People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Tissemsilt

Faculty of Science and Technology

Department of Science and Technology

Graduation thesis for graduation

of an academic Master's degree in

Sector: Electronics

Specialty: Instrumentation

Presented by: **Ait Abdelkader Nour Islam**

Rezzouk Meziene

Thème

Control of dc motor based on atmega328p µc and PID Controller

Defended on, /12 /06/ 2023

Before the Jury:

Hamdani Mostefa	President	M.C.A	Univ-Tissemsilt
Taibi Ahmed	Examiner	M.C.B	Univ-Tissemsilt
Nail Bachir	Supervisor	M.C.B	Univ-Tissemsilt
Tibermacine Imad Eddine	Co-Supervisor	Doctorate	Univ-Rome

Academic year: 2022-2023

DEDICATED

To

I am delighted to dedicate this project,

To the invaluable support and encouragement I
received.

No combination of words can adequately convey the depth of my heartfelt affection for thosewho hold a special place in my heart.

To my beloved Mother, whose unwavering affection and unwavering belief in me have been asource of solace and inspiration. Your unwavering support, guidance, and unwavering faith inme have been instrumental in my journey. May this endeavor serve as a testament to your unwavering dedication and countless sacrifices.

To my cherished Father, whose constant drive and unwavering support have fueled my academic pursuits. It is my fervent desire that this undertaking stands as a testament to thecountless sacrifices you have made.
To my big brother amine and my wonderful sisters, , who were always there for me.

To all the friends we made during our time at the Faculty of Science and Technology, University of Tissmsilt, and beyond, I want to convey my sincere gratitude for the wonderfulmoments we shared.

To my dearest friends, barber rami, si bachir bilal whose constant encouragement meant theworld to me.
May they all find immense joy and unparalleled success.

To the cherished individuals who fill my life with love and foster an environment,Where joy thrives and a favorable atmosphere prevails.

Ait Abdelkader Nour Islam

DEDICATED

I dedicate this project to my family, whose unwavering support and encouragement have been instrumental in my journey. Your belief in my abilities and constant motivation have pushed me to strive for excellence and pursue my dreams. Thank you for always being there for me, providing a loving and nurturing environment that has allowed me to focus on my education.

I also dedicate this project to my friends and classmates who have been by my side throughout this academic endeavor. Your camaraderie, shared experiences, and collaborative spirit have made the journey more enjoyable and memorable. We have grown together, faced challenges together, and celebrated successes together, and I am grateful for the bond we have formed.

Rezzouk Meziene

Acknowledgements

First and foremost, we would like to express our heartfelt gratitude to the Almighty for granting us the strength and perseverance throughout our academic journey.

We extend our sincerest appreciation to our supervisor, Dr. Nail Bachir, a distinguished educator at the Institute of Science and Technology. His invaluable guidance, unwavering support, and insightful directions have played a pivotal role in the successful completion of our Project Graduation.

We would also like to acknowledge the director of the Institute of Science and Technology for granting us the necessary resources and creating a conducive environment for us to carry out our work.

Lastly, we extend our deepest thanks to the esteemed president of the jury and all the members of the jury for kindly agreeing to evaluate our modest end-of-studies project. Your expertise, feedback, and evaluation have immensely contributed to our growth and development.

To all those who have directly or indirectly supported us throughout this journey, we express our heartfelt gratitude and appreciation.

Table of contents

List of figures
List of tables
Glossary
Abstract
GENERAL INTRODUCTION....1

Chapter 1 Arduinouno

1 An Overview of Arduino uno Board....2
1.1 What is Arduino?....2
1.2 Introduction to Arduino uno....3
1.2.1 Arduino uno Pinout....3
1.2.2 Arduino Uno Pinout – Diagram....4
1.2.3 Arduino Uno pinout - Power Supply....4
1.2.4 Arduino Uno Pinout - Analog IN....5
1.2.5 Arduino Uno Pinout - Digital Pins....6
1.2.6 What does digital mean?....6
1.3 Arduino uno specifications....6
2 Breadboard....7
2.1 How to Use a Breadboard to Make a Circuit?....7
2.2 Arduino uno Programming and Communication....7
2.3 Arduino Coding Environment and basic tools....8
2.3.1 What language is Arduino?....8
2.4 Arduino IDE....8
2.5 The Programing Structure of an Arduino Sketch....9
2.5.1 Choose the Correct Arduino Board....10
3 LED Blinking Using ArduinoUNO....11
3.1 Components required....11
3.1.1 Connections....11
3.1.2 Arduino code....12
3.2 Result....12
4 Difference between Arduino UNO and Arduino Nano....13
Conclusion....13

Chapter 2 Types ofcontrol

5 PID control....14
5.1 PID Controller Block Diagram....14
6 Types of PID Controller....16
6.1 ON/OFF Control....16
6.2 Proportional Control....16
6.3 Standard Type PID Controller....16
6.4 Real-Time PID Controllers....16
7 Manual tuning....17

7.1 Manual tuning procedure 18
Conclusion 19
8 PWM theory 19
8.1 Duty Cycle of PWM 19
8.2 Frequency of PWM 20
8.3 Output Voltage of PWM signal 20
9 Types of Pulse Width Modulation Technique 21
Conclusion 21

Chapter3 Methodology

10 DC motor 22
10.1 What is a DC motor? 22
10.2 Different Parts of a DC motor 23
11 DC Motor Working 25
12 What is an L298N Motor Driver ? 26
13 Features of the L298N motor driver: 26
14 L298N Motor Driver Module pinout 27
15 I2C LCD display 29
15.1 What is an I2C LCDdisplay? 29
15.2 Character LCD Display 29
15.3 I2CLCDAdapter 29
16 I2C LCD Display Pinout 30
17 Encoder working principale 30
18 Lm393 Motor Speed 31
18.1 LM393 Sensor Pinout 32
18.2 Required Materials 32
18.3 Hardware Components 32
18.3.1 Software Apps 33
18.4 Interfacing LM393 Infrared Speed Sensor with Arduino 33
18.4.1 Step 1: Circuit 33
18.4.2 Step 2: Code 33
19 Features 34

Chapter 4 Results and discussion

DC Motor Speed Regulation: System Modeling 36
20 Physical setup 36
21 System equations 37
21.1 . Transfer Function 38
21.2 State-Space 38
21.3 Design requirement 39
21.4 MATLAB representation 39
21.5 State Space 40
22 Open-loop response 41
23 PID Controller Design 42
24 Proportional control 44

25 PID control 45
26 Tuning the gains 47
27 Results and Analysis of real implementation of the system 49
27.1 Step 1-Hardware and Software needed 49
27.2 Step 2- Hardware connection 50
27.3 Step 3- Arduino Code 51
27.3.1 Arduino code will do: 51
27.3.2 Step 4 - Code works at Computer 51
Conclusion 56
Bibliography and Webography 58

List of figures

Figure 1-Types of Arduino Boards 2
Figure 2- Layout of Arduino uno Board 3
Figure 3 Arduino Uno Pinout – Diagram 4
Figure 4-Breadboard Internal Connections 7
Figure 5- Simple LED Circuit on Breadboard 7
Figure 6- Arduino IDE 8
Figure 7-Arduino Program structure. 10
Figure 8-Choose the Correct Arduino Board 10
Figure 9- LED Blinking Using Arduino uno 11
Figure 10-LED Blinking 12
Figure 11- Difference between Arduino UNO and Arduino Nano 13
Figure 12-Working of PID controller 14
Figure 13PID Controller parameter characteristics 14
Figure 14-Closed Loop System with PID Controller 15
Figure 15-Behavior and comments in the response time of the PID control system 18
Figure 16-pulse-width-modulation 19
Figure 17-duty-cycle-of-pulse-width-modulation 20
Figure 18-motor dc 22
Figure 19-Construction of DC Motor 24
Figure 20- Fleming's left hand rule 25
Figure 21- Production of torque in a DC motor 25
Figure 22- L298N driver 26
Figure 23- pin diagram of L298N 27
Figure 24-Pinouts of L298N Motor driver Module 27
Figure 25-L2CLCD screen 29
Figure 26-SerialI 2CLCD Display Adapter 29
Figure 27-I2C LCD Display Pinout 30
Figure 28-The rotor disc of the sensor 31
Figure 29-The stator of the sensor 31
Figure 30-l393 pin 32
Figure 31-lm393 Required Materials 32
Figure 32-Lm393 wire 33
Figure 33-lm393 serial 34
Figure 34-Schematic diagram of a DC Motor 36
Figure 35--Open-loop response. ..... **Erreur ! Signet non défini.**
Figure 36-The structure of the control system. 42

Figure 37-step response with propotional control..... 45
Figure 38-PID Control With Small Ki and Small Kd..... 46
Figure 39-PID Control With Small Ki and Small Kd..... 48
Figure 40--PID Control With Large Ki and Large Kd..... 49
Figure 41-Components Required to Build a PID Enabled Encoder Motor Controller..... 50
Figure 42- Experiment for DC motor speed control..... 50
Figure 43- System's response with various state feedback control parameters. 52
Figure 44- closed loop response with kp=1..... 52
Figure 45- losed loop response with PI Controller..... 53
Figure 46-System's response with Kp =0.3 and Ki= 0.6..... 54
Figure 47-System's response with Kp =0.5 and Ki= 0.5..... 54
Figure 48-Effects of PWM signal on the speed,rpm=300..... 55

Glossary

IC	Integrated Circuit
DC	direct current
PMW	pulse width modulation
USB	universal serial bus
ISCP	In Circuit Serial Programming
I/O	Input/ Output
EMF	Electromotive force
LED	light emitting diode.
GND	mass of the control part Ground
PID	Proportional Integral Derivative
RPM	Revolutions per Minute
IDE	Integrated development environment
IOT	Internet of Things
DIP	Dual in-line package
USART	Universal Synchronous-Asynchronous Receiver- Transmitter
I2C	Inter-Integrated Circuit
SDA	serial data line
SCL	serial clock line (SCL).
SPI	Serial Peripheral Interface
SS	Slave Select
MOSI	Master Out Slave In
MISO	Master In Slave Out
SCK	Serial Clock
SD	Secure Digital
FTDI	Future Technology Devices International Limited
ICSP	In-circuit serial programming header
COM	communication port
LCD	liquid-crystal display

Abstract : This abstract describes the implementation of a control system for a DC motor using a PID controller, PWM, an L298N motor driver, a speed sensor, and an Arduino Uno microcontroller. The system aims to achieve accurate and stable control of the motor's speed. The Arduino Uno receives speed feedback from the sensor, computes the PID control algorithm, and generates the PWM signal. The L298N motor driver amplifies the PWM signal to drive the DC motor. The PID controller continuously adjusts the PWM duty cycle based on the difference between the desired and measured speed, minimizing the error and maintaining the motor's speed at the setpoint. The system enables bidirectional motor control and is suitable for applications requiring precise motor control.

Keywords. PID, H-bridge, ATmega328P, DC motor, PWM. L298N

Résumé Cet abstract décrit la mise en œuvre d'un système de contrôle pour un moteur à courant continu utilisant un contrôleur PID, la modulation de largeur d'impulsion (PWM), un pilote de moteur L298N, un capteur de vitesse et une carte microcontrôleur Arduino Uno. Le système vise à obtenir un contrôle précis et stable de la vitesse du moteur. L'Arduino Uno reçoit les retours de vitesse du capteur, calcule l'algorithme de contrôle PID et génère le signal PWM. Le pilote de moteur L298N amplifie le signal PWM pour entraîner le moteur à courant continu. Le contrôleur PID ajuste continuellement le cycle de travail du PWM en fonction de la différence entre la vitesse souhaitée et la vitesse mesurée, minimisant ainsi l'erreur et maintenant la vitesse du moteur à la consigne. Le système permet un contrôle bidirectionnel du moteur et convient aux applications nécessitant un contrôle précis du moteur.

Mots clés : PID, moteur à courant continu, PWM., moteur L298N

الملخص : المتحكم الصغير.يهدف النظام الي تحقيق تحكم دقيق و تنفيذ نظام تحكم لمحرك تيار مستمر باستخدام وحدة ردود الفعل حول السرعة من الجهاز الاستشعاري, ويحسب Arduino Uno ومستقر في سرعة المحرك. يستقبل الجهاز لتشغيل المحركPWM علي تضخيم اشارة L298N يعمل المحرك بالقيادة PWM , ويولد اشارة PID خوارزمية المتحكم باستمرار بناء علي الفريق بين السرعة المطلوبة والمقاسة , PWM بتعديل دورة عمل PID بالطريقة المطلوبة .يقوم المتحرك مما يقلل من الخطاء ويحافظ علي سرعة المتحرك عند القيمة المحددة . يتيح النظام التحكم ثنائي الاتجاه في المحرك وهوا مناسب للتطبيقات التي تتطلب تحكما دقيقا في المحرك

الكلمات المفتاحية : اشارة التعديل عرض النبضة ,PWM المتحكم التناسبي التكاملي التفاضلي , PID محرك التيار المستمر

GENERAL INTRODUCTION

In the past, mechanical systems were used for speed control of DC drives, requiring large hardware. However, with the development of semiconductor technology, smaller and faster microprocessors have become available at reduced costs. This has led to the popularity of PID controllers, which offer a wide range of operating conditions and functional simplicity. Motor speed controllers are used to drive a motor at the desired speed, and speed control differs from speed regulation, which accounts for natural speed changes due to varying load on the motor shaft**.** **[**1**]**

Description of research work

The primary objective of this project is to develop an efficient and straightforward method for controlling the speed of a DC motor using pulse width modulation (PWM) technique and a PID controller. The project utilizes a pulse width generator within the ATmega 328p Microcontroller to achieve the modulation of pulse width.

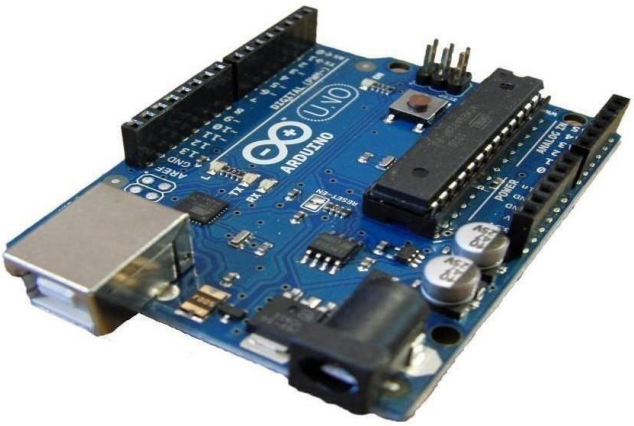
The main purpose of employing PWM is to enable precise control over the power supplied to electrical devices, with a particular focus on inertial loads like DC motors. By utilizing a PID controller in conjunction with PWM, the project aims to regulate the speed of the DC motor accurately.

In summary, this project seeks to provide a concise and effective solution for speed control of DC motors by utilizing the PWM technique and implementing a PID controller. The utilization of pulse width modulation and the integration of a PID controller contribute to the efficient management of power supplied to electrical devices, specifically for inertial loads such as DC motors.
In this project, the Proportional Integral Differential (PID) controller is utilized for designing and selecting appropriate control parameters based on the system's response. To implement the controller in hardware, Infrared (IR) sensors and Arduino Nano are employed to measure the Rotations per Minute (RPM) of the DC motor, and the measured RPM is displayed on an LCD. [2]

For controlling the motor's direction we use a motor driver, an H-bridge is incorporated into the application, enabling the motor to run both forwards and backwards. The H-bridge is driven by a high-frequency Pulse Width Modulation (PWM) signal. By controlling the duty cycle of the PWM signal, the motor's terminal voltage is effectively regulated, thus directly adjusting the motor's speed.

To summarize, this project utilizes a PID controller with carefully selected control parameters for controlling the DC motor's speed. The hardware implementation involves the use of IR sensors and an Arduino Nano to measure RPM, with the results displayed on an LCD. The H-bridge, driven by a high-frequency PWM signal, allows for bi-directional motor control by adjusting the duty cycle, thereby regulating the motor's speed.

Chapter 1
Arduino uno



Introduction

This article gives detailed information about an Arduino uno board, one kind of microcontroller board based on ATmega328, and Uno is an Italian term which means one.Arduino Uno is named for marking the upcoming release of microcontroller board namely Arduino Uno Board 1.0. This board includes digital I/O pins-14, a power jack, analog i/ps-6, ceramic resonatorA16 MHz, a USB connection, an RST button, and an ICSP header. All these can support the microcontroller for further operation by connecting this board to the computer. The power supply of this board can be done with the help of an AC to DC adapter, a USB cable, otherwise a battery. [3]

1 An Overview of Arduino uno Board

1.1 What is Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing [4]. The Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board , you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program [5]. As Arduino is open source, there are a large number of compatible Arduino boards, just as there are many official Arduino boards with special functions as shown below.[6] . Arduino Uno is the most popular

Figure 1-Types of Arduino Boards

1.2 Introduction to Arduino uno

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. [7]

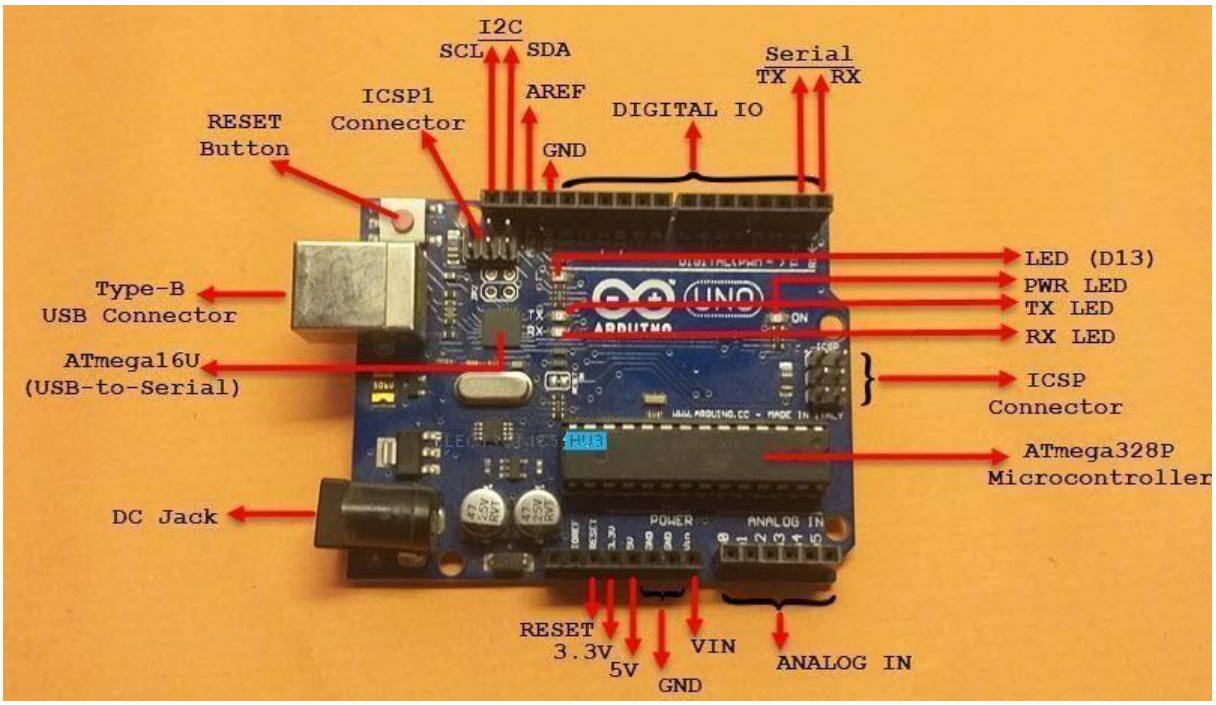


Figure 2- Layout of Arduino uno Board

1.2.1 Arduino uno Pinout

Arduino Uno is based on the ATmega328 by Atmel. The Arduino Uno pinout consists of 14 digital pins, 6 analog inputs, a power jack, USB connection and ICSP header. The versatilit y of the pinout provides many different options such as driving motors, LEDs, reading sensors and more. In this post, we'll go over the capabilities of the Arduino Uno pinout. [8]

