République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Tissemsilt

Faculté des Sciences et de la Technologie
Département des Sciences et de la Technologie

Mémoire de fin d'études pour l'obtention du diplôme
de Master académique en

Filière : **Electronique**

Spécialité : **Instrumentation**

Présentée par : **Mahloul Rabab**
Larbi Nassira

Thème

Real-time speed and direction control of dc motor based on ATMEGA328P µc

---

Soutenu le 19 juin 2022

Devant le Jury :

Djoudi Lakhdar	Président	Prof.	Univ-Tissemsilt
Nail Bachir	Encadrant	M.C.A.	Univ-Tissemsilt
Taibi Ahmed	Examinateur	M.A.B.	Univ-Tissemsilt

Année universitaire : 2021-2022

DEDICATED

To

With great pleasure I dedicate this project.

To those who helped and encouraged me.

To my parents Mahloul Hadj and Zineb who supported me

To all my sisters Arbia; Aicha; Amira and who were always present with me.

To my best friend Abid Malika who encouraged me all the time.

, I wish them joy and success

To all those who I love and who try to create this environment in which I

Live, full of joy and favorable atmosphere.

Mahloul Rabab

DEDICATED

To

How beautiful it is for a person to give his most precious

Possessions and the most beautiful to gift the dear

to the most precious they are the fruit of my labors.

Today's earnings are a gift that

I dedicate to: _my dear father, May God protect him _ my dear mother, may God

prolong her life _All my brothers and sisters _and to those who

supported me in this work

Larbi Nassira

Acknowledgements

First, we want to thank "GOD" for giving us courage and patience during our studies.

Secondly, we would like to express our gratitude to our supervisor Dr. Nail Bachir, a teacher

At Institute of Science and Technology for his precious help, his quality supervision and his directives which have been of great help to us in fulfilling our Project Graduation.

We also wanted to thank director of Institute of Science and Technology, for allowing this work to be carried out in good conditions.

Finally, we express our thanks, the most devoted, to the president of the jury and the jury membersFor accepting the judgment of our modest end of studies project

Table of contents

Chapter3 Methodology

Chapter 4 Results and discussion

List of figures

List of tables

Glossary

BJT :	bipolar junction transistor
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:	**U**niversal **S**ynchronous-**A**synchronous **R**eceiver-**T**ransmitter
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
ASCII:	American Standard Code for Information Interchange,
Hfe :	the current gain or amplification factor of a transistor

Abstract. *This paper concerns with the design and implementation of a microcontroller (ATmega328P) based PID controller for the purpose of controlling DC motor speed.*
The design had been done using MATLAB Simulink and IDE software, in addition to a practical physical system. The controller was implemented on an Arduino Nano board to control the speed of the motor at a desired value with the possibility of changing it and its direction of rotation.
The proposed system use an optical encoder as feedback sensor of motor speed which is compared with reference speed to produce error signal for the controller input, the control signal then sets the PWM duty cycle to change the motor speed which controlled via H-bridge.
The controller was able to track the input regardless of friction, disturbance and medium changes of load on motor shaft.

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

Résumé. *Cet article concerne la conception et la mise en œuvre d'un contrôleur PID basé sur un microcontrôleur (ATmega328P) dans le but de contrôler la vitesse d'un moteur à courant continu. La conception a été réalisée à l'aide du logiciel MATLAB Simulink et IDE, en plus d'un système physique pratique. Le contrôleur a été implémenté sur une carte Arduino Nano pour contrôler la vitesse du moteur à une valeur souhaitée avec la possibilité de la changer ainsi que son sens de rotation. Le système proposé utilise un codeur optique comme capteur de rétroaction de la vitesse du moteur qui est comparée à la vitesse de référence pour produire un signal d'erreur pour l'entrée du contrôleur, le signal de commande définit ensuite le rapport cyclique de PWM pour modifier la vitesse du moteur qui est contrôlée via le H-Bridge*
Le contrôleur a pu suivre l'entrée indépendamment du frottement, des perturbations et des changements moyens de charge sur l'arbre du moteur.

Mots clés : PID, pont en H, ATmega328P, moteur à courant continu, PWM.

ملخص. *يتعلق هذا البحث بتصميم وتنفيذ المتحكم التناسبي التكاملي التفاضلي PID عن طريق وحدة التحكم* ATmega*328P باستخدام برنامج MATLAB Simulink و IDE، بالإضافة إلى نظام فيزيائي عملي. تم تنفيذ المتحكم على لوحة اردوينو نانو للتحكم في سرعة المحرك بالقيمة المرغوبة مع امكانية تغييره واتجاه دورانه، يستخدم النظام المقترح مشفرًا ضوئيًا كمستشعر لقياس سرعة المحرك والتي تتم مقارنتها بالسرعة المرجعية لإنتاج إشارة الخطأ لمدخل وحدة التحكم، حيث تقوم الاخيرة بإرسال اشارة تعديل عرض النبضة PWM الى جسر H-Bridge الذي يتحكم بفرق الكمون المطبق على طرفي المحرك وبالتالي التحكم في سرعته واتجاهه.*
المتحكم PID قادر على تتبع المدخلات بغض النظر عن الاحتكاك والاضطراب والتغيرات المتوسطة للحمل على عمود المحرك.

