print name (first last): _____

course: ECET 337 2025-07-27

Motor Characteristics & the Arduino Motor Driver and Sensors Interface

Should be completed by two students working together.

Prelab Activities

_____ Answer the questions on the following page. (20%)

Performance Checks

			In lab	or	Late
_____	2.	Initial Functionality	(30%	or	15%)
_____	4.	Step Response	(50%	or	25%)

Lab Performance Scoresheet

Prelab … (20%)→ _____

All required signed ***Performance check offs*** **(above)** …… (80%)→ _____

NO report is required.

Total→ (out of 100%) _____

Comments:

Prelab Activity

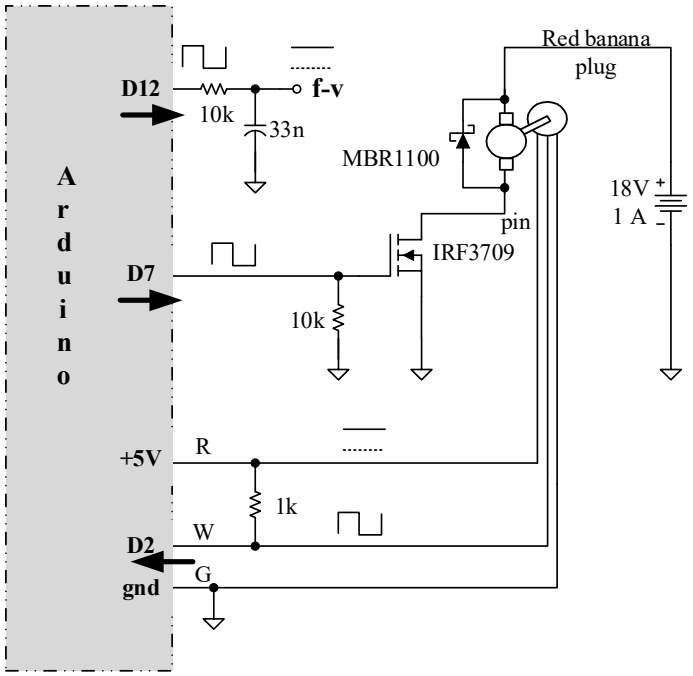

Figure 1 Arduino motor driver and sensors interface schematic

Prelab Activity

1. Assuming that the motor's maximum speed is 28.3 rev/sec (1700 RPM) and that the encoder provides 2048 pulses per revolution as a 50% duty cycle logic signal, prove that the maximum input frequency from the encoder at A or B is approximately 58 kHz. Show all of your work and be sure to include units, showing how RPM turns into rev/sec turns into pulses per second or Hz.

2. The pulses from the encode are counted for 10 ms. That count is accumulated in the variable **encoder** by the program.
   a. When the motor is turning at full speed (1700 RPM and 58 kHz from the encoder), what value is held in encoder at the end of the 10 ms counting interval?

      encoder = _____

   b. How can the shaft speed in RPM be calculated using this value in **encoder**?

      RPM = **encoder** _____

3. What is the waveshape at the Arduino's PWM output, pin 7? _____

4. What are the software instructions necessary to set the pin 7 PWM output to 50%? {Hint: Look at the code on page 6&7.}

5. When the signal from the PWM pin is high, what is the condition of the
   a. MOSFET? On or off (circle one)

   b. Voltage at the Motor top (red banana plug) terminal? _____
      bottom (pin) terminal? _____

   c. Is the motor spinning?

6. a. What is the purpose of the diode?

   b. What would happen if it were missing?

   c. What would happen if it were reversed?

Objectives

Configure and verify the performance of an Arduino driving a DC motor with the Purdue-ECET-I/O-Shield.

Hardware Approach and Results

1. System Configuration

a. Get your Purdue-ECET-I/O-Shield. Carefully align the pins with your Arduino and then mate the two, as shown in Figure 3.

b. Do *not* force the mating. Get help if you are at all uncertain.

