build and prototype circuits. It contains lots of pinholes into which you can plug wire-leads. Each row of five pinholes, indicated in yellow on the diagram on the right, is electrically connected, allowing you to connect up to five leads to a single node. Each row is electrically isolated from the rest of the board, allowing you to prototype large, multi-node circuits.



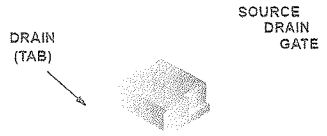
In this lab, due to the limited space available inside your box, we'll use the smallest breadboards.

2.2 MOS transistors

A metal-oxide semiconductor field effect transistor (MOSFET), or MOS transistor, is a three-terminal device that can be thought of as an electrically-controlled switch. (It's more complicated than that, but for our purposes, this is a good model.) We've studied this in class, so for details, refer to the course reader. To refresh your memory:

- An nMOS turns on when the gate is *higher* than the source by at least the threshold voltage.
- An pMOS turns on when the gate is *lower* than the source by at least the threshold voltage.
- Typically, we connect the source of an nMOS to ground, and the source of a pMOS to V_{DD}.

MOSFETs come in lots of different forms. We'll be using power transistors, designed to deliver high current: the RFP12N10L (nMOS) and NDP6020P (pMOS). Both devices use the *TO-220 package*, so they look similar—read them to tell them apart. The pinout is shown below, and is the same for both devices.

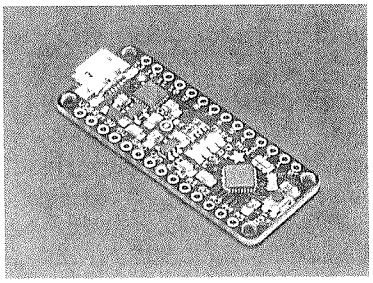


Some words of caution:

1. Transistors can make a short circuit if you connect them incorrectly. If this happens, your computer will (hopefully!) shut down its USB port in order to protect itself. So if nothing looks like it's working, measure V_{DD} to see if that's what happened.
2. **Floating gates can turn a transistor on!** We saw this in homework: Many students connect the source and drain appropriately, but then leave the gate disconnected. Don't do this! This can cause both the pMOS and nMOS to turn on simultaneously, short-circuiting your power supply. Always be sure to **actively** drive the gate of every transistor in your circuit, even during testing.[1]

   In particular, this includes *Arduino pins configured as inputs*, since they look like an open circuit.

2.3 Arduino



Arduino is a popular open-source electronics platform that eases the process of programming a microcontroller to interact with your circuit and the physical world. We'll be using a variant called the *Adafruit Metro Mini*, manufactured by Adafruit.

You'll need to install the Arduino IDE, the software that allows you to load code onto the Arduino, from `http://arduino.cc/`. The Metro Mini derives from the Arduino Uno, so select "Arduino Uno" in the IDE.

The Arduino interfaces with electric circuits via *input/output pins*, or *I/O pins*. In your code, you use the `digitalRead()` and `digitalWrite()` functions to interact with them. Input pins take almost no current, like a voltmeter. As for output pins, we'll learn about their *output resistance* in the prelab.