الكلمات المفتاحية: اشارة تعديل عرض النبضة PWM, *المتحكم التناسبي التكاملي التفاضلي* PID, جسر H , وحدة التحكم ATmega328P, محرك التيار المستمر .

GENERAL INTRODUCTION

GENERAL INTRODUCTION

In the past, speed Controls of dc drives are mostly mechanical and requiring large size hardware to implement.

Speed controllers of DC motor are very useful for controlling the robotic motion and automation systems, PID controllers are most popular and most often used controllers in Industry.

Popularity of the PID controllers are due to their wide range of operating conditions and functional simplicity.

Recent developments in the area of semiconductor technology have made smaller, faster microprocessors and microcontrollers available at reduced cost. The potential use of microprocessors to control some or all electronic functions justifies their use. [1]

The purpose of a motor speed controller is to take a signal representing the required speed, and to drive a motor at that speed.

Through this concept we can control speed of a motor on its running condition. Speed control is a different concept from speed regulation where there is natural change in speed due change in load on the shaft.

Description of research work

The aim of development of this project is towards providing efficient and simple method for control speed of DC motor using pulse width modulation (PWM) technique and PID controller.

The modulation of pulse width is obtained using pulse width generator in ATmega 328p Microcontroller.

Its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as DC motors. [2]

Due to shortcomings of low accuracy and response lag in various methods of the DC motor speed control,

We use The Proportional Integral Differential (PID) controller design and selection of various Proportional, Integral and Differential control parameters according to various system responses.

Hardware implementation requires the use of Infrared (IR) sensors and Arduino Nano for measuring the Rotations per minute (RPM) of the DC motor, measured RPM was displayed on LCD. An H bridge is used in this Application to allow DC motor to run forwards and backwards.

The H-bridge is driven by a high frequency PWM signal. Controlling the PWM duty cycle is equivalent to controlling the motor terminal voltage, which in turn adjusts directly the motor speed?

The desired objective is to achieve a system with the constant speed at any load condition. That means motor will run at a fixed speed instead of varying with amount of load.

The applications of our research could be in conveyors, turntables and others for which adjustable speed and constant or low-speed torque are required.

It also works well in dynamic braking and reversing applications, which are common in many Industrial machines. [3]

This project shows that precise and accurate control of small DC motors can be done efficiently without using costly components and complicated circuit [4]

My report is structured in **four chapters** which reflect the approach I adopted for the development Of this project.

The first chapter "Arduino Nano ", in which I will describe the Arduino Nano pins out and specifications, Arduino code (IDE)

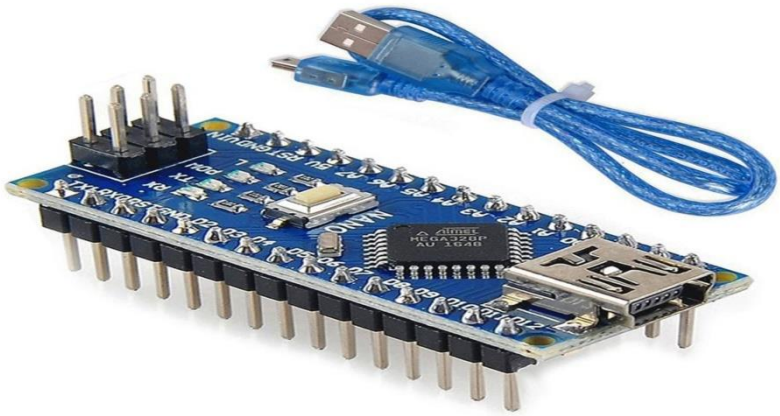
In **the second chapter** "types of control'', there is a brief review on the theoretical part behind the controller, PWM theory and PID controllers, the origins, the adjustment methods, the stability and the limitations

In **the third chapter** "Methodology" includes the principle of work of all the fundamental devices in our project, and after each explanation, a description of the device and circuit that we chose to use will be included

In **the fourth chapter "**Results and discussion", the realized model was validate in laboratory performing some measurements of actuation on the real system , and the DC motor modeling

The project ends with "General conclusion and perspectives" presenting a synthesis of my work, as well as the perspectives allowing to improve this project .

Chapter 1 Arduino Nano



1.1Introduction

This article gives detailed information about an Arduino Nano board, and it is one kind of microcontroller board which is designed by the Arduino team. This microcontroller is based on Atmega168 or Atmega328p. This Nano board has replaced Arduino Uno due to small in size. As we know that while designing an embedded system small size components are preferred. Arduino boards are mainly used to build electronic projects. Embedded systems, robotics, automation, Internet of Things (IoT), Medical Instruments, Virtual Reality Applications Real-Time Face Detection, Arduino Metal Detector, Android Applications [5].

