LAB 4 - DC MOTOR CONTROL (H)

ITEMS NEEDED

1. Computer with Arduino IDE, Arduino, USB cable
2. DC motor - E-S motor (Mfr #: 22PG 2230 4.75 EN 12V)
   12V, 22 mm planetary gear box, 1900 rpm no load speed; 4.75:1 gear ratio; 4 mm shaft
   16 pulse/revolution encoder, 0.2 kg-cm torque, 1.3 kg-cm stall torque
   2.5 A stall current, 90 mA no load current, 330 mA "rated" current
   Note - this motor is geared for more speed, less torque so it gives use more encoder resolution

3. Motor driver/controller (Cytron MDD10 Shield, dual channel (2 motors), 10A, 7-30V; $21)
4. Power supply (12VDC)
5. Potentiometer
6. Resistor?

INTRO / EXERCISE

We will control a "larger" brushed DC motor using Arduino. This motor requires far more current than Arduino can provide so we will use a motor controller. The motor controller will also control motor direction. The power supply is needed to power the driver/controller. In this lab, we will move the motor without using any encoder data information.

BACKGROUND

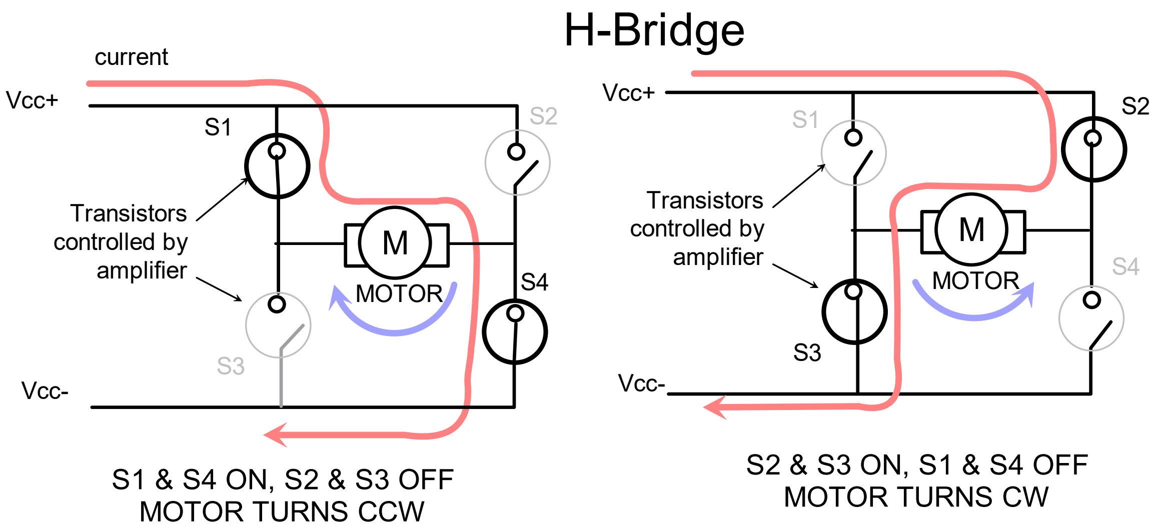
Here we will control the speed of a brushed DC motor using Arduino. Brushed DC motors have a **commutator** that switches the direction of the current in the motor's windings so that the motor always turns in one direction. We have used DC SERVO motors before. A DC servo has a DC motor in it, but it also has control circuitry built into it. A DC motor has no control circuitry. A key specification for our motor is INPUT VOLTAGE.

SUPER IMPORTANT - NEVER apply voltage above the input voltage range.

REFER to the CT GUIDE chapter on motors.

Our DC motor will have a **gear head** to reduce speed and increase torque. GEAR RATIO is the ratio of the rotor speed to output shaft speed, or ratio of output torque to input torque.
It also has an **encoder** to sense the motor's angular position (though we won't use it in this lab). A key encoder spec is encoder CPR (counts (or pulses) per rotor revolution).
You may have to look that up the encoder CPR and gear ratio for your specific motor.

Our DC motors will draw far more current than Arduino can supply (~ 20 mA/pin). For this reason we must use a motor driver/controller. A motor controller plugs into a power supply (DC supply or battery), a controller (Arduino in our case), and the motor. It routes substantial electric current from the power source to the motor based on small CONTROL SIGNAL from Arduino. It also controls the motor's direction. It does this using a so-called **H-bridge**. An H-bridge is a circuit configuration where 4 transistors control the direction of current through the motor. If transistors S1 and S4 are turned on, then the motor turns one way. If transistors S2 and S3 are turned on, the motor turns the other way.



4.1 CONSTANT SPEED

We will command a DC motor to move at constant speed. Please watch this useful video on youtube: https://www.youtube.com/watch?v=Rc892r--njw. (DroneBot workshop). This exercise will not use the encoders at all (encoders come in a later lab).

