**Lab 4: Controlling Position**

I. OBJECTIVES

The object of this lab is to control the speed of the motor using PI control.

II. EQUIPMENT

1. Elegoo MEGA2560
2. Aslong DC motor JGA25-371-12V-95RMP
3. L298N Motor Drive Controller Board Module Dual H Bridge DC Stepper For Arduino
4. Jumper wires - 6
5. Computer/laptop.

III. BACKGROUND

PID (Proportional-Integral-Derivative) position control is a widely used technique for achieving accurate and precise control of servo motors in various applications such as robotics, automation, and manufacturing. The PID controller continuously monitors and adjusts the motor position based on feedback from the motor position sensor, making it an effective method for achieving accurate and responsive motor control.

The PID controller consists of three main components: the proportional, integral, and derivative terms. The proportional term provides a direct and proportional response to the difference between the desired and actual motor position, while the integral term corrects for any steady-state error and the derivative term predicts and corrects for any future changes in the motor position.

To implement PID position control for a servo motor, the controller first reads the current motor position from a position sensor, and then calculates the error between the desired and actual positions. The controller then uses the proportional, integral, and derivative terms to adjust the motor position, based on the magnitude and direction of the error, as well as the rate of change of the error.

PID position control can be implemented using an Arduino or other microcontroller, along with a servo motor and a position sensor such as an encoder or potentiometer. The microcontroller generates a PWM signal to control the motor position and uses the PID controller to adjust the duty cycle of the PWM signal based on the motor position feedback.

1. *L298N Motor Drive*

The L298N motor drive is a popular integrated circuit (IC) used for controlling DC motors and stepper motors in a wide range of applications, such as robotics, automation, and motor control. The main purpose of the L298N motor drive is to provide a convenient and efficient way of driving motors using a microcontroller or other digital circuitry.

The L298N motor drive consists of two H-bridge circuits that can drive two DC motors or one bipolar stepper motor and can handle a wide range of voltages and currents. The H-bridge circuits allow the direction of the motors to be controlled by changing the polarity of the voltage applied to the motor terminals, and the speed of the motors can be controlled by adjusting the voltage amplitude or using pulse width modulation (PWM).

The L298N motor drive is designed to simplify the task of driving motors by providing a compact and reliable IC that can be easily interfaced with a microcontroller or other digital circuitry. The IC contains built-in protection features such as thermal shutdown, overcurrent protection, and undervoltage lockout, which help to ensure safe and reliable operation of the motors.

Diagram

Description automatically generated

Figure 1: L298N motor driver pin layout.

* 1. *Power pins*

The L298N motor driver module receives power from a 3-pin, 3.5mm-pitch screw terminal. The L298N motor driver has two input power pins: VS and VSS.VS pin powers the IC’s internal H-Bridge, which drives the motors. This pin accepts input voltages ranging from 5 to 12V. VSS is used to power the logic circuitry within the L298N IC and can range between 5V and 7V.GND is the common ground pin.

* 1. *Output pins*

The output channels of the L298N motor driver, OUT1 and OUT2 for motor A and OUT3 and OUT4 for motor B, are broken out to the edge of the module with two 3.5mm-pitch screw terminals. You can connect two 5-12V DC motors to these terminals. Each channel on the module can supply up to 2A to the DC motor. The amount of current supplied to the motor, however, depends on the capacity of the motor power supply.

* 1. *Direction pins*

The direction control pins allow you to control whether the motor rotates forward or backward. These pins control the switches of the H-Bridge circuit within the L298N chip. The module has two direction control pins. The IN1 and IN2 pins control the spinning direction of motor A; While IN3 and IN4 control the spinning direction of motor B. The spinning direction of the motor can be controlled by applying logic HIGH (5V) or logic LOW (Ground) to these inputs. The chart below shows various combinations and their outcomes.

|  |  |  |
| --- | --- | --- |
| **Input1** | **Input2** | **Direction** |
| Low | Low | OFF |
| High | Low | Forward |
| Low | High | Backward |
| High | High | OFF |

Figure 2: L298N input values that determine direction.

