POLITECNICO DI TORINO

Master degree In automotive engineering

Master Thesis

Design and development of the position control of a motorized axis with Arduino

Supervisor

Prof. Andrea Mura

Candidate

Zhao Hanqing

A.A. 2020/21

Abstract

This thesis is mainly about researching a motorized axis's position control based on Arduino, including both hardware and software.

This thesis aims to design a control system, which could make a slider precisely at the position following the user's order. This thesis proposes a hardware design and then a software design to achieve this task.

Compared to the traditional method, the **Single-Chip Microcomputer**, the method in this thesis is more straightforward, thanks to the **Arduino** board and its open-source software.

The motor employed is a 24V DC motor. Thus, the circuit design is also included in this thesis. The first function of the circuit is the motor rotation direction control, which is accomplished by H-bridge. The second function is motor rotation speed control, which is performed by the PWM method.

Moreover, this thesis also improves the control preciseness by the PID method. It's also interesting to see how the PID parameters affect the system.

Content

List of figures

List of tables

Chapter 1 Introduction

1.1 Overview

Nowadays, with the development of power electronics technology and microprocessor technology, the electric transmission system is developing towards high-performance, digital control, and mechatronic direction.

The electric transmission systems include DC, AC, and servo systems. The DC system is a significant part of the transmission system due to its control simplicity and fast, wide range, smooth speed adjustment characteristics. Thanks to the automatic control theory and application, the DC motor control system becomes the majority in the market.

Thus, the demands for high-performance DC motor system in the various field is increasing. And the market set higher requirements for DC motor system. Therefore, researching high preciseness and reliability DC motor control system is meaningful[1].

Combined with the microprocessor and computer programming, complex algorithms are realistic. And it makes motor control much more manageable and affordable.

This thesis is based on the microprocessor Arduino Uno R3 and its open-source software. With the PWM, pulse width modulation, and H bridge, the DC motor's speed, and direction control is achieved.

1.2 A general view of the whole system

This system is, generally, a closed-loop control system. The control loop is shown in figure 1 below.

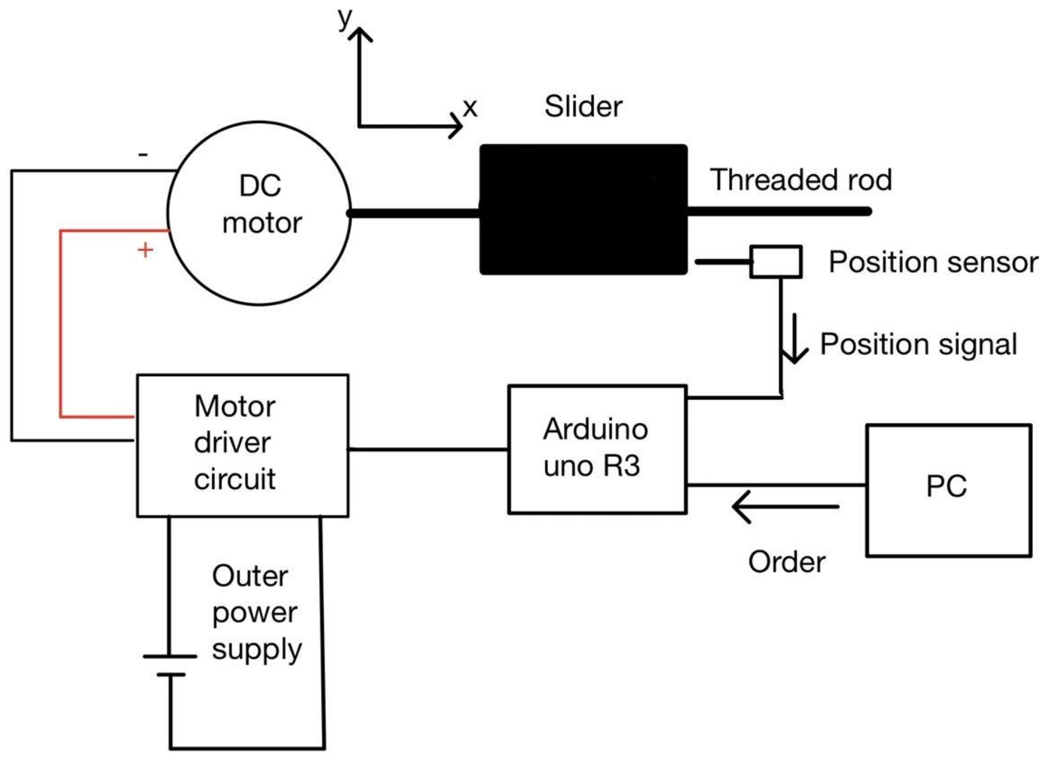


Figure 1 the general view of the system

The motor used in this experiment is a 24V DC motor. Rotation of the DC motor leads to a linear displacement of the slider on the axis, which is the same axis as the motor. Thus, by making the motor rotate both forward and reversely at a different speed, the slider's position could be adjusted.

The Input of the system is the order from the user by the keyboard. Then the system works to keep the slider's position at the ordered one. A potentiometer is working as a position sensor. This position sensor provides the voltage data, which represents the real position of the slider. Thanks to the Arduino Uno R3, the voltage could be read, and by programming, the Arduino can control the DC motor. A driver circuit is necessary to receive the signal from Arduino and then drive the DC motor's direction and speed. The driver has two main functions, direction control and rotation speed control. The H bridge circuit layout provides the direction control function, and PWM, pulse width modulation accomplishes the speed control function.

The control preciseness is guaranteed thanks to the PID method. PID controller has three parameters, which affect the system's performance significantly. Thus, the best PID parameters should be obtained. In this thesis, the best PID parameters are found by the variable control method. It's also interesting to see how the three parameters affect the system, respectively.

Chapter 2 Hardware design

2.1 design overview

As mentioned in Chapter 1, the hardware should make the slider move to the ordered position. The power which drives the slider is from a 24V DC motor with a threaded axis.

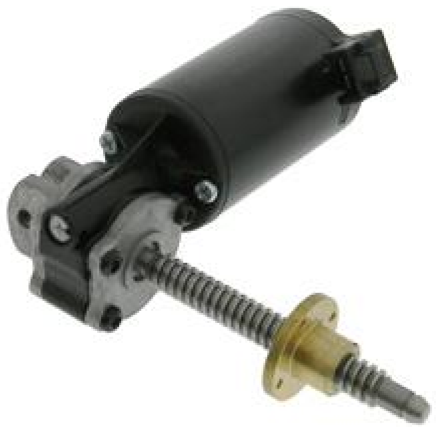


Figure 2 A screwed DC motor

This threaded axis is connected to the slider. With the bidirectional rotation of the DC motor, the slider moves back and forth.

Moreover, the slider still needs side support to make it stable. The linear guide rails can provide the side support for the slider. They are screwed to the slider.

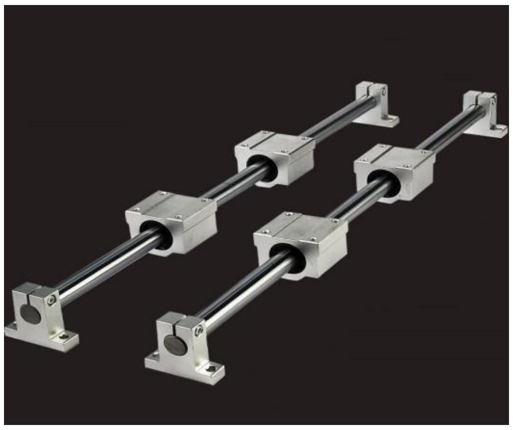


Figure 3 Linear guide rail

Besides, the hardware system should have:

- Proper safety factor, which means the hardware should be safe for people, should be economical to avoid using the components beyond expectations.

- Simplicity, which means it should be as simple as possible to reduce unnecessary steps to assembly and use.

- Low friction, which means the hardware should work with a low friction factor. Because this system is precise, a low friction factor helps to control it more precisely.

- Robustness, which means the life of the hardware system should be as long as possible.[2]

2.2 Components

In this part, the components which contain the hardware will be introduced. They are base, screw axis, slider, guide rail, guide blocks, coupling, and bearings. The unit in this chapter is in millimeters.

2.2.1 Base

The base is a square plate, which supports the system.

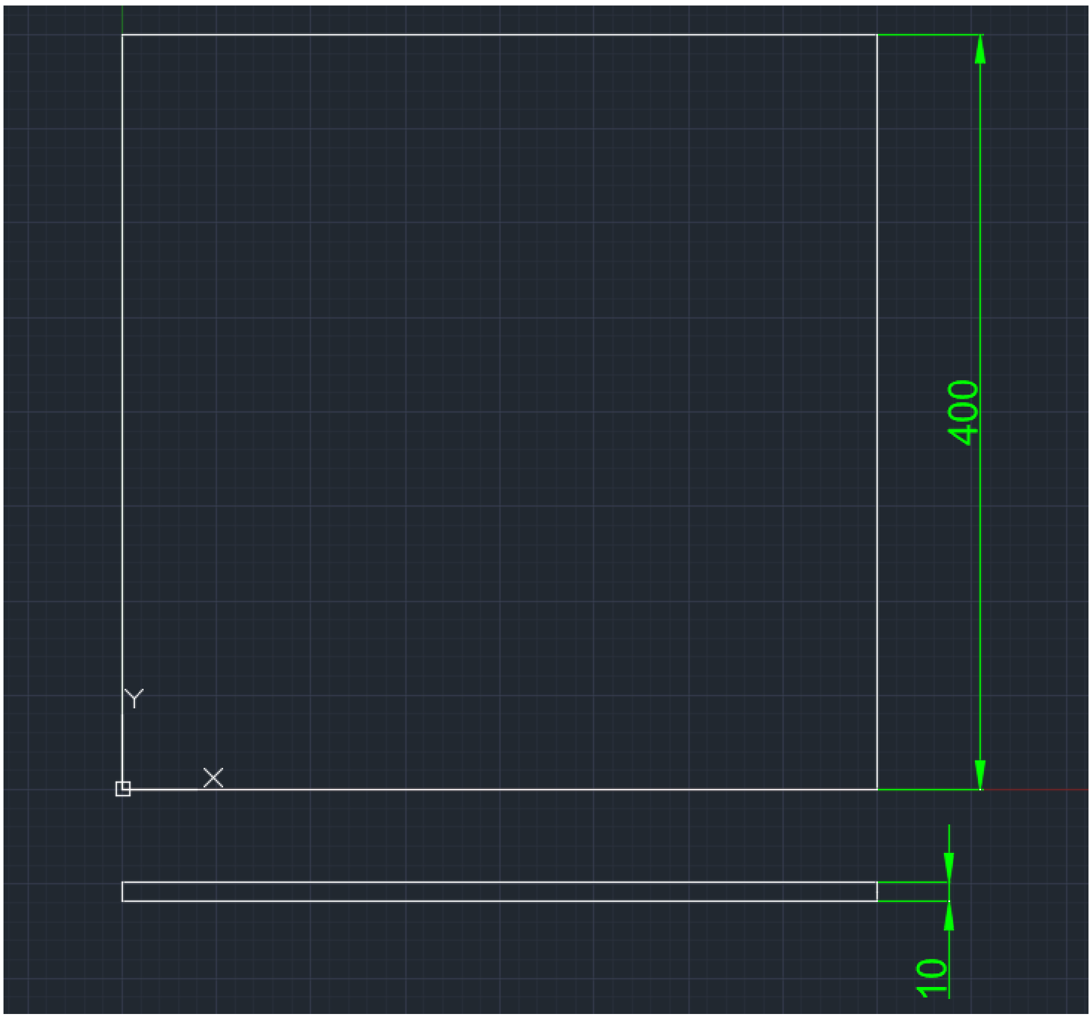


Figure 4 2D drawing of the base

2.2.2 Screw rod

The screw rod is connected to the DC motor and the slider. When screwed to the slider, It can transmit the rotation motion of the DC motor to a linear displacement of the slider.

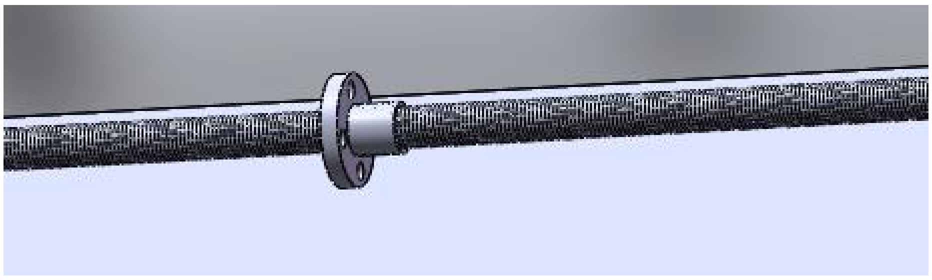


Figure 5 Screw rod

2.2.3 Slider

The slider is the core of the hardware. It's screwed to the screw rod and side guide block.

The position of the screw holes is very accurate to make sure of a perfect assembly.

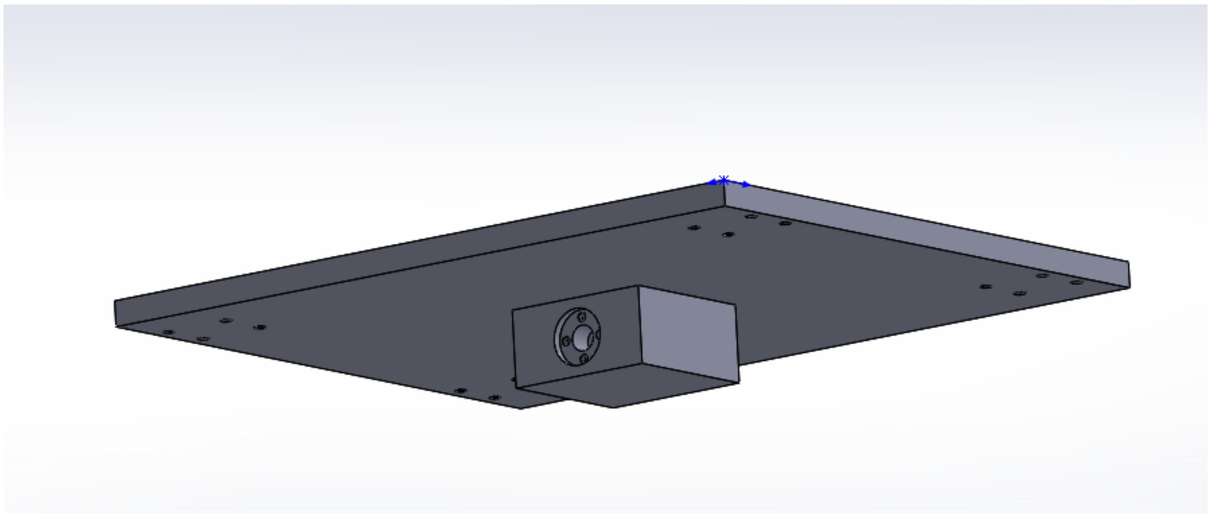


Figure 6 Slider

Moreover, the position sensor should be carefully assembled to make sure that the sensor touches the slider.