These boards were initially introduced for the students and non-technical users but nowadays Arduino boards are widely used in industrial projects .

1.2 An Overview of Arduino Nano Board

1.2.1 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 [6].

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 [7].

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.[8] . Arduino Uno is the most popular

Fig .1- Types of Arduino Boards

1.2.2 Introduction to Arduino Nano

Arduino Nano is a small complete chip board based on Atmega 328 (v3.0) or Atmega 168 (v2.0). Every Arduino has the same functionality and the same features except the number of pins and size. One of the major flaws of this board is that it doesn't have any power jack. So, you can't supply power from any external power source like a battery. This board is quite similar to any Arduino board [9] developed by Arduino.cc in Italy in 2008 and contains 30 male I/O headers, configured in a DIP30 style.

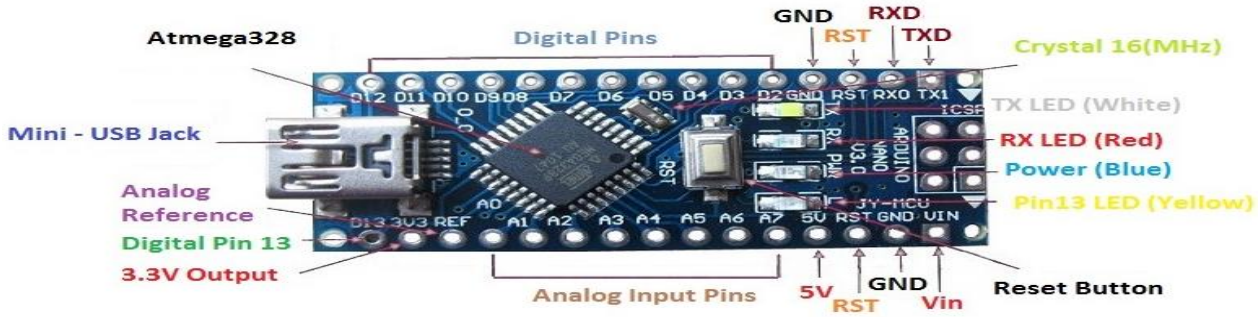


Fig.2 - Layout of Arduino Nano Board

You can use a power source using the Vin Pin, but you need to take a few precautions. Otherwise, it will burn the board. This Nano board is different in packaging. The power supply can be give using a small USB port otherwise straight connecting to the pins like VCC and GND [10].

1.3 Arduino Nano Pinout

- Contains 14 digital pins, 8 analog Pins, 2 Reset Pins and 6 Power Pins.
- The Nano has 8 analog inputs, each of which provide 10 bits of resolution (1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the analog Reference () function. Additionally, some pins have specialized functionality.
- It comes with an operating voltage of 5V, however, the input voltage can vary from 7 to 12V.
- Arduino Nano's maximum current rating is 40mA, so the load attached to its pins shouldn't draw current more than that.
- Each of these Digital and Analog Pins is assigned with multiple functions but their main function is to be configured as Input/output.
- Arduino Pins are acted as Input Pins when they are interfaced with sensors, but if you are driving some load then we need to use them as an Output Pin.
- Functions like pin Mode () and digital Write () are used to control the operations of digital pins while analog Read () is used to control analog pins.
- Arduino Nano comes with a crystal oscillator of frequency 16 MHz it is used to produce a clock of precise frequency using constant voltage.

- This board doesn't use standard USB for connection with a computer, instead, it comes with Type-B Micro USB [11] .

The following figure shows the Pinout diagram of the Arduino Nano board [12].

