Electric Motor Control with Arduino

GNG1103 – Engineering Design

Objective

This lab provides an overview of the electric motors used in most of the GNG 1103 and 2101 course design projects. Emphasizing the best value for money and ease of use, the Arduino control of DC motors, stepper motors and servomotors will be discussed. By the end of this lab, you will be able to:

- Adjust the direction of rotation of the motor
- Increase or decrease the rotation speed of the motor
- Determine the distance traveled according to the number of rotations
  Identify scenarios where the use of a motor controller/driver is necessary.

Background

An electric motor is a machine that transforms electrical energy into mechanical energy. There is a variety of electric motors available on the market today. *Direct current motors* are the most popular in the category of electric motors. Typically, they consist of a stator, a rotor, a collector and brushes. The speed of a DC motor can be controlled by varying the voltage at its terminals. The two most common types of DC motors are: Brushed motors and Brushless motors. DC motors are widely adopted for applications that require high RPM (revolutions per minute). For example: mobile wheels, fans, etc.

Servomotors are a subcategory of brushless DC motors. It consists of a DC motor and a rotating axis (standard variation between 0° and 180°) coupled to a sensor (often a potentiometer) for position feedback. It additionally requires a servo drive to complete the system. A servomotor allows precise control of angular or linear position. It is noteworthy that the term “servo” refers to a function rather than a specific type of device. A servo motor, then, is one that incorporates a feedback device (such as an encoder) whose output is compared to the command signals sent to the motor by the controller. These motors are designed for tasks where a motor position needs to be precise; for instance, moving a robotic arm or controlling the rudder on a boat within a particular range. Moreover, servo motors provide a high-performance alternative to stepper motors.

A *stepper motor* is fundamentally a servo motor that employs a different method of motorization. Where notched electromagnets arranged around central equipment to describe the position. They typically have a considerable number of magnetic pole pairs (50 to 100) generated by permanent magnets or current. They require a control circuit that possesses two levels, namely: logical and power components. The logical component determines for each step what are the powered coils and the direction of rotation. The power component level controls the speed of the motor. The benefits of this technology are long life (bearings are the unique wear-out mechanism.), little maintenance required and high performance (85-90%). Stepper motors are used primarily for

position control in an open-loop system, with applications ranging from 3d printers, fans, pumps and compressors, also in industrial applications like high-speed placement equipment.

During this lab, several experiments will be carried out for each of these motors coupled with an Arduino microcontroller. However, it is not possible to connect a motor directly to an Arduino output. This is due to the low voltage and current output of the Arduino compared to the motor characteristics. Therefore, the incorporation of an external power supply is critical to ensure the connection with the microcontroller. This component might be a transistor (bipolar, MOSFET), a relay, a motor driver or a motor controller. The main concept behind some of the power interfaces mentioned is explained in the following lines.

A vast majority of the electric motors require a specific *motor driver* to be operational. This type of driver is frequently constituted of an H-bridge, a voltage control circuit, and a few connectors to connect the motors and an external battery. Hence, a motor driver can be seen as a kind of amplifier that helps the microcontroller to control a motor.

In addition to performing the same functions as a motor driver, a motor controller has the ability to provide feedback, error detection, and other interesting features. It uses several standard communications methods like UART (also called a serial port) and/or PWM to interact with the microcontroller. In addition, some motor controllers can be manually controlled by mean of an analog signal (usually created by means potentiometer).

Lastly, the *motor shield* is a driver module for motors that allows you to use Arduino to control the working speed and direction of the motor. Based on the Dual Full-Bridge Drive Chip L293D, it is able to drive four DC motors, two step motor or two servomotors at a time. So the idea of a motor shield is that you can just connect your motor and power supply directly to an Arduino without the needs for wiring.

Apparatus and Equipment Overview

The equipment used in this lab includes:

- 1 x Arduino Uno R3 Microcontroller
- 1 x USB 2.0 Type A to USB 2.0 Type B cable
- 1 x Lab or Personal computer (with Windows, Macintosh, or Linux OS)
- 1 x Adafruit Motor Shield V1 (L293D chipset)
- 1 x USB flash drive
- 1 x breadboard
- 1 x potentiometer
- 1 x DC motor in the body of a Micro Servo
- 1 x MOSFET Canal-N
- 1 x servomotor
- 1 x stepper motor
- Connector wires

- Arduino 1.8.2 IDE Software
- Corresponding Arduino IDE libraries (outlined below)

Pre-lab Preparation

Before arriving to the lab, you should read this manual to familiarize yourself with the topics and procedures of this laboratory. Use of your personal computer is allowed, provided you have the Arduino IDE software and the required libraries. Revision of the Arduino IDE syntax as well as trying to write some programs in advance is encouraged. A useful list of commands used in Arduino can be found here:
https://www.arduino.cc/en/Reference/HomePage.