2.2.4 Guide rails and guide blocks

As mentioned before, the slider needs side supports. The guide rails and blocks are designed for this purpose. The blocks provide the holes for the slider to screw on them.

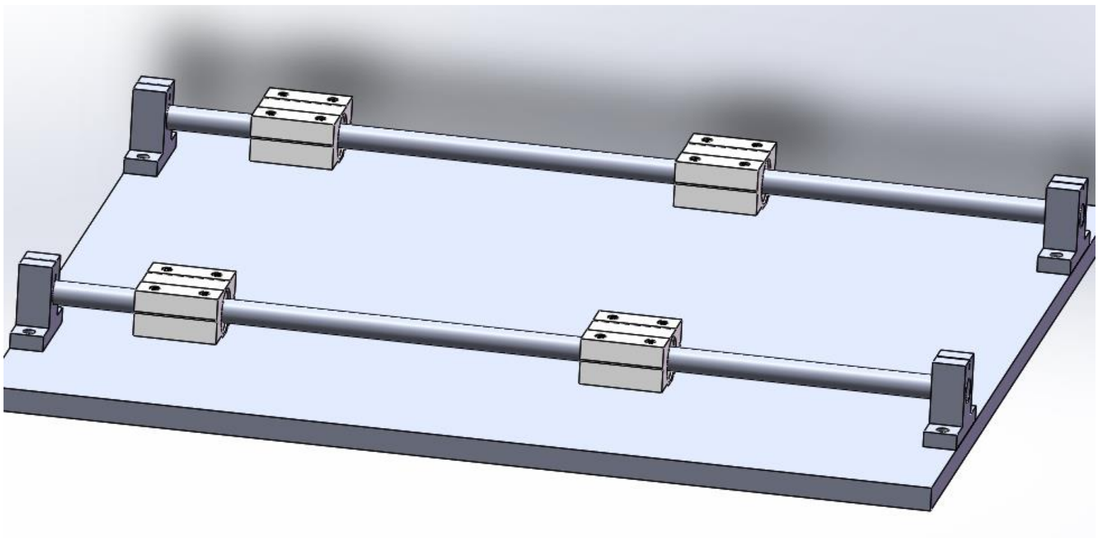


Figure 7 Guide rails and blocks

2.2.5 Coupling and bearing

The coupling is a device that connects the end of the DC motor's shafts and the screw rod to transmit power. In this thesis, the load is a torque subjected to the connection between the screw rod and the slider.

The beam flexible coupling is suitable for this situation because it's flexible and can handle a fair amount of angular misalignment.

Figure 8 Coupling

2.3 The 3D model of the hardware

The 3D model is shown in the figure below:

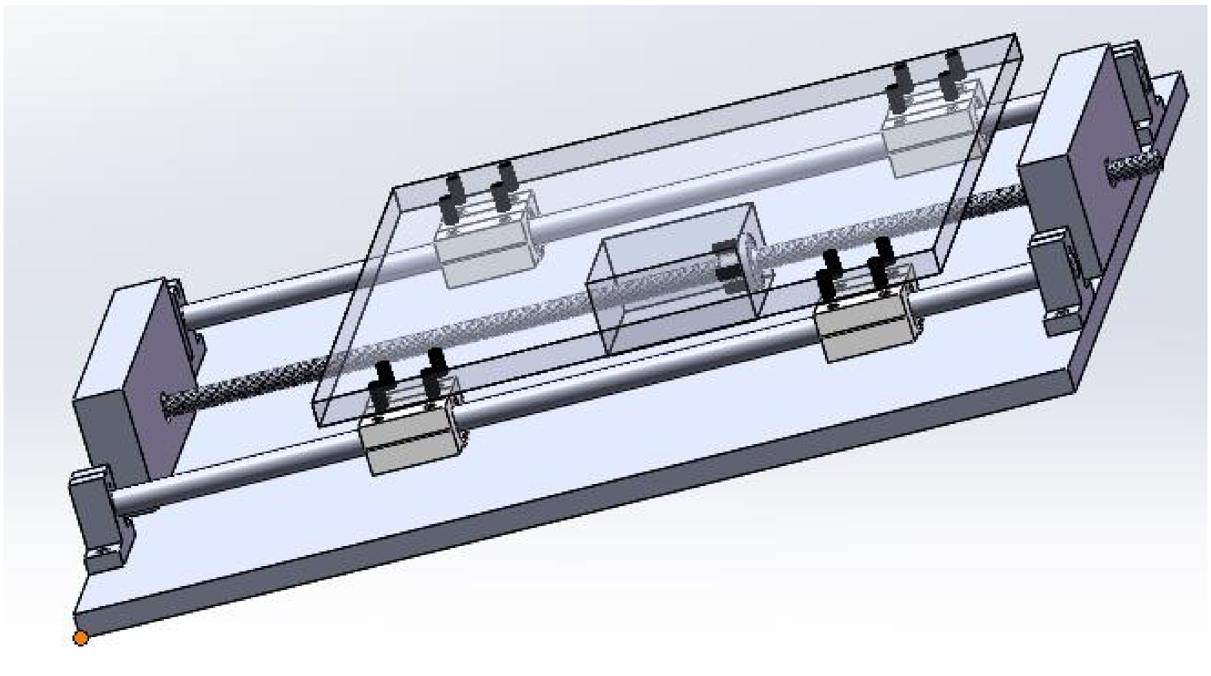


Figure 9 3D Assembly of the hardware

Chapter 3 Method to control the system

3.1 Main functions of the control system

Before describing the methods to exploit the system, it's necessary to tell the functions needed to achieve the tasks.

The core of the control system is controlling the DC motor. DC motor control includes speed control and direction control. It's quite crucial to maintain the DC motor to make the rod screw rotate to the desired position.

Therefore, the control system provides the functions below:

1. Ability to adjust the rotation speed
2. Ability to change the rotation direction

Moreover, the control system should be:

3. Fast to reach the setpoint
4. Able to continuous control, which means the user can give a new order at any time to command the slider move to a new position.
5. Precise

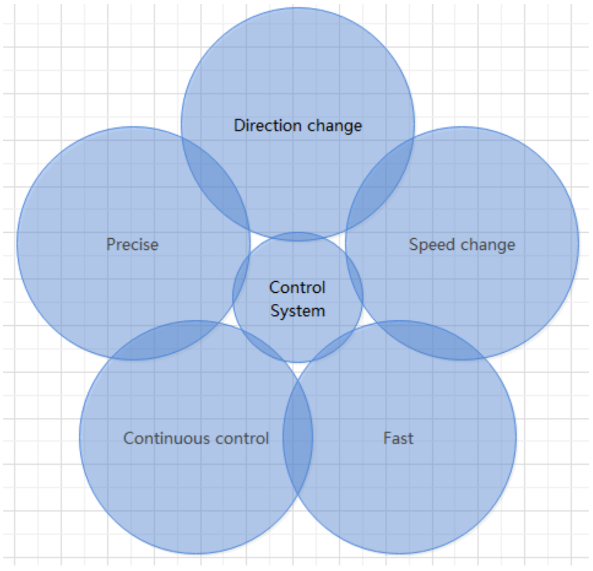


Figure 10 Control system characteristics

3.2 Method to control the DC motor: PWM

The equation of a DC motor's rpm n is:

$$n = \frac{V_m - R_m I_a}{k_b \Phi} \tag{3.1}$$

Where:

n: rotation speed of the motor (r/min)

V_m: motor input voltage (V)

R_m: motor resistance (Ω)

I_a: armatrue current (A)

k_b: counter EMF equation constant

Φ: machine's total flux (Wb)

According to equation (3.1), by varying the motor input voltage V_m, the rotation speed of the motor could be adjusted. The figure below shows the general relationship between the motor input voltage and motor rotation speed:

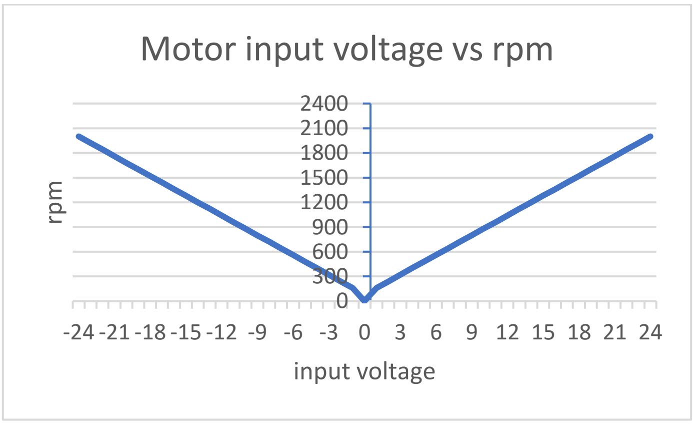


Figure 11 relationship between motor voltage and rpm

According to the figure above, the rotation speed of a motor is approximately proportional to the input voltage. Thus, the rotation speed could be controlled by varying the input voltage.

PWM, pulse width modulation, is based on the principle above. The figure below is the wave shape of the PWM[3].

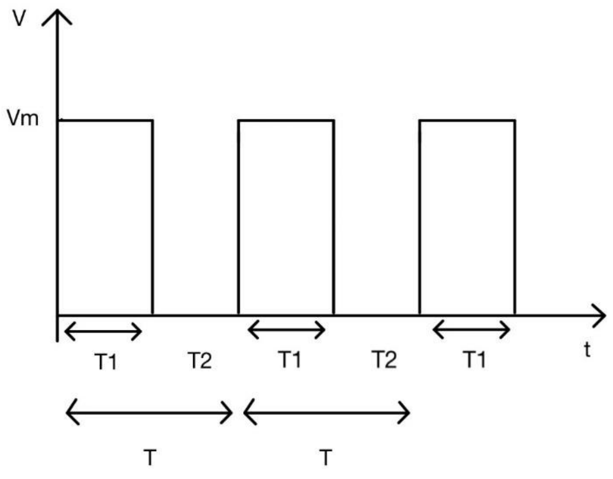


Figure 12 PWM wave shape

As shown in the figure above, at the time t=0, the input voltage is Va for a duration of T1. Then the input voltage becomes 0 for a duration of T2. And duration T is a circle for PWM. We have:

$$T_1 + T_2 = T \tag{2.2}$$

$$V = \frac{T_1}{T_1 + T_2} V_m = \frac{T_1}{T} V_m = \alpha V_m \tag{2.3}$$

$$\alpha = \frac{T_1}{T} \tag{2.4}$$

Where:

V: average voltage

V_m: input voltage

The PWM frequency is usually several thousand Herz, which means a circle is less than1 ms. The PWM works due to the high frequency. In Arduino, the PWM frequency is 1000 Herz by default.

Thus, by adjusting the parameter α, the voltage applied to the DC motor can be controlled. The rotation speed can be controlled finally.

3.3 Method to control the DC motor: H-bridge

According to 3.1, the rotation speed control has been settled.

Figure 10 shows that if the motor is applied the negative voltage, the direction of rotation will reverse.

H-bridge is usually used for the direction control of a motor. An H-bridge circuit is shown in the figure below:

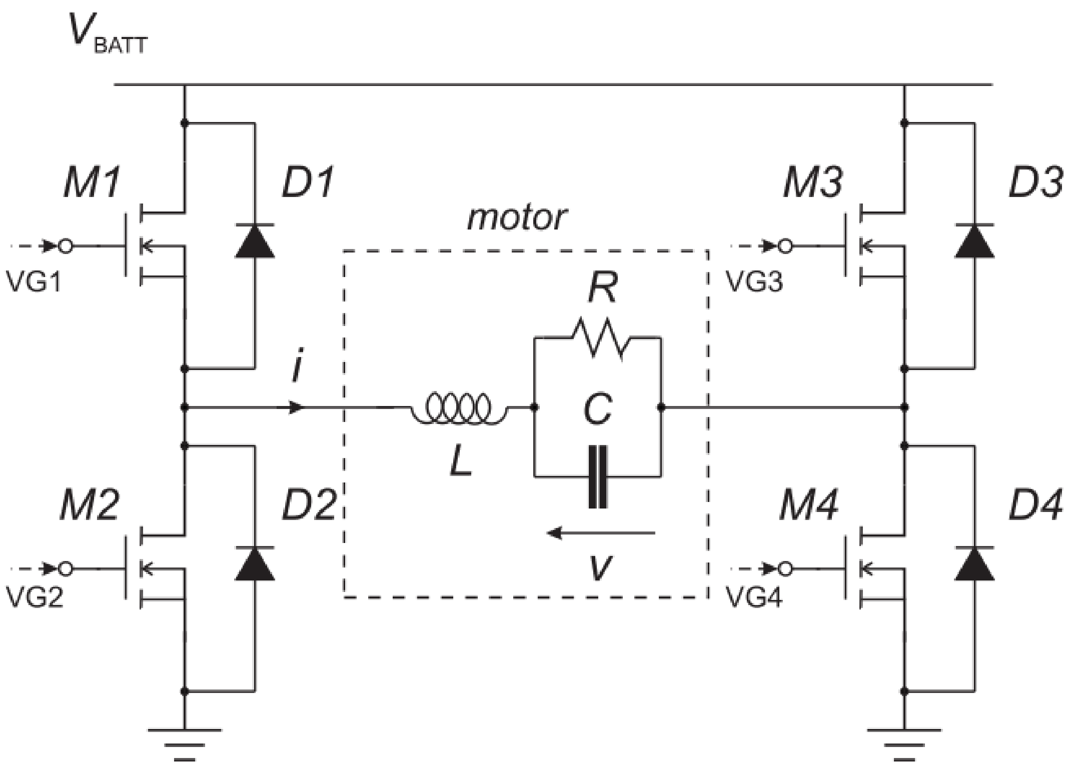


Figure 13 H-bridge circuit

With reference to such a circuit, whenever the M1 and M4 are ON, meanwhile, the M2 and M3 are OFF, a positive voltage can be provided to the load so that a driving torque in a direction (for example, the clockwise direction) is applied.

