MOSFET application for control

The purpose of this exercise is to explore the application of MOSFETs in controlling the power setting of a device. This exercise provides hands-on experience with MOSFETs for managing the state, (on/off) brightness of an LED light source, and speed control for a DC motor. You will use Gradescope (GS) to enter your responses.

Figure 1 depicts a circuit with a 20 milliamp (mA) LED in series with a resistor powered by a 5 V from Arduino. The LED chip has a rating of 20 mA and a forward voltage (V_f) of around 3 volts (V). This means, the LED requires a minimum of 3 V to activate at sufficient brightness, drawing a current (I) of 20 mA. The resistor exists to limit the current and ensure that you do not burn out the LED. Once you choose a known voltage, (V_s – supplied voltage), the resistor value can be calculated using the formula:

$$R \text{ (ohms)} = \frac{V_S - V_f \text{ (Volts)}}{I \text{ (Amps)}}$$

(GS) When using 5V from the Arduino with the LED that draws 20 mA and has a forward voltage of 3V, what resistor should you use?. R = 100Ω **(ohms)**

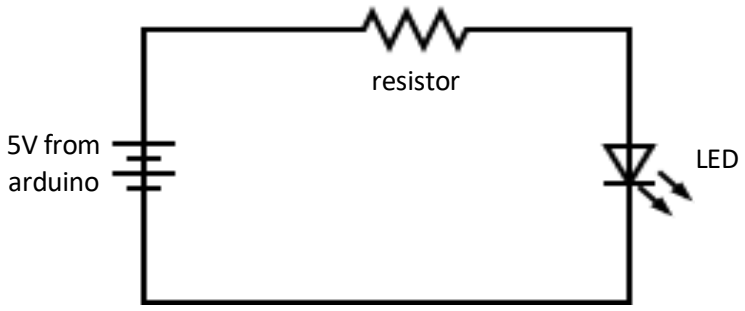

Figure 1.

Set up the circuit in Figure 2, below, using the **1.5-volt battery**. (GS) When using a very simple circuit and the 1.5V battery, does the LED work? (GS) Explain what is happening with the simple circuit powered by the 1.5V battery. Why is the LED doing what it is doing.?

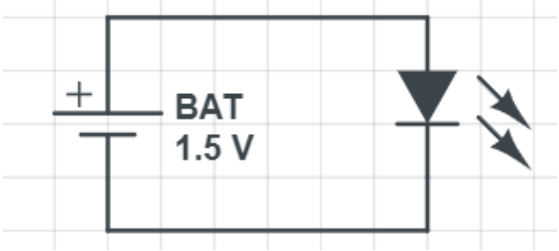

Figure 2

MOSFET application for control

Set up the circuit as shown in Figure 1 to power the circuit with 5V from Arduino and introduce the 100-ohm resistor.

a. Use the multimeter: (GS) what is the voltage drop across the resistor?
b. Use the multimeter: (GS) what is the voltage drop across the LED?
c. (GS) Why is the LED now lit properly?

To regulate the LED's brightness, we manage the current passing through the LED chip. This can be achieved by incorporating a potentiometer (100 ohms) in series, as depicted in Figure 3. By turning the knob such that it increases its resistance value, we elevate the overall resistance in the circuit, effectively governing the current flow and consequently adjusting the brightness. However, using a potentiometer necessitates manual intervention and lacks automation. To automate this procedure, an Arduino board and a MOSFET can be employed, as illustrated in Figure 4.