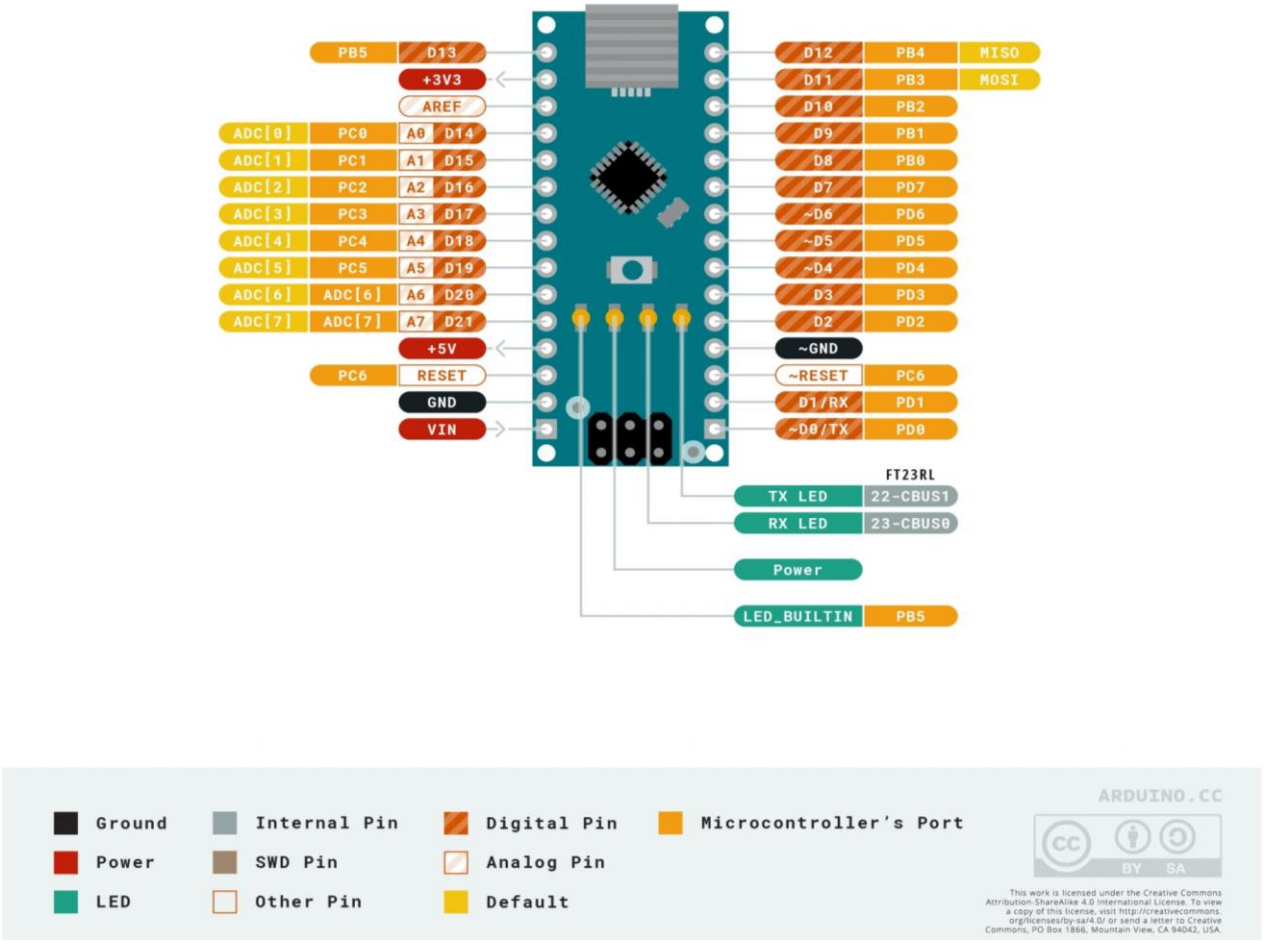


Fig.3 - Arduino NANO Pinout Diagram.

Table .1 - Arduino Nano specifications [12]

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 MA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70". Length: 45 mm Width: 18 mm

1.4 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.

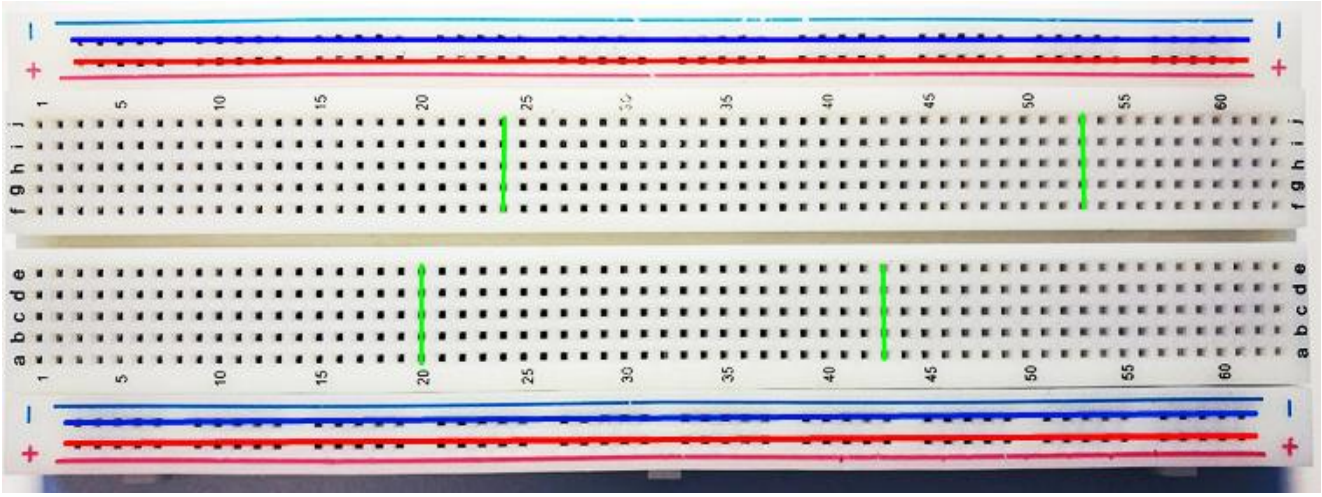


Fig.4 -Breadboard Internal Connections.