Whenever M2 and M3 are ON, and M1 and M4 are OFF. Differently, the motor is connected to the supply with an opposite layout so that a driving torque in the opposite direction (for example, counterclockwise) is obtained. If M1 and M3 are ON (while M2 and M4 are OFF) or if M2 and M4 are ON (while M1 and M3 are OFF), the load is short-circuited so that to brake the moving mass, independently of its direction of rotation, as described with reference to the half-bridge circuit.

The control logic is as below:

$M1$	$M2$	$M3$	$M4$	State
ON	OFF	OFF	ON	Driving Torque, clockwise
OFF	ON	ON	OFF	Driving Torque, counterclockwise
ON	OFF	ON	OFF	Braking
OFF	ON	OFF	ON	Braking
OFF	OFF	OFF	OFF	Idle
ON	ON	-	-	Short Circuit - **to be avoided !**
-	-	ON	ON	Short Circuit - **to be avoided !**

Table 1 control logic of H-bridge

The switches M1, M2, M3, and M4 are usually transistors. It could be BJT or MOSFET. In figure 4, the switches are MOSFET. The transistor is convenient for a control system circuit because the on/off can be easily controlled by signals. The transistor will be discussed in 3.4.

This is just a general concept of direction control. In a real case, the control circuit is more complicated. This will be discussed in chapter 4.2.

3.4 Method to control the DC motor: Transistor

The symbols of an n-channel(nMOS) and of a p-channel (pMOS) MOS transistor are reported in the figure below.

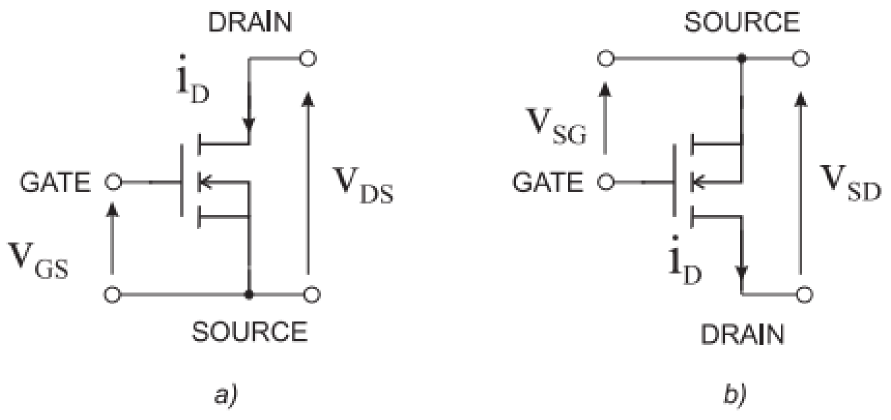


Figure 14 MOSFET

In such devices, the drain and source terminals' electrical conduction characteristics are controlled by the voltage V_{GS} between the gate and the source terminals. In the figure below, in particular, the current i_D flowing through the drain terminal of an nMOS transistor is plotted versus the voltage V_{DS} applied between the drain and source terminals for different values of the gate-source control voltage,V_{GS}. The electrical characteristics of a pMOS transistor are qualitatively similar to those reported in figure 6, provided that V_{DS} is replaced by V_{SD} and V_{GS} is replaced with V_{SG}.

This means, by controlling the gate-source voltage, the drain current will follow a curve such as $V_{GS} = 5V$ curve. Then, by controlling the drain-source voltage, which is the input voltage, the drain current can be controlled. As mentioned before, the drain-source voltage can be governed by the PWM method. And the gate-source voltage is also adjustable by Arduino, which provides a voltage varying from 0V to 5V.

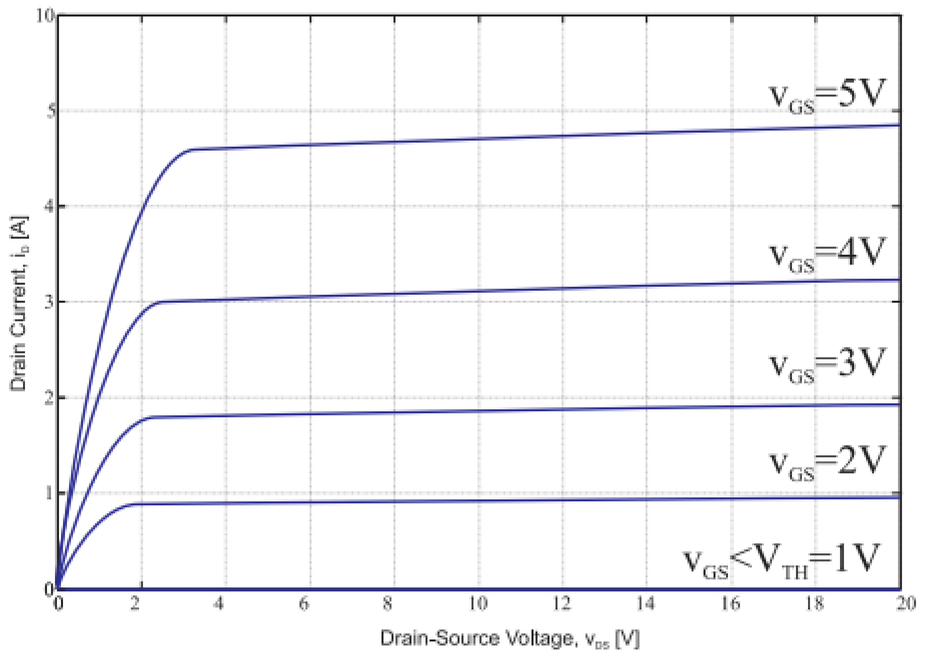


Figure 15 Electrical characteristics i_D (V_D) of an nMOS for different values of the control voltage V_{GS}.

3.5 Method to control the slider's position: PID method

PID represents proportional-integral-derivative controller. It's nowadays widely used in the industrial field. A PID controller keeps calculating an error value e(t), which stands for the difference between the desired setpoint and the measured one (process value). Meanwhile, the PID controller will correct the position based on proportional, integral, and derivatives terms. The following figure shows how PID works[4].

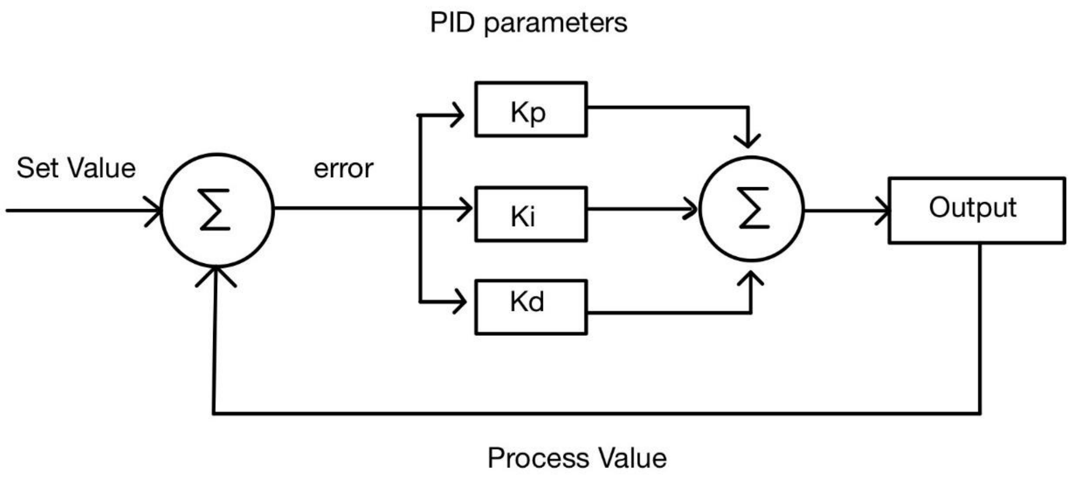


Figure 16 PID Controller

As shown in the figure above, there are several important values: Set position value, process position value, error, and output. At the beginning of the PID process, a set position value is introduced by the user's command. The system will then keep reading the process value, which represents the real position of the controlled item, in this thesis, the slider. PID controller will calculate the output based on the error between the set position value and the process position value. Thus, the value of the output is dependent on the value of the error. There are two types of PID controllers. One is the direct controller, and another one is the indirect controller. Generally speaking, the direct controller gives a positive output if the error is positive. The indirect controller has a negative output if the error is positive. For example, with a car, the PID controller should give a decreased output to bring the vehicle speed down. With a refrigerator, the opposite is correct. The controller should give an increased output to make the temperature down. In this thesis, the direct controller is chosen because the DC motor is typically direct-controlled: the lower current, the lower rotation speed.

There are also three parameters in a PID controller: Kp, Ki, and Kd. These three parameters are tuned parameters that are adjustable to control the performance of the system. By adjusting Kp, Ki, Kd, the response speed, the overshoot and undershoot oscillation can be affected. Thus, for a PID controller, it's quite essential to tune the PID parameters. In this thesis, the variables control method is used to find out suitable PID parameters. Also, the effect of these three parameters will be estimated, respectively.

Chapter 4 Circuit composition

4.1 Microprocessor: Arduino

Arduino contains both software and hardware. The hardware is a microprocessor board. The software is an open-source program that can help users control the board by programming. The language used in Arduino is based on C/C++.

There are many different types of Arduino board. Generally, all the boards are equipped with sets of digital and analog I/O pins. Users can choose proper pins to complete various tasks depends on the user's requirement. The board could also be interfaced to expansion boards, which are called 'shield,' to accomplish more complex tasks.

The boards match many communications interfaces, for example, the USB interface. It means the Arduino board is convenient to be used on a computer.

In this thesis, the board is Arduino Uno R3. Arduino Uno is the first USB-based Arduino board.

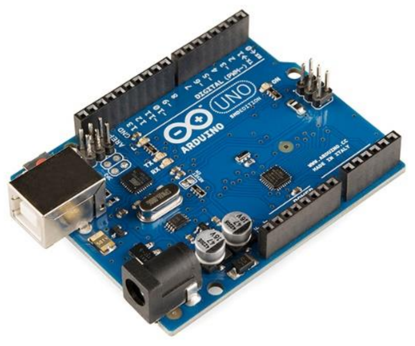


Figure 17 Arduino Uno R3

Arduino Uno R3 has 14 digital input/output pins. Six of them are capable of PWM output. And it has six analog I/O pins. This board could be powered by a

USB cable or by an external 9-volt battery. In this thesis, the power is from the PC through a USB cable.

The maximum output voltage of this board is 5V, and the maximum output current is 40mA. It means this board could not be a reliable power supply to the circuit. That's why in figure 1, an external power supply is needed to drive the 24V DC motor. This board is just a microcontroller that can control the driver circuit, not the DC motor.

The following figure shows the Arduino Uno R3 in detail. Pin 0~13 are digital pins, and A0~A5 are analog pins. Digital pins with '~' indicate this pin is capable of PWM output. The Arduino Uno can also supply power to a small circuit such as a LED circuit through the pin 5V and GND, which means ground.[5]

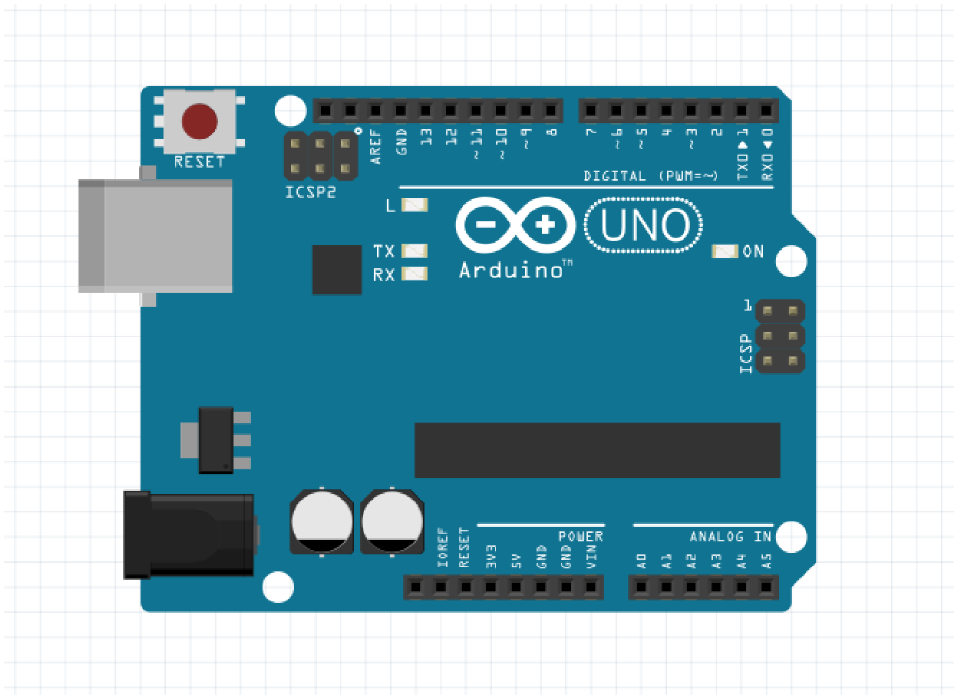


Figure 18 Arduino Uno pin layout

In this thesis, a motor driver shield is assembled on the Arduino board, which will be discussed in 4.2. Some pins' function has been changed due to the motor driver shield. And some pins remain their original function. Such as pin A2 can be used as an analog reader. By using the code AnalogRead(), the position information from the sensor can be read by Arduino.

Figure 19 An example of an Arduino shield

The figure above shows an Arduino shield. With a shield, it's pretty convenient to extend the functions of an Arduino board. There are many types of Arduino shields. In this thesis, the shield is a motor driver.

4.2 The DC motor driver

Since the Arduino Uno board can not directly drive the DC motor, the DC motor drive circuit is essential. As mentioned in chapter 3, the H-bridge used in the control system is complex. Therefore DC motor driver is a quite convenient choice.

The DC motor driver is a drive circuit, which connects the Arduino Uno and the DC motor. Through this drive circuit, Arduino Uno is able to control the DC motor by controlling the drive circuit.

As mentioned in 3.1, the Arduino board could be interfaced by the expansion board, the shield. The DC motor driver is a shield, which provides a very convenient method to control the motor.

The driver used in this thesis is 'Dual VNH5019 Motor Driver Shield' from the Pololu Corporation. It's an Arduino shield based on the chip VNH5019. This motor driver shield could be assembled on the Arduino Uno board.

