Lab1: Motor Control

Objective: The objective of this lab is to provide you with a solid foundation for the rest of this course as well as the necessary knowledge for your final project for the course. At the end of this lab, you will be able to control the step motor using MATLAB through Arduino and Motor controller.

Before you get started:

1. Get familiar with MATLAB, here are some resources:
   https://www.mathworks.com/videos.html#matlabgetstarted
2. Get familiar with Arduino, here are some resources:
   https://docs.arduino.cc/tutorials/
   we also covered the basic Arduino functions in lecture 2, please refer to the lecture materials

Hardware:

- Arduino Mega
- Breadboard
- STEPPERONLINE Nema 17
- A4988 motor driver (datasheet) (Amazon)
- 12V Power supply (There are multiple power supply at The Hive or Invention Studio, we also have some limited power supply as secondary backup)
- 100uF capacitor

  * Be extra careful with the motor driver as it breaks/overheat often if it is being improperly used. Recheck the wiring and try substituting the components if all else fails (breadboard, wires, etc.)

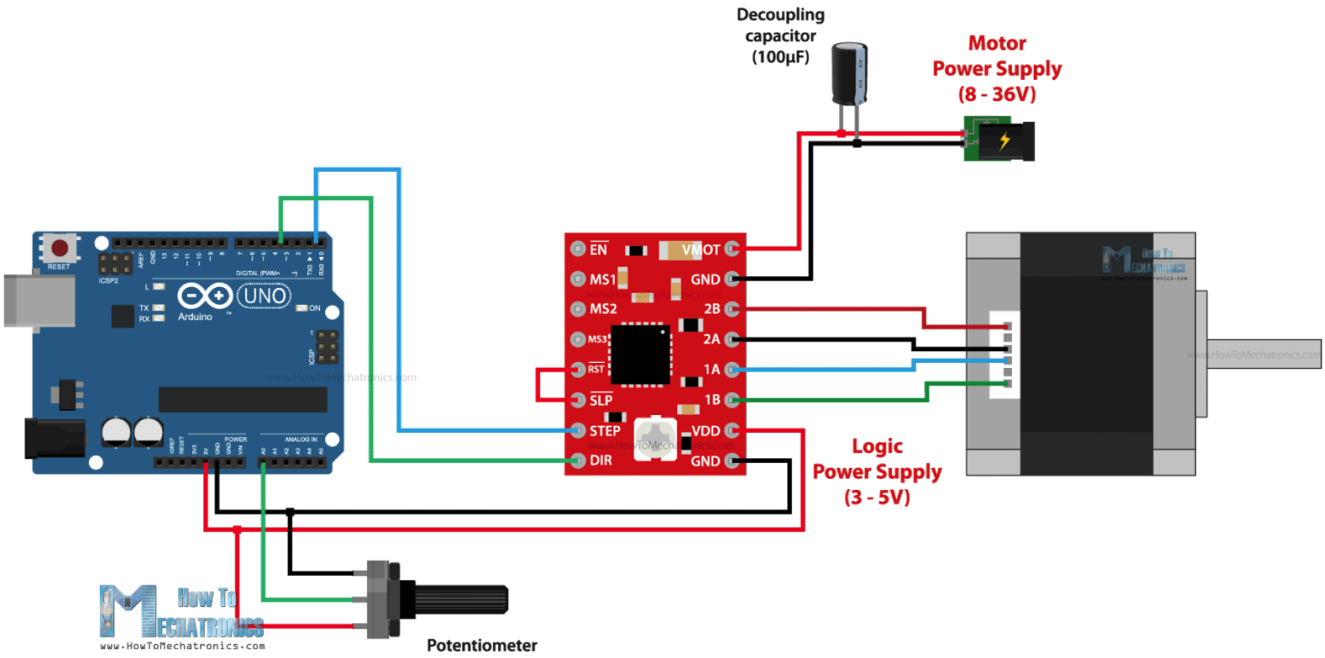
* **DO NOT** rearrange cable while connected to power supply (both Arduino and 12V power)

* When turning off the power after experiment, turn it off in this order:

Turn off 12V Power supply > Plug off Arduino > Rearrange wire

What you need to do:

1. Connect motor and Arduino with motor controller. Here is the minimal wiring diagram for the connection (Arduino connected to your computer). Here is an example video tutorial.