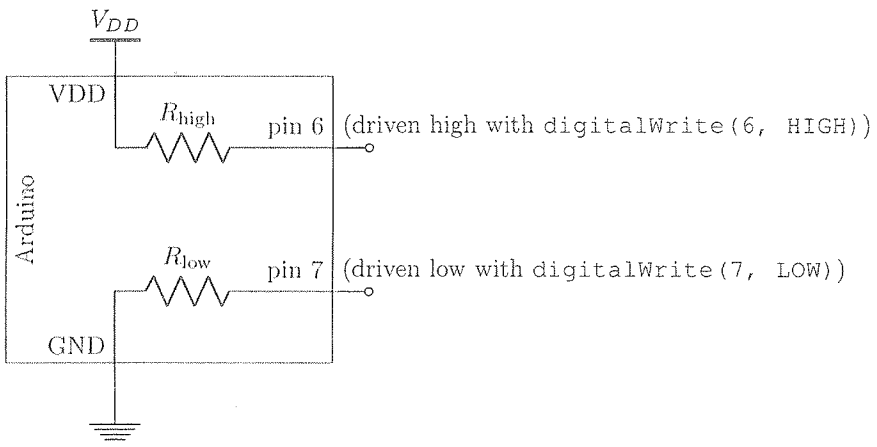
3 Prelab

3.1 Arduino output resistance

In this section, we'll measure the *output resistance* of an Arduino I/O pin. The output resistance is the resistance "seen" by a load on the output, when it is connected to V_{DD} or ground. For example, in the schematic below, pin 6 is *driven high* (set to `1`) and pin 7 is *driven low* (set to `0`). We can think of the output resistance when driven high as R_{high}, and the output resistance when driven low as R_{low}.[2]

[1] In practice, for reasons beyond the scope of ENGR 40M, a floating gate tends to "remember" the last voltage that was connected to it. However, you should never rely on this behavior, because it's unreliable and sensitive to brief accidental contacts.

[2] Actually, the output pins are themselves controlled by pMOS and nMOS transistors, which aren't shown in the figure, and the resistance is just the resistance of those transistors.



To get started, connect your Arduino to a USB port, and try loading one of the sample programs to make sure everything is in order.

P1: Choose your two favorite I/O pins, but don't use pins 0, 1 or 13. Write a simple program that configures both of them to be outputs using `pinMode()`, then drives one of those pins to `HIGH` and the other to `LOW`. Load the program into your Arduino.

This program should be four lines long. You don't need to submit it with your prelab.

P2: Measure the output resistance of the I/O pin that you drove high.

Note: Because this is actually a transistor, the polarity of your ohmmeter matters. The ohmmeter measures resistance by pushing a small test current from the positive lead to the negative lead, so you need to make sure this current would go in the "correct" direction. In this case, this means putting the positive lead on VDD, and the negative lead on your I/O pin, so that the test current goes from VDD to the I/O pin.

4kΩ

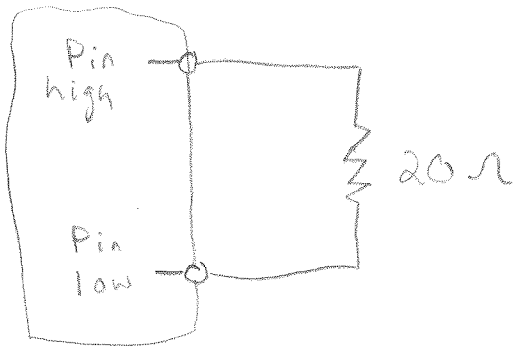
P3: Measure the output resistance of the I/O pin that you drove low.

Again, the polarity of your ohmmeter matters—think through which way the test current should run.

resistance too high to measure

A naïve way to control the direction of your useless box's motor would be to connect it directly between two I/O pins. To drive the motor one way, you set one pin to `HIGH` and the other to `LOW`; to drive it the other way, you set the pins the other way round.

P4: Draw a model of the circuit that would be formed if you connected the motor directly between two I/O pins, one of which was driven to high and the other of which was driven to low. Model the motor as a 20 Ω resistor.[a]



[a]It's not actually a resistor, but this gives us some idea of the load it provides when it's running continuously.

P5: Use the model you just drew to estimate the maximum current that this design would be able to provide to the motor. Compare this to your measurement of motor current from prelab 2a. Would this design work? Why, or why not?