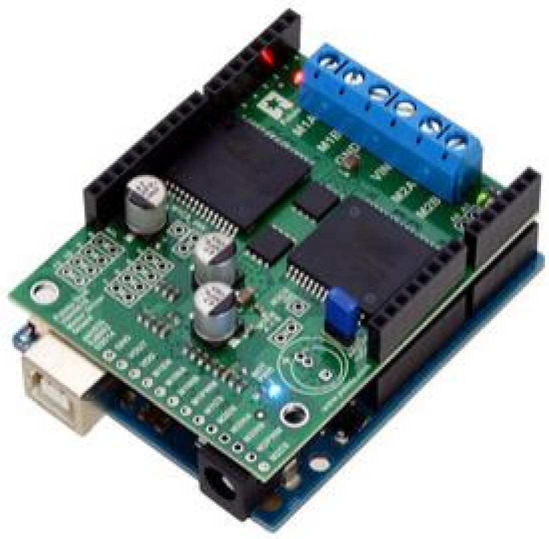


Figure 20 Dual VNH5019 Motor Driver Shield assembled on Arduino Uno

The dual VNH5019 motor driver shield and its corresponding Arduino library make it easy to control two bidirectional, high-power DC motors with an Arduino. This driver operates from 5.5 to 24V and can deliver a continuous

12A(30A peak) per channel. This driver also provides an H-bridge function to help the user control the motor in a different direction. The PWM function is also available. Again, this driver can survive input voltages up to 41V, has Undervoltage and overvoltage shutdown, high-side and low-side thermal shutdown, and short circuit protection.

Since the DC motor is 24V/10A, this motor driver is the right choice for this motor.

When Arduino Uno is assembled with a VNH5019 motor driver, the following figure shows how to connect the circuit[6].

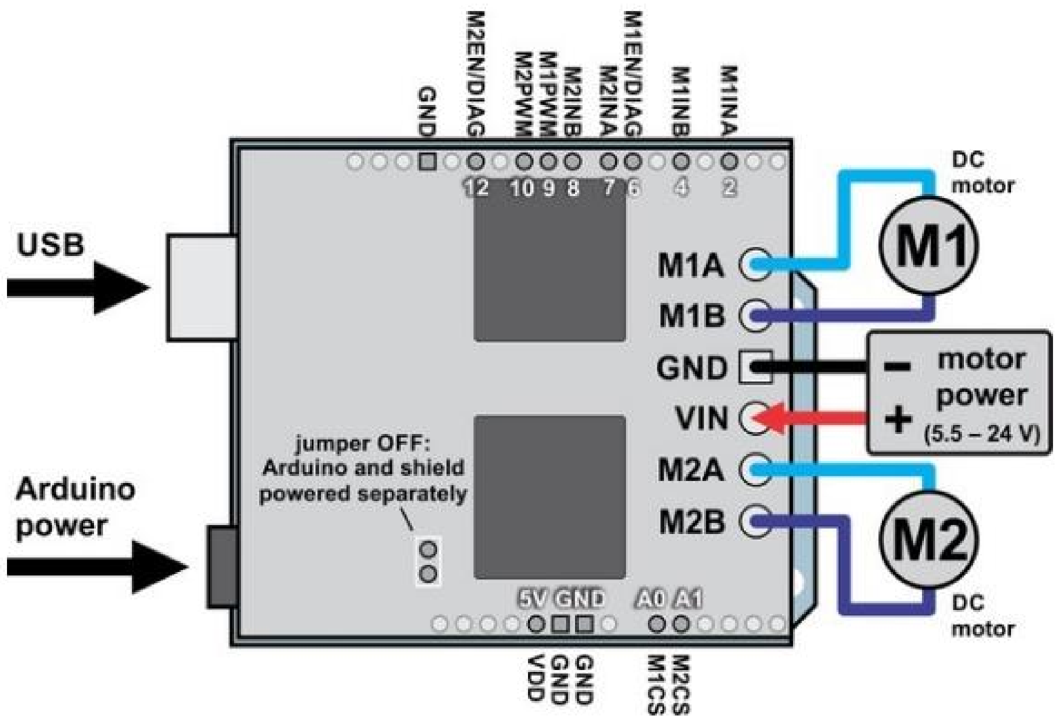
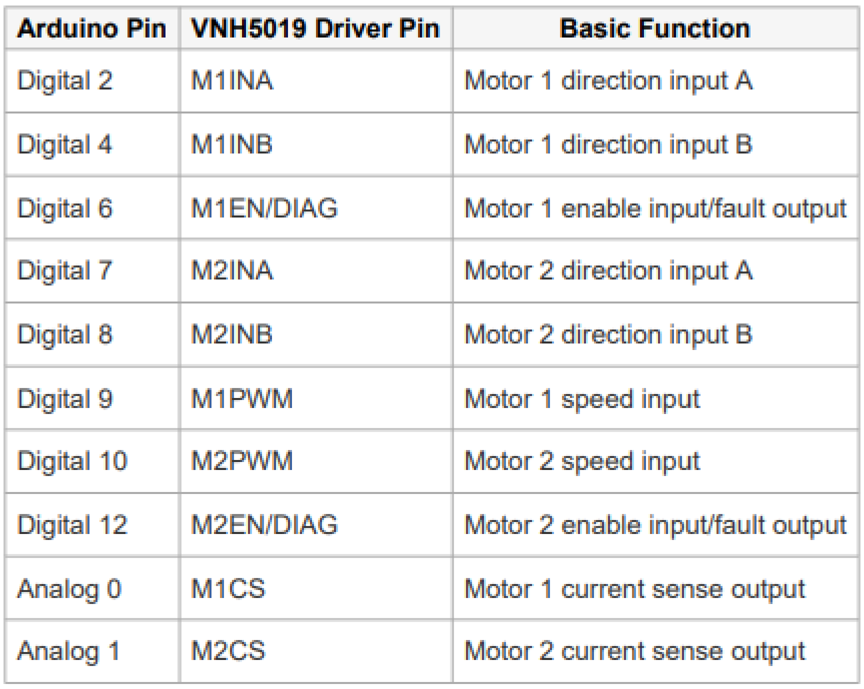


Figure 21 Dual VNH5019 motor driver shield with an Arduino Uno (shield and Arduino powered separately)

Figure 5 shows that pin 2/4/6/7/8/9/10/12/A0/A1/5V/GND obtain a new function. The other pins keep their original function. The 24V power supply is connected to VIN and GND. The motor is connected to M1A and M1B. Since only one motor is used in this thesis, M2A and M2B are left float.

The following table figure shows how the shield connects Arduino's pins to the motor driver's pins:



Arduino Pin	VNH5019 Driver Pin	Basic Function
Digital 2	M1INA	Motor 1 direction input A
Digital 4	M1INB	Motor 1 direction input B
Digital 6	M1EN/DIAG	Motor 1 enable input/fault output
Digital 7	M2INA	Motor 2 direction input A
Digital 8	M2INB	Motor 2 direction input B
Digital 9	M1PWM	Motor 1 speed input
Digital 10	M2PWM	Motor 2 speed input
Digital 12	M2EN/DIAG	Motor 2 enable input/fault output
Analog 0	M1CS	Motor 1 current sense output
Analog 1	M2CS	Motor 2 current sense output

Table 2 Function of VNH5019

The dual VNH5019 motor driver shield uses two VNH5019 motor driver ICs to enable independent control of two bidirectional brushed DC motors. Each channel is capable of delivering up to 12A of continuous current and 30A of peak current. This driver also provides the combination of two channels into a single one, which can deliver up to a constant 24A(60A peak) current to one DC motor. In this thesis, 12A is enough. The combination of the two channels is not considered.

4.3 Power supply

Many circuits in this control system need a power supply, such as Arduino board, DC motor, motor driver, and position sensor.

The Arduino's power is provided by a PC using a USB cable.

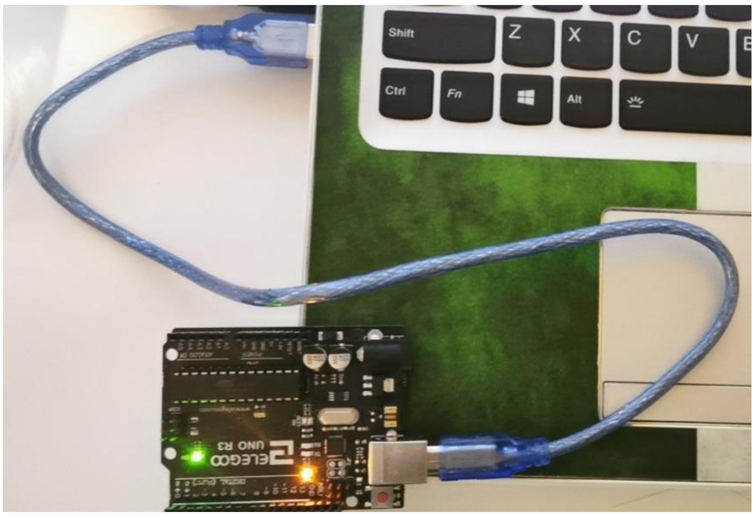


Figure 22 Arduino's power supply

And the others' power supply is from a laboratory power source 0-30V/0-5A:

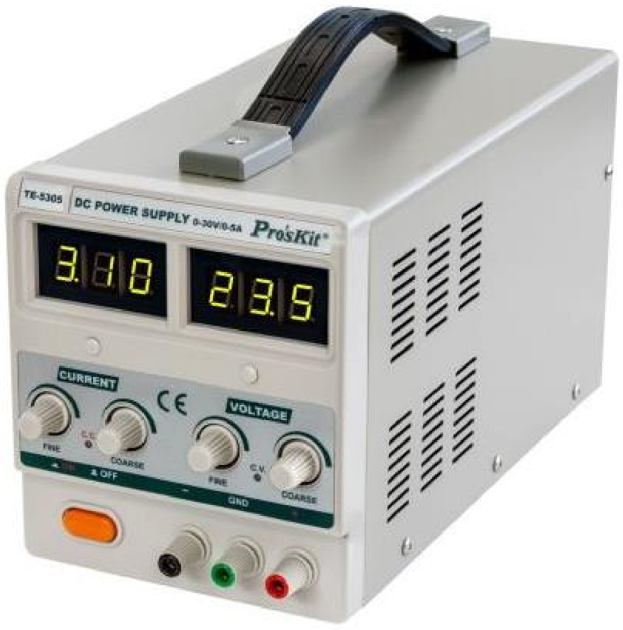


Figure 23Power source

This power source is connected to a breadboard, which can conveniently provide the power supply to many circuits.

4.4 Breadboard

A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes make it easy to insert electronic components to prototype (meaning to build and test an early version of) an electronic circuit like a battery, switch, resistor, and an LED.

The 24V voltage source is connected to the breadboard. Thus, all the circuits can use the power supply conveniently.

The figure below shows how a breadboard supplies power to different circuits. It's worth noting that this figure is not the circuit for this thesis.

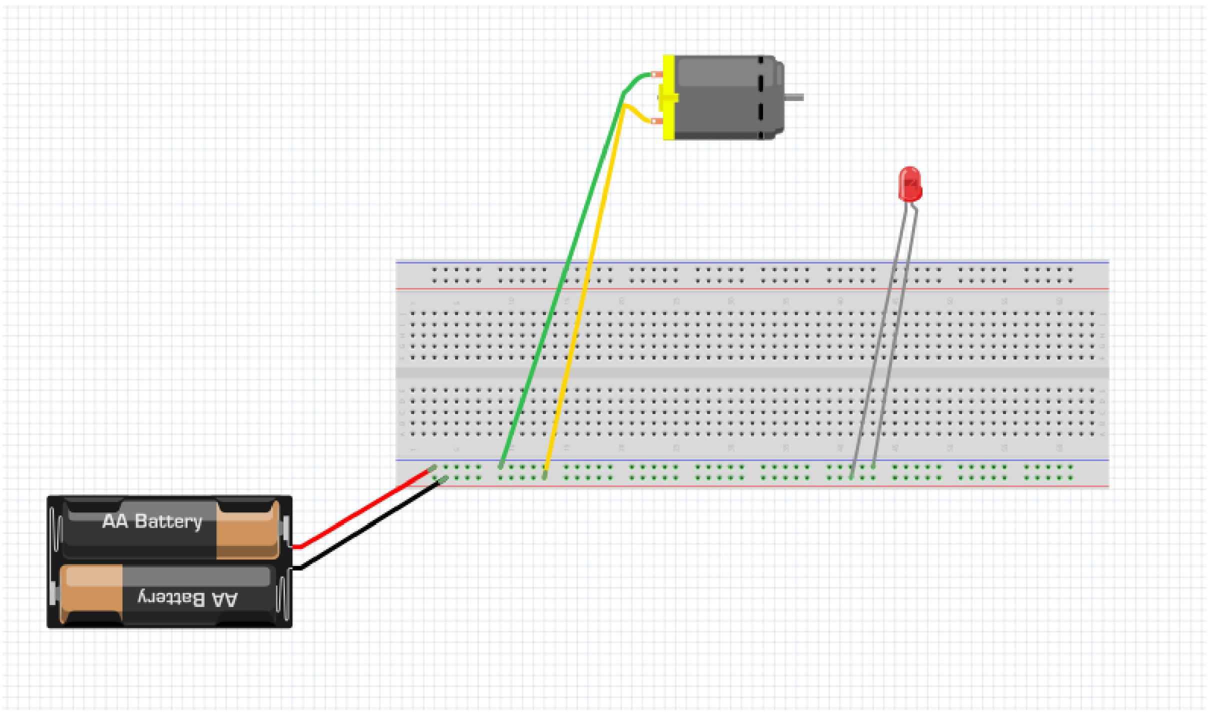


Figure 24 Breadboard and multi-power supply

4.5 Position sensor

Since this is a closed-loop system, a position sensor must provide the process position value to the microprocessor.

There are many position sensors in the market with different principles. In this thesis, the motion is linear displacement. Therefore a linear position sensor is necessary. And since the Arduino can read analog information such as voltage, the sensor should provide position data by means of the voltage value.

A potentiometer is a proper choice. The position sensor used in this thesis is a spring return linear displacement sensor from Penny & Giles: the HLP190 linear potentiometer[7].