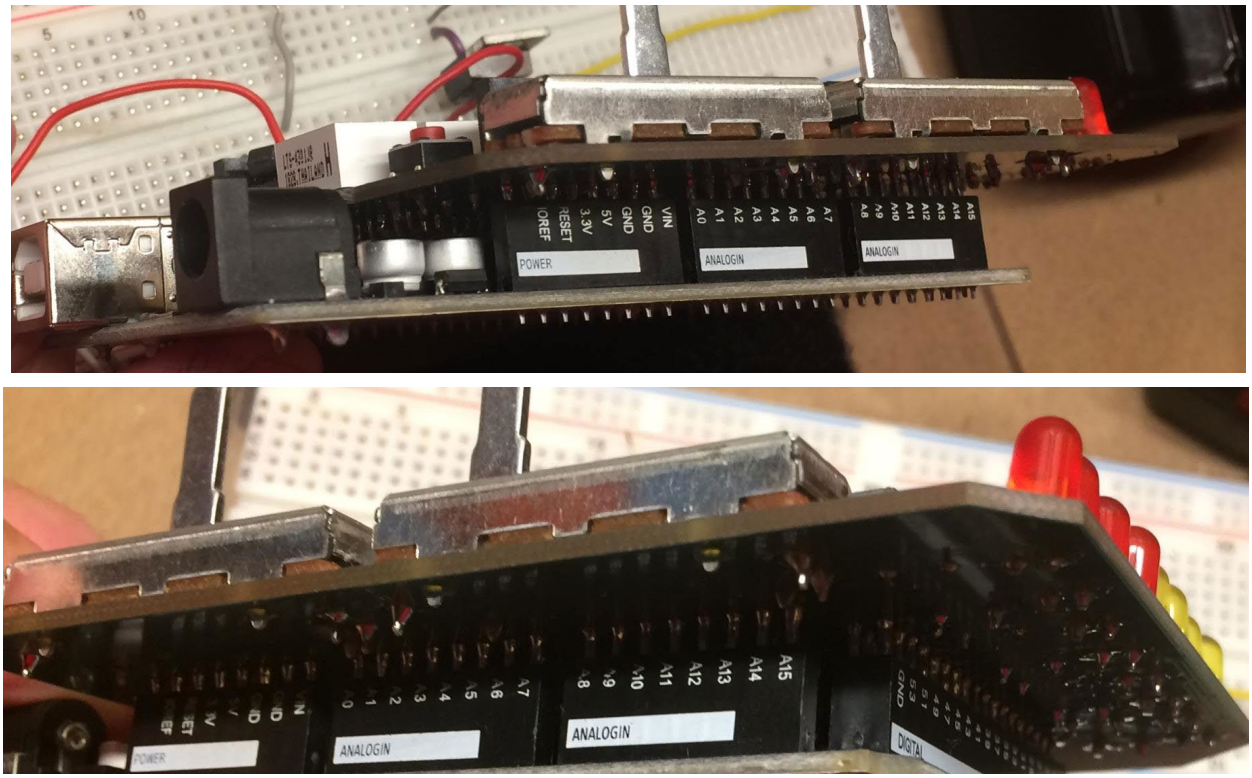

Figure 3 Aligning the pins

c. If you have a base to hold the Arduino, connect the Arduino-and-Shield into the base. This will provide a stable assembly. Otherwise, securely tape the Arduino-and-Shield to the middle of your workspace.

d. Assure that the potentiometer on the shield is fully to the left. This sets the A0 voltage to 0 V_{DC}, which will cause 0 PWM when the software is loaded and begins running.

e. Assure that the four white switches on the shield are to the ***right***. This disables them, assuring that they do not interfere with the code's operation.

f. Connect the rest of the hardware as shown in Figure 4, *in this order*:

1) *All* connections between the Arduino and the protoboard's circuitry.

2) The motor's encoder cable into +5 V (red), D2 (white) and common bus (green).

3) The oscilloscope's common, the Arduino common, the power supply common, the multimeter's common, and the oscilloscope probe's common to the GND bus.

4) The oscilloscope probe to the **D7** (MOSFET gate drive) signal. (The photograph shows the scope probe at a different location. You will move the probe to the shown location later.)

5) The 18 V supply's common to the common bus. The +18V from the power supply to the motor's red banana plug with the lead to the flyback diode. Then, turn the +18 V power supply on.

6) The Arduino's USB cable to the computer.