1.2.2 Arduino Uno Pinout – Diagram

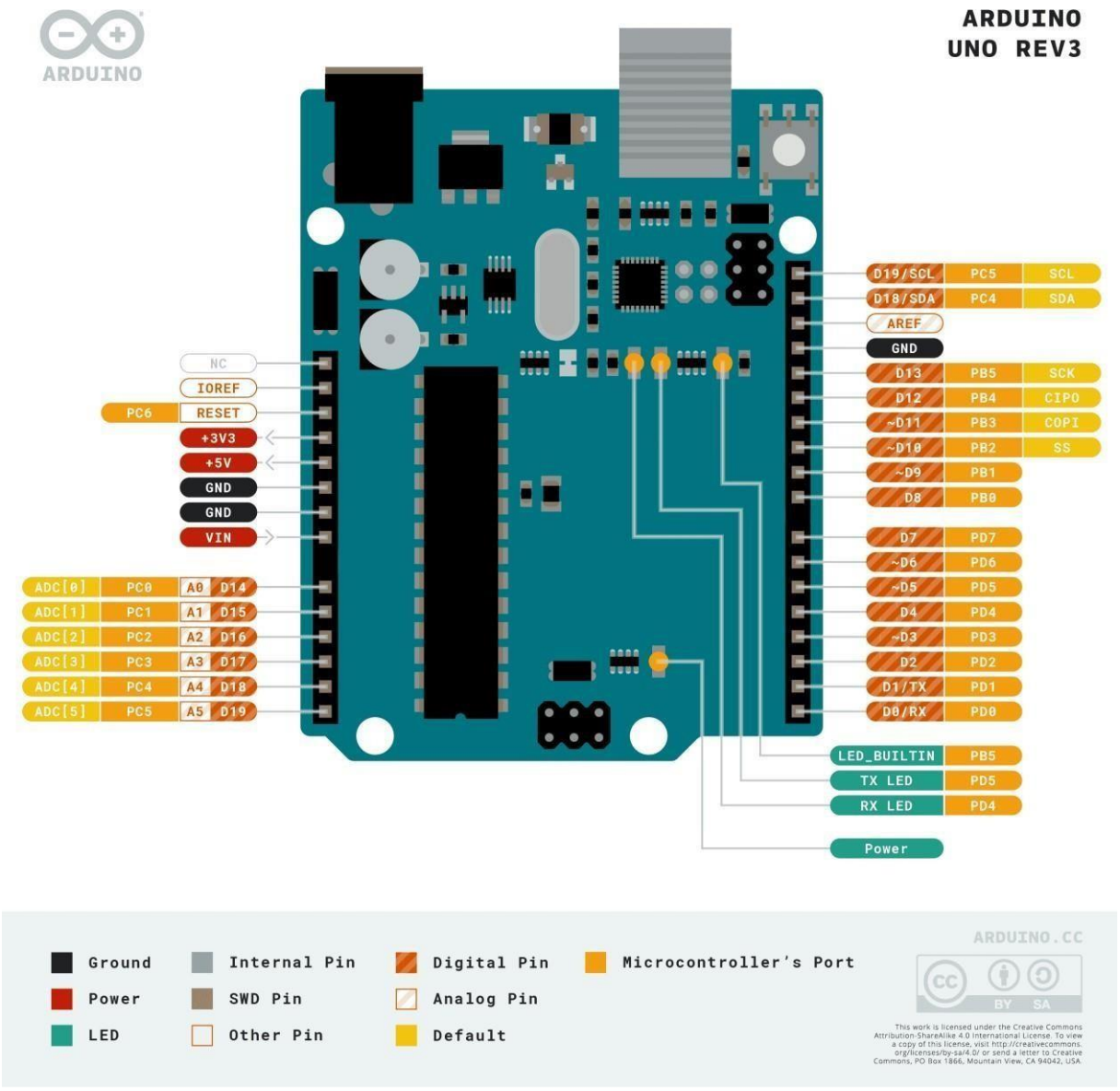


Figure 3 Arduino Uno Pinout – Diagram

1.2.3 Arduino Uno pinout - Power Supply

There are 3 ways to power the Arduino Uno:

Barrel Jack - The Barrel jack, or DC Power Jack can be used to power your Arduino board. The barrel jack is usually connected to a wall adapter. The board can be powered by 5-20 volts but the manufacturer recommends to keep it between 7-12 volts. Above 12 volts, the regulators might overheat, and below 7 volts, might not suffice.

VIN Pin - This pin is used to power the Arduino Uno board using an external power source. The voltage should be within the range mentioned above.

USB cable - when connected to the computer, provides 5 volts at 500mA.

There is a polarity protection diode connecting between the positive of the barrel jack to the VIN pin, rated at 1 Ampere.

The power source you use determines the power you have available for your circuit. For instance, powering the circuit using the USB limits you to 500mA. Take into consideration that this is also used for powering the MCU, its peripherals, the on-board regulators, and the components connected to it. When powering your circuit through the barrel jack or VIN, the maximum capacity available is determined by the 5 and 3.3 volts regulators on-board the Arduino.

5v and 3v3

They provide regulated 5 and 3.3v to power external components according to manufacturer specifications.

GND

In the Arduino Uno pinout, you can find 5 GND pins, which are all interconnected.

The GND pins are used to close the electrical circuit and provide a common logic reference level throughout your circuit. Always make sure that all GNDs (of the Arduino, peripherals and components) are connected to one another and have a common ground.

RESET - resets the Arduino

IOREF - This pin is the input/output reference. It provides the voltage reference with which the microcontroller operates.

1.2.4 Arduino Uno Pinout - Analog IN

The Arduino Uno has 6 analog pins, which utilize ADC (Analog to Digital converter).

These pins serve as analog inputs but can also function as digital inputs or digital outputs.

Analog to Digital Conversion

ADC stands for Analog to Digital Converter. ADC is an electronic circuit used to convert analog signals into digital signals. This digital representation of analog signals allows the processor (which is a digital device) to measure the analog signal and use it through its operation.

Arduino Pins A0-A5 are capable of reading analog voltages. On Arduino the ADC has 10-bit resolution, meaning it can represent analog voltage by 1,024 digital levels. The ADC converts voltage into bits which the microprocessor can understand.

One common example of an ADC is Voice over IP (VoIP). Every smartphone has a microphone that converts sound waves (voice) into analog voltage. This goes through the

device's ADC, gets converted into digital data, which is transmitted to the receiving side over the internet.

1.2.5 Arduino Uno Pinout - Digital Pins

Pins 0-13 of the Arduino Uno serve as digital input/output pins.

Pin 13 of the Arduino Uno is connected to the built-in LED.

In the Arduino Uno - pins 3,5,6,9,10,11 have PWM capability.

It's important to note that:

Each pin can provide/sink up to 40 mA max. But the recommended current is 20 mA.

The absolute max current provided (or sank) from all pins together is 200mA

1.2.6 What does digital mean?

Digital is a way of representing voltage in 1 bit: either 0 or 1. Digital pins on the Arduino are pins designed to be configured as inputs or outputs according to the needs of the user. Digital pins are either on or off. When ON they are in a HIGH voltage state of 5V and when OFF they are in a LOW voltage state of 0V.

On the Arduino, When the digital pins are configured as **output**, they are set to 0 or 5 volts.

When the digital pins are configured as **input,** the voltage is supplied from an external device. This voltage can vary between 0-5 volts which is converted into digital representation (0 or 1). To determine this, there are 2 thresholds:

Below 0.8v - considered as 0.

Above 2v - considered as 1.

When connecting a component to a digital pin, make sure that the logic levels match. If the voltage is in between the thresholds, the returning value will be undefined. [8]

1.3 Arduino uno specifications

- Microcontroller: ATmega328P
- Operating Voltage: 5 volts
- Digital I/O Pins: 14 (6 with PWM)
- Analog Input Pins: 6
- Flash Memory: 32 KB / SRAM: 2 KB /EEPROM: 1 KB

2 Breadboard

A Breadboard is a helpful tool to build circuits without any soldering. Certain contacts are connected with each other. Therefore it is possible to connect many cables with each other without soldering or screwing them together. The image below shows in color, which contacts are connected.

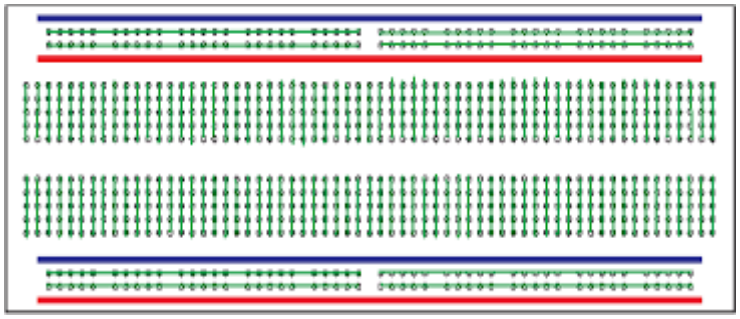


Figure 4-Breadboard Internal Connections

2.1 How to Use a Breadboard to Make a Circuit?

To use a breadboard for your circuit, follow the circuit diagram and connect one component in a line. Always connect the battery at the last after double-check all the connections. Keep an eye out of common mistakes like mixing ground and supply, connect in a wrong rail, ICs not set properly, etc. In the below diagram, we made a circuit for Glowing LED on a breadboard [9].

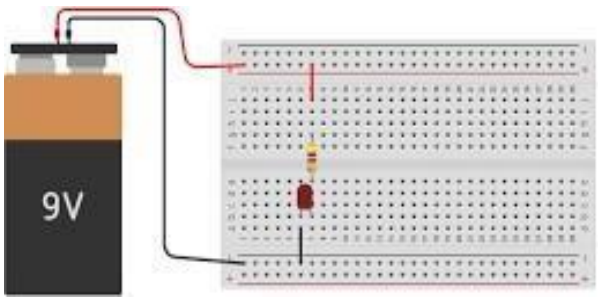


Figure 5- Simple LED Circuit on Breadboard

2.2 Arduino uno Programming and Communication

Arduino has built-in support for UART which enable serial communication. UART as a serial protocol is most useful and famous protocol. The Arduino can transmit and receive data to the PC over USB Cable. This comes handy when we want to send the sensor data from microcontroller to PC. The Arduino IDE has built-in Serial Monitor window, which displays the data sent from Arduino to PC. The same way we can send data/command from Serial Monitor to Arduino. The serial communication enables us to control electronic

devices connected to Arduino board from PC. When comes to interfacing more complicated devices such as LCD, RTC, EEPROM etc. We can use serial communication to debug the code and track errors to interface those devices. [10]

2.3 Arduino Coding Environment and basic tools

2.3.1 What language is Arduino?

Arduino code is written in C++ with an addition of special methods and functions, which we‟ll mention later on. C++ is a human-readable programming language. When you create a „sketch‟ (the name given to Arduino code files), it is processed and compiled to machine language.

2.4 Arduino IDE

The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where you‟ll be typing up your code before uploading it to the board you want to program. Arduino code is referred to as sketches [11]

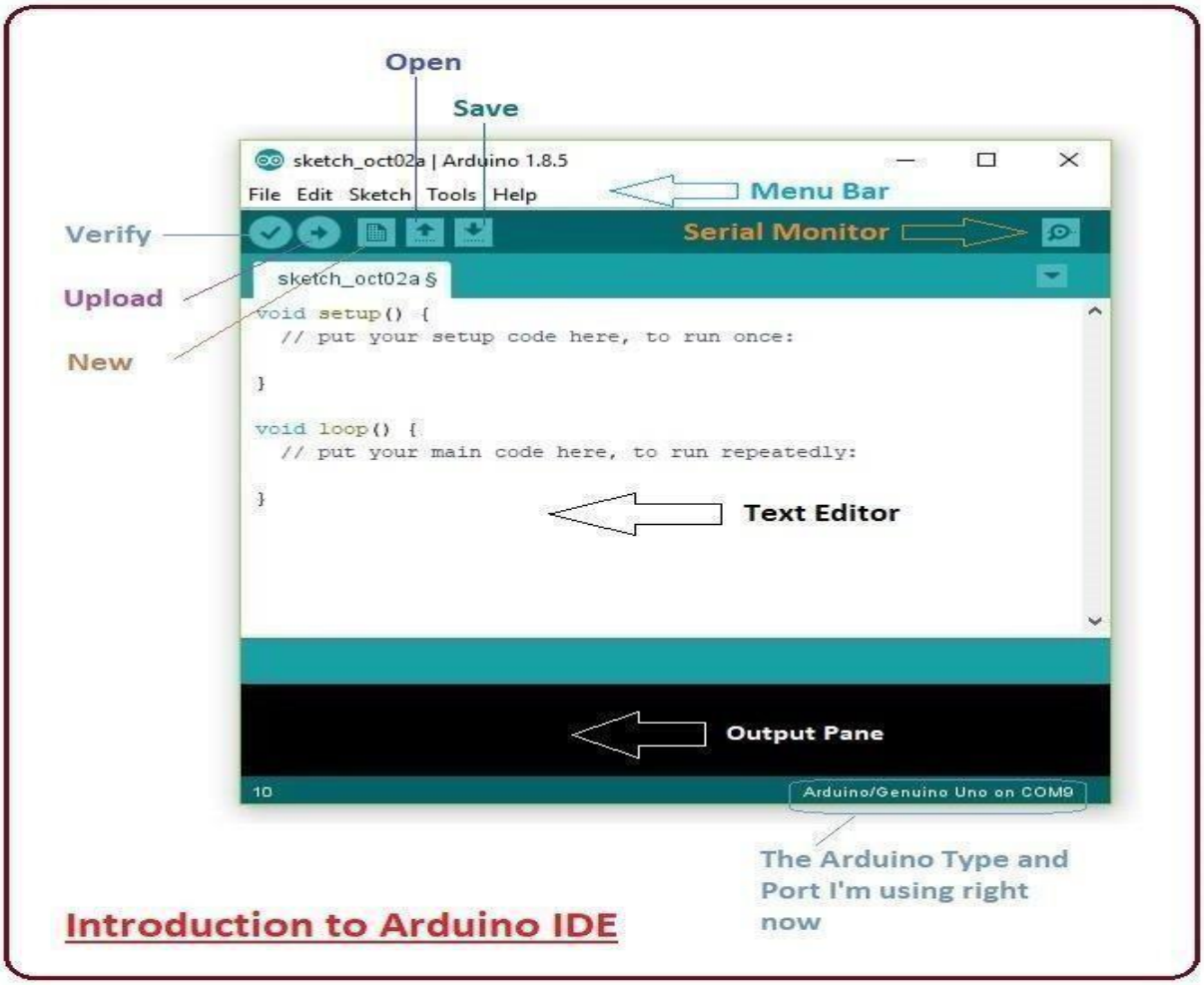


Figure 6- Arduino IDE

- Menu Bar: Gives you accessto the tools needed for creating and saving Arduino sketches. VerifyButton: Compiles your code and checks for errors in spelling or syntax.

- Upload Button: Sends the code to the board that‟s connected such as Arduino Nano in this

- case. Lights on the board will blink rapidly when uploading. New Sketch: Opens up a new window containing a blank sketch.

- Sketch Name: When the sketch is saved, the name ofthe sketch is displayed here

.- Open Existing Sketch: Allows you to open a saved sketchor one fromthe stored examples.

- Save Sketch: This saves the sketch you currentlyhave open

.- Serial Monitor: When the board is connected, this will display the serial information of your

- Arduino Code Area: This area is where you compose the code of the sketch that tells the board what to

- do. Message Area: This area tells you the status on saving, code compiling, errors and more

.- Text Console: Shows the details of an error messages, size of the program that was compiled

- and additional info. Board and Serial Port: Tells you what board is being used and what serial port it‟s connected to [12].

2.5 The Programing Structure of an Arduino Sketch

The structure of an Arduino sketch consist of 3 main sections as explained below: Declaration Section: This section used for declare variables, constants, Pin Numbers and to include Libraries (C++ header files). Setup Section: In this section, the setup () function is used to initialize serials, configure Pin mode, Pin Numbers, using Libraries …etc. This function will only run once after each power up or reset of the Arduino board [13]. Loop Section: The main program of the sketch will runs in this section. The Loop () Function will run infinitely

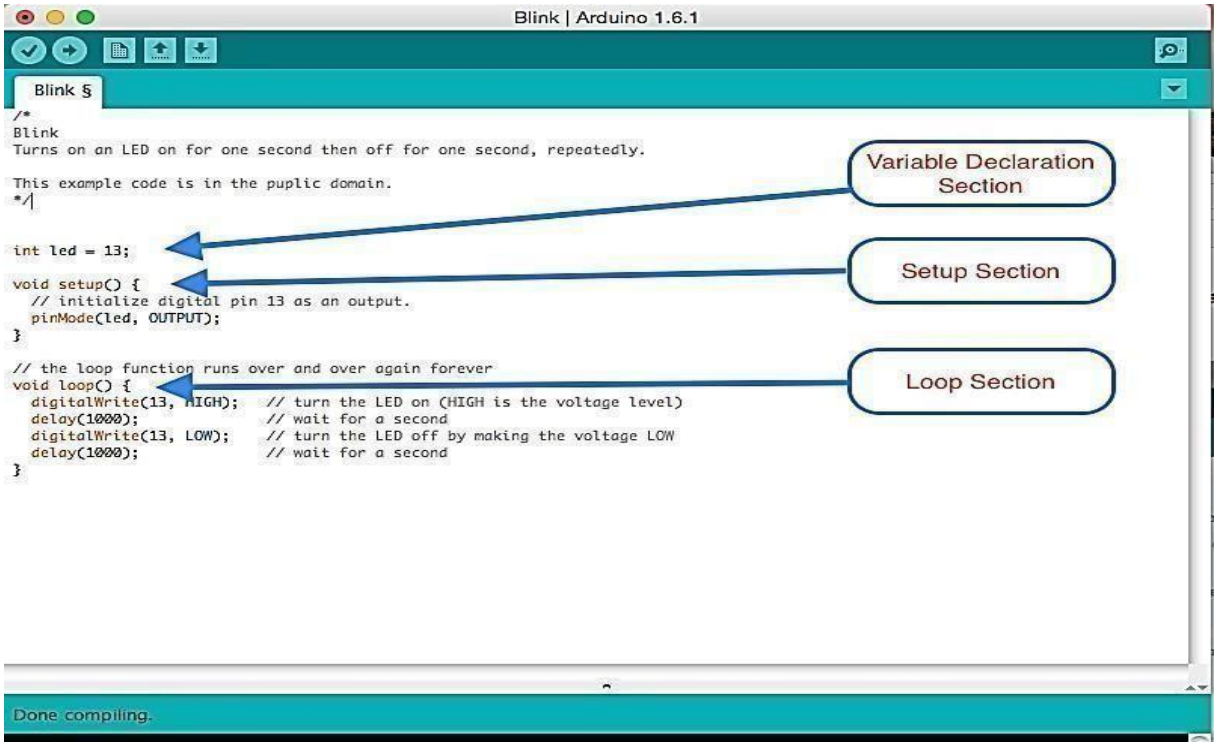


Figure 7-Arduino Program structure.