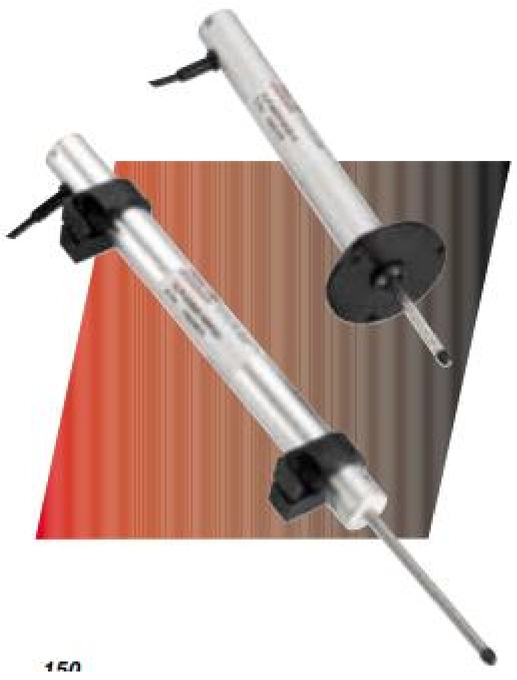
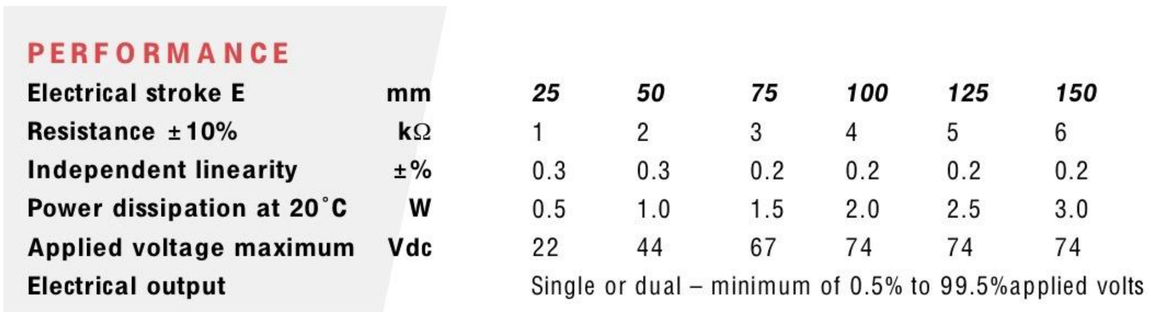


Figure 25 HLP190 linear potentiometer, Penny & Giles

The HLP190 linear potentiometer provides the facility for single or dual electrical output. The stroke length is 50mm, with the body flange mounting.

The table below is the performance data of HLP190. Since it's a linear potentiometer, the Input and Output are both voltage values.



PERFORMANCE							
Electrical stroke E	mm	*25*	*50*	*75*	*100*	*125*	*150*
Resistance ±10%	kΩ	1	2	3	4	5	6
Independent linearity	±%	0.3	0.3	0.2	0.2	0.2	0.2
Power dissipation at 20˚C	W	0.5	1.0	1.5	2.0	2.5	3.0
Applied voltage maximum	Vdc	22	44	67	74	74	74
Electrical output		Single or dual – minimum of 0.5% to 99.5%applied volts					

Table 3 Data of HLP190

Since the 50mm stroke's maximum applied voltage is 44V, it's possible to connect the position sensor to the 24V breadboard voltage source. Therefore, the sensor's output is 0V to 24V, which is not suitable for the Arduino board because the voltage range of the Arduino board is 0V to 5V. A partition circuit is necessary for the position sensor's output. Another limit is that an Arduino board can only tolerate 40mA current.

Considering the limits above, the resistance of the position sensor is:

$$R = 5V \div 40mA = 1250\Omega \tag{4.1}$$

Thus, the resistances for the partition circuit are:

$$R_1 = 1250 \times \frac{5}{24} = 260\Omega \tag{4.2}$$

$$R_2 = 1250 \times \frac{19}{24} = 990\Omega \tag{4.3}$$

The position sensor partition circuit is shown in figure 24.

Right now, the Arduino board can read the voltage value from the sensor. It means that the Arduino board can read the position information.

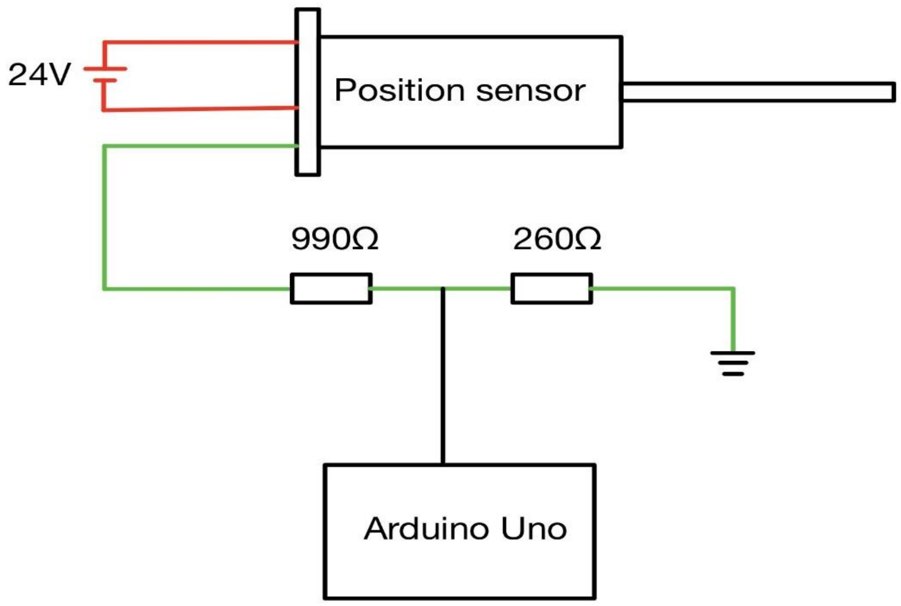


Figure 26 Position sensor circuit

Chapter 5 Arduino coding

5.1 Introduction

Arduino programming is based on the language C/C++. Generally, the Arduino language is a group of API(Application programming interface). Therefore, the Arduino programming is more straightforward than the traditional microprocessor.

The basic structure of Arduino programming is the following:

1.Void setup(){

2. }

3.Void loop(){

4. }

The code in the setup will run once. And the code in the loop will run repeatedly.

The flow chart is the following:

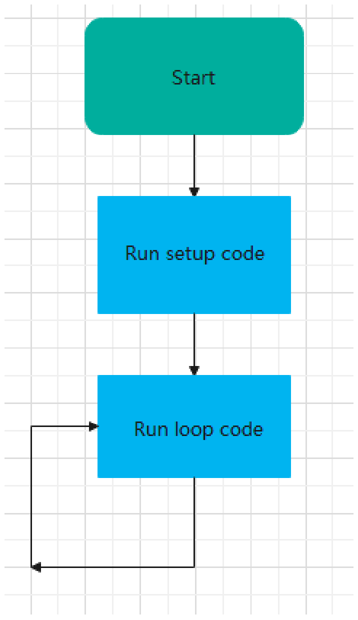


Figure 27 Arduino code flow chart

5.2 Position data read

It's quite crucial to read the data from the position sensor. As mentioned before, the position sensor data is a voltage signal with a range of 0-5V. Arduino can deal with a voltage signal that is smaller than 5V.

Since the voltage signal is analog, the code utilized to read this signal is AnalogRead(A), where A is the pin that can read the analog signal. In this thesis, the pin to read the position signal is A4.

The code AnalogRead reads the value from the specified analog pin. The Arduino board contains a six-channel, 10-bit analog to digital converter, which means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of 5 volts / 1024 units or 0.0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

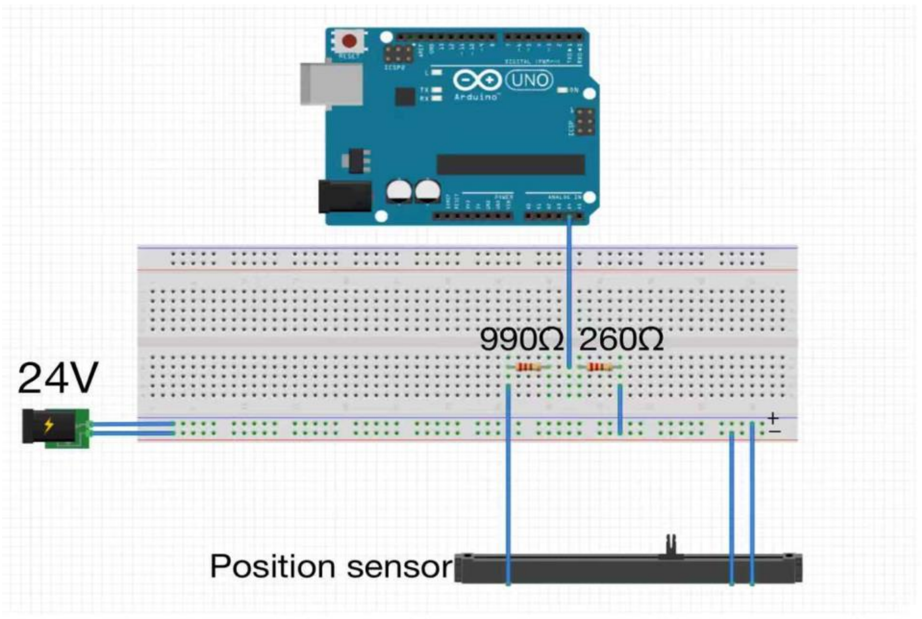


Figure 28 Circuit of position read

5.3 Position data remap

As mentioned before, the position data is a number in the range [0,1024]. It doesn't show the real position of the slider.

A code is introduced for this purpose: map(A,n1,n2,n3,n4). A is the variable, n1 and n2 are the range to be remapped, n3 and n4 are the new range of n1 and n2.

In this thesis, variable A is the data from the position sensor. Range n1 to n2 is 0 to 1024. And range n3 to n4 is 0 to 50, representing the position sensor's stroke, 0mm to 50mm.

Finally, by using the code map(), the position data can provide real position information.

5.4 Serial communication

The Input of this system is the user's order. Therefore, it's necessary to build the serial communication that makes Arduino know the expected position.

The codes used in serial communication are:

Serial.begin(): Open the serial communication and Sets the data rate in bits per second for serial data transmission. In this thesis, the bit rate is 115200.

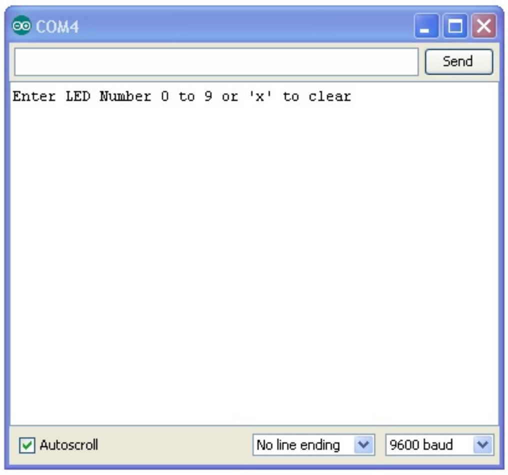
Serial.available(): Check if there is data in the serial receive buffer or not. If there is data in the buffer, the return value will be larger than 0. Therefore, if the user inputs an order, the code Serial.available() will be larger than 0. In this way, the Arduino knows if there is an order from the user.

Serial.parseInt(): Looks for the next valid integer in the incoming serial stream. parseInt() inherits from the Stream utility class. In particular: Initial characters that are not digits or a minus sign are skipped; Parsing stops when no characters have been read for a configurable time-out value or a non-digit is read.

Thus, the system only receives the numbers. For example, if user input is a3b7c, the system will receive 37 as the order. The system obtains high anti-interference ability.

Serial.print(): The code prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as-is. This code can provide the information we are interested in the serial window, such as the current position.

A serial monitor is shown as following (note: just a general monitor, not used in this thesis):



```
Enter LED Number 0 to 9 or 'x' to clear
```

Figure 29 Serial monitor

The order can be sent on the top.

5.5 PID library

The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches. Many libraries come installed with the IDE. Other users could also create a library.

There are many different PID libraries in Arduino. The most popular one is the library created by Brett Beauregard.

After downloading this PID library in Arduino IDE, the code #include <PID_v1.h> should be stated at the beginning of the program to utilize this library. Then, the functions of this PID library can be easily used in programming.

The functions of this library used in this thesis are:

1.PID(&Input, & Output, &Setpoint, Kp, Ki, Kd, Direction)

Where:

Input: The variable we're trying to control (double)

Output: The variable that will be adjusted by the PID (double)

Setpoint: The value we want to Input to maintain (double)

Kp, Ki, Kd: Tuning Parameters. these affect how the PID will change the output. (double>=0)

Direction: Either DIRECT or REVERSE. Determines which direction the output will move when faced with a given error. DIRECT is most common.

2. Compute()

This function contains the PID algorithm. It should be in the loop and return a computed value every loop.

3. **SetOutputLimits**()

The Arduino has a PWM range 0-255 by default. This code makes the PID controller vary its output within a given range. In the motor driver's library, the speed range is [-400,400]. -400 means rotating reversely at the maximum speed, and 400 means rotating forward at the maximum speed. It’s necessary to use SetOutputLimits(-400,400) to vary the PWM range.

4. **SetMode**()

There are two modes in this PID library, automatic and manual. In this thesis, automatic mode is selected.

5. **SetSampleTime**()

This code will determine how often the PID algorithm evaluates. The default is 200ms. It depends on different applications. For a DC motor, 50ms is a proper choice.

6. **SetControllerDirection**()

As mentioned before, the PID controller has two directions. For example, the refrigerator is a typical reverse control system. In this thesis, the system is direct controlled.

5.6 Motor driver library

It's similar to the PID library. The motor driver has its library, which makes it easy to control the motor driver.

The code #include <DualVNH5019MotorShield.h> should be stated at the beginning of the program to utilize this library. This library creates several functions.

The functions of this library are:

1.void init():

Initialize pinModes and timer.

2.void setM1Speed(int speed):
The code is to set speed and direction for motor 1. Speed should be between -400 and 400. 400 corresponds to motor current flowing from M1A to M1B. -400 corresponds to motor current flowing from M1B to M1A. 0 corresponds to full coast.

And this is the reason to set the output limit as [-400,400] in the PID library.

The code setM2Speed(int speed) is not shown because the motor driver is on the single-channel mode.

The motor driver library provides the system with the continuous adjustment ability because it's easy to adjust the motor's speed by varying the number from 0 to 400 and adjusting the direction of rotation by changing the number's sign.