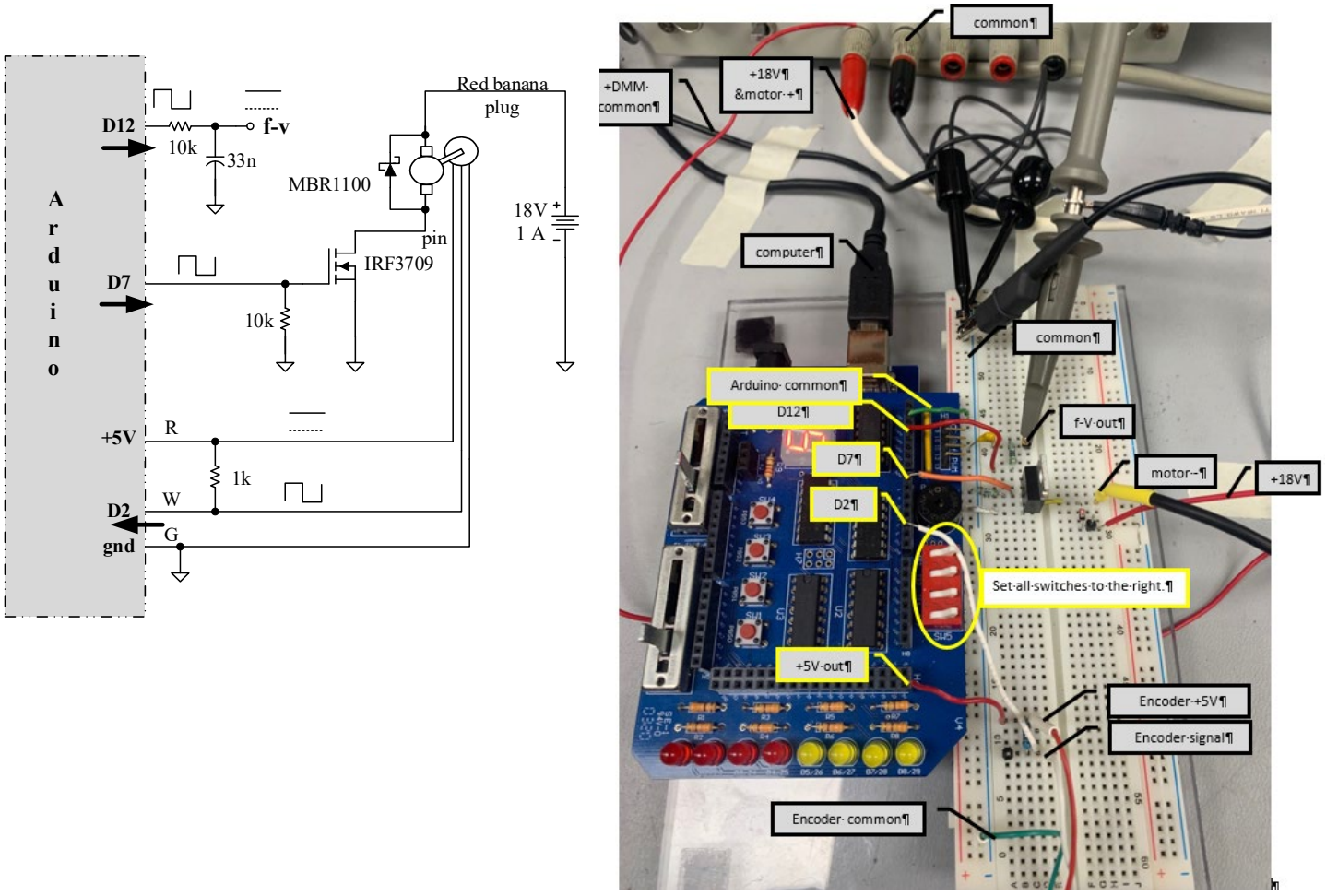

Figure 4 Hardware and instrument connections

g. Download the Sketch_Board_Static Motor Driver program from the course website, in this lab's folder.

```
/*****************************************************************
  Author: Joshua Bell
  Modified by: Baylen Jobe, Joe Bormann
  Frequency generator with PWM D12
  Display Duty cycle on 7 seg display
  PWM to drive motor D7 @ 1kHz
  Return from encoder to D2 (input)

*****************************************************************/

//// set this value to desired high frequency (50kHz) pwm of fV frequency
#define Frequency 50000  /// must be defined as:50000

int encoder=0;       //encoder value counting for 10 ms
int RPM=0;           //calculated from the value in the encoder variable
int count;           //used in the 50 kHz f-V routine
int fV = 50;         //sets the pw of the 50 kHz for f-v   0 to 100 for 0V to 5V out.  Keep max at or below 80
int Duty_cycle;     //desired dutycycle to motor from A0 potread
int Perdisplay;     // 0 to 99  %  value to 7 segment displays

int CO = 0;
const int LBL = 44; //left !blank
const int LLT = 43; //left !lamp test
const int LLE = 45; //left latch enable
const int RBL = 41; //right !blank
const int RLT = 40; //right !lamp test
const int RLE = 42; //right latch enable



void setup()
{
 pinMode(7,OUTPUT);   //pwm to drive motor (1kHz)
 pinMode(2,INPUT);      //Encoder Channel A 2048 pulses per rev

 pinMode(LBL,OUTPUT);
 pinMode(LLT,OUTPUT);
 pinMode(LLE,OUTPUT);
 pinMode(RBL,OUTPUT);
 pinMode(RLT,OUTPUT);
 pinMode(RLE,OUTPUT);

 attachInterrupt(0,freq,RISING);   //interrupt to count every rise on encoder channel A step up
 Serial.begin(9600);


timer_setup();
 DDRC = 0xFF; //pins 30 - 33 is MSB 7seg pins 34-37 is LSB 7seg
 digitalWrite(RLT, HIGH);  //setting !BL and !LT high for both ICs
 digitalWrite(RBL,HIGH);
 digitalWrite(LLT,HIGH);
 digitalWrite(LBL,HIGH);
 digitalWrite(LLE,LOW);  //setting LE LOW for both
 digitalWrite(RLE,LOW);
 PORTC = 0xFF; //displays a blank on 7seg
}



void loop()
{
//get and report the motor's speed
  encoder=0;
  delay(10);                    //wait for 10 msec while routine below counts pulses from the encoder