Potentiometer can be removed for our purpose but make sure to have the **100uF capacitor** to prevent voltage spike from the 12V power supply which can damage your motor driver, Arduino, or even your computer.

There is a knob on the A4988 driver that can be turned with a screwdriver, it is used to adjust the reference voltage (Vref) of the driver. I will not go in depth on it for this lab, but if your motor is not running, try adjusting the knob to see if it helps. Here is an article about adjusting Vref in A4988

For the Stepper motor (Nema 17) there are four cables connected to the motors and they form 2 pairs, in the diagram above as 1A 1B and 2A 2B. If the pair is connected incorrectly, the motor will not turn. One way to find the pair is by using a Multimeter to check the resistance. The correct pair should have a few ohms resistance; if the wires are not a pair, it will have no continuity. Another way is by using an LED, once an LED is connected to a pair of wire, manually turn the stepper motor. If it is the correct pair, the LED will light up.

2. Use Arduino to control stepper motor. Here is an example code for a basic rotation of the stepper motor.

```
#include <Arduino.h>
#include "A4988.h"

// using a 200-step motor Nema 17 is also 200 steps/revolution. Make sure to
recalculate if using gearbox
#define MOTOR_STEPS 200
// configure the pins connected
#define DIR 2
#define STEP 3
A4988 stepper(MOTOR_STEPS, DIR, STEP);

void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
  stepper.begin(20,1); // 20 is the rpm, 1 is the microstep
}

void loop() {
  stepper.rotate(90);
  delay(1000);
}
```

There are multiple other ways to run a stepper motor in Arduino (AccelStepper library, using `digitalWrite()` on the pins, etc.) You may use whichever you are most comfortable with as long as you properly comment on your code.

* If your stepper motor is vibrating instead of moving (with the above code), try to lower the rpm in `stepper.begin()`

3. One way to establish communication between Arduino and MATLAB is through serial communication. Download the MATLAB add-ons for Arduino. Read more here, here.

```
% Initialize the serial communication with serialport(). Input COM with the
% port your arduino is using and adjust the BaudRate corresponding to the
% arduino
arduino = serial(COM,'BaudRate',9600);

% Open the serial port
fopen(arduino)

% You can use fprintf to send values that can be read by the arduino. %f is
% for float. If you are sending multiple values (in a loop for example) it
% is recommended to add a pause() to make sure your arduino finishes
% reading the first value you send before receiving another.
fprintf(arduino,'%f',0.1)
pause(0.1) % seconds
```

On the Arduino IDE side, you can use

`Serial.parseFloat()` or `Serial.read()`

to let the Arduino read the serial communication from MATLAB.

Again, there are other ways to establish this MATLAB-Arduino connection, you may use whichever you are most familiar with as long as it is reproducible and documented properly.

- Note: When jumping back and forth between Arduino IDE and MATLAB serial communication, you may receive some errors when trying to upload a code in Arduino IDE or starting the serial communication in MATLAB. This is because the USB port is still connected to one program or another. Thus, before uploading Arduino IDE code, in MATLAB, run:

  ```
  delete(instrfindall);%delete any residual serial connection
  ```

  And before running MATLAB serial communication, close Arduino IDE.

Deliverables:

1. In MATLAB (not through Arduino IDE), create a constant motor rotation angle (45 degrees), and send the desired rotation angle to the motor to create the motion. Record a video of the stepper motor rotating to the desired angle.
2. In MATLAB, create a variety of motor rotation angles subject to the time change (theta=A*sin(omega*t), you are free to specify the coefficient, such as A, and omega), and send the desired rotation angles to the motor to create the motion.
   Hint: when you send the command series to Arduino, you need to make sure the previous command has been executed before new command arrives, aka, change the pause time to achieve this. Also, please attach a tap to the motor shaft so that we can easily visualize the motor is running.

Submission

For this lab, you need to submit **five** files.

1. A clear **picture** of the wiring on the breadboard (include the Arduino and motor)
2. Record and submit **two videos** with team to run MATLAB script to control the motor at desired control command (constant rotation angle, and variable rotation angle).
3. Submit the MATLAB **.m script** and the Arduino IDE **.ino script** you used to control the stepper motor.

Put all the files in a .zip file with the format **LastName_FirstName_GroupNumber.zip** and upload to canvas, one submission per group.

Appendix:

The command flow is shown below. In your final robot project, the motor command data will be generated in MATLAB, and then sent to Arduino to drive the motor.