20Ω + 4kΩ + high resistor = very low current

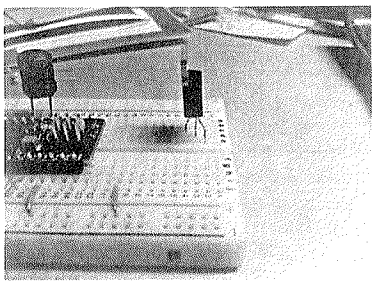
The model wouldn't work

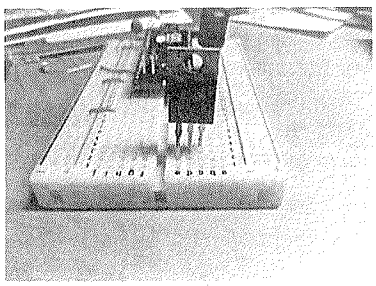
3.2 MOS transistor resistance

Our goal in this section is to measure the *on resistance* and *off resistance* of your power MOS transistors.

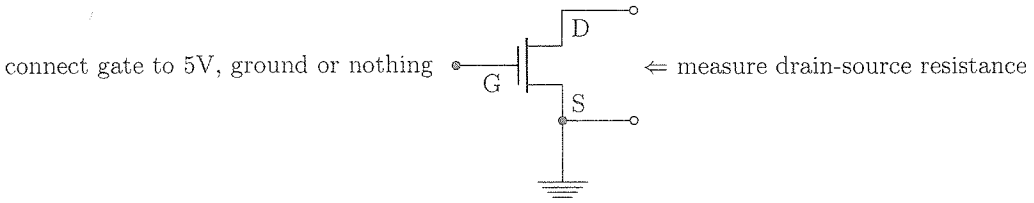
To do this, you'll need to connect a breadboard to a power supply. The easiest way to do this is to use the USB cable you cut in half during lab 1 to power the breadboard directly from any USB charging port (*e.g.* on your computer, a phone charger or your solar charger from lab 1). Alternatively, you could plug your Arduino into the breadboard, connect the Arduino to your computer, and connect the VDD and GND pins of the Arduino to the side rails on the breadboard.

Plug your nMOS power transistor (RFP12N10L) into the breadboard. *Tip: You might find that they plug in easier if you plug the device in so its wide body is parallel with the rows of five, and you push one lead forward and one lead back so the different pins go into different rows, as in the picture below.*





To measure the on and off resistances of the nMOS transistor, we'll want to connect the source to ground, so that we can control the gate-source voltage. The gate connects to 5V, GND or nothing (floating), depending on the measurement. Do **not** connect the drain to anything other than the ohmmeter.



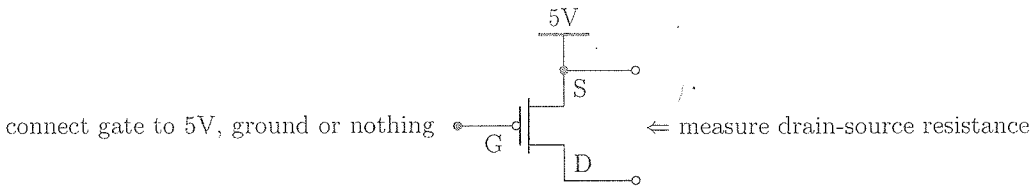
Important measurement technique: All wires have some resistance, including the leads of your multimeter. This means that even when you touch the two probes together, you won't measure zero resistance. Since this is typically less than 1 Ω, it can usually be ignored. But if you are measuring a small resistance, you should measure this baseline resistance first, and subtract it from the resistance you measure.

P6: What is the drain-source resistance of the nMOS transistor when the gate is (a) connected to 5V, (b) floating and (c) connected to GND?

As for earlier questions, the polarity of the ohmmeter matters: you should ensure the test current would go in the "correct" direction for the transistor.

a. 1.2Ω − .9 = .3Ω
b. greater than 2000Ω
c. greater than 2000Ω

Now, do the same thing with the pMOS (NDP6020P). Recall that pMOS sources connect to V_{DD}.