  RPM=encoder*2.93;       //scale 580 pulses/rev to 1700 RPM
  fV=encoder*0.14;          //1700RPM=>580pulses/10msec  This sets f-V to 80%
```

```
  OCR1B = (uint32_t)count * fV / 100; // Use this line to set 50kHz pulse width to speed

//read pot, report to 7seg display & serial monitor, drive motor
  Duty_cycle = analogRead(A0);   //read pot from 0-1023
  analogWrite(7,Duty_cycle/4);     //sets pwn pin 7 at pot value to drive the motor
  Perdisplay=Duty_cycle/10.24;   //convert that reading into a % 0-99
  display7seg(Perdisplay);

  Serial.print(Perdisplay);         //report pot position as %
  Serial.print("\t");                //tab
  Serial.println(RPM);              //report motor speed in RPM

}



// increment every encoder CH A rise to measure motor speed
void freq()
{
  encoder=encoder+1;
}




//set 50kHz pulse width as indication of motor speed
void timer_setup(void)
{
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _BV(WGM10) | _BV(WGM11);
  TCCR1B = _BV(CS11) | _BV(WGM12) | _BV(WGM13);
  count = (16000000UL / (Frequency)) - 1;
  TCCR1B = _BV(CS10) | _BV(WGM12) | _BV(WGM13);
  OCR1A = count;
}



//write % to the 7 segment display
void display7seg(uint8_t data)//load BCD and toggle !BL    Places % on 7 seg displays
{
  uint8_t rightnum = 0;
  uint8_t leftnum = 0;

  if(data > 9)
  {
    leftnum = data/10;
    rightnum = data%10;
  }
  else
  {
    leftnum = 0;
    rightnum = data;
  }
  leftnum = leftnum<<4; //sliding leftnum to most sig 4 bits
  PORTC &=0x00;
  PORTC |= leftnum|rightnum; //outputs to PORTC (7 Segs)

  digitalWrite(RLE, LOW);
  digitalWrite(LLE,LOW);  //toggle control pins to try to load value to 7seg
}
```

Figure 5 Sketch_Board_Static Motor Driver software

h. Compile and upload the program into your Arduino.

i. Open the Arduino's Serial Monitor on the pc.

j. Adjust the A0 potentiometer to 33%, then 67%, then 100%. At each, verify that the motor's speed increases and that the Serial Monitor displays appropriate data. (~ 850 RPM, ~ 1300 RPM, ~1700 RPM)

2. Initial System Functionality

a. Move the potentiometer to produce 50% on the display.

b. Measure with the multimeter and record the voltage at the Red motor terminal and at the $+5V_{DC}$ pin. Move on, only when these are correct.