2.5.1 Choose the Correct Arduino Board

If this is the first time you"ve used Arduino IDE or the first time you"ve built code for a Nano, we need to set the Arduino environment so it knows to build the code for the specific board Just go to the Tools menu at the top of the IDE and then slide down to the [Board >] menu item , when you slide over that item a menu will pop out with a large number of choices. Slide down the Arduino Nano and click it to select it [14].

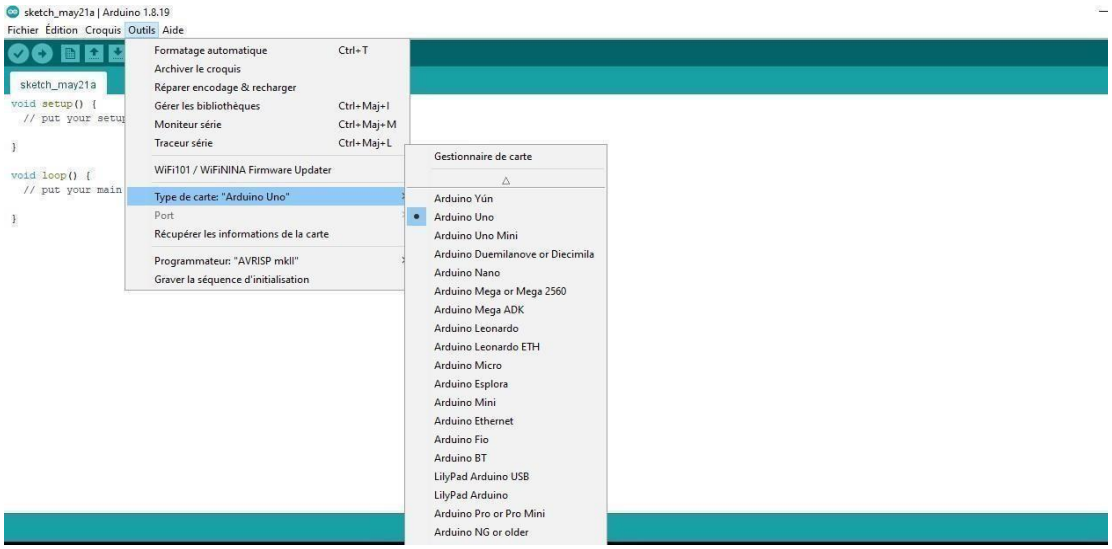


Figure 8-Choose the Correct Arduino Board

The IDE should automatically select the "Processor: ATMega328P" for that board, because it is the newer processor used on these boards. However, you want to make sure it is selected properly. The next image shows you howto select the proper processor. These two options insure that the IDE will build the code that is correct for your board. Notice that the bottom status area of the IDE (shown in previous image) confirms that you are building for the Arduino uno with an ATMega328 and the device is connected to COM1

3 LED Blinking Using ArduinoUNO

3.1 Components required

- 1 × Breadboard
- 1 × Arduino uno
- 1 × LED
- 1 × 220Ω Resistor

3.1.1 Connections

Connect the LED to the Arduino Uno:

Connect the longer leg (anode) of the LED to digital pin 13 on the Arduino Uno.

Connect the shorter leg (cathode) of the LED to a current-limiting resistor (around 220-470 ohms).

Connect the other end of the resistor to the ground (GND) pin on the Arduino Uno.

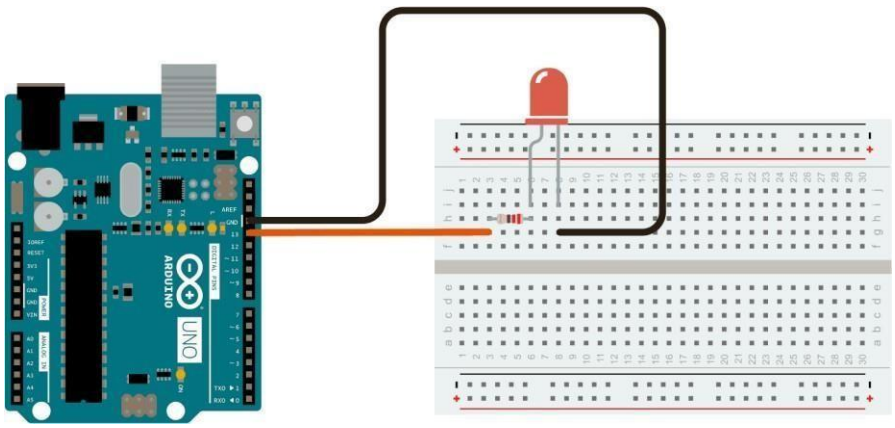


Figure 9- LED Blinking Using Arduino uno

3.1.2 Arduino code

Tableau 1-Arduino code for LED blinking with 1s interval

```
// Pin connected to the
 LEDconst int ledPin = 13;
// Setup function runs
 oncevoid setup()
{ // Initialize the digital pin
 pinMode(ledPin, OUTPUT);
 }
 // Loop function runs repeatedly
| | void loop()
 { | | // Turn the LED on
{ | | // Turn the LED on
 | | digitalWrite(ledPin, HIGH);
 | | delay(1000);
// Wait for 1 sec | |
 | // Turn the LED off
 | | digitalWrite(ledPin, LOW);
 | | delay(1000); // Wait for 1 sec
```

3.2 Result

After uploading the Arduino Sketch, you should see your LED being turn on and off at every 1 second. If the required output is not seen, make sure you have assembled the circuit correctly, andverified and uploaded the code to your board.

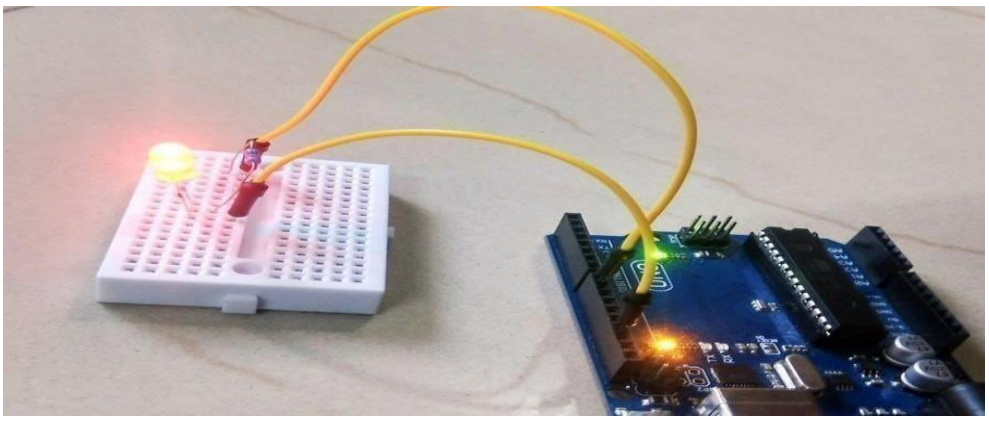


Figure 10-LED Blinking

4 Difference between Arduino UNO and Arduino Nano

The main difference between these two is the size. Because Arduino Uno size is double to Nano board. So Uno boards use more space on the system. The programming of UNO can be done with a USB cable where as Nano uses the mini USB cable. The main differences are listed inthe following table [15].

Projectiotl23.com

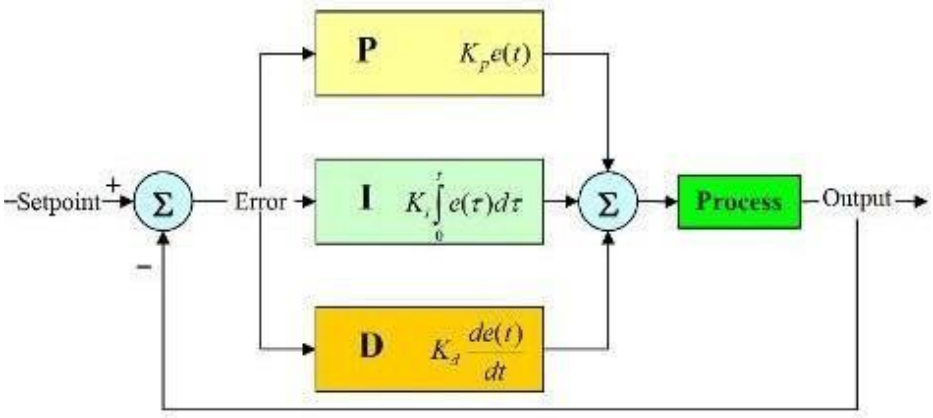
Name	Arduino Nano	Arduino UNO
MCU	Atmega328p/Atmega 168.	Atmega328p
Power	5V	5V
Input Voltage	7 -12 V	7 – 12 V
Maximum Current Rating	40mA	40mA
Clock Frequency	16MHz	16MHz
Flash Memory	16KB/32KB	32KB
USB	Mini CHANGE	Standard
USART	Yes	Yes
SRAM	1KB/2KB	2KB
PWM	6 out of 14 digital pins	6 out of 14 digital pins
GPIO	14	14
Analog Pins	8 CHANGE	6
EEPROM	512bytes/1KB CHANGE	1KB

Figure 11- Difference between Arduino UNO and Arduino Nano

Conclusion

The best thing about Arduino boards is they can work as a stand-alone project or as a part of other electronic projects. You can interface Arduino Nano with other Arduino boards and Raspberry Pi boards. No technical expertise is required to use Arduino boards and anyone with little to no technical knowledge can make amazing projects with these units

Chapter 2 Types of control



5 PID control

The term PID stands for proportional integral derivative and it is one kind of device used to control different process variables like pressure, flow, temperature, and speed in industrial applications. In this controller, a control loop feedback device is used to regulate all the process variables.

This type of control is used to drive a system in the direction of an objective location otherwise level. It is almost everywhere for temperature control and used in scientific processes, automation & myriad chemical. In this controller, closed-loop feedback is used to maintain the real output from a method like close to the objective otherwise output at the fixe point if possible. In this article, the PID controller design with control modes used in them like P, I & D are discussed. [16]

5.1 PID Controller Block Diagram

A closed-loop system like a PID controller includes a feedback control system. This system evaluates the feedback variable using a fixed point to generate an error signal. Based on that,it alters the system output. This procedure will continue till the error reaches Zero otherwise the value of the feedback variable becomes equivalent to a fixed point.

This controller provides good results as compared with the ON/OFF type controller. In the ON/OFF type controller, simply two conditions are obtainable to manage the system. Once the process value is lower than the fixed point, then it will turn ON. Similarly, it will turn OFF once the value is higher than a fixed value. The output is not stable in this kind of controller and it will swing frequently in the region of the fixed point. However, this controller is more steady & accurate as compared to the ON/OFF type controller. [16]

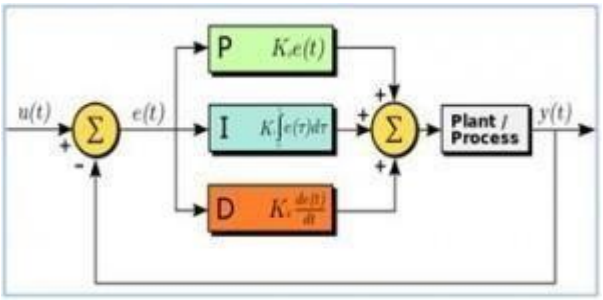


Figure 12-Working of PID controller

Parameter	Rise-time	Overshoot	Settling time	Steady State Error	Stability
K_p	Decrease	Increase	Small Change	Decrease	Decrease
K_i	Decrease	Increase	Increase	Eliminate	Decrease
K_d	Minor changes	Decrease	Decrease	No effect	Improve if Kd is small

Figure 13-PID Controller parameter characteristics

The mathematical form of a PID controller is:

$$U(t) = k\rho^{e} + k_i \cdot \int_0^t e\,dt_{+kd}\frac{d}{dt}e$$

Kp → Proportional Gain

Ki → integral Gain

Kd →derivative Gain

A PID controller is a closed-loop control mechanism that uses the error signal between the desired setpoint and the actual output to calculate a control action. The controller adjusts the process input based on the proportional, integral, and derivative terms of the error signal. By tuning the PID gains, the controller aims to minimize the tracking error and bring the system output closer to the desired setpoint. [17]

The alternative to a closed loop control scheme such as the PID controller is an open loop controller (no feedback) that is in many cases not satisfactory, and is often impossible due to the system properties. By adding feedback from the system output, performance can be improved In Fig.14 a schematic of a system with a PID controller is shown.

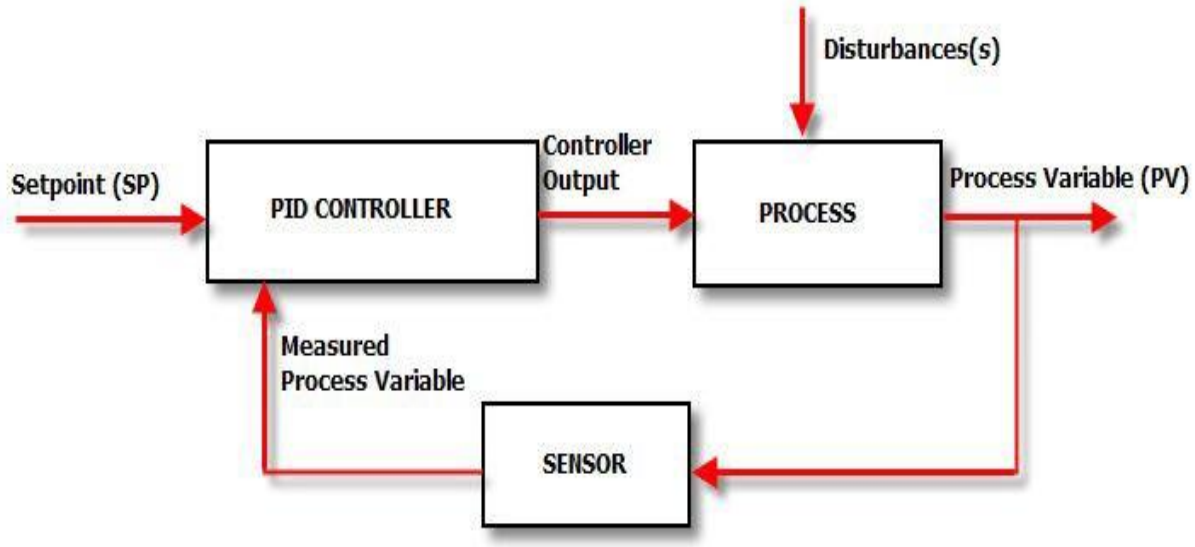


Figure 14-Closed Loop System with PID Controller

A PID controller reads the system state, calculates the error by comparing it with a desired reference, and manages the error in three ways: handling the present through the proportional term, recovering from the past using the integral term, and anticipating the future through the derivative term**.**

6 Types of PID Controller

PID controllers are classified into three types like ON/OFF, proportional, and standard type controllers. These controllers are used based on the control system, the user can be used the controller to regulate the method. [16]

6.1 ON/OFF Control

An on-off control method is the simplest type of device used for temperature control. The device output may be ON/OFF through no center state. This controller will turn ON the output simply once the temperature crosses the fixed point. A limit controller is one particular kind of ON/OFF controller that uses a latching relay. This relay is reset manually and used to turn off a method once a certain temperature is attained.

6.2 Proportional Control

This kind of controller is designed to remove the cycling which is connected through ON/OFF control. This PID controller will reduce the normal power which is supplied toward the heateronce the temperature reaches the fixed point.

This controller has one feature to control the heater so that it will not exceed the fixed point however it will reach the fixed point to maintain a steady temperature. This proportioning act can be achieved through switching ON & OFF the output for small time periods. This time proportioning will change the ratio from ON time to OFF time for controlling the temperature.

6.3 Standard Type PID Controller

This kind of PID controller will merge proportional control through integral & derivative control to automatically assist the unit to compensate modifications within the system. These modifications, integral & derivative are expressed in time-based units.

These controllers are also referred through their reciprocals, RATE & RESET correspondingly. The terms of PID must be adjusted separately otherwise tuned to a specific system with the trial as well as error. These controllers will offer the most precise and steady control of the 3 types of controller.

6.4 Real-Time PID Controllers

At present, there are various kinds of PID controllers are available in the market. These controllers are used for industrial control requirements like pressure, temperature, level, and flow. Once these parameters are controlled through PID, choices comprise utilize a separate PID controller or either PLC. These separate controllers are employed wherever one otherwise two loops are required to be checked as well as controlled otherwise in the conditions wherever it is complex to the right of entry through larger systems.

These control devices provide different choices for solo & twin loop control. The standalone type PID controllers provide several fixed-point configurations to produce the autonomous several alarms.
These standalone controllers mainly comprise PID controllers from Honeywell, temperature controllers from Yokogawa, autotune controllers from OMEGA, Siemens, and ABB controllers. [16]

PLCs are used like PID controllers in most of the industrial control applications The arrangement of PID blocks can be done within PACs or PLCs to give superior choices for an exact PLC control. These controllers are smarter as well as powerful as compared with separate controllers. Each PLC includes the PID block within the software programming.

7 Manual tuning

PID controller tuning is a crucial step to achieve optimal control performance. Various methods are available to tune PID controllers, ensuring they are well-matched with the dynamics of the controlled process. Some common tuning methods include:

1. Trial and Error Method: This straightforward approach involves incrementally adjusting the proportional (Kp), integral (Ki), and derivative (Kd) gains while observing the system's response. The process involves iteratively increasing the Kp to induce oscillations, then adjusting Ki to eliminate oscillations and Kd to enhance response speed.

2. Ziegler-Nichols Method: This method utilizes the system's step response to determine the PID gains. It involves applying a step input and analyzing the resulting response curve to obtain parameters such as the ultimate gain (Ku) and oscillation period (Pu). These values are then used to calculate the appropriate Kp, Ki, and Kd values based on pre-defined tuning rules.

3. Process Reaction Curve Technique: This open-loop method involves applying a manual control output to the system and recording the response curve. By analyzing the curve's characteristics, such as slope, dead time, and rise time, suitable values for Kp, Ki, and Kd can be derived using mathematical equations or empirical rules.

It is essential to note that tuning a PID controller often requires expertise and an understanding of the process dynamics. Different methods may be suitable for different scenarios, and fine-tuning adjustments may be necessary to achieve the desired control performance.