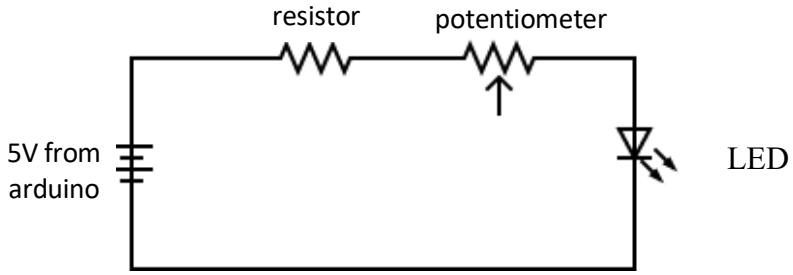

Figure 3

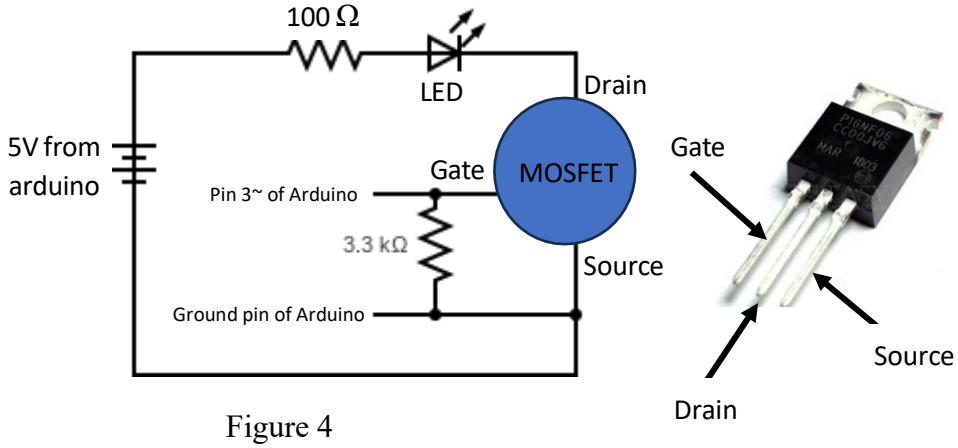

Figure 4

MOSFET application for control

We know the UNO can supply 5V which is more than enough voltage. However, the UNO can only provide a maximum of 500 mA when connected via USB to a laptop. Some of this current is used internally, limiting the available output current to less than 500 mA. The current limit may vary between different manufacturers and models of Arduino boards. Attempting to draw more current can potentially damage the Arduino board. It would probably work for the single LED in this activity which requires 20 mA. However, for high-wattage LEDs, multiple LEDs, LED matrix, motors etc., it is advisable to use an external power supply. As such, we use an external power supply in the motor activity.

Your task is to develop the circuit shown in Figure 4 and to program the Arduino/MOSFET in order to achieve desired outcomes. In the activity box you will find an Arduino, a 100-ohm resistor, a 3.3 kΩ resistor, an N-Channel MOSFET (P16NF06), 1.5X2-volt battery, battery holder, LED, a breadboard, and a DC motor with propeller. Perform the following exercises:

Set up the circuit as shown in Figure 3 utilizing the MOSFET.

a. Write a simple code that sends low and high for some number of seconds to the pin connected to the MOSFET.

```
void setup() {
        pinMode(3, OUTPUT);
}

void loop() {
        digitalWrite(3, LOW);

        delay(5000);

        digitalWrite(3, HIGH);

        delay(5000);

}
```

b. Upload your code. You will power the Arduino with your computer. Check that the LED turns off and on for the correct amount of time.
c. Create your own pulse width modulation (PWM) to get the LED to be about medium brightness. The duty cycle is the ratio of the time the current is flowing to the total time it is flowing and not flowing. This ratio controls the PWM. You can set the delays to be much less than 5000 ms, think on the scales of single ms. Try a few different duty cycles. Record your delay settings and comment on the LED's behavior and any readings that appear on the screen.
    i. (GS) What delay combo gives fully on? Low / High, value in ms
    ii. (GS) What delay combo gives medium (50%) brightness? Low / High

d. There is another option to use rather than manually using the delay command. Arduino includes the analogWrite command, which contrls the PWM of the set pin. Use the PWM capability already available in the analogWrite command. The information provided for analogWrite states:

> Syntax: analogWrite(pin, value)
> // pin: the Arduino pin to write to -- allowed data type: int.
> // value: the duty cycle: between 0 (always off) and 255 (always on) -- allowed data type: int.

The code you should write will look like:

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {       //set up a duty cycle values
analogWrite(3 , 127);  // duty cycle 50%
}
```

(GS) What is the lowest value for analogWrite() that can still illuminate the LED?

MOSFET application for control

Set up the circuit shown in Figure 5 with a 3V external power supply and a DC motor. (NOTE: Please do not use 5V from Arduino and instead only use 3V from an external battery.) Use a tape to mount the fan and its motor on the side of the table as shown in Figure 6.

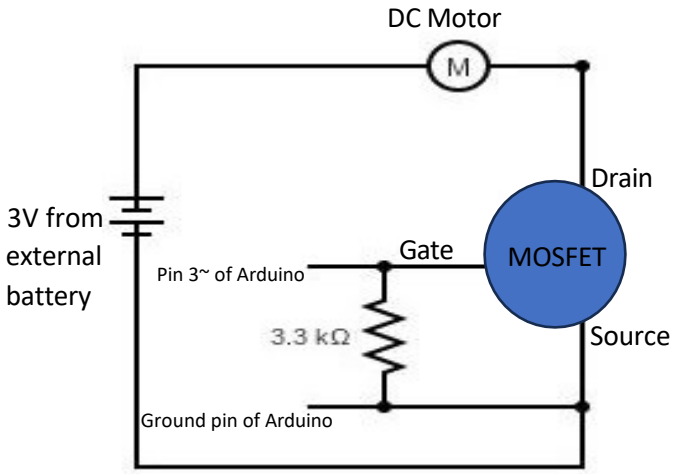

Figure 5

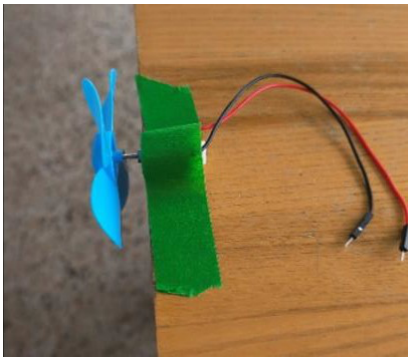

Figure 6

a. (GS) What analogWrite() value gives you the fan's full speed?
b. (GS) What analogWrite() value gives you fan at about half speed? (50%)
c. (GS) What analogWrite() value is the threshold at which the fan barely spins?