$V_{\text{Red motor terminal – actual}}$ = _____ $V_{+5\text{VDC pin}}$= _____

c. Verify that the oscilloscope probe is measuring the PWM **D7** output pin, voltage to the MOS gate. Display two to three cycles of that waveform. Display on the oscilloscope its high level, low level, mean (DC), and frequency.

$V_{\text{PWM-D7 out high}}$ = _____ $V_{\text{PWM-D7 out low}}$ = _____ $V_{\text{PWM-D7 out DC}}$ = _____ $f_{\text{PWM-D7 out}}$ = _____

d. Move the *voltmeter* to the **pin** motor connection. This is the other end of the motor, connected to the MOSFET, not the end with the red banana plug. Record that voltage as the potentiometer moves from 0% to 99%

$V_{\text{motor pin input DC}}$ = _____ to _____

e. With the *multimeter*, measure and record the frequency at D12 and the DC voltage at the **f-*v*** output (after the resistor, across the capacitor) as the motor speed is adjusted from 0% to 99%. Also measure the V_{DC} across the capacitor at the **f-v** output.

$f_{\text{D 12 high speed}}$ = _____ $f_{\text{D 12 low speed}}$ = _____

$V_{\text{DC-f-v output high speed}}$ = _____ $V_{\text{DC-f-v output low speed}}$ = _____

Demonstrate the circuit's performance displays and the data to your lab instructor.

3. Static, Unloaded Performance

a. Complete Table 1. Do this in Excel as you take the data, so you can create two plots.

The frequency and all of the voltages, except V_{motor} are measured with the multimeter's common connected to the shield's GND pin.

To measure V_{motor}, you must move the multimeter's common to the pin from the motor, and measure *across* the motor, at the Red banana plug. Be sure to return the meter's common to the GND pin before moving to the next set of measurements.

Make all of the measurements in one *row* before changing V_{pot} and moving to the next row.

When you have gathered all of the data, set V_{pot} = 0 V, then turn off the 18 V supply.

Table 1 Arduino-shield-motor unloaded performance

	V_{motor}	f_A	speed	**f-v**
	Red to pin	D2	monitor	output
pot position %	V_{DC}	kHz	RPM	V_{DC}
0				
20				
40				
60				
80				
100				

b. Plot V_{motor} (x) versus *speed* (y). Properly label the axes, adjust the decimals on the labels, add grid lines, and a trend line with the equation shown on the plot.

Be sure to *save* this table and this plot. They characterize your motor and are needed later.

