

ME-330

ME 330: Mechatronics – Laboratory 3

Digital Open Loop Motor Driver

Nicholas Agtual, Dalaney Conte, Evaline Cuatok, Josh Gipson

Department of Mechanical Engineering

Spring 2021

**Abstract:**

This lab introduces building circuits while utilizing an Arduino microcontroller and potentiometer. The potentiometer allows for an understanding of analog input signals and how to operate an open loop motor controller pulse width modulation (PWM). Three circuits were constructed during the lab using the Arduino Uno. The first two included the potentiometer, op-am, DC motor, and motor with a flyback diode. The first circuit used a pulldown resistor and BJT transistor, whereas as the second circuit used a MOSFET. For the final configuration, a motor drive controller was used to change the direction of the motor while also using the potentiometer. All circuits were built with accompanying edited sample code to produce instructed outcomes.

Table of Contents

[1 Introduction and Objectives 5](#_Toc98971775)

[1.1 Objectives 5](#_Toc98971776)

[1.2 Required Components 5](#_Toc98971777)

[2 Digital Open-Loop Motor Control: BJT 7](#_Toc98971778)

[2.1 Required Components 8](#_Toc98971779)

[2.2 Physical Circuit 8](#_Toc98971780)

[2.3 Arduino Code 8](#_Toc98971781)

[2.4 Lab Questions 9](#_Toc98971782)

[3 Digital Open-Loop Motor Control: MOSFET 10](#_Toc98971783)

[3.1 Required Components 11](#_Toc98971784)

[3.2 Physical Circuit 11](#_Toc98971785)

[3.3 Arduino Code 12](#_Toc98971786)

[3.4 Lab Questions 12](#_Toc98971787)

[4 Bidirectional Digital Open-Loop Motor Control 14](#_Toc98971788)

[4.1 Required Components 14](#_Toc98971789)

[4.2 Physical Circuit 15](#_Toc98971790)

[4.3 Arduino Code 15](#_Toc98971791)

[4.4 Lab Questions 17](#_Toc98971792)

[5 Conclusion 20](#_Toc98971793)

[6 Appendices 21](#_Toc98971794)

[7 References 22](#_Toc98971795)

List of Figures

[Figure 1: Arduino Uno Microcontroller 5](#_Toc98971816)

[Figure 2: Potentiometer 6](#_Toc98971817)

[Figure 3: L298N Motor Drive Controller 7](#_Toc98971818)

[Figure 4: TB6612 H-Bridge 18](#_Toc98971819)

List of Tables

[Table 1: Specifications for a 2N2222 NPN transistor (from ref. 7) 13](#_Toc98971996)

[Table 2: Specifications for a P30N06LE MOSFET (from ref. 8) 13](#_Toc98971997)

# 1 Introduction and Objectives

## 1.1 Objectives

The objective of the Digital Open Loop Motor Controller is to control the speed and directions of a DC motor. This is performed by including an Arduino microcontroller within circuit to read the input of the positioning of a potentiometer to determine the output of the controller. The first two circuits in the lab focused on controlling the speed of the motor, while the second incorporated the direction as well. Sample code was provided with the instruction for each circuit, which could then be modified to be compatible with the given configuration.

## 1.2 Required Components

Most components used for the circuits constructed have been introduced in previous labs. The new elements introduced are as follows:

**Arduino UNO Microcontroller**

**A close-up of a circuit board

Description automatically generated with medium confidence**

Figure 1: Arduino Uno Microcontroller

The Arduino UNO microcontroller is an electronics platform that reads inputs and transforms it into an output. This can be altered by editing and uploading instructions utilizing the accompanying Arduino program to control isolated portions of the circuit.

**Potentiometer**

Diagram

Description automatically generated

Figure 2: Potentiometer

The potentiometer is a transducer that converts the angular or linear positioning of a shaft/wiper to a variable output. Inside the potentiometer there are three electric terminals connected in series. This component is used in the lab to control the speed of the motor through the positioning of the shaft.

**L298N Motor Drive Controller**

Graphical user interface, schematic

Description automatically generated

Figure 3: L298N Motor Drive Controller

The motor drive controller is used to control the direction of the motor. The logic pins of the motor control the forward/reverse directions by changing the polarity of the input voltage. This component is used in place of an H-bridge, which would perform the same outcome by wiring a combination of transistors and diodes.