5.7 The code

This part will show the Arduino IDE code with explanations[8].

```
// Start coding

#include <PID_v1.h>  //Open the PID library

#include <DualVNH5019MotorShield.h> //Open the motor driver library


DualVNH5019MotorShield md; //Name the motor as md


double Kp=1, Ki=0.1, Kd=0.01; //Define PID parameters

double input , output , setpoint; //Define PID values

PID myPID(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT); //Create a new
PID

int User_input; //Define the user’s input

int i,a; //Define count number

int realposition; //Define real position

void setup() {

Serial.begin(115200); //Open serial communication with bit rate 115200

md.init(); // Initialize pinModes and timer

myPID.SetMode(AUTOMATIC); //set pid mode as automotic

myPID.SetSampleTime(50); //In ms, 50ms is good for DC motor

myPID.SetOutputLimits(-400,400); // Since the speed range for VNH5019 is [-
400,400],so we reflect this range.
```

```
}

void loop() {
  if(Serial.available()>0) { //Check if there is data input in buffer
    User_input=Serial.parseInt(); //Get the setpoint from the input
    Serial.print("The user input is: ");
    Serial.println(User_input); //Print user input in serial window
    setpoint=map(User_input,0,48,1024,0); // Remap position data and real
position
    i=0;
    a=0;//Count number
  }
  if(User_input>5&User_input<48){  // Define the position range to avoid
possible accidence
    i=i+1;
    input=analogRead(A4); //Read the position data from sensor
    realposition=map(input,0,1024,48,0); //Remap the sensor data and real
postion
    if(i%1700==0){ //Give the current position every 1700 loop, about 100ms
      Serial.println(realposition);
      a+=1;
    }
    if(a==120){
      Serial.println("over"); //Limit the total time to test if the system is swift
      }
    myPID.Compute(); // Compute the PID output
    md.setM1Speed(output); // Define the output as PWM value
  } else{ // If the user's input is out of [0,48],remind user to input again
    Serial.println("Please input the position you want (from 0mm to 48mm):");
    delay(5000);
    }
}
```

The process is shown in the following figure:

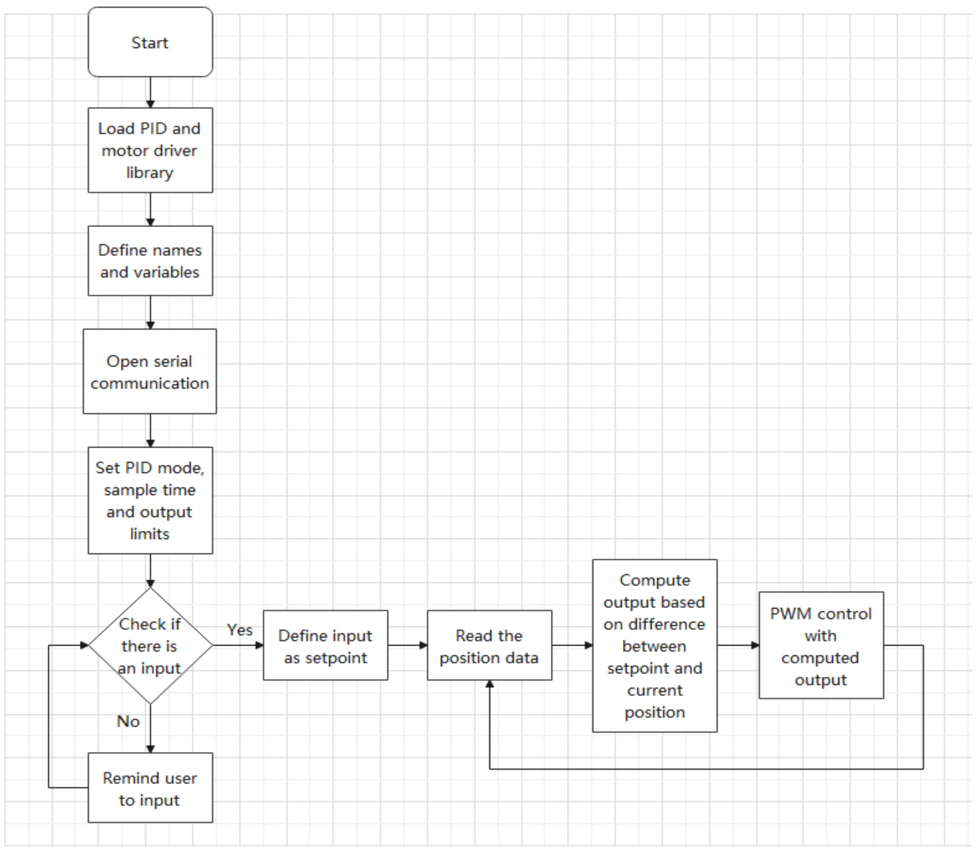

Start
Load PID and motor driver library
Define names and variables
Open serial communication
Set PID mode, sample time and output limits
Check if there is an input
Yes
No
Remind user to input
Define input as setpoint
Read the position data
Compute output based on difference between setpoint and current position
PWM control with computed output

Figure 30 Flow chart of the process

Chapter 6 Result analysis

6.1 Setup

This project is directly operated in the laboratory. The PID parameters are set as Kp=1, Ki=0.1, Kd=0.01 by default. It's not essential if the parameters are good enough to achieve the goals, but it's interesting to check if the system works well.

Since the slider's position range is [5,48], the slider may oscillate near the setpoint, the start point is at 5mm, and the goal is at 35mm to give the slider some space to oscillate.

6.2 Result

The first test has PID parameters (1, 0.1, 0.01), which means Kp=1, Ki=0.1, Kd=0.01. As mentioned before, the test will finish after 120 steps. And, one step is 1700 loops, which means around 100ms for one step. Thus, the slider must reach the setpoint within 12s to guarantee the sensitivity of the system.

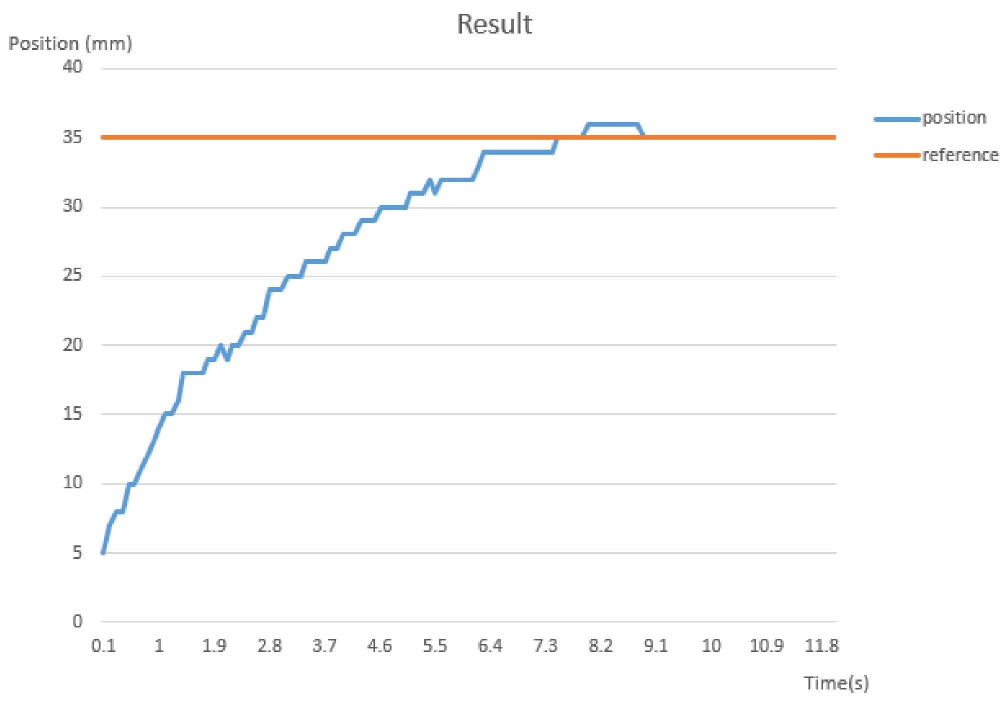


Figure 31 First test result

As shown in the result, the slider reaches the setpoint at 6.1s. The system is high-speed that can satisfy many industry applications. About the preciseness, the result shows that the slider stops at 35mm, which is precisely the setpoint.

Here is the second test with PID parameters (5, 0.1, 0.01) to double-check the system performance.

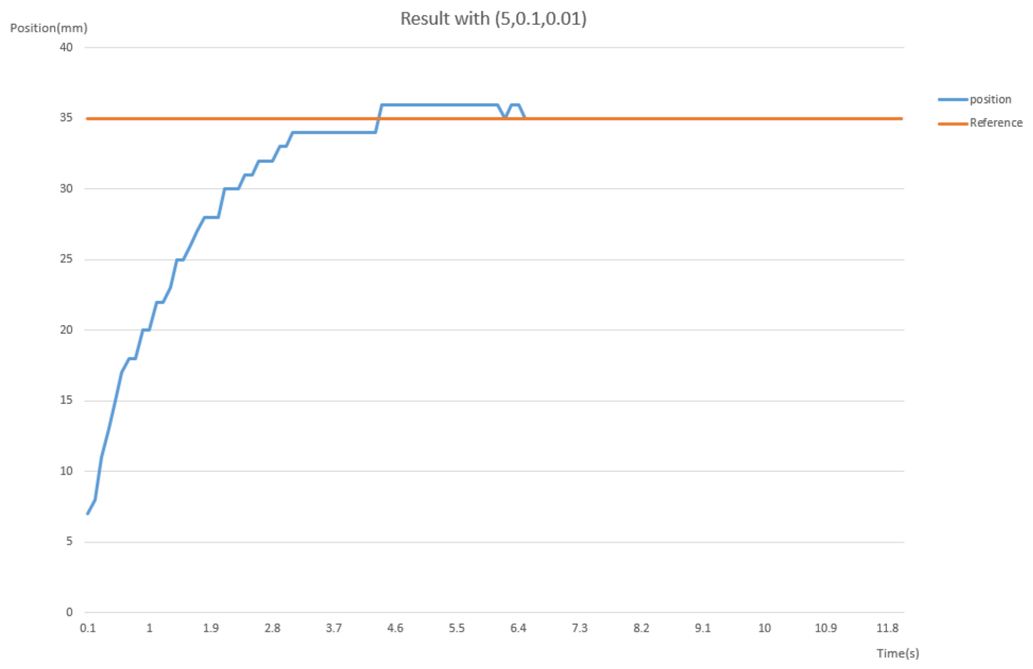


Figure 32 Result with (5, 0.1, 0.01)

The figure shows that the system reaches setpoint at step 30 and then oscillate. It's even faster than test 1. Therefore, the speed of the control system has been double-checked. And the slider finally stops at 35mm, which means the system is precise.

To check if the system is able to continuous control, first, let the slider move from 5mm to 35mm, then 35mm to 5mm. This time the test has 240 steps, which means around 24s.

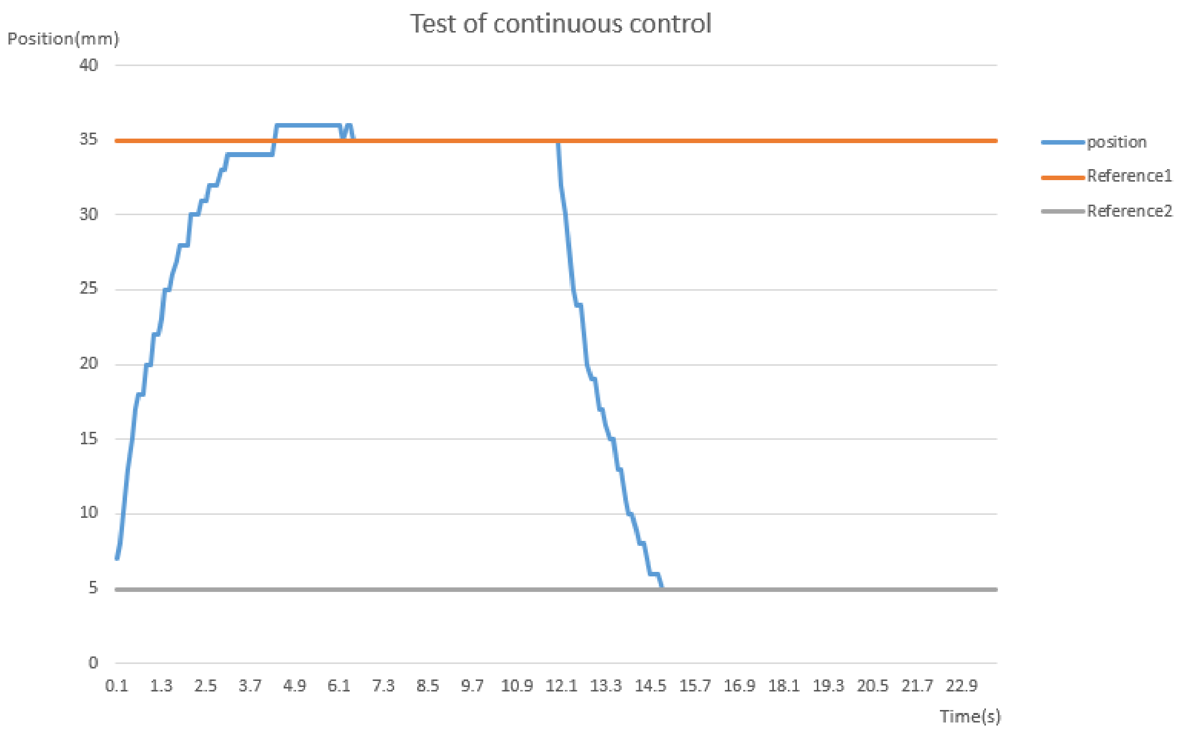


Figure 33 Continuous control test

The continuous control test result shows that the control system can work continuously.

Thus, the requests of the control system have been achieved.

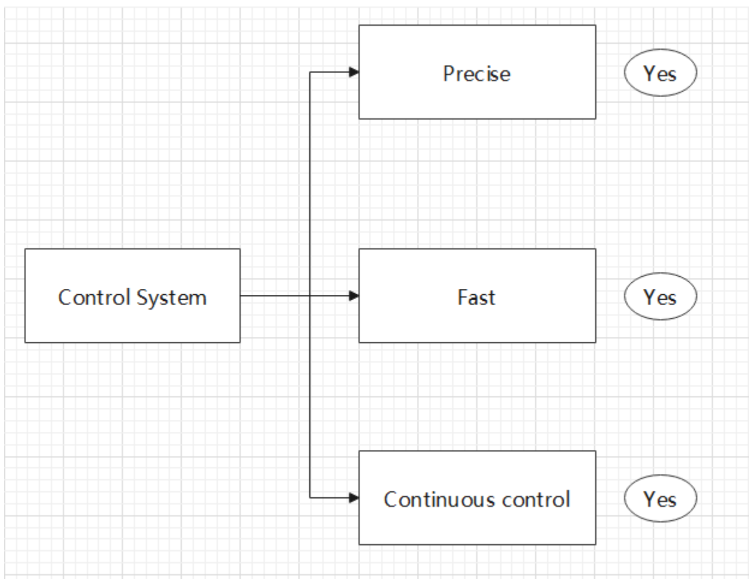


Figure 34 The characteristics of the system

6.3 Method to evaluate suitable PID parameters

It's significant to improve the performance of the system by assessing appropriate PID parameters.