Pre-lab Questions

What are the two major types of DC motors?

_____

_____

What is the rotor of a DC motor?

_____

_____

Why should a power interface be used?

_____

_____

Name some examples of power interfaces between a motor and a microcontroller.

_____

_____

What is the difference between a *motor driver* et un *motor controller* ?

_____

_____

Warnings

Sensitive electrical components like the control boards and actuators used in this lab can be damaged or destroyed from static discharge from mishandling during assembly and use. Before

setting up this lab, students should ensure that their workstation is statically prepared (non-metal workstation, no carpet/other fibrous materials), and they have grounded themselves by touching a large metal object before handling the components. There are accessories to improve the assembly workstation for static resistance, such as an antistatic mat and static wrist/ankle straps, which should be used if available. Before starting the lab with the experimentations make sure the Arduino is not connected to any power supply (computer or wall outlet).

Part A – DC motor control using a MOSFET

This program will test and control a circuit composed of a DC motor and a MOSFET using a pre-set sequence of commands. The MOSFET is very far the most common transistor and can be used in both analog and digital circuits. The MOSFET works by electronically varying the width of a channel along which charge carriers flow (electrons or holes). Therefore, it has a crucial role in dealing with high current when the motor first begins to rotate.

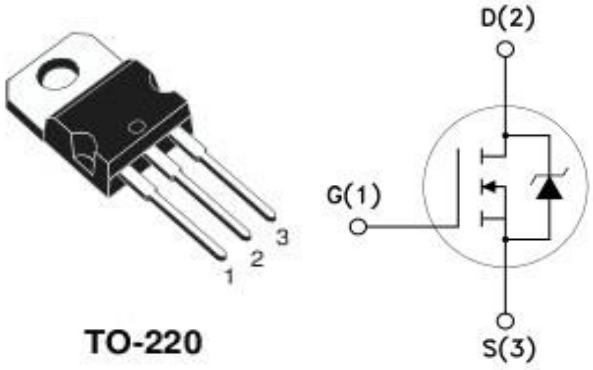


Figure 1: N-channel MOSFET

- G or 1 represents the GATE: Control pin
- D or 2 represents the DRAIN: for N-channel, it is connected to the load
- S or 3 represents the SOURCE: for N-channel, it is connected the ground.

Figure 2 shows the wiring diagram of a N-channel MOSFET with a motor and an Arduino microcontroller. Frequently, the computer is unable to supply enough power for a motor. In these cases, it is possible to power the motor via the *Vin* pin of the Arduino if it is powered by a wall outlet or an external power supply. Moreover, a Schottky diode can be used to prevent a return of current due to the inertia of the motor while stopping.

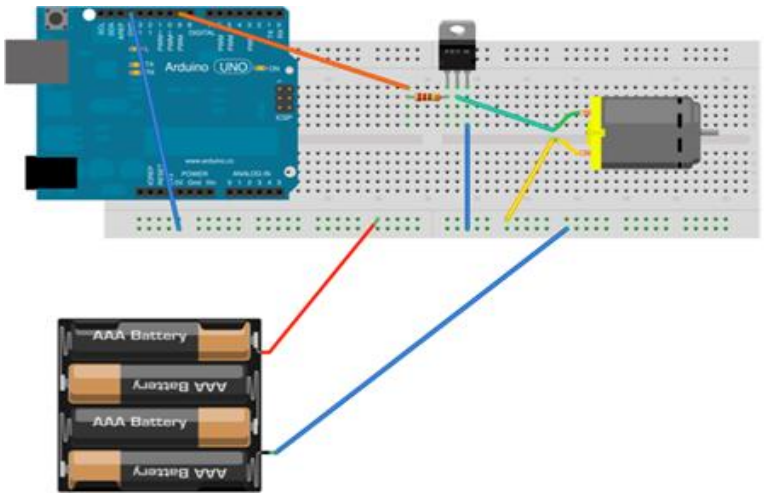


Figure 2: Wiring diagram of a *MOSFET Motor Switch*

Beyond the basic commands such as “FORWARD and BACKWARD” some applications may require a speed variation. To do so, some of the pins of digital outputs of the Arduino board will be used. They are rated PWM and are recognized by the symbol “~” next to their number. These pins offer the possibility of sending a sequence of variable pulse between 0v and + 5v. When applying a potential to the Gate the MOSFET will switch the current through the Source/Drain junction. So you need to connect the gate to the Arduino through a **220 ohms** current limiting resistor to protect the Arduino. Then you can now connect the motor to the external power supply and the Drain of the MOSFET. We complete the circuit by connecting up the grounds. In this case the MOSFET controlling the motor is driven from Arduino pin **9**.