* 1. *Speed control pins*

The speed control pins ENA and ENB are used to turn on/off the motors and control their speed. Pulling these pins HIGH will cause the motors to spin, while pulling them LOW will stop them. However, with Pulse Width Modulation (PWM), the speed of the motors can be controlled. The module usually comes with a jumper on these pins. When this jumper is in place, the motor spins at full speed. If you want to control the speed of the motors programmatically, remove the jumpers and connect them to the Arduino’s PWM-enabled pins.

1. *Pulse Width Modulation*

Pulse Width Modulation (PWM) is a method of controlling the power supplied to an electrical device by adjusting the width of a train of pulses of electrical energy. In other words, PWM is a way of regulating the average voltage or current that is supplied to a load, by varying the duty cycle of a periodic waveform.

The duty cycle is the fraction of time that the signal is on, or the pulse width divided by the period of the waveform. By varying the duty cycle of the waveform, we can control the amount of power that is supplied to the load.

In practice, PWM is implemented using a microcontroller or other digital circuitry that generates a periodic waveform with a fixed frequency and variable duty cycle. The waveform is then applied to a switch or driver circuit, which controls the flow of current to the load. By adjusting the duty cycle of the waveform, the amount of current that flows through the load can be adjusted, and hence control its power consumption or output.

A picture containing diagram

Description automatically generated

Figure 3 PWM waveform and calculations.

1. *Calculating Position*

The user will need to do additional work by accounting for the counts per motor shaft rotation, accounting for any gear ratios, and converting data into revolutions, radians, or degrees. This can be calculated by looking at the datasheet specifications. This motor has 12 pulses per revolution and a gear ratio of 1:45. Revolutions can be calculated with

where and are the output and input of the gear ratio, respectively. Here, revolutions can be easily measured in radians or degrees with the relationship

1. *Programming*

To program PID position control, the motor driver and servo motor can be set by writing code to send a PWM signal to control the motor speed and direction. The analogWrite() function is used in Arduino to generate a PWM signal. Connect the position sensor to the Arduino and write code to read the current position of the servo motor from the sensor.

The PID controller algorithm is coded to calculate the desired motor position and the motor speed required to reach the desired position. The PID algorithm uses the difference between the desired position and the current position to calculate the motor speed needed to move the servo motor towards the desired position. Then the parameters, such as the proportional, integral, and derivative gains, can be tuned to achieve the desired performance of the servo motor. The motor position is looped to reiterate the motor position and adjust the PWM signal to maintain the desired position using the PID controller.

IV. PROCEDURE

1. *Connecting the gearmotor and Arduino*
2. Connect the yellow wire to pin 2.
3. Connect the green wire to pin 3.
4. Connect blue wire to 5V.
5. Connect the black wire to GND.
6. *Connecting the motor to the L298N*
7. Connect the red wire to OUT1.
8. Connect the white wire to OUT2.
9. *Connecting the L298N to the Arduino*
10. Connect EN1 from the L298N to pin 9 of the Arduino.
11. Connect IN1 from the L298N to pin 10 of the Arduino.
12. Connect IN2 from the L298N to pin 11 of the Arduino.
13. Connect GND from the L298N to the GND pin of the Arduino.
14. *Connecting the power supply to the L298N*
15. Connect the positive terminal of the power supply to the +12V pin of the L298N.
16. Connect the negative terminal of the power supply to the GND pin of the L298N.
17. *Connecting the Arduino to the interface*
18. Make the USB connection on the Arduino and open the Arduino IDE.
19. Save the sketch as *Lab2\_Measuring\_Speed*.
20. Click on the board drop box.
21. Under the Boards, select Arduino Mega or Mega 2560.
22. Under Ports, select COM3 Serial Port (USB).

A close-up of a circuit board

Description automatically generated with medium confidence

1. *The program*

V. QUESTIONS