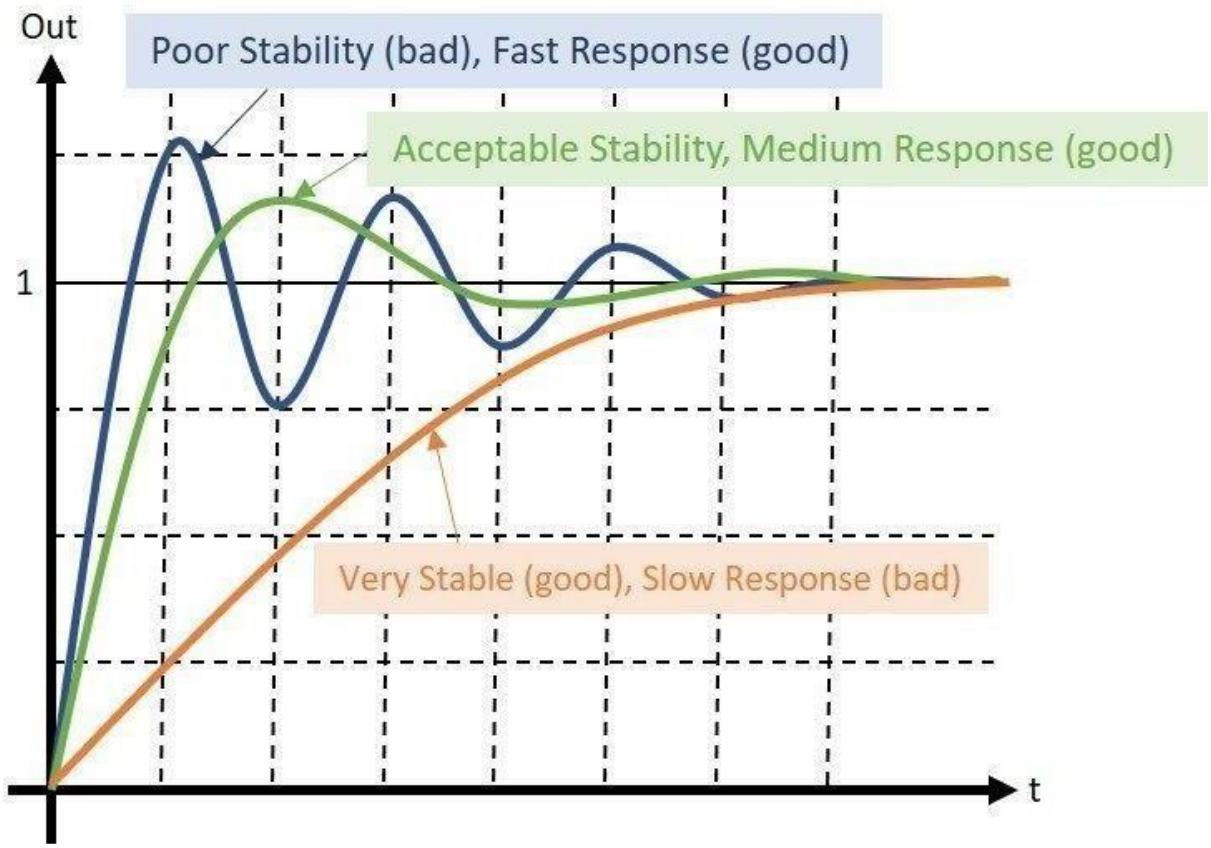


Figure 15-Behavior and comments in the response time of the PID control system

7.1 Manual tuning procedure

1. Start by setting a low value for the proportional gain (P), typically around 1.
2. Gradually increase the value of P until you observe oscillations in the system's response.
3. The amplitude of these oscillations reflects the error in your system, providing a measure of the controller's performance.
4. Set the value of P to approximately half of the oscillation-inducing value determined in the previous step.
5. Incrementally increase the integral gain (I) by a small amount and repeat step 2. Continue this process until further improvements cannot be found.
6. Similarly, increase the derivative gain (D) by a small amount and repeat step 2. Keep iterating until no more improvements are observed and the system adequately responds to reference changes after a load disturbance.

Conclusion

This paper presents the results of our analysis of the performance of PID controller parameters. Our findings suggest that the design of a PID controller for any given system involves several steps that are essential to achieve the desired response. To improve the rise time, it is necessary to incorporate a proportional gain (Kp), while an overshoot can be reduced by including a derivative gain (Kd). To eliminate steady-state error, an internal gain (Ki) must also be included. These parameters (Kp, Ki, and Kd) must be carefully analyzed and optimized until the desired overall response is achieved. It is important to note that all three controllers need not be implemented into a single systemunless necessary

8 PWM theory

Pulse width modulation reduces the average power delivered by an electrical signal by converting the signal into discrete parts. In the PWM technique, the signal's energy is distributed through a series of pulses rather than a continuously varying (analogue) signal. [18]

A pulse width modulating signal is generated using a comparator. The modulating signal forms one part of the input to the comparator, while the non-sinusoidal wave or sawtooth wave forms the other part of the input. The comparator compares two signals and generates a PWM signal as its output waveform.

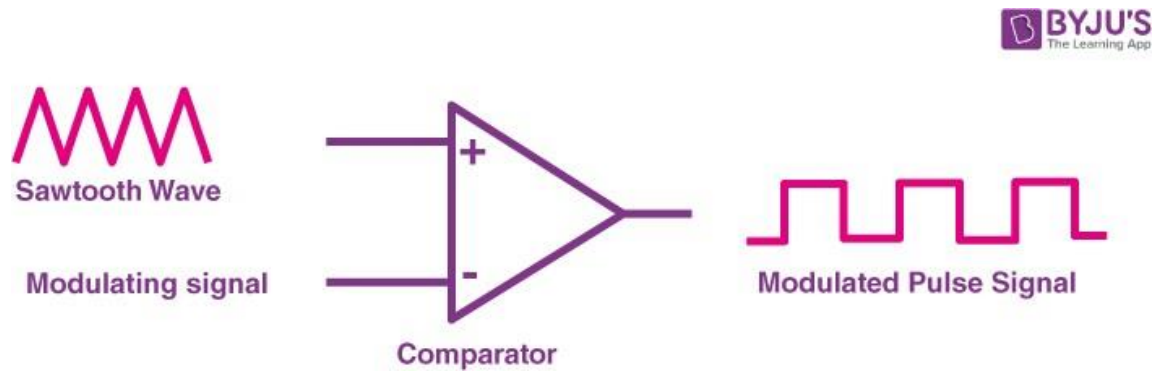


Figure 16-pulse-width-modulation

If the sawtooth signal is more than the modulating signal, then the output signal is in a "High" state. The value of the magnitude determines the comparator output which defines the width of the pulse generated at the output.

8.1 Duty Cycle of PWM

a PWM signal stays "*ON*" for a given time and stays "*OFF*" for a certain time. The percentage of time for which the signal remains "*ON*" is known as the duty cycle. If the signal is always "*ON*," then the signal must have a 100 % duty cycle. The formula to calculate the duty cycle is given as follows:

$$DUTY\ cycle = \frac{\text{turn on time}}{turn\ on\ time + turn\ off\ time}$$

The average value of the voltage depends on the duty cycle. As a result, the average value can be varied by controlling the width of the "ON" of a pulse.

8.2 Frequency of PWM

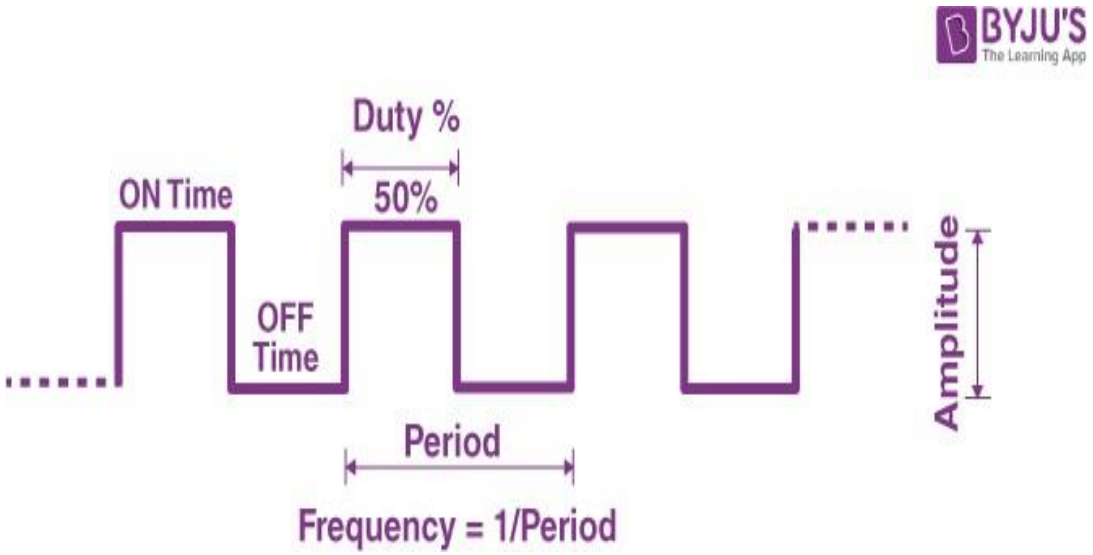


Figure 17-duty-cycle-of-pulse-width-modulation

The frequency of PWM determines how fast a PWM completes a period. The frequency of a pulse is shown in the figure above.

The frequency of PWM can be calculated as follows:

$$frequency = \frac{1}{time\ period}$$

$$\text{Time Period} = \text{on time} + \text{off time}$$

8.3 Output Voltage of PWM signal

The output voltage of the PWM signal will be the percentage of the duty cycle. For example, for a 100% duty cycle, if the operating voltage is 5 V then the output voltage will also be 5 V.If the duty cycle is 50%, then the output voltage will be 2.5 V.

9 Types of Pulse Width Modulation Technique

There are three conventional types of pulse width modulation technique and they are namedas follows:

- Trail Edge Modulation – In this technique, the signal's lead edge is modulated, andthe trailing edge is kept fixed.

- Lead Edge Modulation – In this technique, the signal's lead edge is fixed, and the

trailing edge is modulated.

- **Pulse Center Two Edge Modulation –** In this technique, the pulse centre is fixed and both edges of the pulse are modulated.

Conclusion

Switch mode controllers are widely preferred in various applications such as audio power amplifiers, inverters, and motors due to their advantages in terms of high efficiency and low system design costs. These controllers utilize PWM (Pulse Width Modulation) signals to control the active components in a switching converter. The duration of the switches being turned on or off is determined by the duty cycle of the PWM signals. This duty cycle plays a crucial role in regulating the amount of energy delivered to the load [31]. To ensure that the majority of energy received by the load is influenced mainly by the modulating signal, it is essential to maintain a significantly higher switching frequency compared to the frequency of the modulating signal. By guaranteeing a much higher switching frequency, the switching converter can effectively modulate the energy delivered to the load based on the desired output. [19]

Chapter3 Methodology

Abstract

A DC motor can be controlled using an Arduino board and a motor driver, such as the L293D. The Atmega328P microcontroller on the Arduino can be programmed to output PWM signals that control the motor speed, while the motor driver handles the current and voltage requirements of the motor. By varying the duty cycle of the PWM signal, the speed of the motor can be adjusted. Additionally, the motor direction can be controlled by changing the direction of the current flowing through the motor. This setup allows for precise control of the motor speed and direction, making it ideal for a variety of applications such as robotics, automation, and hobby projects.

10 DC motor

10.1 What is a DC motor

A DC motor or direct current motor is an electrical machine that transforms electrical energy into mechanical energy by creating a magnetic field that is powered by direct current. When a DC motor is powered, a magnetic field is created in its stator. The field attracts and repels magnets on the rotor; this causes the rotor to rotate. To keep the rotor continually rotating, the commutator that is attached to brushes connected to the power source supply current to the motors wire windings.

Figure 18-motor dc

One of the reasons DC motors are preferred over other types of motors is their ability to precision control their speed, which is a necessity for industrial machinery. DC motors are able to immediately start, stop, and reverse—an essential factor for controlling the operationof production equipment [20]

10.2 Different Parts of a DC motor

If you have read about DC motor explanation, you may find a different number of the parts of a DC motor. The most common mentioned parts are rotor, stator, brush, commutator, and armature. [21]

1. Rotor:

Rotor comes from the "rotate" meaning it is the electrical rotating part of a dc motor. Rotor is the moving parts of a dc motor. It dynamically moves when the voltage is applied to the armature winding. This will produce mechanical movement for a dc motor.

Rotor is built from:

- Shaft
- Armature core
- Brush
- Commutator
- Armature windings

2. Stator:

Stator comes from the "stationary" meaning it is the electrical stationary parts of a dc motor. Stator does not move and only produces a magnetic field around the rotor to make the rotor rotating when the voltage is applied to it.

Stator is built from:

- Yoke or frame
- Field windings
- Poles

3. Brush:

Brushes are attached to the commutator as a bridge to deliver the electrical energy from the supply circuit to the rotor. Brushes are usually made from Carbon or Graphite material

4. Commutator:

Commutator has the form of a split ring. The ring is made from copper and split in 2 or more depending on the number of armature windings. The split segment is connected to the armature winding.

5. Field Windings:

The field windings are made from copper wire and circle around the Pole Shoes. Field winding is used to energize the static magnetic field in the stator. We install the field windings around the slot of the Pole Shoes. We do not need field windings if we use permanent magnets like in a Permanent Magnet Motor or PMDC motor.

6. Yoke or Frame:

Yoke is an iron frame as a protective cover for both rotor and stator. This part protects everything inside it, supports the armature, and the house of the magnetic poles, field windings, and the pole to provide magnetic fields for the rotor.

7. Poles:

Poles in the stator are used to energize a specific sequence of magnetic poles to make sure therotor is rotating. It is divided into Pole Core and Pole Shoes.

For a dc motor we need the magnetic fields to make the rotor start rotating. In order to generate magnetic fields, we put field windings around the Pole Shoe which is attached to the Pole Core in the Yoke inner part. These Pole Shoe and Pole Core parts are attached to each other using hydraulic pressure. The structure is the yoke holding the Pole Core which carries the Pole Shoes carrying field windings. This pole unit generates flux spread out into the air gap between rotor and stator. [21]

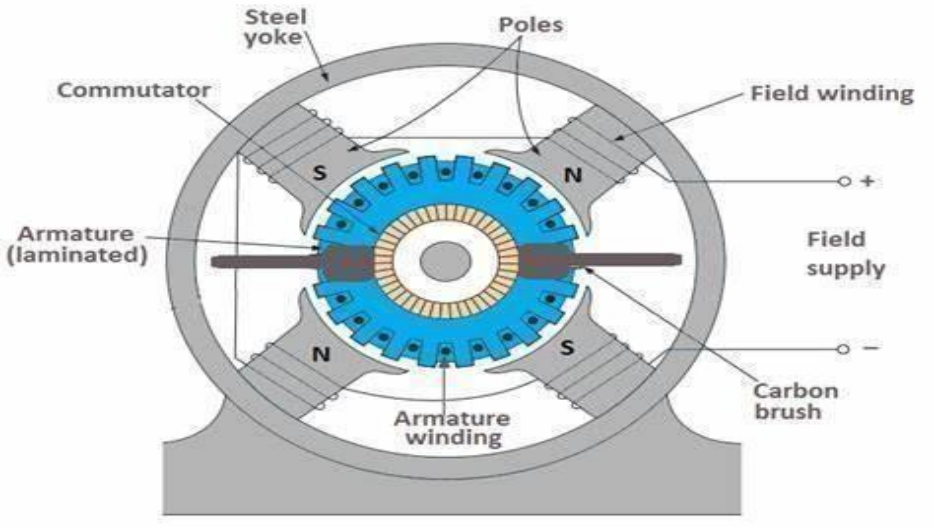


Figure 19-Construction of DC Motor

11 DC Motor Working

A DC motor works on the principle that whenever a current-carrying conductor is placed inside a magnetic field, it experiences a magnetic force whose direction is given by Fleming's Left-hand Rule. In other words, the DC motor spins due to the interaction of the permanent magnet's magnetic field with the magnetic field of the current-carrying electromagnet

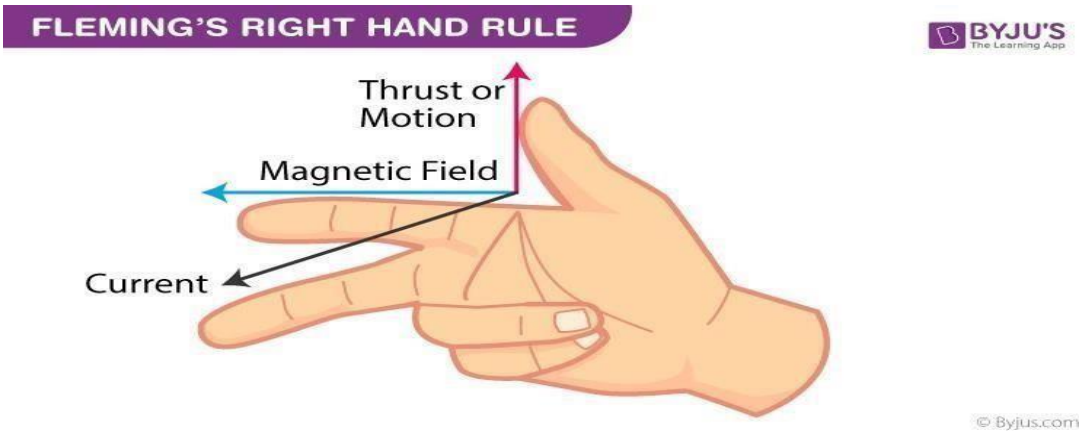


Figure 20- Fleming's left hand rule

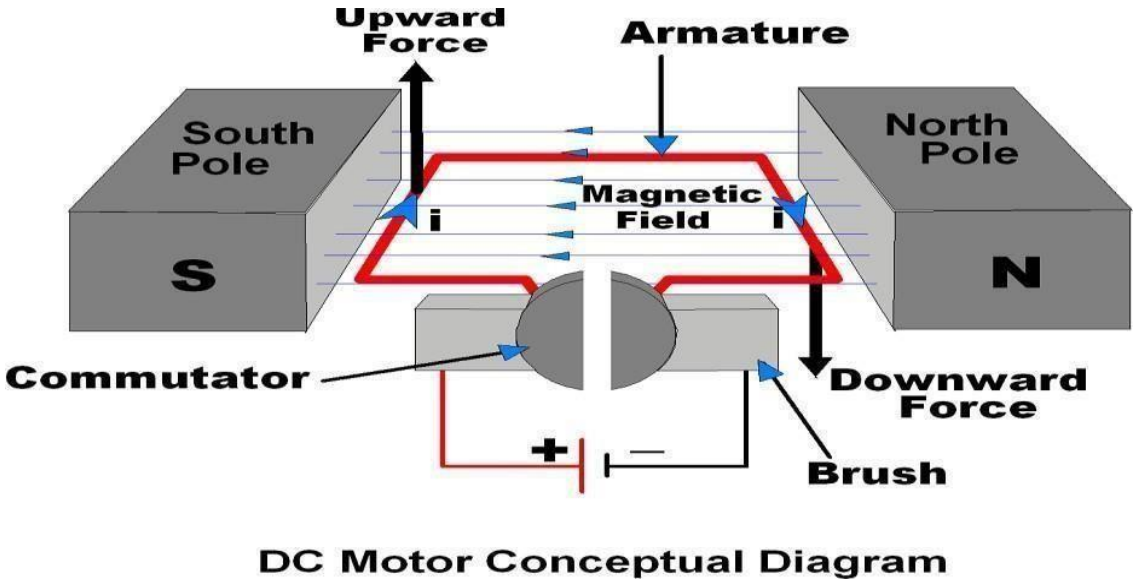


Figure 21- Production of torque in a DC motor

When a current-carrying conductor is placed in an external magnetic field, the conductor experiences a force perpendicular to both the field and the current flow's direction. Fleming's left-hand rule is used to find the direction of the force acting on the current carrying conductorplaced in a magnetic field. [22]

12 What is an L298N Motor Driver

L298N motor driver IC has many applications in the embedded field, especially on the robotics side. Most of the microcontrollers operate on very low voltage (5v) and current while the motors

require higher voltages and current So, the microcontrollers cannot provide them such higher current. For this purpose, we use motor driver ICs.