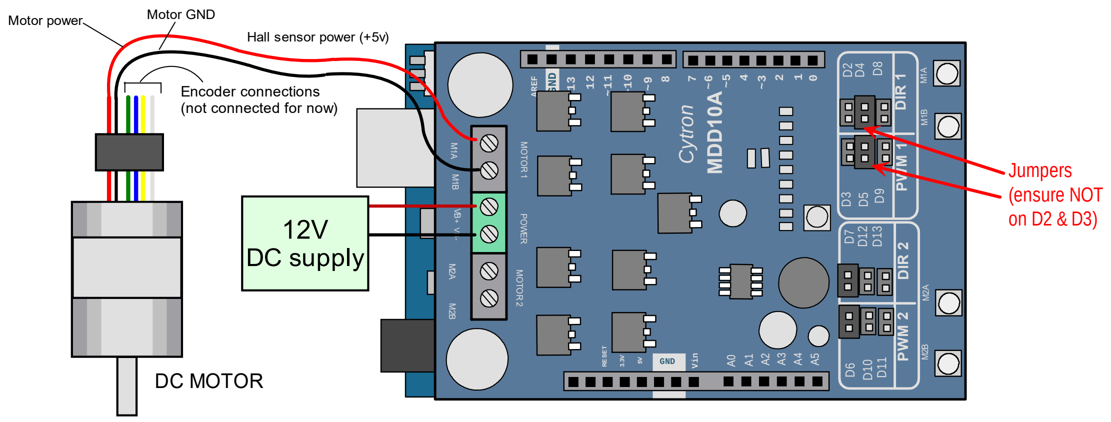
After successfully turning the motor in one direction, alter the direction to see if it works in that direction too. You change the direction by either commanding a HIGH (or 1) or a LOW (or 0) to the direction pin. This is how the Cytron controller is set up. Also, while the motor is turning use a DMM to check the output of the controller. Is it analog or PWM?

CONNECTIONS

Power must come from a DC power brick, a bench-top DC power supply, or a LiPo battery. Set the bench-top DC supply to 7 – 11 volts and current of 1A. The DC motor with quadrature encoder has 6 connectors. The red and black are for the motor. The remaining 4 are for the encoder. Only use the 2 motor connections for now. Connections between Arduino and the driver/controller are:

PWM control signal (use a PWM pin) – to control motor speed (0 to 255)
Direction signal (use any digital pin) – to control motor direction (0 or 1)
GND (ground)

For the Cytron SHIELD option - which is sandwiched onto the top of the Arduino UNO, there is an alternative wiring as shown. Here, the PWM and DIR connections are made via jumpers placed in the desired position. The advantage is that you don't have to explicitly hook up wires for PWM (control signal) and direction. The disadvantage is that you are limited to the specified I/O pins indicated by the jumpers' positions. Important - do NOT use the PWM and DIR pins for the encoder output!



CYTRON MOTOR CONTROLLER SHIELD
INSERTED ON ARDUINO

CODE

Note that the servo library is not needed for this exercise.

```
// DCmotor1_constSpeed.ino -------------------------------------
// command constant speed of DC motor
// vid name: DCmotor1_constSpeed.mp4 (if submitting digitally)

int motorCmdPin = 9;       // pwm pin
int dirPin = 8;

int speedVal = 10;         // 0 - 255

void setup()
{
  pinMode(motorCmdPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop()
{
  digitalWrite(dirPin, 0);                // set motor direction
  analogWrite(motorCmdPin, speedVal);     // speedVal = 0 to 255
}
```

4.2 BACK & FORTH

Here we will move the motor back and forth (2 directions). Connections are the same as the last exercise.

CODE

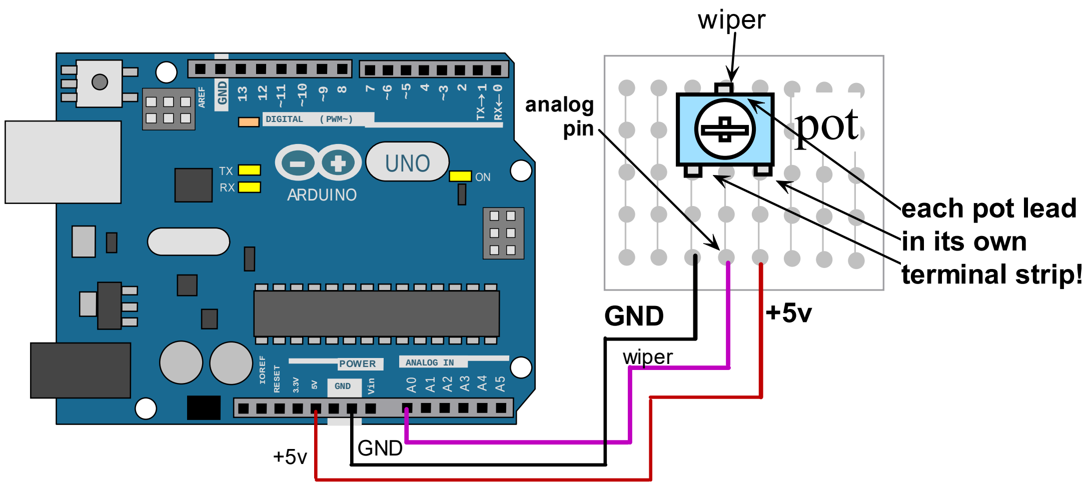
```
// DCmotor2_back-forth.ino --------------------------------------
// move motor back and forth
// hardware: arduino, cytron, power supply
// vid name: DCmotor2_back-forth.mp4 (if submitting digitally)

int motorCmdPin = 9;    // pwm pin
int dirPin = 8;
int topSpeed = 50;

int speedVal = 0;

void setup() // ------------------------------------------------------
{
  pinMode(motorCmdPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop() // ---------------------------------------------
{
  digitalWrite(dirPin, 0);                      //speed up in 1 dir
  for (speedVal = 0; speedVal < topSpeed; speedVal++)
  {
    analogWrite(motorCmdPin, speedVal);
    delay(50);
  }

  analogWrite(motorCmdPin, 0);
  delay(50);
  digitalWrite(dirPin, 1);
  for (speedVal = 0; speedVal < topSpeed; speedVal++)
  {
    analogWrite(motorCmdPin, speedVal);
    delay(50);
  }
  analogWrite(motorCmdPin, 0);
  delay(40);
}

// end -------------------------------------------------------------
```