Lab_moteur_CC_MOSFET §

```
int pinMotor=9;

void setup(){
    pinMode(pinMotor,OUTPUT); // set the the pinMotor as an output
}

void loop(){

//======================================== Part 1 ====================================

    for (int percentage=0;percentage<=100;percentage+=25){//variations in %
    int valeur=map(percentage,0,100,0,255); //change a variable number from one range to another
    analogWrite(pinMotor,valeur);
    delay (2000); // wait for 2 seconds
  }

//======================================== Part 2 ====================================
    digitalWrite(pinMotor,HIGH); //Motor start
    delay(1000);

//======================================== Part 3 ====================================
    for (int *** ; *** ;percentage***25){//variations in %
    int valeur=map(percentage,0,100,0,255); //change a variable number from one range to another
    analogWrite(pinMotor,valeur);
    delay (***); // wait for 0.5 seconde
  }
    digitalWrite(pinMotor,LOW); // Motor stop
    delay(1000); // delay in between reads for stability
    }
```

Figure 3: DC motor Arduino code

Explain what each part of the code does.

_____

_____

_____

1. Set up the circuit like in Figure 2.
2. Taking into consideration that we want to decrease the speed gradually, replace all the *** in the code in Figure 3 with the appropriate values.
3. Show the TA/PM your work.

Compile and upload the sketch if your work is right. Note your observations

_____

Explain in your own words the expression "percentage+=25"

Part B – Servo motor control

This part of the lab will be a guide to understand how to control servo motors. As we saw above, a servo motor is connected with three wires: two for the power and ground and the last for the command signal. Remember that a servo motor generally accepts a range between 4.5V and 6V of power. In that case, don't forget to connect the ground of external power and the Arduino to keep a common electrical reference. Considering the servo motors already have an onboard control system a power interface is usually not necessary. In this experiment, we will connect the power and ground pins directly to the Arduino 5V and GND pins. The PWM input will be connected to one of the Arduino's digital output pins.

Caution: Do not try to rotate the servo motor by hand, as you may damage the motor.

1. Connect a servo motor and a potentiometer to the Arduino board by using the wiring diagram below in Figure 4.
2. Open a new example sketch in the Arduino IDE. The specific example will be included in: File > Examples > Servo > Knob.
3. Show the TA/PM your work.

Which library should be included to run this code?

What are all the variables declared in this code?