The method is variable control. A variable in the test keeps constant to check the relationship between multiple variables because its unchanging state allows the relationship between the other variables being tested to be better understood.

Therefore, each parameter will be checked one by one to evaluate the effect of the parameter. Finally, the best combination will be tested to check if it leads to the best performance.

What influences the system more is the proportional parameter and the integral parameter. It's not very obvious to check the influence of the derivative parameter. Therefore, the tests are mainly on PI control.

6.4 Evaluation index

In general, there are some indexes to evaluate the performance of a PID system.

They are:

1. Response time: the first moment that the slider reaches the position which is near the setpoint. It's evident that the faster the response time, the better the system behaves.
2. Oscillation amplitude: the system may oscillate before it reaches a stable state. It's meaningful to check the oscillation amplitude.
3. Precision: under some PID parameters, the system may never reach the setpoint. Precision means the difference between the final position and the setpoint. It's essential to check its accuracy because some applications need a high preciseness level.

6.5 Result diagrams and table

The following figures show the effect of the proportional parameter. The integral parameter and the derivative parameter are zero.

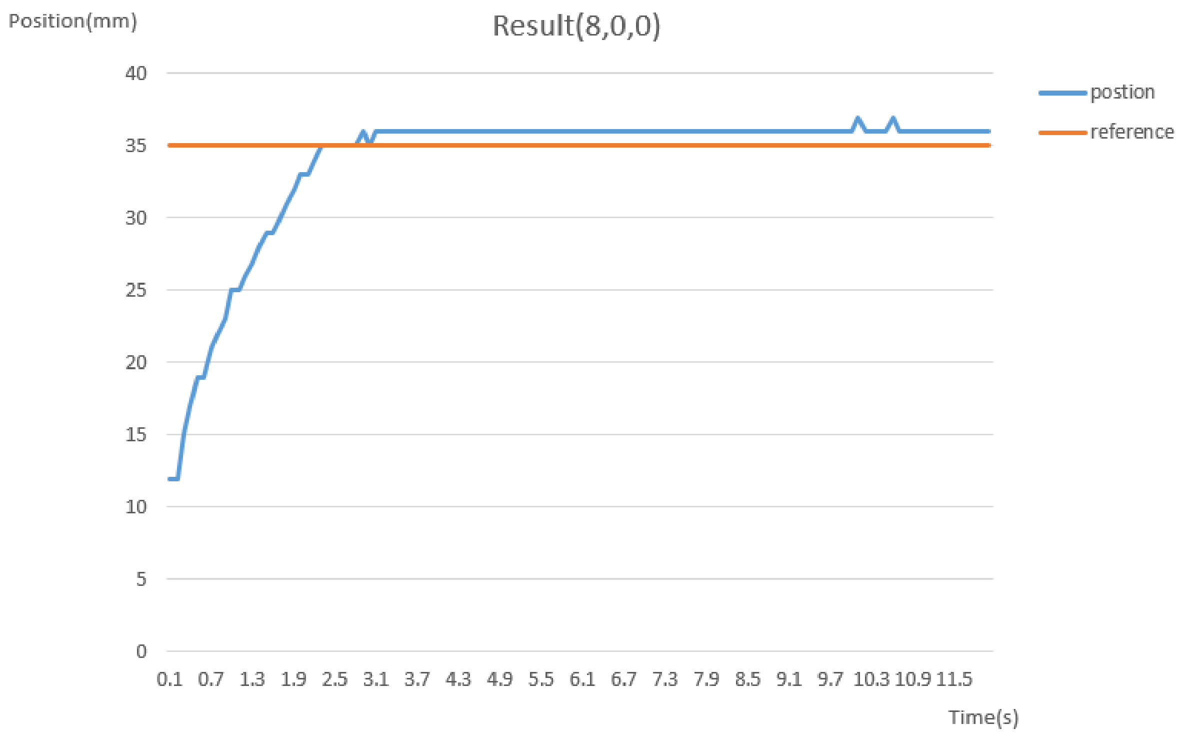


Figure 35 Kp=8, Ki=0, Kd=0

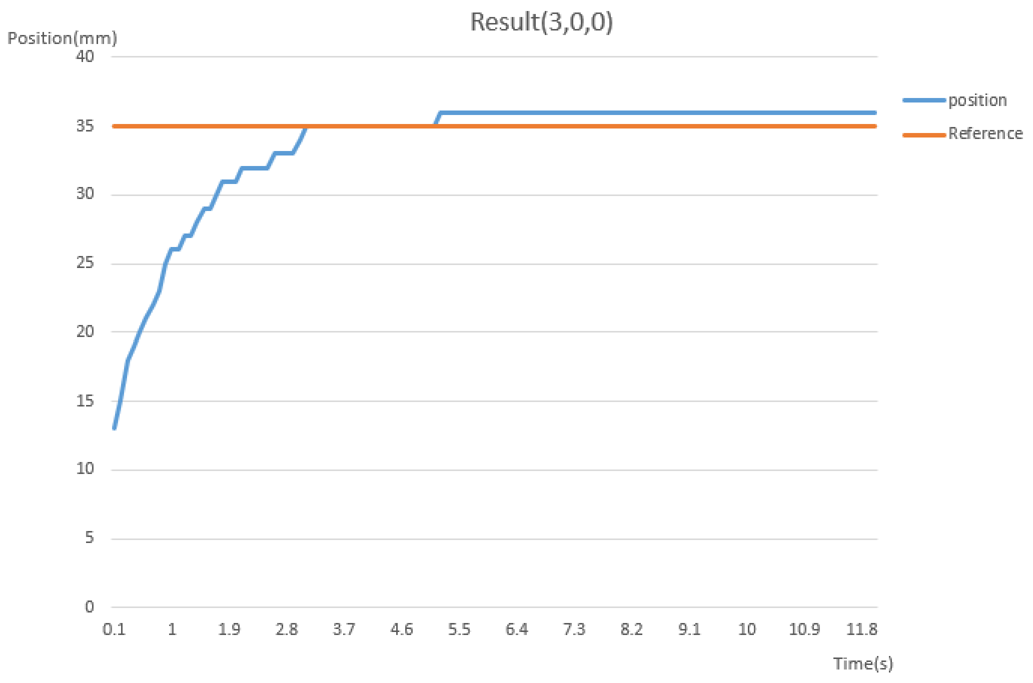


Figure 36 Kp=3, Ki=0, Kd=0

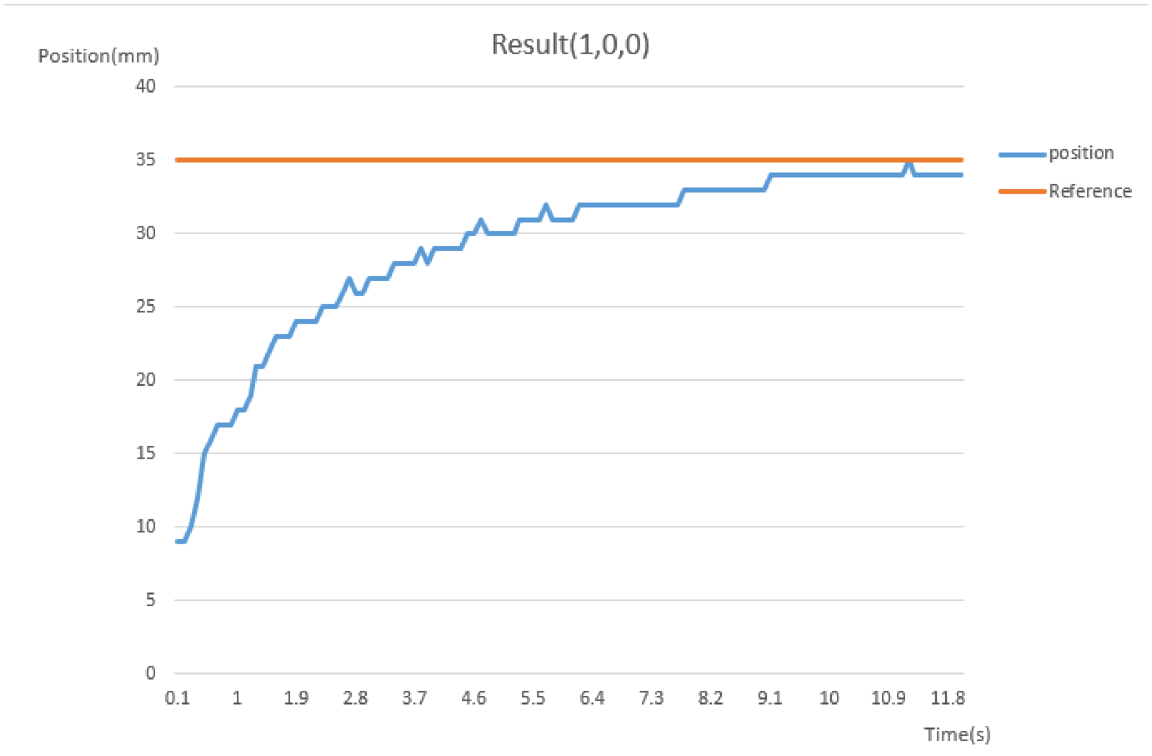


Figure 37 Kp=1, Ki=0, Kd=0

The table:

Test	Kp	Ki	Kd	Response time(s)	Amplitude(mm)	Precision(mm)
1	8	0	0	2.1	0	1
2	3	0	0	3.1	0	1
3	1	0	0	9	0	1

Table 4 Evaluation of Kp

Then, the following figures show the effect of the integral parameter. The proportional parameter and the derivative parameter are zero.

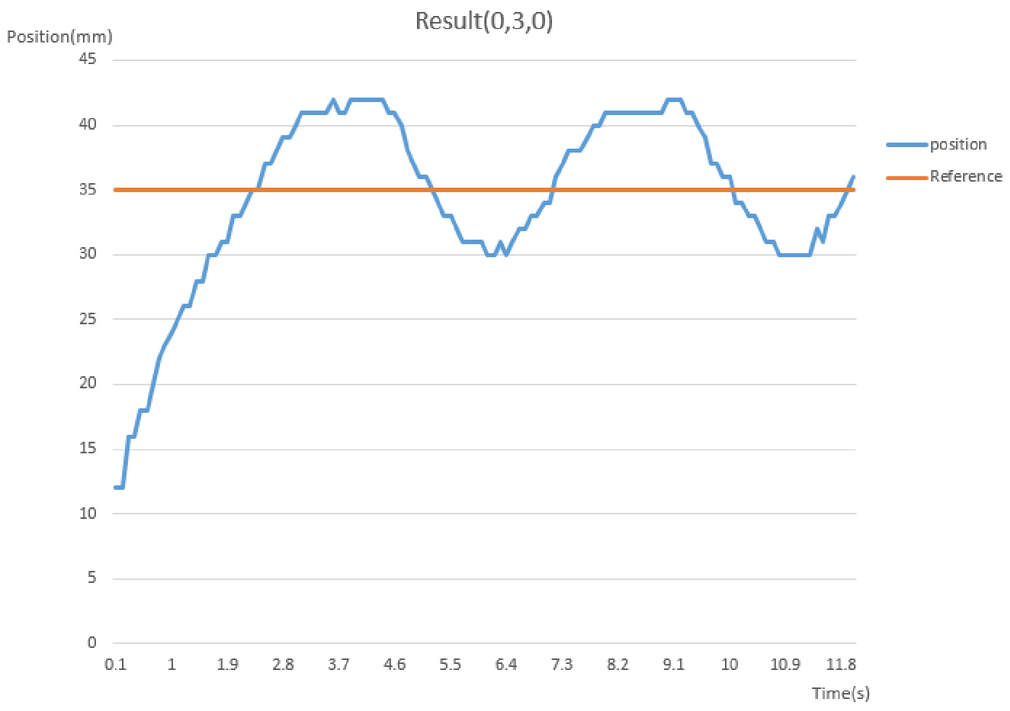


Figure 38 Kp=0, Ki=3, Kd=0

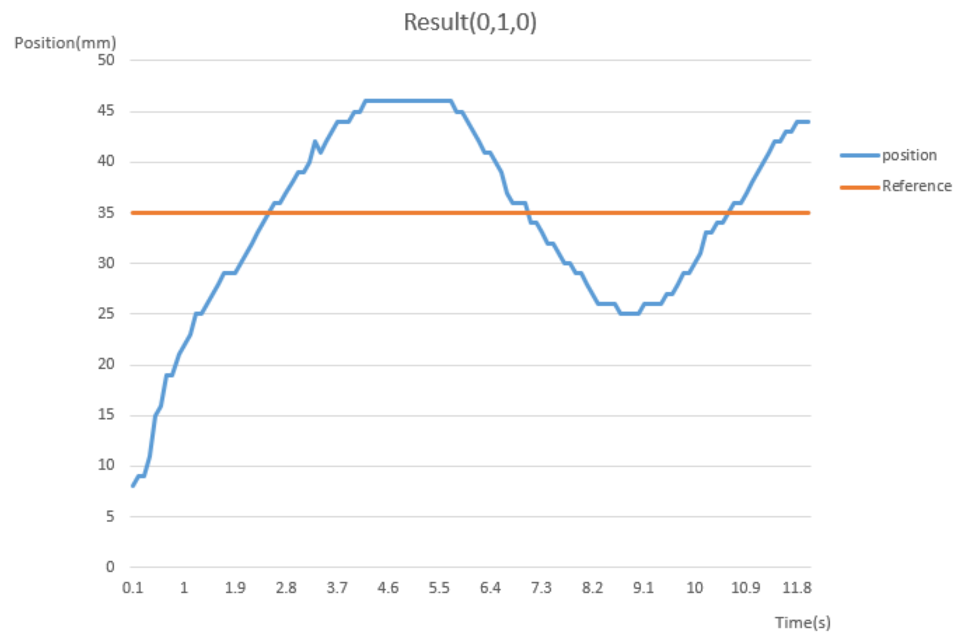


Figure 39 Kp=0, Ki=1, Kd=0

The table:

Test	Kp	Ki	Kd	Response time(s)	Amplitude(mm)	Precision(mm)
4	0	3	0	2.5	6	3 in average
5	0	1	0	2.8	10	5 in average

Table 5 Evaluation of Ki

It's not possible to evaluate the derivative parameter separately because the system will crash with only the derivative parameter. However, it's possible to check the derivative parameter with the other parameters non-zero.