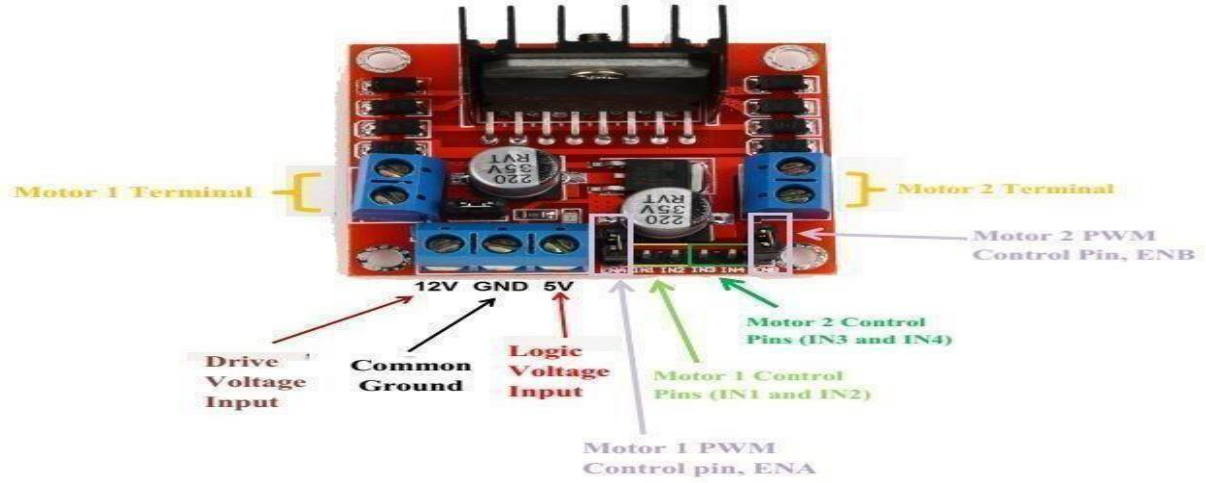


Figure 22- L298N driver

The motor driver is a little current amplifier. It takes a low current signal and gives out a high current signal which can drive a motor. It can also control the direction of the motor. Motor drives are of many kinds depending upon the maximum supply voltage, maximum output current, rated power dissipation, load voltage, and number outputs, etc. Here we are going to discuss motor driver L298N. It is used in dc motor speed control project and you can interface dc motor easy with microcontroller using this motor driver. and also in Bluetooth controlled robot using a pic microcontroller. you can check the line follower robot for more about its applications.

13 Features of the L298N motor driver:

L298N is an integrated circuit multi watt 15 package and capable of giving high voltage. It is a high current dual full-bridge driver that is designed to accept standard TTL logic levels. It can drive inductive loads e.g relays, solenoids, motors (DC and stepping motor), etc.
Its basic features are:

- Maximum supply voltage 46V
- Maximum output DC current 4A
- Low saturation voltage
- Over-temperature protection
- Logical "0" Input Voltage up to 1.5 V . [23]

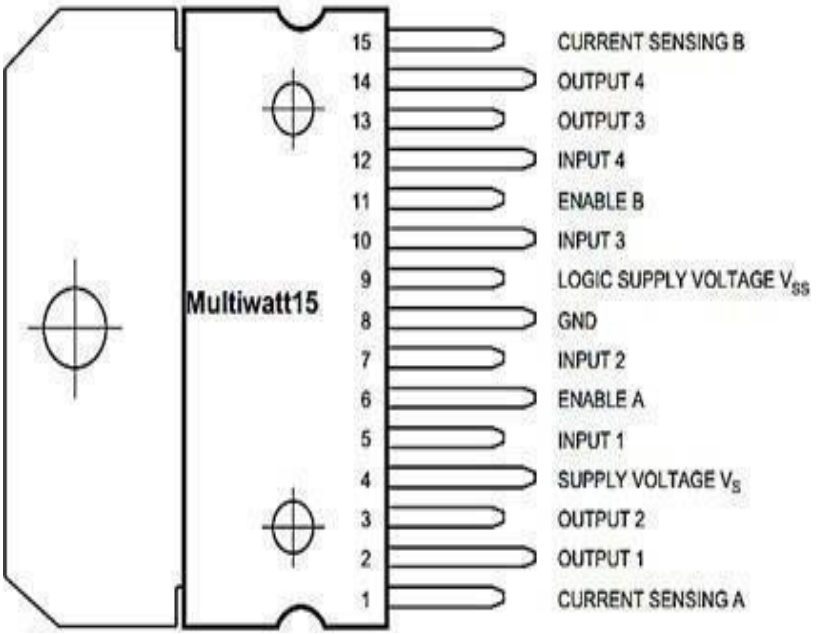


Figure 23- pin diagram of L298N

14 L298N Motor Driver Module pinout

The L298N Motor Driver module consists of an L298 IC Dual H-bridge, 5V Voltage Regulator, resistors, capacitor, Power LED, 5V jumper.

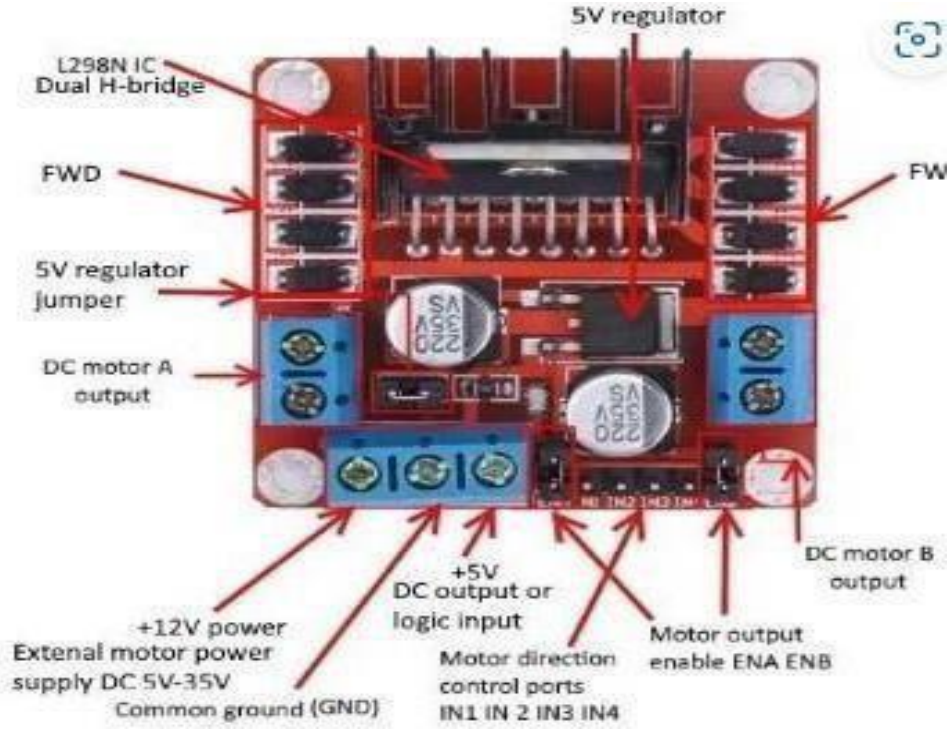


Figure 24-Pinouts of L298N Motor driver Module

DC motor output pins, 12-volt external motor power supply, motor direction control pins (IN1,IN2, IN3, IN4), motor output enable pins (ENA, ENB), and a heat sink.

- **VCC** pin supplies power to the motor. Voltage anywhere between 5 to 35V can be applied. Remember, if the 5V-EN jumper is in place, you need to supply 2 extra volts than the motor's actual voltage requirement, in order to run the motor at its maximum speed.

- **GND** is the common ground pin.

- **5V** pin supplies power to the switching logic circuitry inside the L298N IC. If the 5V-EN jumper is in place, this pin acts as output and can be used to power up the Arduino. If the 5V- EN jumper is removed, you need to connect it to the 5V pin on Arduino.

- **ENA** pins are utilized to control the speed of Motor A. Supplying this pin with HIGH logic makes the Motor A rotate, supplying it with LOW logic causes the motor to stop. Removing the jumper and connecting this pin to the PWM input let us control the speed of the Motor A.

- **IN1 & IN2** pins are used to control the direction of Motor A. If IN1 is HIGH and IN2 is LOW, Motor A spins in a certain direction. To change the direction, make IN1 LOW and IN2 HIGH. If both the inputs are either HIGH or LOW, the Motor A stops.

- **IN3 & IN4** pins are used to control the direction of the Motor B. If IN3 is HIGH and IN4 is LOW, Motor B spins in a certain direction. To change the direction, make IN3 LOW and IN4 HIGH. If both the inputs are either HIGH or LOW, the Motor B stops.

- **ENB** pin can be used to control the speed of Motor B. Supplying this pin with the HIGH signal makes the Motor B turn, supplying it LOW cause the motor to stop. Eliminating the jumper and interfacing this pin to PWM information let us control the speed of Motor B.

- **OUT1 & OUT2** pins are connected to Motor A.

- **OUT3 & OUT4** pins are connected to Motor B. [24]

15 I2C LCD display

15.1 What is an I2C LCDdisplay?

A typical I2C LCD display consists of an HD44780-based character LCD display and an I2C LCD adapter. Let's learn more about them. [26]

Figure 25-L2CLCD screen

15.2 Character LCD Display

As the name suggests, these LCDs are ideal for displaying only characters. A 16×2 character LCD, for example, candisplay32 ASCII characters acrosstwo rows

15.3 I2CLCDAdapter

This passage describes an adapter that utilizes the PCF8574 chip to convert I2C data from an Arduino into parallel data required for an LCD display. The adapter includes a trim-pot to adjust the display's contrast and a jumper to supply power to the backlight. By removing the jumper and providing an external voltage to the header pin marked as LED, the user can regulate the brightness of the backlight.

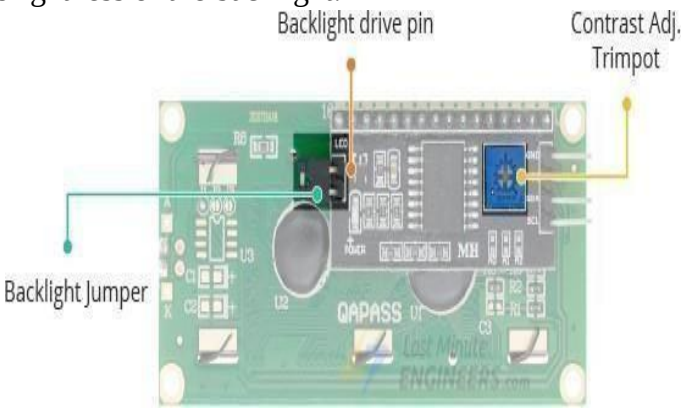


Figure 26-SerialI 2CLCD Display Adapter

16 I2C LCD Display Pinout

The I2C LCD Display has only four pins. The following is the pinout:

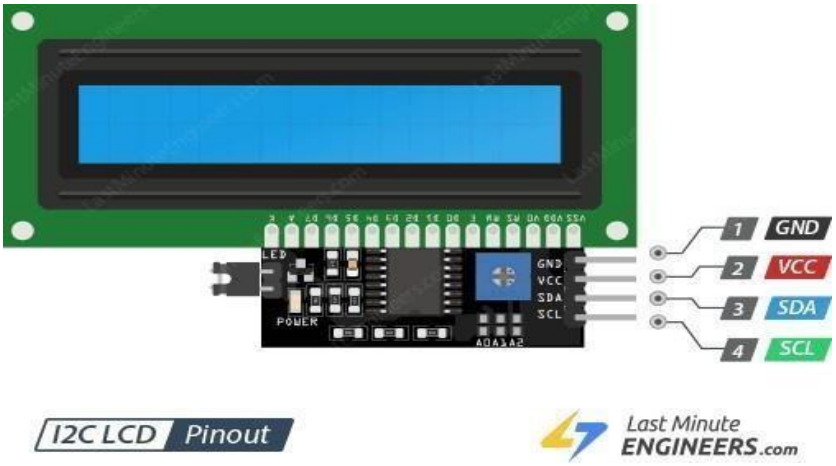


Figure 27-I2C LCD Display Pinout

GND: is a ground pin.

VCC is the power supply pin. Connect it to the 5V output of the Arduino or an external 5Vpower supply.

is the I2C data pin

SCL :is a Serial Clock pin. This is a timing signal supplied by the Bus Master device. Connectto the analog pin A5 on the Arduino.

17 Encoder working principale

Encoders are used to translate rotary or linear motion into a digital signal. Usually this is for the purpose of monitoring or controlling motion parameters such as speed, rate, direction, distance or position.
The Optical Encoders typically consist of a rotating and a stationary electronic circuit. The rotor is usually a metal, glass, or a plastic disc mounted on the encoder shaft. The disc has some kind of optical pattern, which is electronically decoded to generate position information. [25]

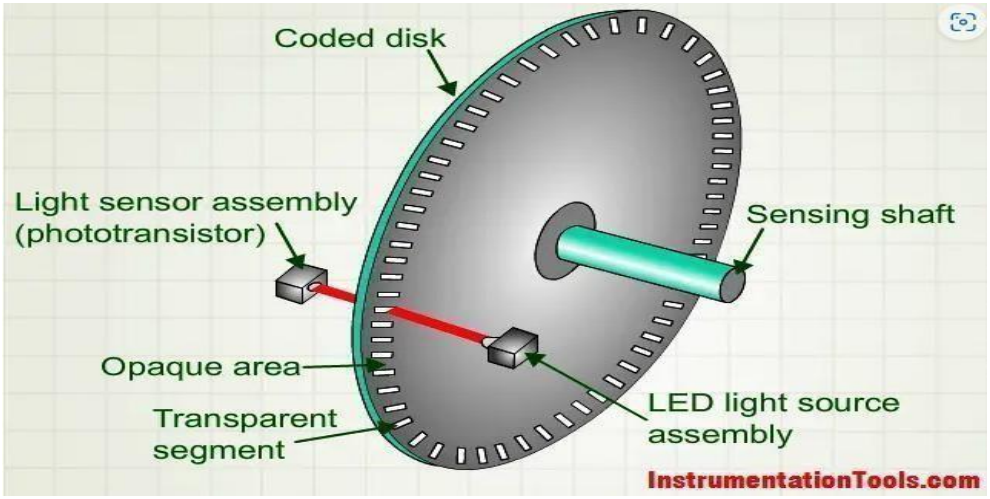


Figure 28-The rotor disc of the sensor

The rotor disc in absolute optical encoder uses opaque and transparent segments arranged in a gray-code pattern. The stator has corresponding pairs of LEDs and photo-transistors arranged so that the LED light shines through the transparent sections of the rotor disc and received by photo-transistors on the other side. After the electronic signals are amplified and converted, they are then available for the evaluation of the position

18 Lm393 Motor Speed

Widely used in motor speed detection, pulse count, the position limit, etc. The DO output interface can be directly connected to a micro-controller IO port, if there is a block detection sensor, such as the speed of the motor encoder can detect. DO modules can be connected to the relay, limit switch, and other functions, it can also with the active buzzer module, compose alarm

Figure 29-The stator of the sensor

The LM393 module is an IR counter that has an IR transmitter and receiver. If any obstacle is placed between these sensors, a signal is sent to the microcontroller. The module can be used in association with a microcontroller for motor speed detection, pulse count, position limit, etc.

18.1 LM393 Sensor Pinout

- VCC: Module power supply – 3-5.5 V
- GND: Ground
- OUT: Digital output data to the microcontroller

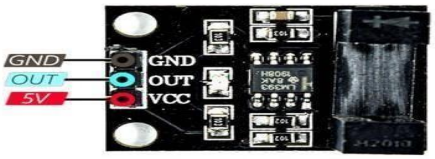


Figure 30-l393 pin

18.2 Required Materials

Figure 31-lm393 Required Materials

18.3 Hardware Components

Arduino UNO R3 × 1
LM393 Infrared Speed Sensor
Module × 1Male Female Jumper
Wire × 1

18.3.1 Software Apps

Arduino IDE

18.4 Interfacing LM393 Infrared Speed Sensor with Arduino

18.4.1 Step 1: Circuit

The following circuit shows how you should connect Arduino to LM393 sensor. Connect wires accordingly.

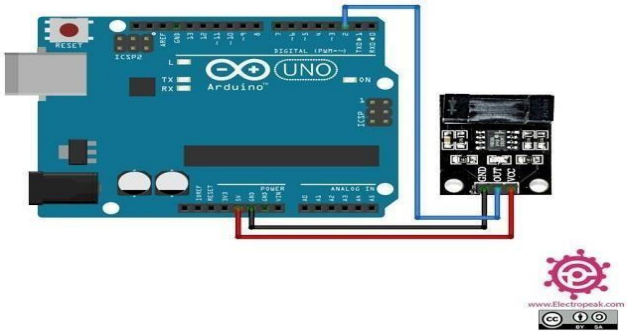


Figure 32-Lm393 wire

18.4.2 Step 2: Code

Upload the following code to your Arduino

```
/*
  by islemandmeziane
  Home
*/
#include "timer.h"

Timer timer;

const int LM393 = 2;
int counter = 0;
void setup() {
  attachInterrupt(digitalPinToInterrupt(LM393), count, RISING);
  Serial.begin(115200);
  timer.setInterval(1000);
  timer.setCallback(RPM);
  timer.start();
}
void count() {
  counter++;
}

void RPM() {
```

```
void loop() {
  timer.update();
}
```

After running the code, you will see the following image in the serial output.

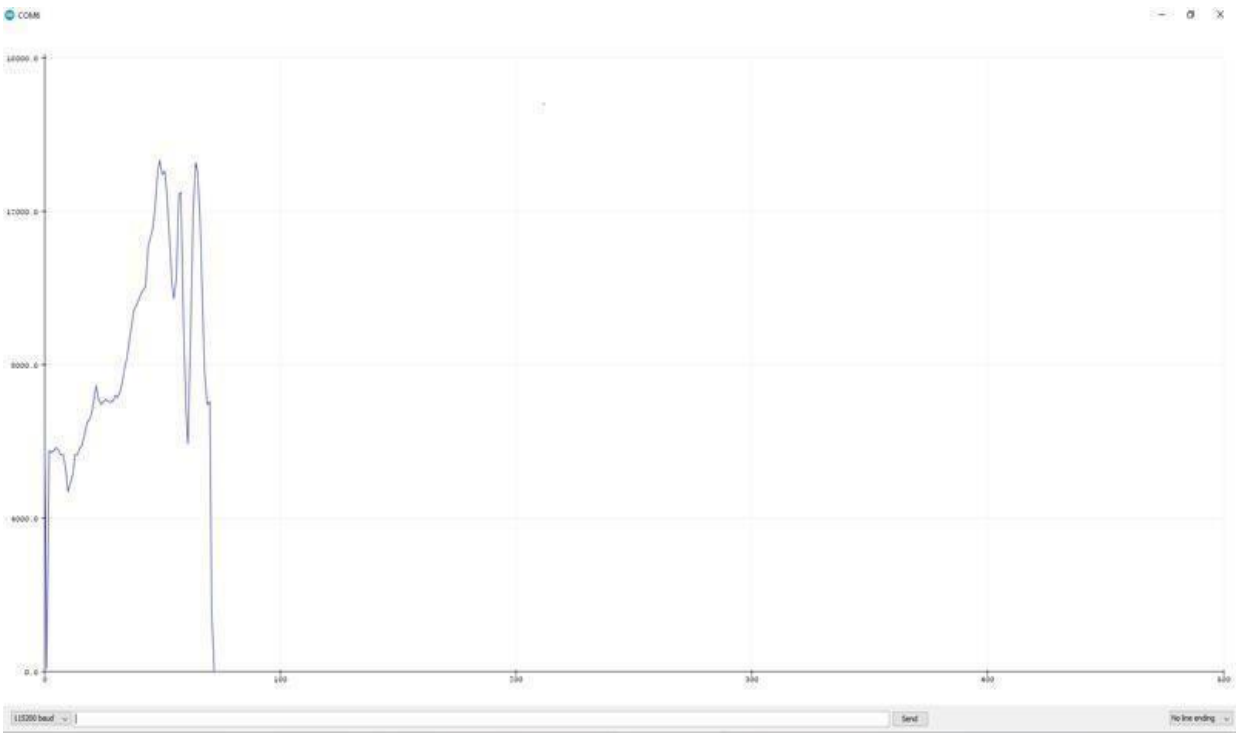


Figure 33-lm393 serial

19 Features

- Using imported trough type optical coupling sensor, groove width 5 mm.
- The output state light, lamp output level, the output low level light.
- Covered : output high level; Without sunscreen : the output low level
- The comparator output, signal clean, good waveform, driving ability is strong, for
- more than 15 ma. The working voltage of 3.3 V to 5 V
- Output form: digital switch output (0 and 1)
- A fixed bolt hole, convenient installation
- Small board PCB size: 3.2 cm x 1.4 cm
- Use the LM393 wide voltage comparator Module [25]

Chapter 4 Results and discussion

DC Motor Speed Regulation: System Modeling

20 Physical setup

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion. The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in the following figure.