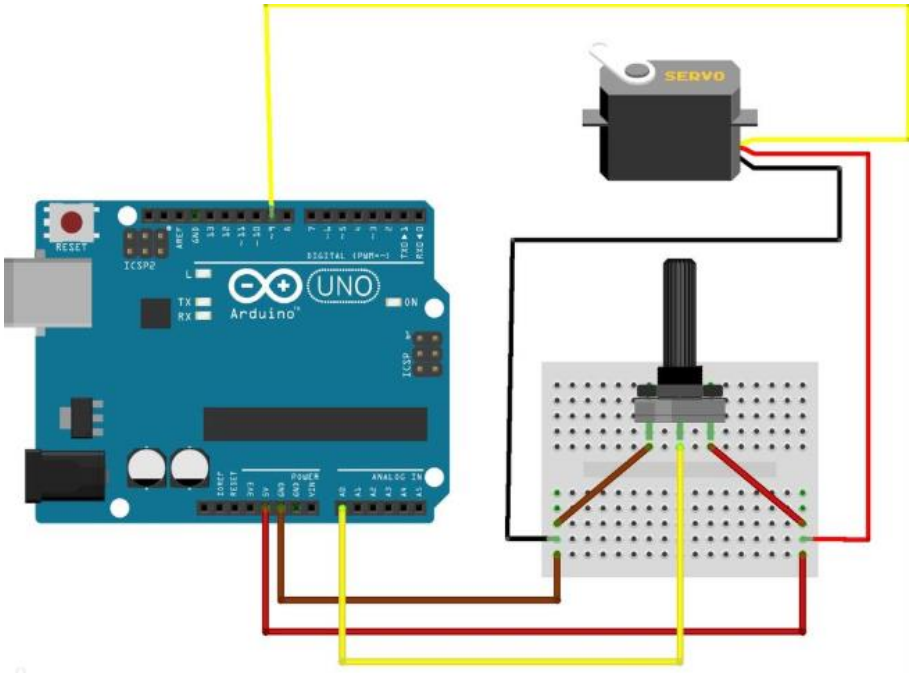


Figure 4: Wiring diagram of servomotor with a potentiometer

Part C – Stepper motor control using a motor shield

This program will test and control two stepper motors mounted on an Arduino through a motor shield using a pre-set sequence of commands. The steps in the code are as follows:

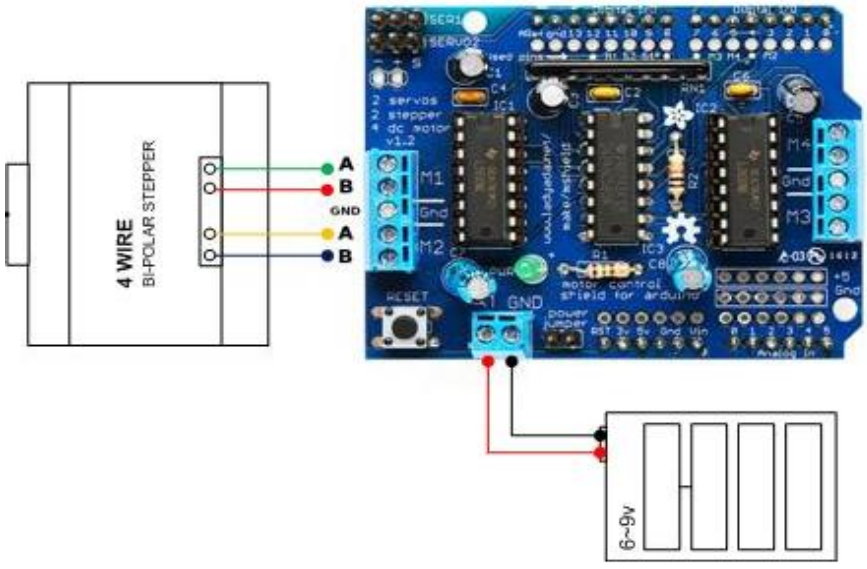
1. Install the library *AFMotor.h*, https://github.com/adafruit/Adafruit-Motor-Shield-library
2. Create the stepper motor object with *AF_Stepper(steps, stepper#) to* setup the motor H-bridge and latches. *Steps* indicates how many steps per revolution the motor has (a 7.5 degree/step motor has 360/7.5 = 48 steps). *Stepper#* is which port it is connected to. If you are using M1 and M2, it is port 1. If you are using M3 and M4 it is port 2
3. Set the speed of the motor using *setSpeed(rpm)* where *rpm* is how many revolutions per minute you want the stepper to turn.
4. Then every time you want the motor to move, call the *step(#steps, direction, steptype)* procedure. *#steps* is how many steps you would like it to take. *Direction* is either FORWARD or BACKWARD and the *steptype* is SINGLE, DOUBLE. INTERLEAVE or MICROSTEP. "Single" means single-coil activation, "double" means 2 coils are activated at once (for higher torque) and "interleave" means that it alternates between single and double to get twice the resolution (but of course its half the speed). "Microstep" is a method where the coils are PWM'd to create smooth motion between steps.

Stepper_motor_Code_En

```
#include <**** > //include library AFMotor


AF_Stepper motor(***, 1);


void setup() {
  Serial.begin(***); // set up Serial library at 9600 bps

  motor.setSpeed(***);  // set the speed to 60 rpm (revolutions per minute)

  motor.step(100, FORWARD, SINGLE);
  motor.release(); // releses the axis after a movement
  delay(1000);
}
```

5. Open the template for the code **"*Stepper_motor_Code_En.ino*"** from brightspace and replace the *** in the code with the appropriate values, based on what you learned in the previous sections.
6. Next, compile and upload your code on the Arudino and observe what happens when the step precision changes. Show the TA/PM your work.

Note that the stepping commands are 'blocking' and will return once the steps have finished.



Additional Ressources

- [Choosing a motor controller]
- [Learn stepper motor control with Arduino motor shield]
- [Arduino Programming Lab Manual - GNG2101]
- [Using Stepper Motors]