1.4.1 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 [13].

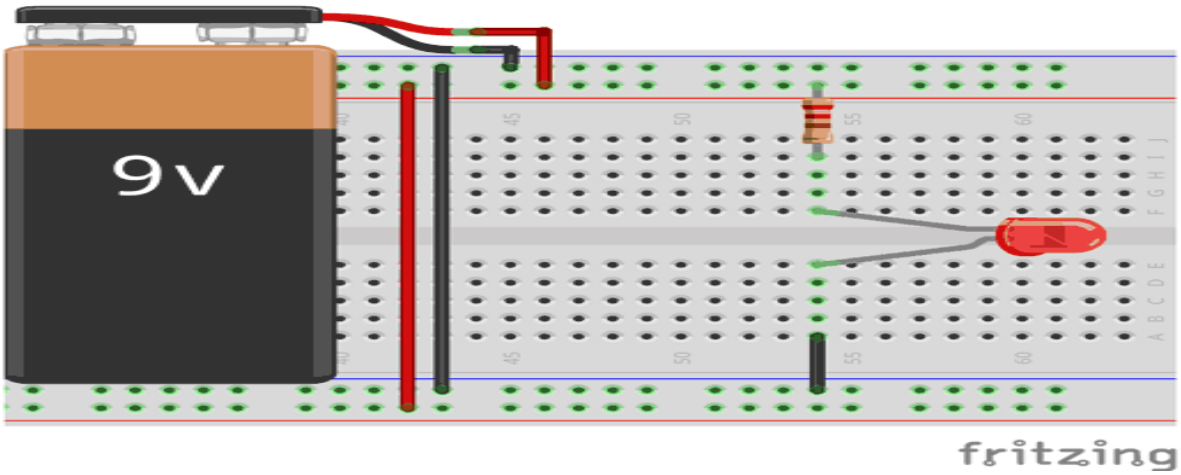


Fig.5 - Simple LED Circuit on Breadboard

1.5 Arduino Nano Pin Description

In this section, we'll cover the Arduino Nano Pinout, we will discuss pin description of each pin integrated on the board.

1.5.1 Arduino Nano Power Pins

VIN: This is an input voltage to the Arduino board when using an external power source (6-12V).

3.3V: It is a minimum voltage produced by the voltage regulator on the board.

5V: Regulated power supply used to power up the controller and other components on board.

GND: Two ground pins are available on the board.

1.5.2 Arduino Nano Function Pins

LED: The unit comes with a built-in LED connected to pin 13 on the board. This LED is used to check the board if it's working fine or not. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

AREF: It is an Analog Reference that is applied to the unit as a reference voltage from an external power supply.

Reset: Two reset pins are integrated on the board. These pins are used to reset the controller internally through software.

1.5.3 Arduino Nano I/O Pins

Digital Pins: There are 14 digital pins on board which is used to connect external component.

Analog Pins: 6 analog pins on board that is used to measure voltage in a range from 0 to 5V.

External Interrupts: Pin 2 and 3 are used to trigger external interrupts. These pins are used in case of emergency, when we need to stop the main program and call important instructions.
The main program resumes once interrupt instruction is called and executed.

PWM Pins: Arduino Nano has 6 PWM pins, which are Pin 3, 5, 6, 9, 10 and 11. (All are digital pins), these pins are used to generate an 8-bit PWM (Pulse Width Modulation) signal.

1.5.4 Arduino Nano Pinout for Communication Protocols

USART: The board supports USART serial communication that carries two pins Rx which is used for receiving the serial data and Tx which is a transmission pin used to transmit serial data.

I2C: The unit comes with an I2C communication protocol where two pins SDA and SCL are used to support this communication. It's developed using A4 and A5 pins, where A4 represents the serial data line (SDA) which carries the data and A5 represents the serial clock line (SCL) used for data synchronization between the devices on the I2C bus. The Wire Library of Arduino Software can be accessed to use the I2C bus.

SPI Protocol: Four pins 10, 11, 12and 13 are used for SPI (Serial Peripheral Interface) Protocol.
SPI is an interface bus and is mainly used to transfer data between microcontrollers and other peripherals like sensors, registers, and SD cards [14].