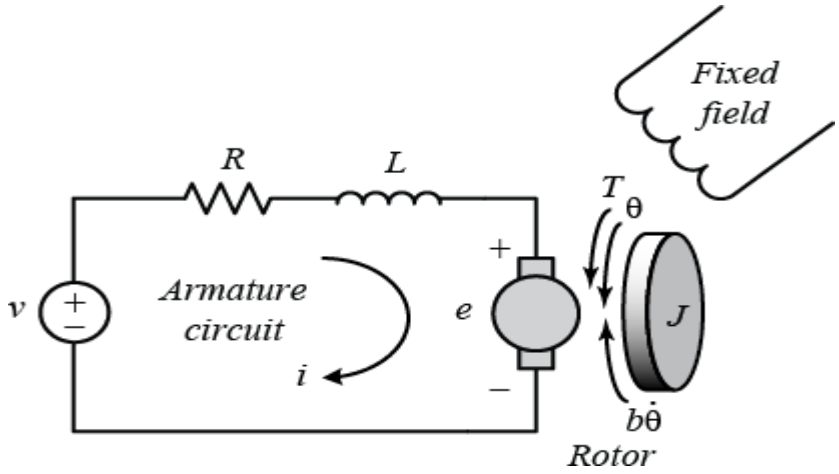


Figure 34-Schematic diagram of a DC Motor

For this example, we will assume that the input of the system is the voltage source (*V*) applied to the motor's armature, while the output is the rotational speed of the shaft d(*theta*)/dt. The rotor and shaft are assumed to be rigid. We further assume a viscous friction model, that is, the friction torque is proportional to shaft angular velocity.

The physical parameters for our example are:

(J)	moment of inertia of the rotor	0.01 kg.m^2
(b)	motor viscous friction constant	0.1 N.m.s
(Ke)	electromotive force constant	0.01 V/rad/sec

(Kt)	motor torque constant	0.01 N.m/Amp
(R)	electric resistance	1 Ohm
(L)	electric inductance	0.5 H

21 System equations

In general, the torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field. In this example we will assume that the magnetic field is constant and, therefore, that the motor torqueis proportional to only the armature current *i* by a constant factor *Kt* as shown in the equation below. This is referred to as an armature-controlled motor.

$t = kti$ (1)

The back emf, *e*, is proportional to the angular velocity of the shaft by

$e = ke\dot{\theta}$ (2)

constant factor *Ke*

In SI units, the motor torque and back emf constants are equal, that is, *Kt* = *Ke*; therefore, we will use *K* to represent both the motor torque constant and the back emf constant.

From the figure above, we can derive the following governing equations based on Newton's 2nd law and Kirchhoff'svoltage law.

$$J\dot{\theta} + b\dot{\theta} = k_i \qquad L\frac{di}{dt} + R_{i=}v_{-k\dot{\theta}}$$

21.1 . Transfer Function

Applying the Laplace transform, the above modeling equations can be expressed in terms of the

$$s(Js + b)\theta(s) = kI(s)$$

$$(Ls + R)I(s) = v(s) - ks\theta(s)$$

Laplace variable *s*.(5)

(6)

We arrive at the following open-loop transfer function by eliminating *I*(*s*) between the two above equations, where the rotational speed is considered the output and the armature voltage is considered the input.

$$p(s) = \frac{\theta(s)}{v(s)} = \frac{k}{(Js+b)+(Ls+R)+k^2} \qquad \left[\frac{rad/sec}{V}\right]$$

21.2 State-Space

In state-space form, the governing equations above can be expressed by choosing the rotational speed and electric current as the state variables. Again the armature voltage is treated as the input and the rotational speed is chosenas the output.

$$\frac{d}{dt}\begin{bmatrix}\theta \\ i\end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{k}{L} & -\frac{R}{L} \end{bmatrix}\begin{bmatrix}\theta \\ i\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{L}\end{bmatrix} V \qquad y = [1 \quad 0]\begin{bmatrix}\theta \\ i\end{bmatrix}$$

$$\frac{d}{dt}\begin{Bmatrix}\theta \\ i\end{Bmatrix} = \begin{bmatrix} \frac{J}{-K} & \frac{J}{-R} \\ L & L \end{bmatrix}\begin{bmatrix}\theta \\ i\end{bmatrix}\begin{bmatrix}0 \\ \frac{1}{L}\end{bmatrix} V \qquad y = [1 \quad 0]\begin{bmatrix}\theta \\ i\end{bmatrix}$$

21.3 Design requirement

First consider that our uncompensated motor rotates at 0.1 rad/sec in steady state for an input voltage of 1 Volt Since the most basic requirement of a motor is that it should rotate at the desired speed, we will require that the steady-state error of the motor speed be less than 1%. Another performance requirement for our motor is that it must accelerate to its steady-state speed as soon asit turns on. In this case, we want it to have a settling time less than 2 seconds. Also, since a speed faster than the reference may damage the equipment, we want to have a step responsewith overshootof less than 5%.

In summary, for a unit step command in motor speed, the control system's output should meet the following requirements.

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

21.4 MATLAB representation

Transfer Function

We can represent the above open-loop transfer function of the motor in MATLAB by defining the parameters andtransfer function as follows. Running this code in the command window

```
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+K^2)
```

$$\textbf{P moto} = \frac{0.01}{0.005\ s\text{^}2 + 0.06\ s + 0.1001x}$$

21.5 State Space

We can also represent the system using the state-space equations. The following additional MATLAB commands create a state-space model of the motor and produce the output shown below when run in the MATLAB command window.

```
A = [-b/J K/J -K/L -R/L];
B = [0 1/L];
C = [1 0];
D = 0;
motor_ss = ss(A,B,C,D)
```

The state space model above can also be generated by transforming your existing transmission Functional models in state-space form. Again, this is done with the ss command as shown Under

```
motor_ss=ss(P_motor);
```

For a 1-rad/sec step reference, the design criteria are the following.

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

22 Open-loop response

First create a new m-file and type in the following commands (refer to the main problem for the details of gettingthese commands).

Now let's see how the original open-loop system performs. Add the following ltiview command onto the end of the m-file and run it in the MATLAB command window. The string 'step' passed to the function specifies to generate a unit step response plot for the system P_motor. The range of numbers 0:0.1:5 specify that the step response plot should include data points for times from 0 to 5 seconds in steps of 0.1 seconds. The resulting plot is shown in the figure below, where you can view some of the system's characteristics by right clicking on the figure and choosing from the Characteristics menu such performance aspects as Settling Time and Steady Stat

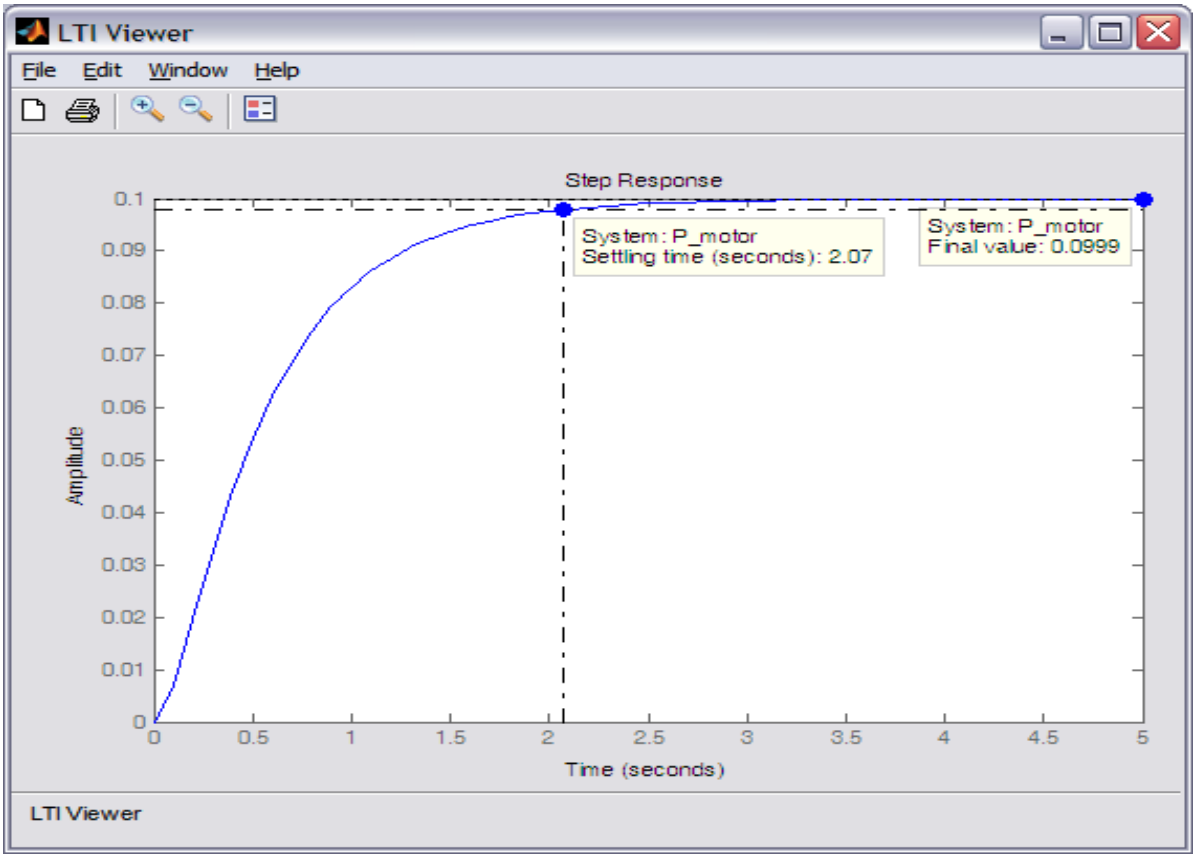


Figure 35--Open-loop response.

From the plot we see that when 1 Volt is applied to the system the motor can only achieve a maximum speed of 0.1 rad/sec, ten times smaller than our desired speed. Also, it takes the motor 2.07 seconds to reach its steady-state speed; this does not satisfy our 2 second settling time criterion.

23 PID Controller Design

From the main problem, the dynamic equations in the Laplace domain and the open-loop transfer function of the DC Motor are the following.

$$s(Js + b)\theta(s) = KI(s)$$

$$p(s) = \frac{\theta(s)}{V(S)} = \frac{K}{(Js+b)(Ls+R)+K^2} \quad \frac{rad/sec}{V} \qquad (Ls + R)I(s) = V(s) - Ks\theta(s)$$

The structure of the control system has the form shown in the figure below.

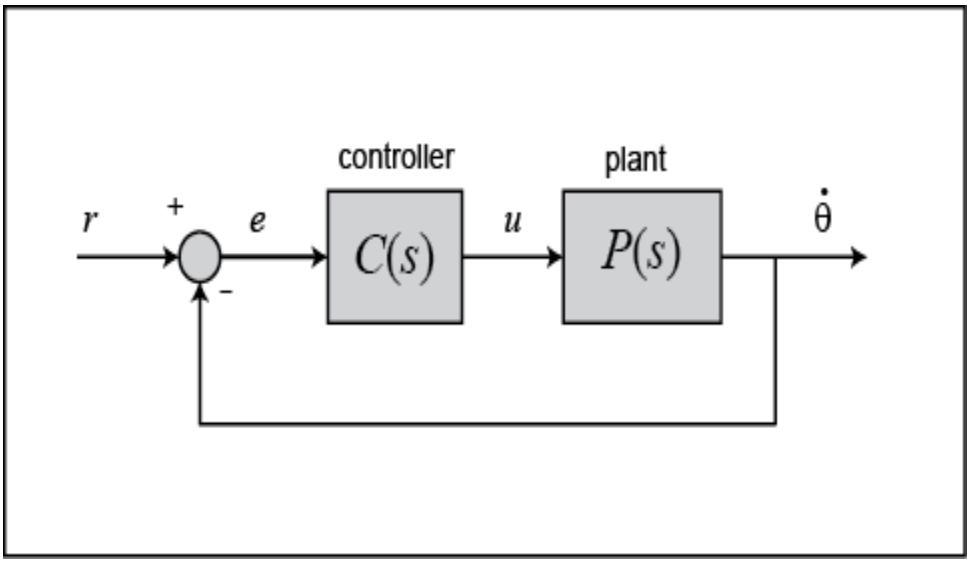


Figure 36-The structure of the control system.

For a 1-rad/sec step reference, the design criteria are the following.

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

```
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+K^2);
```

Now let's design a controller using the methods introduced

Recall that the transfer function for a PID controller is:

$$C(8)K_{p+\frac{K_i}{s}} + K_{d^8} = \frac{K_{d8^2} + K_{p^{8+Ki}}}{8}$$

24 Proportional control

Let's first try employing a proportional controller with a gain of 100, that is, C(s) = 100. To determine the closed-loop transfer function, we use the feedback command. Add the following codeto the end of your m-file.

```
Kp = 100;
C = pid(Kp);
sys_cl = feedback(C*P_motor,1);
```

Now let's examine the closed-loop step response. Add the following commands to the end of your m-file and run it in the command window. You should generate the plot shown below. You can view some of the system's characteristics by right-clicking on the figure and choosing Characteristics from the resulting menu. In the figure below, annotations have specifically been added for Settling Time, Peak Response, and Steady State.

```
t = 0:0.01:5;
step(sys_cl,t)
grid
title('Step Response with Proportional Control')
```

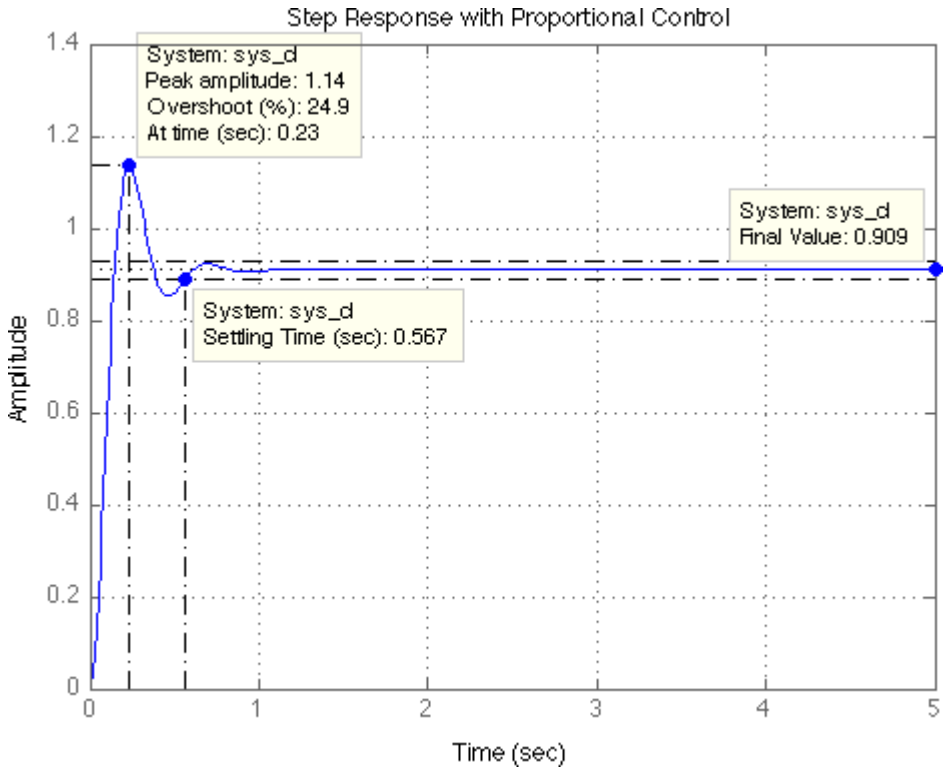


Figure 37-step response with propotional control

From the plot above we see that both the steady-state error and the overshoot are too large. increasing the proportional gain *Kp* will reduce the steady-state error. However, also recall that increasing *Kp* often results in increased overshoot, therefore, it appears that not all of the design requirements can be met with a simple proportional controller.

for meeting the given design requirements; derivative and/or integral terms must be added to the controller.

25 PID control

Recall from the Introduction: PID Controller Design page adding an integral term will eliminate the steady-state error to a step reference and a derivative term will often reduce the overshoot. Let's try a PID controller with small *Ki* and *Kd*. Modify your m-file so that the lines defining your control are as follows. Running this new m-file gives you the plot shown

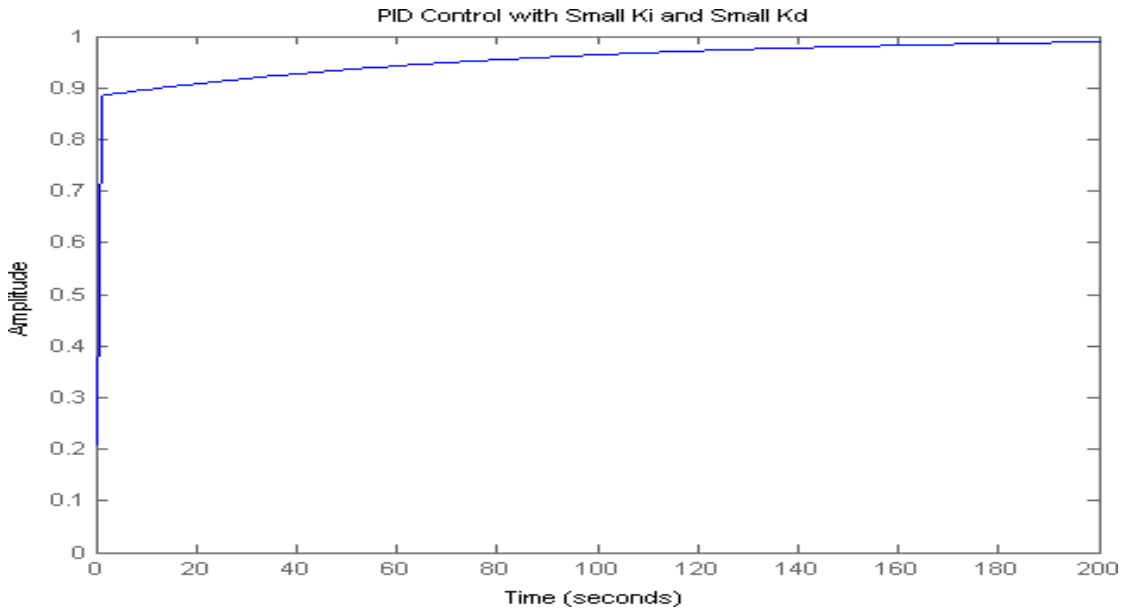


Figure 38-PID Control With Small Ki and Small Kd

```
Ki = 1;
Kd = 1;
C = pid(Kp,Ki,Kd);
sys_cl = feedback(C*P_motor,1);
step(sys_cl,[0:1:200])
title('PID Control with Small Ki and Small Kd')
```

Inspection of the above indicates that the steady-state error does indeed go to zero for a step input. However, thetime it takes to reach steady-state is far larger than the required settling time of 2 seconds.