# 2 Digital Open-Loop Motor Control: BJT

The goal of this circuit is to alter the speed of the DC motor by adjusting the applied voltage. The PWM is used to modify the duty cycle which changes the average voltage across the motor. The Arduino platform utilizes a digital pin as an output with the duty cycle ranging from 0 to 255. The specified duty cycle is then transformed to a reading set by the position of the potentiometer. The potentiometer was scaled down linearly from 1024 to 255 to be compatible with the range of the Arduino.

## 2.1 Required Components

* 2N2222 BJT
* NI ELVIS II Breadboard
* 220 Ω resistor
* LM324AN op-amp
* 3-6V DC motor
* Potentiometer
* Diode
* Arduino microcontroller
* Multimeter (ELVIS DMM)

## 2.2 Physical Circuit

Building this circuit was relatively straightforward as most of the components used have been introduced in previous labs. The incorporation of the Arduino was the only area of troubleshooting due to the inclusion of the ground and power source on the microcontroller itself. However, once the ground of the Arduino was connected to the ground of the breadboard, the motor performed as intended. The minimum PWM value for the circuit was determined to be 27.

## 2.3 Arduino Code

#define PWM 6 // ARDUINO PIN 6 - PWM

#define POT A1 // ARDUINO ANALOG PIN 1 - POTENTIOMETER

int pot\_val; // DEFINING POTENTIOMETER VALUE AS INTEGER

// RANGE = 0 - 1023

int PWM\_val; // DEFINING PWM VALUE AS INTEGER

// RANGE = 0 - 255

void setup() {

// DEFINING PIN MODES

pinMode(PWM, OUTPUT);

pinMode(POT, INPUT);

**Serial**.begin(9600); // STARTING SERIAL MONITOR

}

void loop() {

// READING POTENTIOMETER VALUE AND ASSIGNING TO pot\_val

pot\_val = analogRead(POT);

// MAPPING pot\_val TO PWM\_val

// MINIMUM PWM VALUE = 27 ; THE MOTOR DOES NOT FUNCTION BELOW THAT VALUE

PWM\_val = map(pot\_val, 0 , 1023, 27, 255);

// WRITING PWM VALUE TO PWM PIN

analogWrite(PWM, PWM\_val);

// PRINTING POTENIIOMETER VALUE AND MAPPED PWM VALUE

**Serial**.print("; Potentiometer Value =");

**Serial**.println(pot\_val);

**Serial**.print("PWM Value = ");

**Serial**.print(PWM\_val);

// 10ms DELAY

delay(10);

}

## 2.4 Lab Questions

**1) Why do we need to connect the ground from the power supply to the Arduino?**

Connecting the ground from the Arduino to the ground located on the power supply forms a closed circuit due to the creation of a common ground between the two. For the circuit to be operational, the grounds need to be linked between the components within it.  **2) If we did not connect the ground to the Arduino what would happen? Why?**

This would form an open circuit which would not allow the required power to travel to the components due to the lack of current. The motor would not be operational, and the speed would not be able to be controlled through the Arduino.  **3) In control system terminology, how would you classify the potentiometer? (See Figure 1)**

The potentiometer functions as the control classified by control terminology because it is the component that controls the state of the output. The positioning of the potentiometer determines the value of the duty cycle, thus controlling the average voltage being delivered to the motor.  **4) Considering the physics, why is there a minimum PWM for the motor to begin spinning?   
Hint: Think about a motor being modeled as an inductor and resistor in series – how is the   
magnetic field stabilized in the inductor?**

Inductors use a magnetic field to build and store energy as voltage is being applied to the component. Even as current is continuously applied to the inductor, it can stabilize itself by transferring the excess power stored back into the circuit. A motor is often schematically modeled as a resistor and inductor wired in series, representing the inductive load that travels across a motor. There is a minimum PWM for the motor to begin spinning, because like an inductor, it requires an initial load to store to begin functioning.  **5) How can this circuit or design be improved?**

This circuit can be improved through the use of better equipment and wiring. In a design aspect, the circuit could perform more efficiently if it did not require a pulldown resistor by connecting the Arduino in a more proactive way. A more efficient transistor could be utilized as well, such as the MOSFET, which proved to be the preferred component in the next section.