1.6 Arduino Nano Programming and Communication

- The Nano board comes with the ability to set up communication with other controllers and computers.
- The serial monitor is added to the Arduino IDE, which is used to transmit textual data to or from the board.
- FTDI drivers are also included in the software which behaves as a virtual Com port to the software. The Tx and Rx pins come with an LED which blinks as the data is transmitted between FTDI and USB connection to the computer.
- Arduino Software Serial Library is used for carrying out serial communication between the board and the computer.
- The Arduino Nano is programmed by Arduino Software called IDE which is a common software used for almost all types of board available. Simply download the software and select the board you are using.
- Uploading code to Arduino Nano is quite simple, as there's no need to use any external burner to compile and burn the program into the controller and you can also upload code by using ICSP (In-circuit serial programming header).
- Arduino board software is equally compatible with Windows, Linux or MAC, however, Windows are preferred to use [11].

1.7 Arduino Coding Environment and basic tools

1.7.1 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.

1.7.2 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 [15].

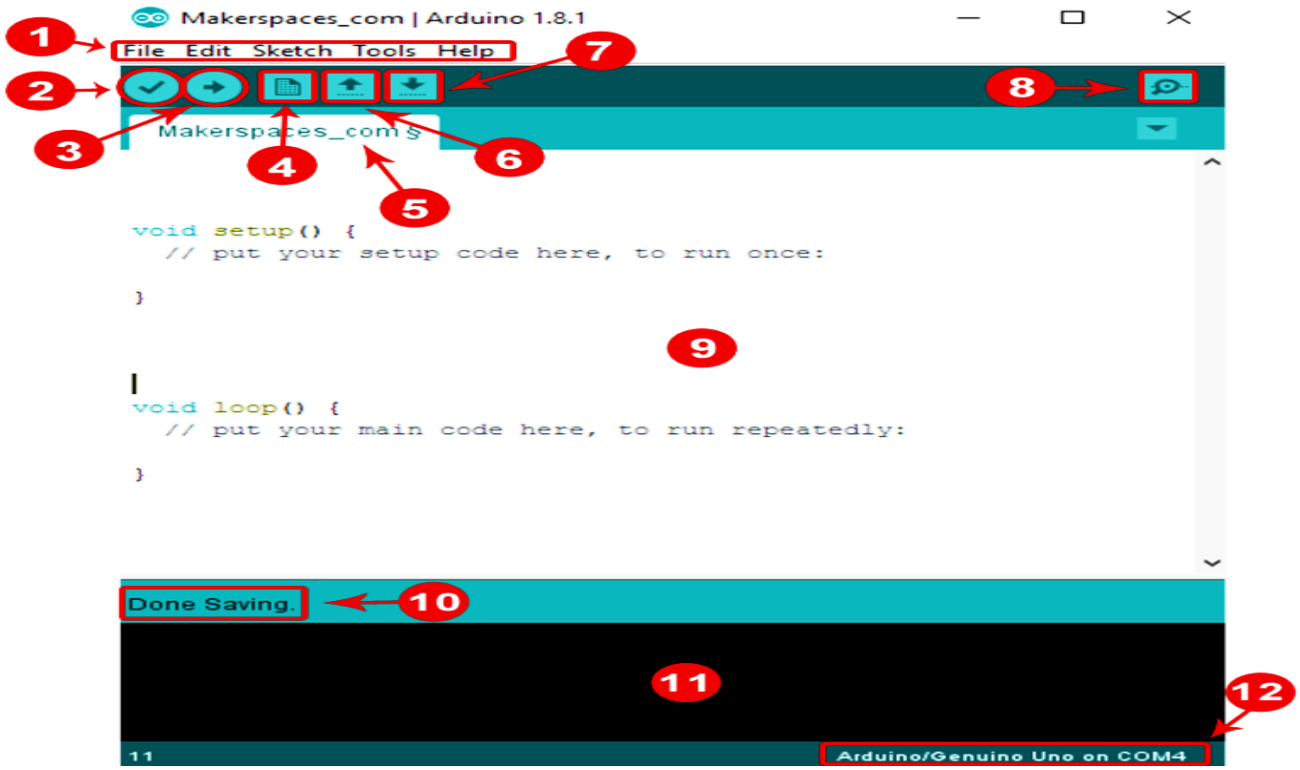


Fig. 6 - Arduino IDE

- Menu Bar: Gives you access to the tools needed for creating and saving Arduino sketches.
- Verify Button: 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 of the sketch is displayed here.
- Open Existing Sketch: Allows you to open a saved sketch or one from the stored examples.
- Save Sketch: This saves the sketch you currently have 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 [16].

1.8 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 [17].