P7: What is the drain-source resistance of the pMOS transistor when the gate is (a) connected to 5V, (b) floating and (c) connected to GND?

a: Greater than 2000Ω
b: 148Ω
c: 148Ω

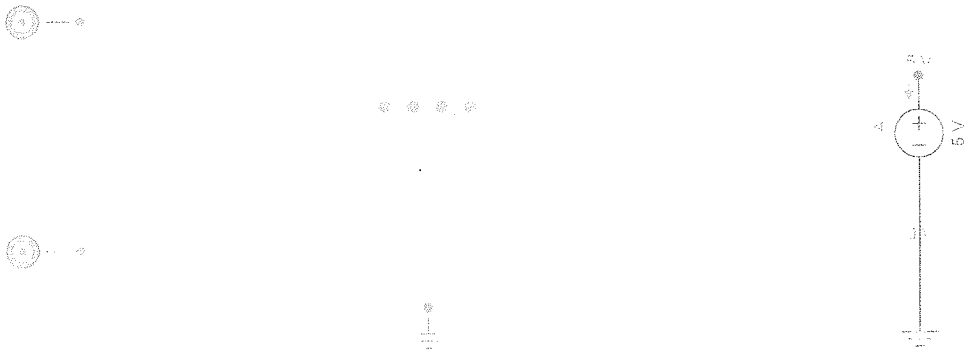
3.3 Designing the circuit

Having established that we need these "power" transistors to interface between the Arduino and the motor, let's design the circuit that drives the motor. The motor needs to be able to do *three* things: go forward, go backward, and stop. This means that we'll need two output pins to drive the motor, via a circuit that uses

power MOS transistors. By changing the states of these output pins (in Arduino code), you should be able get the motor to do each of those three actions.

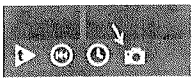
We've created a starter circuit on EveryCircuit that contains the parts you need. You can search for it using the search term "ENGR 40M prelab 2b starter" (or just "ENGR 40M"), or use this URL: `http://everycircuit.com/circuit/5069979055816704`.

You should see the circuit shown below:



- The two circles on the left are "logic sources", which represent your Arduino output pins. If you click on them, they'll change from 0 (low) to 1 (high) and vice versa.
- The circular item in the middle is the motor. When current goes through it, it animates either clockwise or counter-clockwise.
- The 5 V source on the right represents V_{DD}.

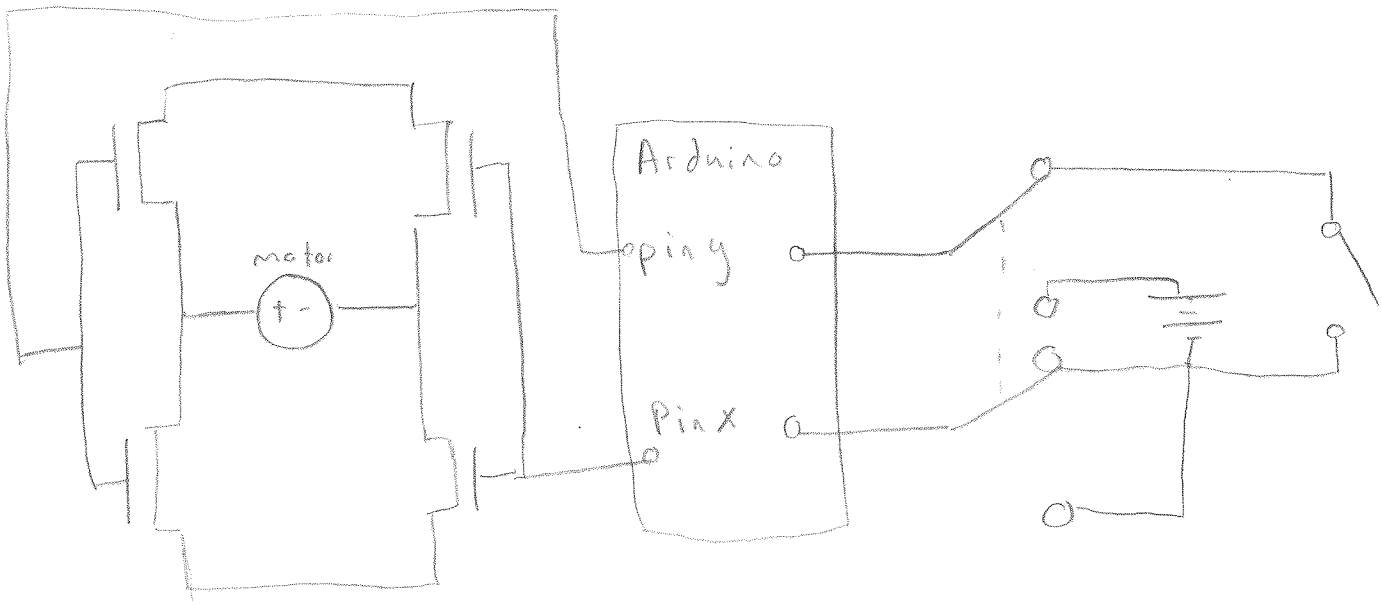
P8: Complete the circuit on EveryCircuit. By clicking the logic sources into appropriate combinations (in the middle of a simulation), you should be able to get the motor to spin clockwise, spin counter-clockwise and stop. It should also be impossible to short-circuit the 5 V source. Please attach a screenshot of your completed circuit to your prelab submission.