# 3 Digital Open-Loop Motor Control: MOSFET

The Metal Oxide Semiconductor Field Effect Transistor, also known as MOSFETs, are a type of semiconductor that contains three main parts: the drain, the gate, and the source [4]. With field-effect transistors, a voltage is applied at the gate producing an electric field which “controls a channel where charge carriers are available and current can flow” [4]. Within the MOSFET, there is an induced channel. When the voltage between the gate and source is increased, the conductivity of the channel is enhanced which increases the drain current and allows it to flow. However, when the gate voltage fails to exceed a threshold value, there is no induced channel and current will not flow. The MOSFET operates in two regions: the active region and the saturation region. In the active region, the current flowing from the drain to the source is controlled by the voltage at the gate. In the saturation region, the current will remain the same even if the applied voltage increases. In this experiment, a P30N06LE MOSFET is used.

## 3.1 Required Components

* P30N06LE MOSFET
* NI Elvis II Breadboard
* LM324AN op-amp
* 3-6V DC motor
* Potentiometer
* Diode
* Arduino microcontroller
* Multimeter (ELVIS DMM)

## 3.2 Physical Circuit

Wiring the circuit for this portion of the lab was very similar to the previous section. The only thing that changed was that the resistor was removed and the BJT was replaced by a MOSFET. In this portion of the experiment, a P30N06LE MOSFET transistor was used. Once the circuit was set up, the minimum PWM was able to be determined using the Arduino code from the previous section. The minimum PWM for this circuit was found to be 39.

## 3.3 Arduino Code

#define PWM 6 // ARDUINO PIN 6 - PWM

#define POT A1 // ARDUINO ANALOG PIN 1 - POTENTIOMETER

int pot\_val; // DEFINING POTENTIOMETER VALUE AS INTEGER

// RANGE = 0 - 1023

int PWM\_val; // DEFINING PWM VALUE AS INTEGER

// RANGE = 0 - 255

void setup() {

pinMode(PWM, OUTPUT); // DEFINING PWM PIN AS OUTPUT

pinMode(POT, INPUT); // DEFINING POT PIN AS INPUT

**Serial**.begin(9600); // STARTING SERIAL MONITOR

}

void loop() {

// READING POTENTIOMETER VALUE AND ASSIGNING TO pot\_val

pot\_val = analogRead(POT);

// MAPPING pot\_val TO PWM\_val

// MINIMUM PWM VALUE = 39 ; THE MOTOR DOES NOT FUNCTION BELOW THAT VALUE

PWM\_val = map(pot\_val, 0 , 1023, 39, 255);

// WRITING PWM VALUE TO PWM PIN

analogWrite(PWM, PWM\_val);

// PRINTING POTENIIOMETER VALUE AND MAPPED PWM VALUE

**Serial**.print("; Potentiometer Value =");

**Serial**.println(pot\_val);

**Serial**.print("PWM Value = ");

**Serial**.print(PWM\_val);

// 10ms DELAY

delay(10);

}

## 3.4 Lab Questions

1. **How did the minimum PWM values compare to the BJT circuit?**

The minimum PWM value that was obtained for the MOSFET was 39 which is considerably higher to the value of 27 that was obtained in the BJT circuit.

1. **Compare the specifications of the MOSFET to the specifications of the BJT create a table and list the maximum operating specifications of each. Write a brief discussion of the comparison results.**

**2N2222 NPN transistor specifications**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Max** |
|  | Collector-base voltage | 60 V |
|  | Collector-emitter voltage | 30 V |
|  | Emitter base voltage | 5 V |
|  | Collector current | 800 mA |
|  | Total power dissipation | 500 mW |

Table 1: Specifications for a 2N2222 NPN transistor (from ref. 7)

**P30N06LE MOSFET specifications**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Max** |
|  | Drain-source voltage | 60 V |
|  | Gate-source voltage | ±20 V |
|  | Continuous Drain Current | 50 A |
|  | Maximum power dissipation | 150 W |

Table 2: Specifications for a P30N06LE MOSFET (from ref. 8)

When looking at the maximum operating specifications for both the MOSFET and the BJT, they both operate under a maximum of 60 V for the drain source and collector base voltage. Additionally, they operate under similar maximum voltage for the gate source and collector emitter voltage. There is only a difference of 10 V between the two values. The largest difference seen amongst these values is the collector current and the continuous drain current as well as the power dissipation. Additionally, the gate source voltage of the MOSFET is able to operate in the negative region whereas the collector-emitter voltage of the BJT can only operate when it is positive.