Loop Section:

The main program of the sketch will runs in this section. The Loop () Function will run infinitely

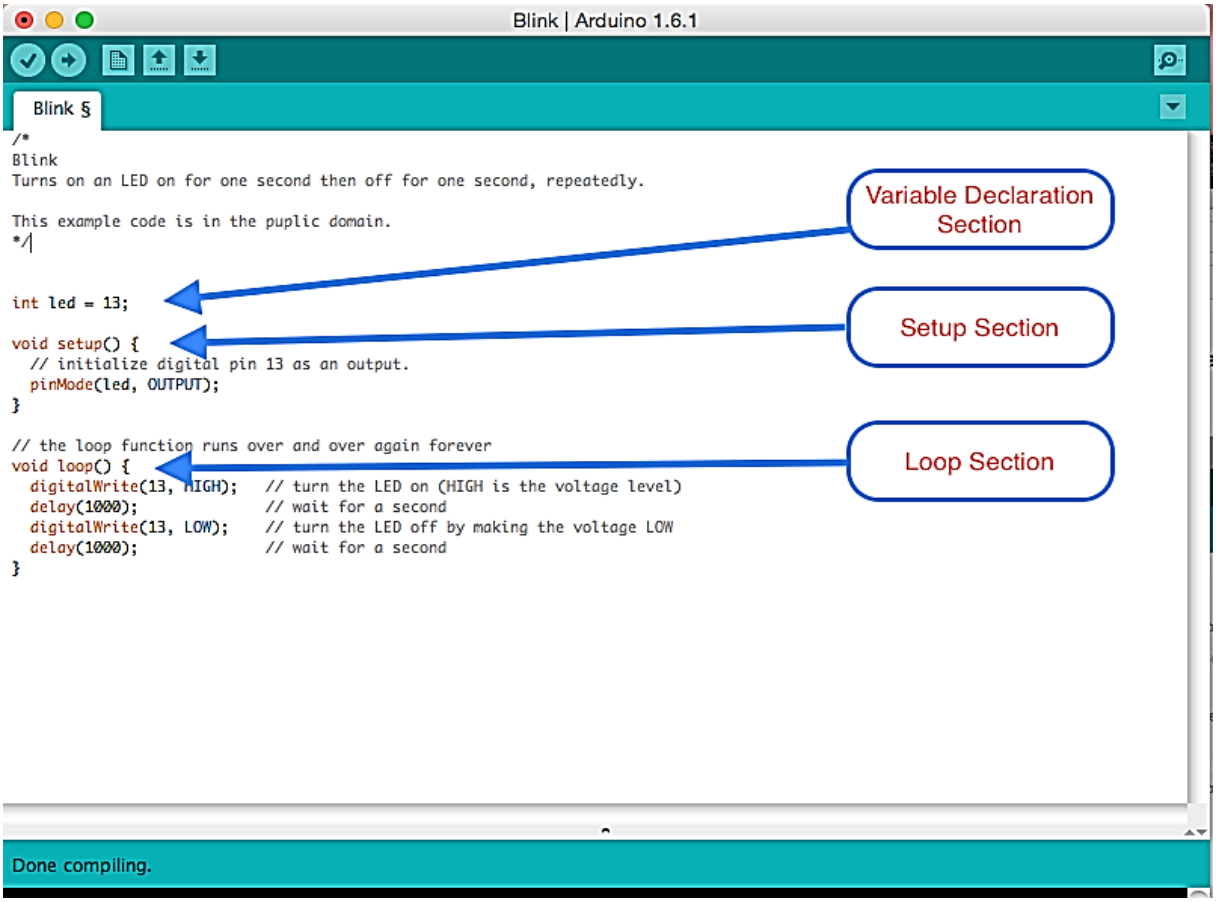


Fig.7 -Arduino Program structure.

1.9 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 [18].