4. Step Response Performance

a. Download the Sketch_Motor_Step program from the course website, in this lab's folder.

```
/*********************************************************************
  Author: Joshua Bell
  Modified by: Baylen Jobe, Joe Bormann
  Frequency generator with PWM D12
  Display Duty cycle on 7 seg display
  PWM to drive motor D7 @ 1kHz
  Return from encoder to D2 (input)
 *********************************************************************/


//// set this value to desired high frequency (50kHz) pwm of fV frequency
#define Frequency 50000  /// must be defined as:50000

int encoder=0;       //encoder period
int RPM=0;
int count;           //used in the 50 kHz f-V routine
int fV = 50;         //sets the pw of the 50 kHz for f-v
int Duty_cycle;    //desired dutycycle to motor from A0 potread
int Perdisplay;    // 0 to 99  %  value to 7 segment displays

int CO = 0;
const int LBL = 44; //left !blank
const int LLT = 43; //left !lamp test
const int LLE = 45; //left latch enable
const int RBL = 41; //right !blank
const int RLT = 40; //right !lamp test
const int RLE = 42; //right latch enable


void setup()
{
 pinMode(7,OUTPUT);   //pwm to drive motor (1kHz)
 pinMode(2,INPUT);    //Encoder Channel A  2048 per rev

 pinMode(LBL,OUTPUT);
 pinMode(LLT,OUTPUT);
 pinMode(LLE,OUTPUT);
 pinMode(RBL,OUTPUT);
 pinMode(RLT,OUTPUT);
 pinMode(RLE,OUTPUT);

 attachInterrupt(0,freq,RISING);   //interrupt process every rise on encoder channel A step up
 Serial.begin(9600);


timer_setup();
 DDRC = 0xFF; //pins 30 - 33 is MSB 7seg pins 34-37 is LSB 7seg
 digitalWrite(RLT, HIGH);  //setting !BL and !LT high for both ICs
 digitalWrite(RBL,HIGH);
 digitalWrite(LLT,HIGH);
 digitalWrite(LBL,HIGH);
 digitalWrite(LLE,LOW);  //setting LE LOW for both
 digitalWrite(RLE,LOW);
 PORTC = 0xFF; //displays a blank on 7seg
}


void loop()
{
//get and report the motor's speed
  encoder=0;
  delay(10);                          //wait for 10 msec while routine below counts pulses from the encoder
  RPM=encoder*2.93;                   //scale 5800 pulses/rev to 1700 RPM
  fV=encoder*0.14;                   //1700RPM=>580pulses/0.1sec    set f-v at 80%
  OCR1B = (uint32_t)count * fV / 100; // Use this line to set 50kHz pulse width to speed
```

```
//read pot, report to 7seg display & serial monitor, drive motor
  Duty_cycle = analogRead(A0);     //read pot from 0-1023

if (analogRead(0) <512)               //use the following for the step test or steady 25% or 75%
  {
      Duty_cycle=255;                 //pot set below half => setpoint=25%
  }
  else                                //pot > half => setpoint = 75%
  {                                   // moving pot across mid point produces a step in SP fro 25% to 75%
      Duty_cycle=768;
  }

  analogWrite(7,Duty_cycle/4);    //sets pwn pin 7 at pot value to drive the motor
  Perdisplay=Duty_cycle/10.24;   //convert that reading into a % 0-99
  display7seg(Perdisplay);

  Serial.print(Perdisplay);        //report pot position as %
  Serial.print("\t");               //tab
  Serial.println(RPM);            //report motor speed in RPM

}


// increment every encoder CH A rise to measure motor speed
void freq()
{
  encoder=encoder+1;
}


//set 50kHz pulse width as indication of motor speed
void timer_setup(void)
{
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _BV(WGM10) | _BV(WGM11);
  TCCR1B = _BV(CS11) | _BV(WGM12) | _BV(WGM13);
  count = (16000000UL / (Frequency)) - 1;
  TCCR1B = _BV(CS10) | _BV(WGM12) | _BV(WGM13);
  OCR1A = count;
}


//write % to the 7 segment display
void display7seg(uint8_t data)//load BCD and toggle !BL    Places % on 7 seg displays
{
  uint8_t rightnum = 0;
  uint8_t leftnum = 0;

  if(data > 9)
  {
    leftnum = data/10;
    rightnum = data%10;
  }
  else
  {
    leftnum = 0;
    rightnum = data;
  }
  leftnum = leftnum<<4; //sliding leftnum to most sig 4 bits
  PORTC &=0x00;
  PORTC |= leftnum|rightnum; //outputs to PORTC (7 Segs)

  digitalWrite(RLE, LOW);
  digitalWrite(LLE,LOW);  //toggle control pins to try to load value to 7seg
}
```

Figure 6 Sketch_Motor_Step program

b. Compile and upload the program into your Arduino.

c. Open the Arduino's Serial Monitor on the pc.

d. Turn the 18 V power supply on.

When the potentiometer is below half-way, the motor should be driven at 25%. When the potentiometer crosses above half-way, the motor drive should step up to 75%.

e. Adjust the A0 potentiometer to about 1/3, and wait for the system to stabilize. Then, adjust the potentiometer to about 2/3, and wait for the system to stabilize. Finally adjust the potentiometer back to about 1/3. Verify that the motor's speed steps up, then back down and that the Serial Monitor displays appropriate data.

f. Move the oscilloscope probe to the **f-v** signal. Adjust the oscilloscope controls so that as you move the potentiometer above and below half-way, you can display a voltage near the bottom of the screen that rises to near the top of the screen. Adjust the time/div so that the rise fills about 2/3 of the horizontal (sweep time), with a little flat before and after each edge.

g. When the oscilloscope is properly set, perform a **single sweep**, to capture and hold the motor's speed rise.

h. Use cursors to measure the motor's rising time constant. Capture this display for your report.

Since the ***f-v*** output is in steps, you may have to estimate where in the proper step the time constant occurs.

τ_{rising} = _____

i. Adjust the oscilloscope time/div control to capture and hold the motor's speed fall.

$\tau_{falling}$ = _____

$\tau_{average}$ = _____

Demonstrate the circuit's performance, the oscilloscope displays, the data, and the captures.