1. **Independent of specifications, explain fundamentally why the MOSFET would be preferred over a BJT in microcontroller applications?**

The MOSFET would be preferred over the BJT because of its structure. The gate terminal of the MOSFET is composed of metal and is isolated from the drain and source terminals by an oxide layer which acts as an insulator. Due to this isolation, the MOSFET consumes less power than the BJT where the base, collector, and emitter are not isolated from each other. Additionally, MOSFETs move faster in terms of switching and it has fewer switching losses compared to BJTs.

1. **Why is the pulldown resister no longer necessary in this circuit?**

A pulldown resistor ensures that a wire is pulled to the low logical level when there is no input signal. This ensures that the microcontroller does not misinterpret the input value and it gives the input pin a default state regardless of whether the switch is open or not. In this circuit, the pulldown resistor is not necessary because the pin is directly connected to ground. When this happens, the inputs of the logic gates cannot float. Instead, there will be a constant low which is desired.

# 4 Bidirectional Digital Open-Loop Motor Control

The bidirectional digital open-loop motor control is intended to allow for control the speed of the DC motor in both directions using a potentiometer. This is accomplished with the use of the LN298N integrated circuit chip motor driver.

## 4.1 Required Components

* L298N Motor Drive Controller – 1X
* ELVIS board (For Power Supply) – 1X
* 3-6V DC Motor – 1X
* Potentiometer – 1X
* Arduino Microcontroller – 1X
* Multimeter (ELVIS DMM)

## 4.2 Physical Circuit

**Find the datasheet for the L298N circuit.**

<https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf>

**Describe the operation of the IC. Be sure to include the limitations of this IC.**

The L298N requires 4V-7V and can operate from 0-36mA. This IC gains heat very fast and requires a heat sink and a low current. The IC can take a voltage up to 46V and is driven by a dual H-bridge system. It does consume a lot of power as it isn’t as efficient as other ICs and that’s why it requires a heat sink.

**Determine if it is okay to run a motor on this circuit without external diodes? Why?**

It’s recommended to use an external diode to protect the circuit from voltage spikes. This is key especially when using an inductive load like the motor. With other ICs the diode may be built into the IC but the L298 requires an external diode to handle flyback voltage.

**What if the IC was an L293 instead?**

If the IC was changed from an L298 to a L293 IC, the motor would have to operate a lower current. One of the largest differences between the two ICs and is the difference in H-driver designs. The L293 has a quadruple H-drive meaning the inputs and outputs are independent of each other but with the L298 has a dual H-drive meaning the inputs depend on each other.

## 4.3 Arduino Code

#define ENABLE 5 // ARDUINO PIN 5 – L298N ENABLE A used for MOTOR ENABLE AND PWM

#define DIRA 3 // ARDUINO PIN 3 – L298N IN1 USED FOR MOTOR DIRECTION

#define DIRB 4 // ARDUINO PIN 4 – L298N IN2 USED FOR MOTOR DIRECTION

#define POT A1 // ARDUINO ANALOG PIN A1 - POTENTIOMETER

void setup() {

// DEFINING PIN MODES

pinMode(ENABLE,OUTPUT);

pinMode(DIRA,OUTPUT);

pinMode(DIRB,OUTPUT);

**Serial**.begin(9600); // STARTING SERIAL MONITOR

}

void loop() {

// DEFINING read\_position as integer & setting equal to potentiometer value (0 - 1023)

int read\_position=analogRead(POT);

// DEFINING converted\_position AS INTEGER AND MAPPING TO (-255:255)

int converted\_position = map(read\_position,0,1023,-255,255);

// DEFINING forward\_stall\_PWM & reverse\_stall\_PWM as integers & setting values

// MOTOR DOES NOT OPERATRE WITHIN THIS RANGE (converted\_position)

int forward\_stall\_PWM = 21;

int reverse\_stall\_PWM = -11;