This button might be useful: 

That's most of the work needed to design the circuit. Now we just need to add the switches and be a bit more explicit about how the Arduino fits in.

P9: Draw a complete schematic of the entire circuit you need to build in the lab to make your More Awesome Useless Box. This should include the motor driver, the two switches in your basic useless box, the Arduino, batteries and it should indicate which Arduino pins the motor driver and switches connect to.

Please submit this with your prelab, and bring a copy to lab, since it will guide you as you make your More Awesome Useless Box a reality.



3.4 Programming the Arduino

It makes your life easier as an engineer—and as a maker—if you build things up step by step. This makes debugging much easier: if something doesn't work, you know it should be the thing you just added. As far as the Arduino goes, here's our plan of attack:

1. First, we'll verify that we can read our switches correctly (without touching the motor).
2. Then, we'll verify that we can drive our motor correctly (without touching the switches).
3. Then, we'll write a program that replicates our original useless box, incorporating both the switches and the motor.
4. Finally, we'll extend the program to make the useless box more awesome.

Before you come to lab, we ask you to write two simple programs, for steps 1 and 2 respectively. Once you come to lab and have built the improved box, we'll continue with step 3. Your homework for the week after the lab is to do step 4.

A useful reference for the Arduino language can be found at `https://www.arduino.cc/en/Reference/HomePage`.

A few reminders:

- You need to use `pinMode()` to set whether each pin is an input or an output, and if it's an input, whether the pull-up resistor is enabled.
- You have two output pins to your motor driver circuit, which **both** need to be set whenever you want to change direction or stop the motor.

```
void loop() {
  byte user_input = 0;

  if (Serial.available() > 0) {
    user_input = Serial.read();

    if (user_input == 'f') {
      Serial.println("Forward");
      pinMode(3,HIGH);
      pinMode(2,LOW);
    } else if (user_input == 'r') {
      Serial.println("Reverse");
      pinMode(2,HIGH);
      pinMode(3,LOW);
    } else if (user_input == 's') {
      Serial.println("Stop");
      pinMode(2,LOW);
      pinMode(3,LOW);
    } else {
      Serial.println("Invalid character");
    }
  }
}
```

```
void setup() {
  // put your setup code here, to run once:
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
}

void loop() { }
```

```
void setup() {
  Serial.begin(115200);
  Serial.println("Setup completed");
}


void loop() {
  // put your main code here, to run repeatedly:
  int val2 = 0;
  int val3 = 0;
  val2 = digitalRead(2);
  val3 = digitalRead(3);
  Serial.println(val2);
  Serial.println(val3);
}
```