26 Tuning the gains

In this case, the long tail on the step response graph is due to the fact that the integral gain is small and, therefore, it takes a long time for the integral action to build up and eliminate the steady-state error. This process can be sped up by increasing the value of *Ki*. Go back to your file and change *Ki* to 200 as in the following.

```
Kp = 100;
Ki = 200;
Kd = 1;
C = pid(Kp,Ki,Kd);
sys_cl = feedback(C*P_motor,1);
step(sys_cl, 0:0.01:4)
grid
title('PID Control with Large Ki and Small Kd')
```

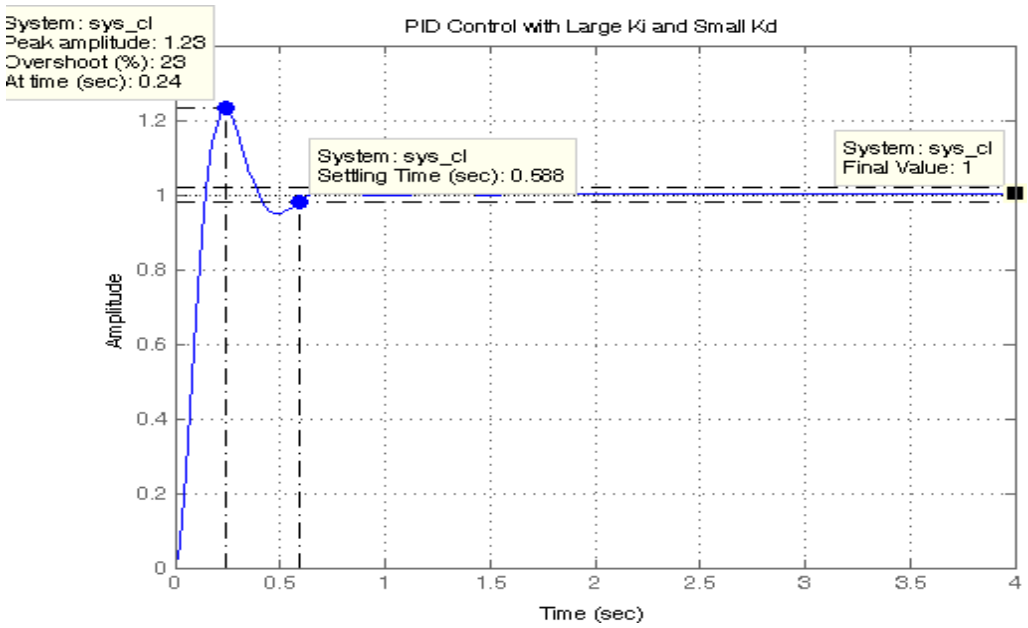


Figure 39-PID Control With Small Ki and Small Kd

As expected, the steady-state error is now eliminated much more quickly than before. However, the large *Ki* has greatly increased the overshoot. Let's increase *Kd* in an attempt to reduce the overshoot. Go back to the m-file and change *Kd* to 10 as shown in the following.

```
Kp = 100;
Ki = 200;
Kd = 10;
C = pid(Kp,Ki,Kd);
sys_cl = feedback(C*P_motor,1);
step(sys_cl, 0:0.01:4)
grid
```

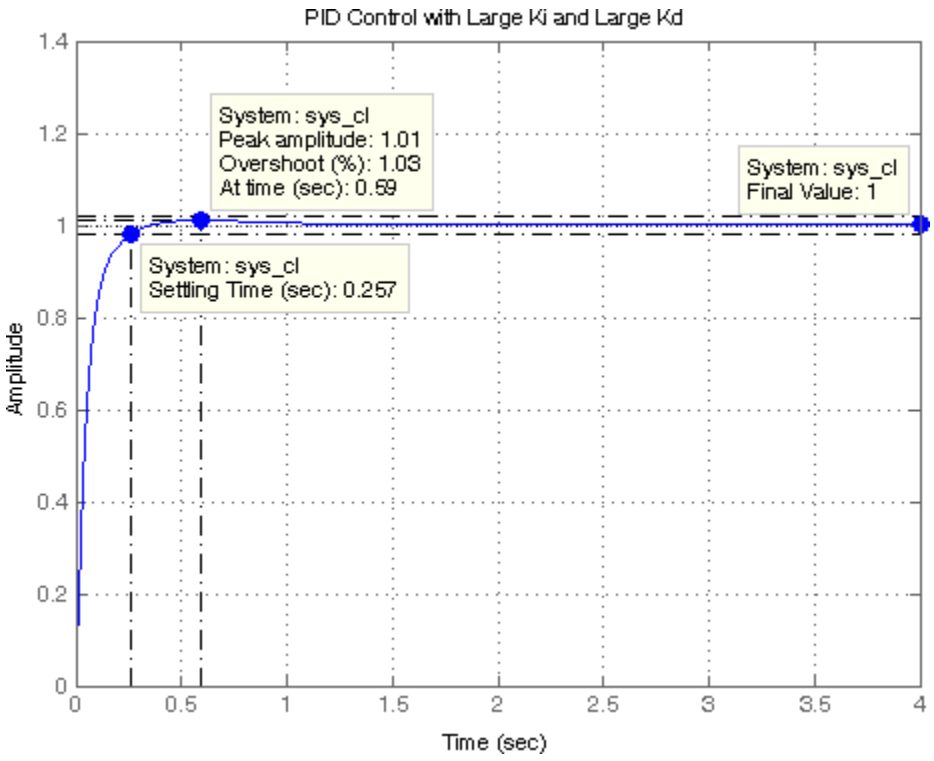


Figure 40--PID Control With Large Ki and Large Kd

As we had hoped, the increased *Kd* reduced the resulting overshoot. Now we know that if we use a PID controllerwith

Kp = 100, *Ki* = 200, and *Kd* = 10,

all of our design requirements will be satisfied.

27 Results and Analysis of real implementation of the system

27.1 Step 1-Hardware and Software needed

The successful completion of this project necessitates the utilization of the following hardware components:

1. DC Motor with encoder LM393
2. L298N motor driver
3. Arduino UNO
4. Breadboard
5. Jumper Wires
6. LCD
7. BUTTONS

Below is a compilation of the accompanying elements:

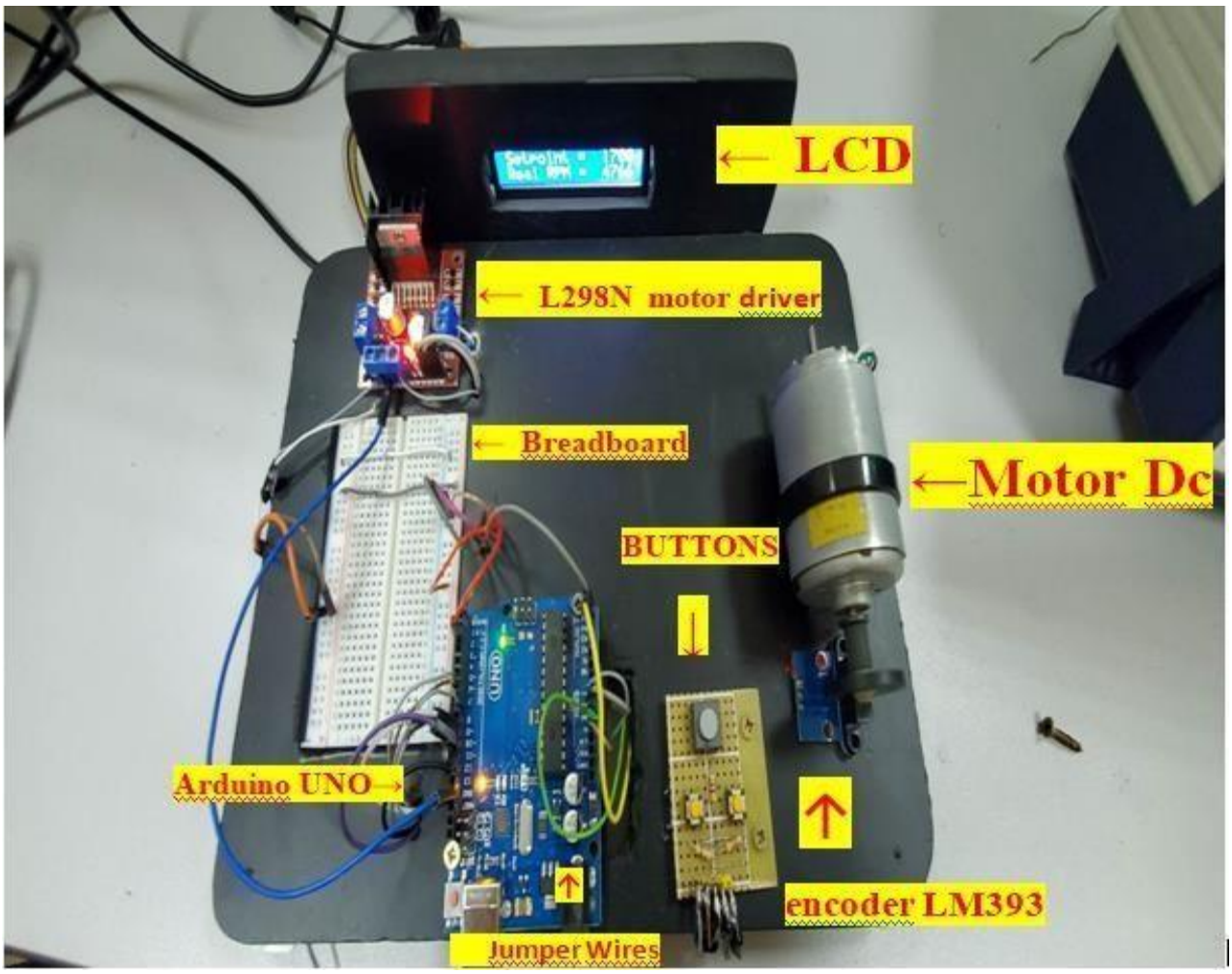


Figure 41-Components Required to Build a PID Enabled Encoder Motor Controller

27.2 Step 2- Hardware connection

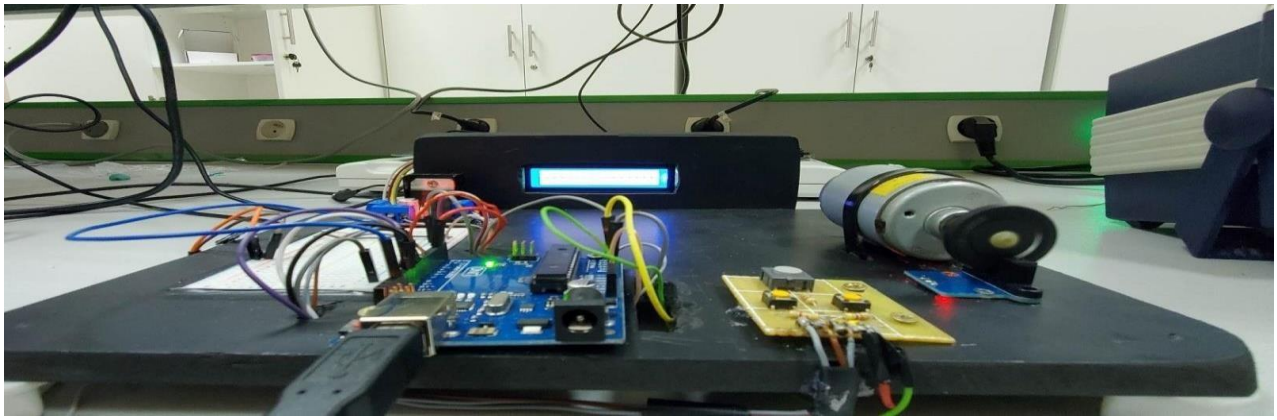


Figure 42- Experiment for DC motor speed control

In this circuit, we have a DC motor connected to an encoder. The encoder's output is connected to pin 2 of a microcontroller or another controlling device. To control the motor's speed, we use motor driver l298N an H-bridge, which allows us to change the direction and adjust the speed of the motor. Pins 5, 6, and 7 of the controlling device are connected to the H-bridge to control its operation.
Additionally, we have an IR sensor module positioned in front of the disc attached to the motor. As the motor and disc rotate, the IR sensor detects changes in the intercepted infrared values. These changes can be used to estimate the current RPM of the motor. The RPM estimation can be displayed on an LCD or any other suitable output display.
Once we have the motor's RPM, we can use this information to manage the motor's speed. By comparing the desired RPM with the measured RPM, we can dynamically adjust the control signals sent to the H-bridge to achieve the desired speed.

To maintain a consistent speed for the motor, the RPM value obtained from the IR sensor is sent to a microcontroller, such as an Arduino, as feedback. The microcontroller then compares this feedback value with a predefined set point, which represents the desired speed.

The difference between the feedback RPM and the set point RPM is calculated, resulting in an error value. To regulate the motor speed and reduce this error, a PID (Proportional-Integral-Derivative) control algorithm is implemented in the Arduino code. The PID algorithm analyzes the error value and generates a control signal.

The control signal, typically in the form of a PWM (Pulse Width Modulation) signal, is sent from the Arduino to the H-Bridge. The H-Bridge acts as a motor driver and adjusts the power supplied to the DC motor based on the received control signal.

If there is an increase in load on the motor, the motor's RPM will decrease. The Arduino code, by continuously monitoring the RPM, can detect this change. In response, the Arduino adjusts the PWM signal sent to the H-Bridge, increasing the power supplied to the motor to compensate for the decreased RPM and maintain the desired speed.

Similarly, if the load on the motor decreases, the RPM will increase. The Arduino senses this change by monitoring the RPM value and adjusts the PWM signal to reduce the power supplied to the motor, thereby compensating for the increased RPM and maintaining the desired speed.

27.3 Step 3- Arduino Code

27.3.1 Arduino code will do:

(1) Calculate motor speed
(2) Send motor speed to Computer
(3) Calculate PWM pulse (base on PID algorithm)
(4) Push result of PWM to H-brigde

27.3.2 Step 4 - Code works at Computer

(1) Send speed setting to Arduino
(2) Send PID gain to Arduino
(3) Receive motor speed, show on graph

Figure 43 shows the comparison between the desired speed and the actual speed. after that number is The speed of a DC motor with an encoder can be controlled by a computer using PID gains.

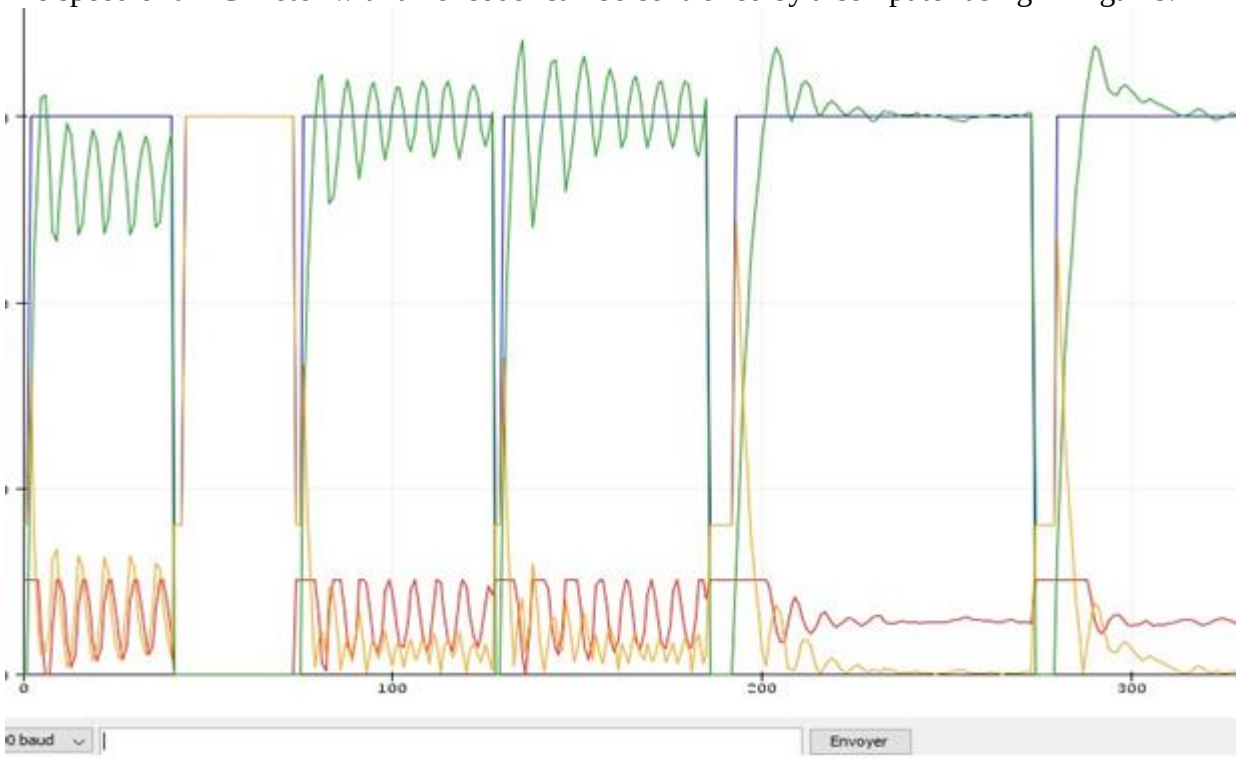


Figure 43- System's response with various state feedback control parameters.