// DISPLAYING MAPPED PWM VALUE

**Serial**.print("PWM VALUE =");

**Serial**.println(converted\_position);

// IF THE CONVERTED VALUE FALLS WITHIN THE MOTOR'S INOPERABLE RANGE

if ((converted\_position > reverse\_stall\_PWM )&&(converted\_position < forward\_stall\_PWM)){

 // TURNING MOTOR OFF

 digitalWrite(DIRA, LOW);

 digitalWrite(DIRB, LOW);

 // DISPAYING BOOLEAN VALUES FOR DIRECTION OF MOTOR

**Serial**.print("DIRA = ");

**Serial**.println(0);

**Serial**.print("DIRB = ");

**Serial**.println(0);

}

// IF converted\_position IS GREATER THAN 0

else if (converted\_position > 0){

 // WRITING CONVERTED POSITION VALUE TO MOTOR

 analogWrite(ENABLE, converted\_position);

 // SETTING DIRECTION OF MOTOR SPIN

 digitalWrite(DIRA, LOW);

 digitalWrite(DIRB, HIGH);

 //DISPLAYING BOOLEAN VALUES FOR DIRECTION OF MOTOR

**Serial**.print("DIRA = ");

**Serial**.println(0);

**Serial**.print("DIRB = ");

**Serial**.println(1);

   }

// IF converted\_position IS LESS THAN 0

else {

 // WRITING CONVERTED POSITION VALUE TO MOTOR

 analogWrite(ENABLE, -converted\_position);

 // SETTING DIRECTION OF MOTOR SPIN

 digitalWrite(DIRA, HIGH);

 digitalWrite(DIRB, LOW);

 // DISPLAYING BOOLEAN VALUES FOR DIRECTION OF MOTOR

**Serial**.print("DIRA = ");

**Serial**.println(1);

**Serial**.print("DIRB = ");

**Serial**.println(0);

   }

   // 50ms DELAY

   delay(50);

## 4.4 Lab Questions

1. **If a motor required 5V and 1A to operate, could the L298N IC be used?**

Yes, the L298 could be used as it can provide the current and voltage needed for that specific voltage and current.

1. **Look online for another IC that could be used instead of L298N. Attach the datasheet in the Appendix of the lab report. Describe the operating parameters of the IC and how it is a suitable replacement.**

A suitable replacement for the IC would be the TB6612 H-Bridge. The TB6612 operates within low voltages (2.5V – 13.5V) and can operate at lower currents. It is a suitable replacement because it operates at lower voltages and is more efficient then the L298. The L298 has a voltage drop of 1.4 VDC and whereas the TB6612 has a maximum voltage drop 0.13 VDC. The TB6612 can do the same functions as the L298 but at lower voltages and lower currents.

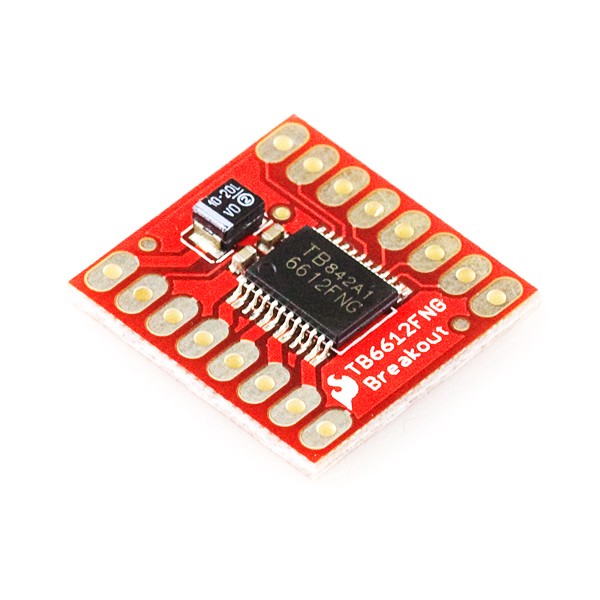


Figure 4: TB6612 H-Bridge

1. **In the code, from the reverse stall PWM to forward stall PWM value, the motor is motionless. How could you modify the code such that this motionless region is removed, and used for speed control (i.e., from 1024 to 512, the motor PWM ranges from 255 to 100 and from 512 to 0 the PWM ranges from -100 to -255)?**

Removing this content within this if statement of the code would remove the motionless portion.