The results are as following:

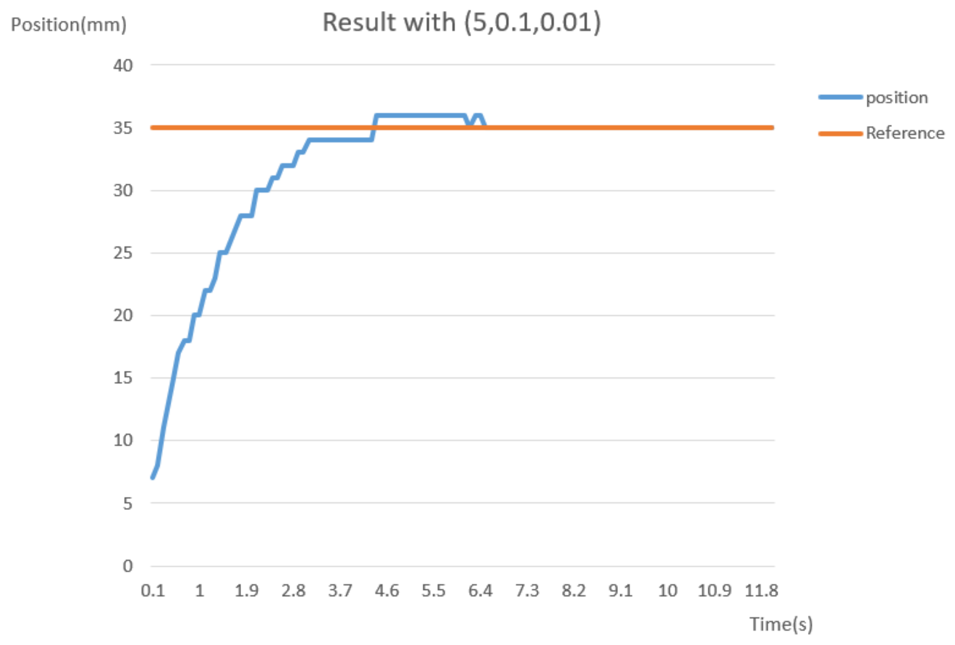


Figure 40 Kp=5, Ki=0.1, Kd=0.01

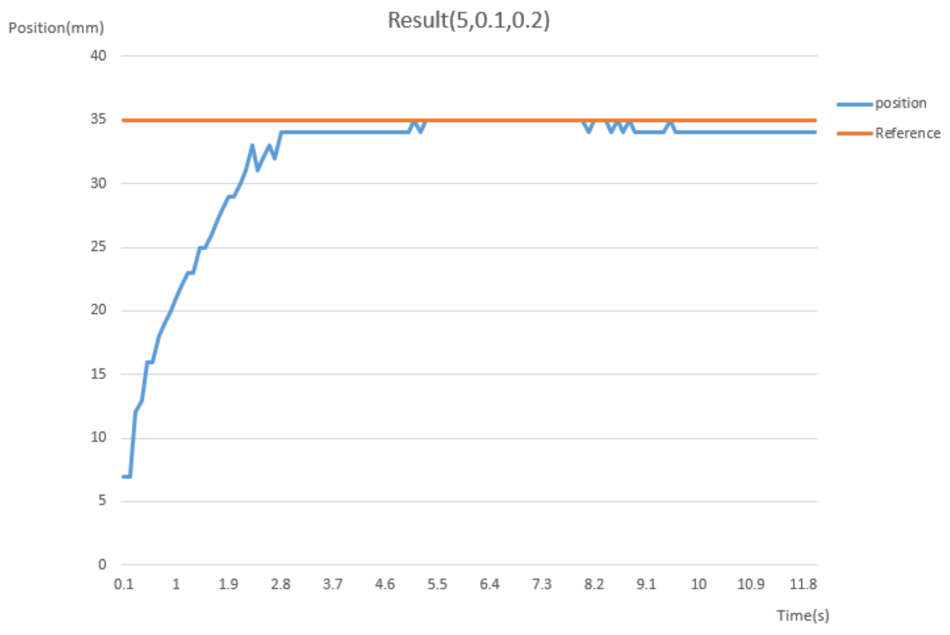


Figure 41Kp=5, Ki=0.1, Kd=0.2

The table:

Test	Kp	Ki	Kd	Response time(s)	Amplitude(mm)	Precision(mm)
6	5	0.1	0.01	3	1	0
7	5	0.1	0.2	2.9	1	0

Table 6 Evaluation of Kd

It's also necessary to evaluate the PI control method because they may be coupled with each other.

Here is the result with Kp=5, Ki=0.5, Kd=0.01 that can be compared to the test Kp=5, Ki=0.1, Kd=0.01 to evaluate how integral parameters influence the system.

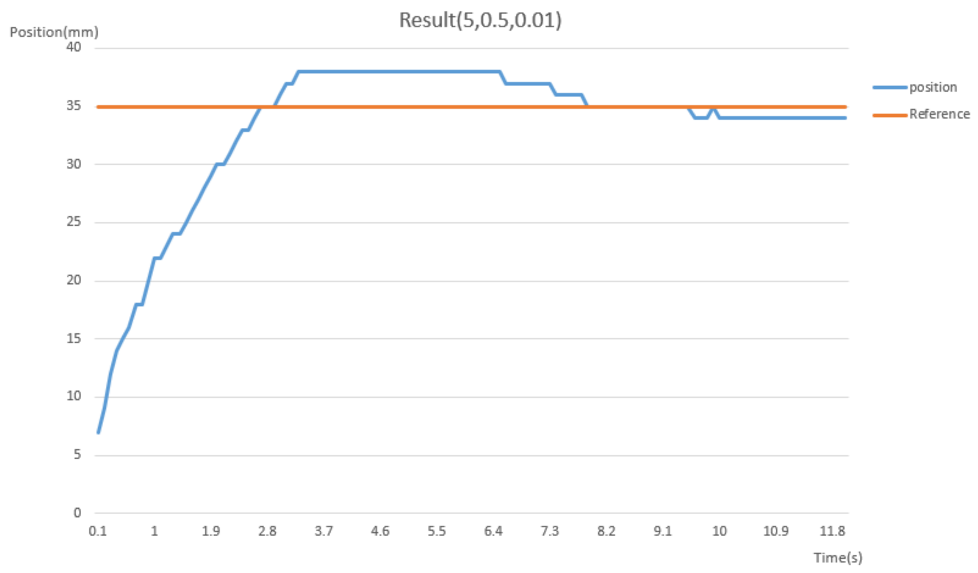


Figure 42 Kp=5, Ki=0.5, Kd=0.01

Compared to (5, 0.1, 0.01), here is the table:

Test	Kp	Ki	Kd	Response time(s)	Amplitude(mm)	Precision(mm)
6	5	0.1	0.01	3	1	0
8	5	0.5	0.01	3	3	0

Table 7 Evaluation of PI coupled parameter

6.6 Suitable PID parameters

The conclusions in chapter 6.5 are the guideline to find a suitable PID parameter. This parameter guarantees a fast response, low oscillation, and high precision system.

The proportional parameter should be large for quickly reaching the setpoint. The integral parameter should be small but not zero for smaller oscillation amplitude and eliminating the steady-state error.

The derivative parameter should be small but not zero for a better oscillation performance.

Thus, with the guideline, the result of Kp=10, Ki=0.1, Kd=0.01 could be a suitable combination.

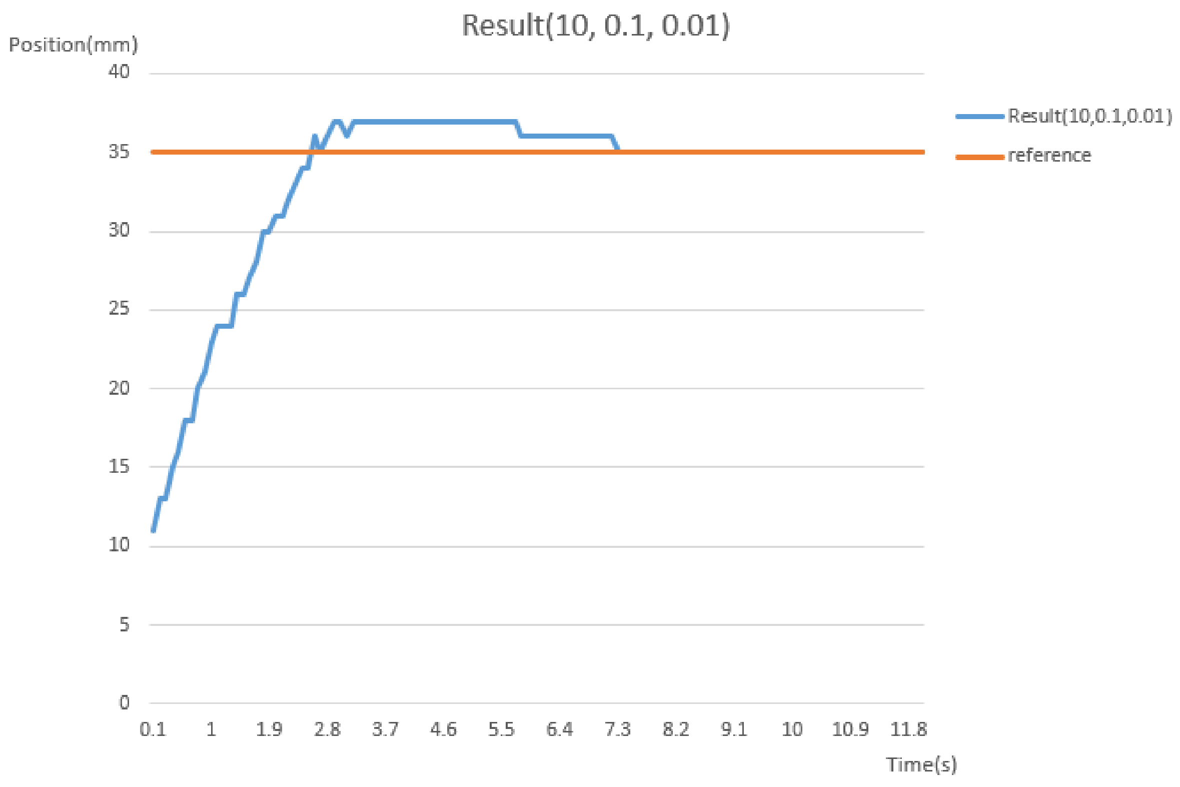


Figure 43 Kp=10, Ki=0.1, Kd=0.01

With this PID parameter, the system has a quick response time of 2.5s, a low oscillation amplitude of 2mm, and a very stable final result with high precision.

It's not recommended to further increase the proportional parameter because, with proportional parameter 10, the current has already reached the peak value at the beginning. It won't be faster with a larger proportional parameter.

Chapter 7 Conclusions

According to the results in the previous chapter, it’s apparently to say the system is good to achieve all the tasks. Also, it’s clear to see how the PID parameters affect the performance of the system.

Based on figure 31, 32, the system is fast, accurate and based on figure 33, the system is able to work continuously.

Based on table 4, the conclusions for Kp are:

1. The larger the proportional parameter, the faster the response time to reach the setpoint.
2. With only a proportional parameter, there is a steady-state error that is not avoidable. Therefore, the precision, the difference between the setpoint and the final position is always 1mm.
3. With only a proportional parameter, the system doesn't oscillate.

Based on table 5, the conclusions for Ki are:

1. The larger the integral parameter, the faster the response time. This characteristic is similar to the proportional parameter.
2. With only an integral parameter, the system will oscillate forever. The amplitude keeps constant. The slider will never stop at the setpoint. Moreover, the larger the integral parameter, the smaller the oscillation amplitude.
3. The precision is awful, with only an integral parameter. The difference between the setpoint and the slider position is not constant because of oscillation. However, the average difference can be calculated. And the larger the integral parameter, the smaller the difference between the setpoint and the slider position.

Based on table 6, the conclusions for Kd are:

1. The derivative parameter has a lower influence on the system performance compared to the other parameters.
2. Derivative parameter mainly changes the oscillation parameters. In this test, because the current flow in the DC motor is small, the influence of Kd is not apparent. However, it still shows that the system oscillates a little bit more with the higher derivative parameter.

The conclusions above only consider the independent parameter. Thus, it doesn't show how the parameters influence each other. Therefore, table 7 represents the coupled effect of Kp and Ki:

1. If we consider the integral parameter coupled with the proportional parameter, the conclusion is different than before.
2. The larger the integral parameter, the larger the oscillation amplitude.
3. The integral parameter has almost no influence on precision if considering three parameters together.

Finally, according to the conclusions all above, a PID parameter Kp=10, Ki=0.1, Kd=0.01 leads to a quite good performance of the system.

Reference

[1]Xia, Chang-liang. Permanent Magnet Brushless DC Motor Drives and Controls. 1. Aufl. ed. Singapore: Wiley, 2012. pp.1-2.
[2]Woo. Reliability Design of Mechanical Systems. Springer Singapore, 2020. pp.1-3.
[3]Kumar Peddapelli, Satish. Pulse Width Modulation. De Gruyter Oldenbourg, 2016. pp.1-6.
[4]Kim, Sang-Hoon. Electric Motor Control. Saint Louis: Elsevier Science & Technology, 2017. pp.56-75.
[5] D'Ausilio, Alessandro, and D'Ausilio, Alessandro. "Arduino: A Low-cost Multipurpose Lab Equipment." Behavior Research Methods 44.2 (2012): 305-13.
[6] Pololu company. Pololu Dual VNH5019 Motor Driver Shield User's Guide. 12/2021. https://www.pololu.com/docs/0J49/all
[7] Curtiss-Wright company. HLP190FS(BS)-Spring-Loaded Linear Potentiometer User's Guide. 12/2021. https://www.cw-industrialgroup.com/Products/Sensors/Linear-Position-Sensors-Transducers/Spring-Loaded-Linear-Potentiometer-HLP190FSBS
[8] Cullen, Charlie. Learn Audio Electronics with Arduino. Taylor and Francis, 2020.