In this closed-loop system analysis, the PID parameters are tuned manually in the sketch. this

The purpose of parameter tuning is to find the best parameters for PID. good system

Response should degrade in time slope, settling time, overshoot (%) and steady state error

- Let's try a proportional controller first; so in the program we set **kp=1, ki=0, kd=0**

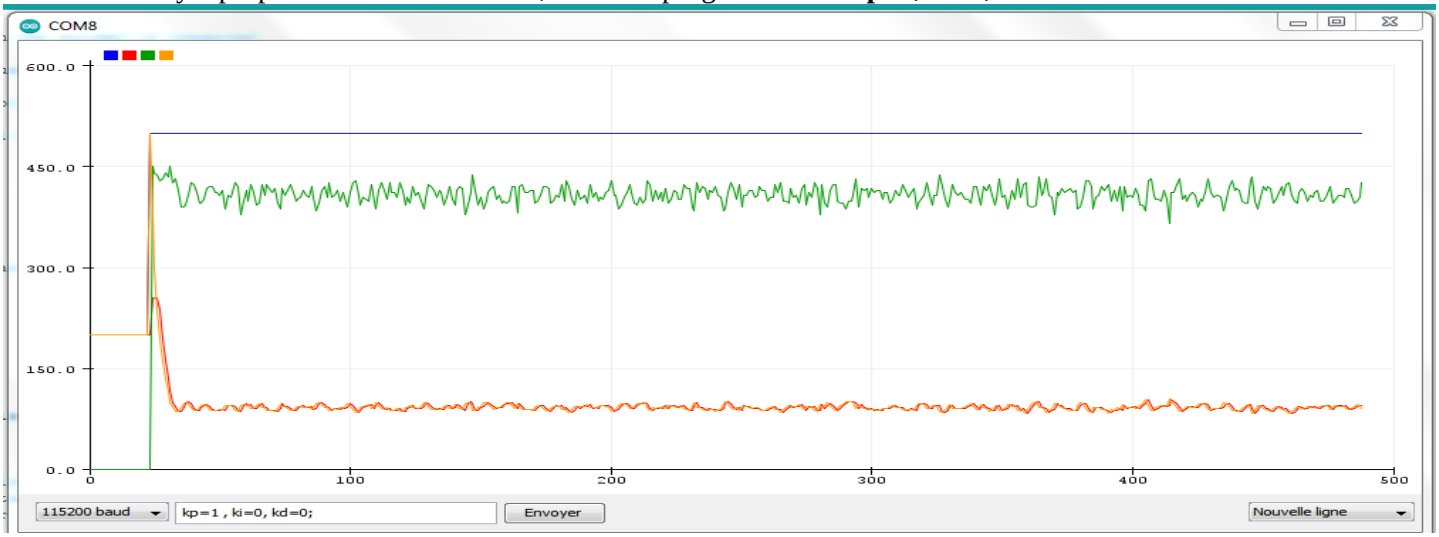


Figure 44- closed loop response with kp=1

The controller response parameters are described as follows:

- The steady state error SSE slope for this control mode is SEE = 100 RPM
- System becomes unstable due to vibration

We now try to increase the gain Ki to 0.8, with KP=0.1, the result is shown in Figure 45

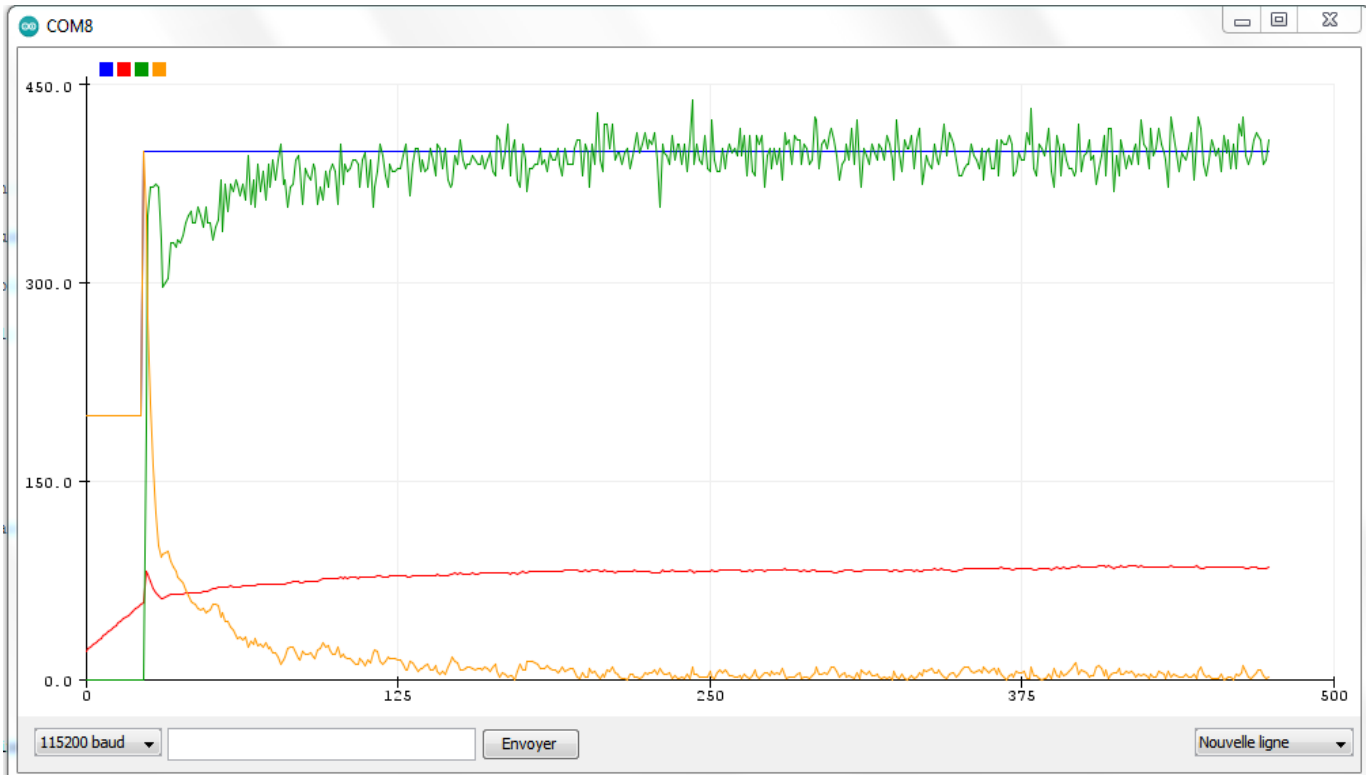


Figure 45- losed loop response with PI Controller

We notice that :

- SSE starts to decrease, becomes small and disappears
- However, the system experienced a significant overshoot.
- We find that the system become a little stable than before

After we change pi parameter now we did it again ,and we try to increase the the KP to 0.3 with Ki = 0.6

Figure 46-System's response with Kp =0.3 and Ki= 0.6

We notice that :

- The steady state error still decrease more and more
- Still have a small overshot in system
- We notice also that the system start to become more stable than before

Now we decrease the KP to 0.2 and the Ki to 0.5

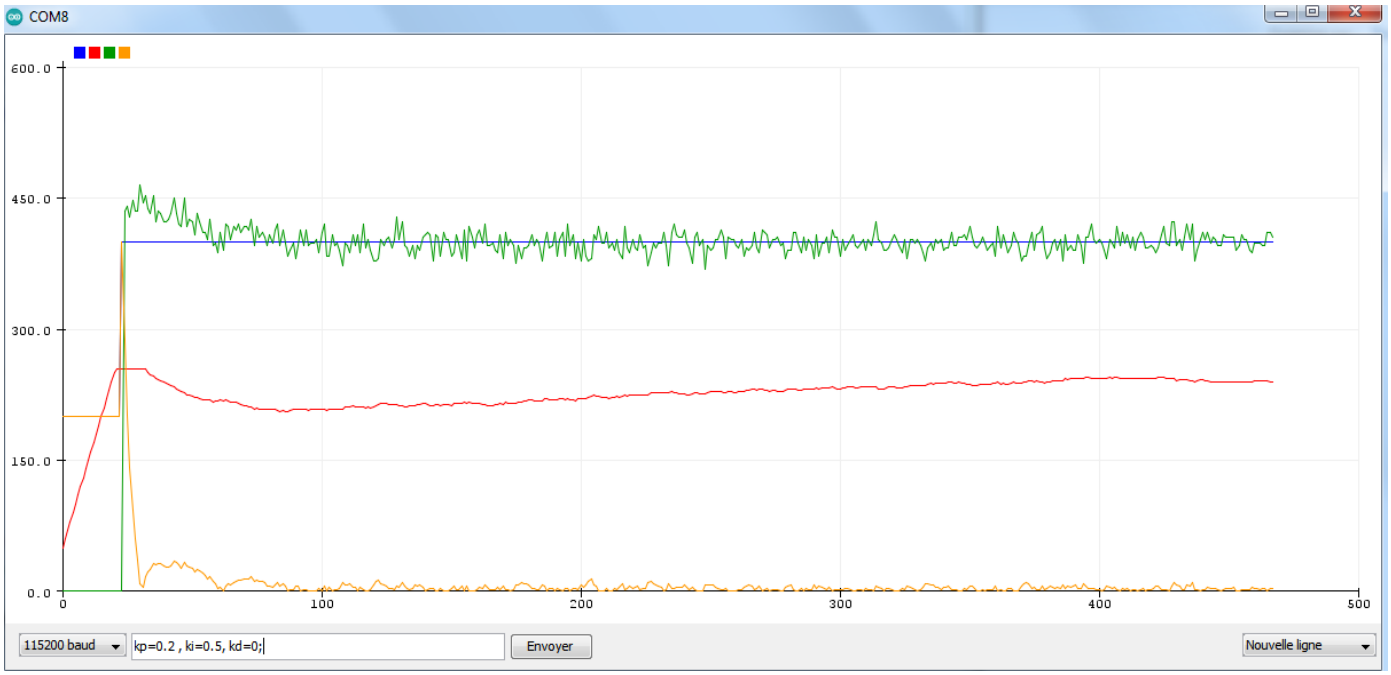


Figure 47-System's response with Kp =0.5 and Ki= 0.5

We find that the system :

- We notice that the reponse get more better and quick then before

- the Steady State Error SSE =0
- still have a little overshot in our system
- However, there is still a noticeable delay between the intended speed and the actual speed during performance.

the speed increases above the set value (300 rpm), the duty cycle is PWM down to compensate for the actual glitch, then immediately return

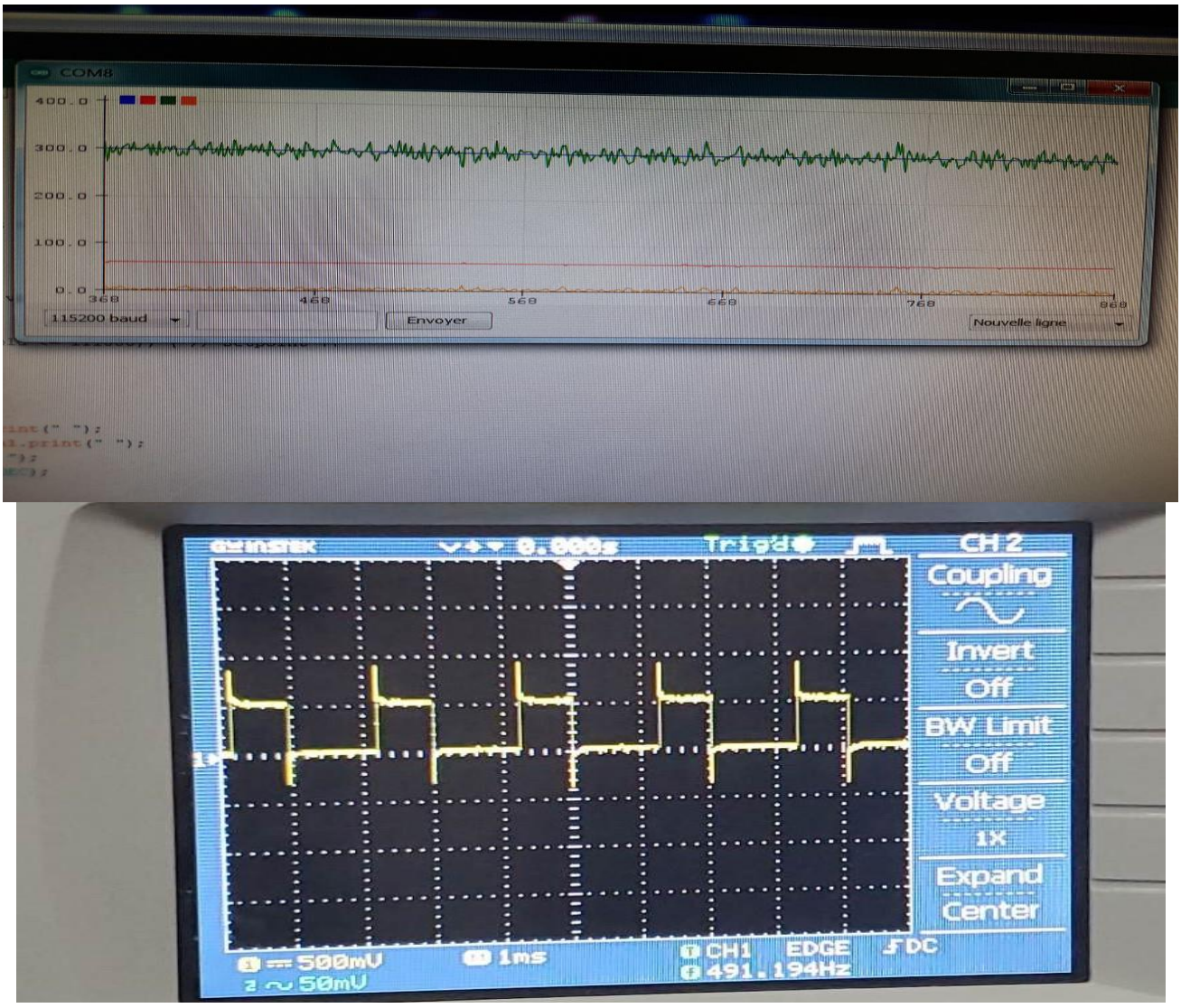


Figure 48-Effects of PWM signal on the speed,rpm=300

As the speed decreases, the duty cycle of the PWM decreases
Add 100% to immediately compensate for actual disturbances, thus compensating for speed Increase
When the speed reaches the target value and the system is stable, the PWM duty cycle is also the same Stable.

Conclusion

Experiment with different PID parameter values to achieve the desired speed regulation characteristics. Optimize the PID controller's gains for stability, responsiveness, and minimal overshoot or steady-state error.
Develop the PID control algorithm in software using the Atmega328P. Tune the PID parameters to achieve optimal performance in terms of speed regulation, stability, and response time
The simulation outcomes closely match the practical measurements.. in our case Overshoot come from the sensor used in the control system has a slow response or exhibits hysteresis. Delayed or nonlinear responses that lead to little overshoot as the controller compensates for the perceived error. We try to find the best parameter of pid to make the system much better and stable .

In conclusion, the combination of a PID controller, Atmega328P microcontroller, L298N motor driver, Kalman filter, and L393 speed sensor provides an effective solution for DC motor control.

The PID controller allows precise control of the motor by continuously adjusting the input voltage based on the error between the desired and measured speed or position. It uses three components: the proportional term (P), which responds to the current error; the integral term (I), which corrects any steady-state error over time; and the derivative term (D), which predicts and adjusts for future changes in the error.

The Atmega328P microcontroller serves as the brain of the control system, responsible for executing the PID algorithm, reading sensor data, and generating appropriate control signals. It offers a wide range of input/output capabilities, making it suitable for motor control applications.

The L298N motor driver provides the interface between the microcontroller and the motor. It allows bidirectional control of the motor's rotation and manages the power supply to the motor, ensuring efficient and safe operation.

The Kalman filter is a mathematical algorithm used to estimate the true state of a system by combining noisy measurements with a model of the system. In the context of DC motor control, the Kalman filter can be used to improve the accuracy of speed or position estimation by reducing the effects of sensor noise and disturbances.

The L393 speed sensor is used to measure the actual speed of the motor. It provides feedback to the PID controller, enabling it to adjust the motor's input voltage to reach and maintain the desired speed or position.

By combining these components, the DC motor control system can achieve precise speed or position control, respond to dynamic changes, and compensate for external disturbances or measurement errors, resulting in improved overall performance and reliability.

Future works of the research are open to find more stable system and and a better control of dc motor we apply the kalman filter and we still have a small over shot maybe in the future research they can Implement anti-windup mechanisms or techniques to prevent integrator windup. Integrator windup can occur when the controller output saturates, causing the integral term to accumulate excessive error. Anti-windup methods help to limit the integrator's contribution during such situations, and reducing overshoot

the PID control system offers a reliable and efficient method for controlling DC motors, providing precise and responsive control across a wide range of applications. Its versatility, adaptability, and ability to handle disturbances make it a popular choice for motor control in industries such as robotics, automation, and manufacturing. With ongoing advancements in control algorithms and hardware capabilities, the future holds even greater potential for further improvements and optimizations in PID-based motor control systems.

Bibliography and Webography

[1] International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056 Volume: 03 Issue: 09 | Sep-2016 www.irjet.net p-ISSN: 2395-0072© 2016, IRJET | Impact Factor value: 4.45 | ISO 9001:2008 Certified Journal | Page 791 [25/02/2023]

[2]International Journal of Advanced Trends in Computer Science and Engineering, Vol.5 , No.1, Pages : 124 -127 (2016) Special Issue of ICACEC 2016 - Held during 23-24 January, 2016 in Institute of

[3]Aeronautical Engineering, Quthbullapur, Telangana-43, India [27/02/2023]

International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-6, April 2019 [27/02/2023]

[4]https://www.arduino.cc/en/Guide/Introduction/ [28/02/2023]

[5]https://learn.sparkfun.com/tutorials/what-is-an-arduino/all [03/03/2023]

[6]https://predictabledesigns.com/how-to-choose-the-best-development-kit-the-ultimate-guide-for-beginners/ [03/03/2023]

[7]https://docs.arduino.cc/hardware/uno-rev3 [04/03/2023]

[8]https://www.circuito.io/blog/arduino-uno-pinout/ [06/03/2023]

[9]https://www.farnell.com/datasheets/1682238.pdf [07/03/2023]

[10]https://binaryupdates.com/serial-communication-in-arduino-uno/ [07/03/2023]

[11]https://www.circuito.io/blog/arduino-code/ [11/03/2023]

[12]https://www.makerspaces.com/simple-arduino-projects-beginners/ [20/03/2023]

[13]https://stemify.weebly.com/programming-structure.html [21/03/2023]

[14]https://stemify.weebly.com/programming-structure.html [23/03/2023]

[15]https://projectiot123.com/2019/04/08/arduino-nano-for-beginners/ [01/04/2023]

[16]https://www.elprocus.com/the-working-of-a-pid-controller/ [03/04/2023]

[17]Farouk Zouari, et.al., “Ädaptive Internal Model Control of a DC Motor Drive System Using Dynamic Neural Network" Journal of Software [05/04/2023]

[18]Pulse width modulation reduces the average power delivered by an electrical signal by converting the signal into [18]discrete parts. In the PWM technique, the signal's energy is distributed through a series of pulses rather than a continuously varying (analogue) signal. [10/04/2023]

[19] AdityaPratap Singh, "Speed Control of DC Motor PID Controller Based on Mat lab" International Conference on Recent Trends in Applied Sciences with Engineering Applications, Vol.4, No.6, 2013. [19]

[20] https://www.iqsdirectory.com/articles/electric-motor/dc-motors.html [12/04/2023]

[21] https://wiraelectrical.com/parts-of-a-dc-motor/ [13/04/2023]

[22] https://www.linquip.com/blog/working-principle-of-dc-motor/ [16/04/2023]

[23] https://microcontrollerslab.com/l298n-motor-driver-circuit/ [17/04/2023]

[25]https://www.etechnophiles.com/l298n-motor-driver-pin-diagram/#:~:text=The%20L298N%20motor%20driver%20is%20based%20on%20the,driver%20constructed%20to%20receive%20standard%20TTL%20logic%20levels. [19/04/2023]

[25]https://electropeak.com/learn/interfacing-lm393-infrared-speed-sensor-with-arduino/ [22/04/2023]

[26] https://docs.arduino.cc/learn/electronics/lcd-displays [24/04/2023]