// IF THE CONVERTED VALUE FALLS WITHIN THE MOTOR'S INOPERABLE RANGE

if ((converted\_position > reverse\_stall\_PWM )&&(converted\_position < forward\_stall\_PWM)){

 // TURNING MOTOR OFF

 digitalWrite(DIRA, LOW);

 digitalWrite(DIRB, LOW);]

Replacing this if statement with code that reads the potentiometer and set the minimum converted value to 0 causing the motor to start. In doing this, the reverse direction will behave in the same manner.

1. **What are some ways this circuit or design could be improved?**

One way the circuit could be improved is to use a DC voltage supply. In our experiment, we found that using the current supplied from the Arduino worked better than the AC current coming from the building. As well, a flyback diode could be placed parallel to the motor in order to avoid voltage spikes.

1. **In this lab we were able to control the speed of the motor, but we do not know what the speed of the motor is. How could we determine the motor speed?**

The speed of the motor can be controlled by analyzing the PWM values and the Arduino output. If 100% PWM is 100% speed, a linear relationship can be assumed.

1. **If we did not have the Arduino, how could a controller be implemented? Is an Arduino preferable to this alternative?**

If an Arduino was not used in this system, a controller could be implemented via the use of a speed controller in conjunction with a potentiometer. Essentially, this would take over the role of the Arduino, adjusting the PWM values sent to the motor, thereby adjusting the speed.

1. **We do not have the specifications for the motor provided, how could the values for the inductor and resistor in the model be determined? (Hint: think of a step response)**

If the motor specifications are not provided, the values for the inductor and resistor in the model can be determined using a step response. By analyzing the voltage required to power the motor and using a multimeter to measure the resistance, the current can be calculated using Ohm’s law. With the known, KVL can be used to calculate the impedance.

1. **Could we determine the motor power from the answers to the previous question? How?**

Using the answer from the previous question the motor power can be determined using the following equations.

1. **Could we determine the torque?**

The torque can be determined using the following equation

# 5 Conclusion

During the completion of this lab, we learned more about the programming of Arduinos and open loop motor controller with PMW. Building the circuit allowed for a deeper understanding of how all these components interact with one another. In this lab issues were faced with both the physical circuit and with the programming of the Arduino. With the physical circuit, multiple iterations of the circuit were performed, and different components used to get the circuit to work properly. In one of the iterations the circuit was wired differently and yet the circuit still ran but was different from what was specified in the lab. With the program we had to go through multiple settings and had to comb through our results to find the correct zero position. The observations in this lab corroborated the theory behind the differences between the opamp running the motor and the IC. Seeing the H-Bridge in effect during the lab deepened our understanding when it helped understand the theory behind the design of the H-Bridge.

# 6 Appendices

BJT & MOSFET CODE

https://components101.com/sites/default/files/component\_datasheet/L298N-Motor-Driver-Datasheet.pdf

DATA SHEET FOR L298N CIRCUIT

1. CW FAST
2. CCW FAST
3. CW W/ PWM
4. CCW W/ PWM
5. OFF

# 7 References

1. Escalona, L. “Electrical Circuits and Components.” ME 330 – Mechatronics. San Diego State University. Lecture
2. Escalona, L. “Analysis of Electrical Networks.” ME 330 – Mechatronics. San Diego State University. Lecture
3. Escalona, L. “Sensors and Actuators Used in Mechatronics Systems.” ME 330 – Mechatronics. San Diego State University. Lecture
4. Escalona, L. “Semiconductor Devices: The Field Effect Transmitter.” ME 330 – Mechatronics. San Diego State University. Lecture
5. Escalona, L. “Operational Amplifier Configurations.” ME 330 – Mechatronics. San Diego State University. Lecture
6. Escalona, L. “Electric Motors.” ME 330 – Mechatronics. San Diego State University. Lecture
7. Philips Semiconductors. “Discrete Semiconductors: 2N2222 NPN switching transistors.” <https://netsonic.fi/en/files/2n2222-datasheet.pdf>
8. VBsemi. “N-Channel 60 V (D-S) MOSFET: P30N06LE.” <https://pdf1.alldatasheet.com/datasheet-pdf/view/1274976/VBSEMI/P30N06LE.html>