4.3 MOTOR SPEED VIA POT

Here we will command a DC motor's speed by turning a potentiometer (pot).

CONNECTIONS

All prior connections with the Arduino, motor, and controller remain the same. Now just ADD the connections for the potentiometer as shown in the figure. Two wiring options are shown. One shows a panel mount pot, and the other shows a breadboard mount pot.



CODE

```
// DCmotor3_pot.ino ---------------------------------------
// command motor speed with a pot
// vid name: DCmotor3_pot.mp4 (if submitting digitally)

int motorCmdPin = 9;    // pwm pin
int dirPin = 8;
int potPin = A0;
int motorSpeed;
int topSpeed = 80;


void setup() // -----------------------------------------------
{
  pinMode(motorCmdPin, OUTPUT);
  Serial.begin(9600);
}

void loop() // --------------------------------------------------
{
  int potVal;

  potVal = analogRead(potPin);        // 0 to 1023
  motorSpeed = map(potVal, 0, 1023, -topSpeed, topSpeed);
  driveMotor(motorSpeed);
}

void driveMotor(int motorSpeed) // ----------------------------------------------
{
  int motorDir;
  int speedVal;
  if (motorSpeed >= 0)
  {
    motorDir = 1;
  }
  else if (motorSpeed < 0)
  {
    motorDir = 0;
  }

  motorSpeed = abs(motorSpeed);
  digitalWrite(dirPin, motorDir);
  analogWrite(motorCmdPin, abs(motorSpeed));            // 0-255
}


//end -----------------------------------------------------------------
```

4.4 DC MOTOR SINE OSCILLATION

Now we will command a sinusoidal oscillation of the DC motor. Connections are unchanged.

CODE

```
  // DCmotor4_sineOscill.ino --------------------------------------
  // oscillate DC motor using sin ()
  // vid name: DCmotor4_sinOscill.mp4 (if submitting digitally)

  int motorCmdPin = 9;    // pwm pin
  int dirPin = 8;
  int speedVal = 0;

  float ampl = 50.0;     //amplitude of sin wave
  float f = 0.3;        //freq in Hz (1/period T)


  void setup() // -----------------------------------------------
  {
    pinMode(motorCmdPin, OUTPUT);
    Serial.begin(9600);
  }

  void loop() // -------------------------------------------------
  {
    float t;
    float motorSpeed;

    t = millis() / 1000.0;

    motorSpeed = ampl * sin(2 * PI * f * t);       // -100 to 100
    motorSpeed = round(motorSpeed);
    motorSpeed = (int) motorSpeed;

    driveMotor(motorSpeed);

  }

  void driveMotor(int motorSpeed) // ----------------------------------------------
  {
    int motorDir;
    int speedVal;
    if (motorSpeed >= 0)
    {
      motorDir = 1;
    }
    else if (motorSpeed < 0)
    {
      motorDir = 0;
    }

    motorSpeed = abs(motorSpeed);
    digitalWrite(dirPin, motorDir);
    analogWrite(motorCmdPin, abs(motorSpeed));            // 0-255
  }


  //end -----------------------------------------------------------------
```

SUBMISSION

Either demo to instructor or video upload.
The instructor will let you know which to do.

For video uploads, the file names should be same as indicated on provided code
A merged video is acceptable, but video must be annotated with the exercises being done.

Note - prior motor used
Pololu 4752: 12V, 30:1 gear, 37Dx68L, 64 CPR encoder; no load: 330 rpm, 150 mA;
stall: 190 oz in/ 14 kg-cm; 5.5A; 6 mm D shaft, $40