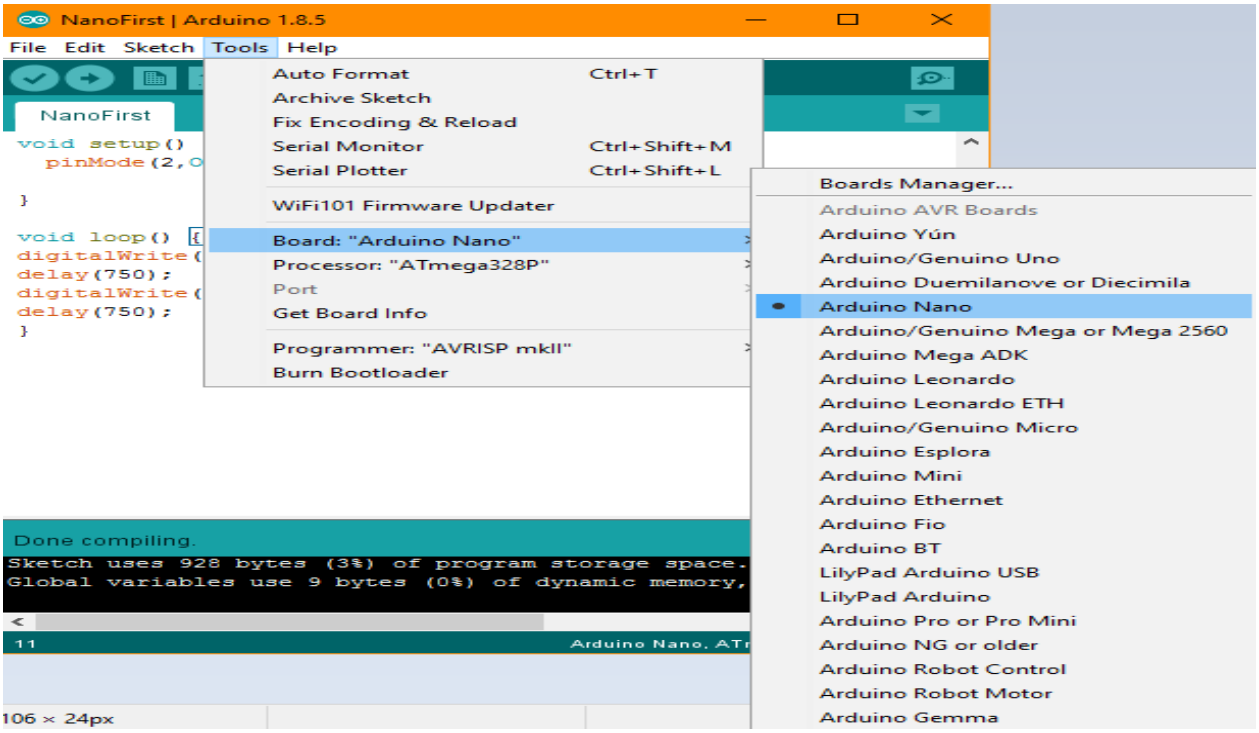


Fig.8- Choose the Correct Arduino Board.

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 how to select the proper processor.

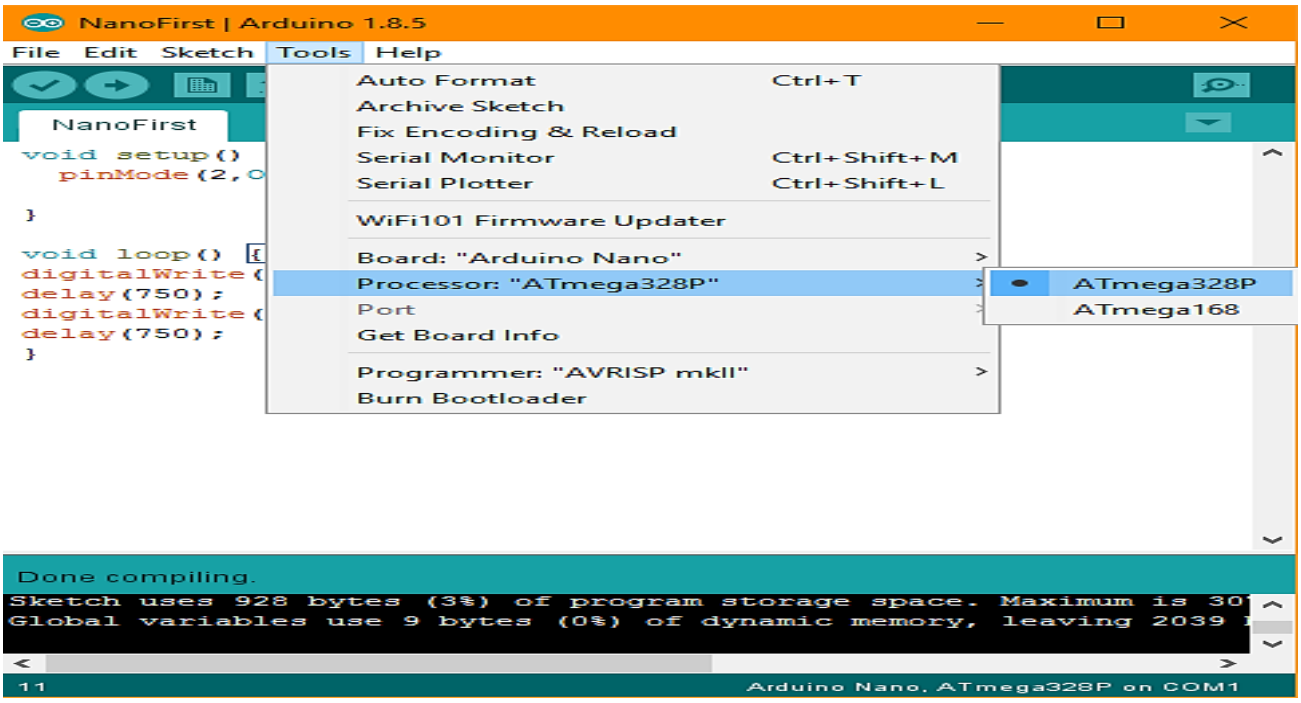


Fig. 9 - Select the right processor for Arduino Nano

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 Nano with an ATMega328P and the device is connected to COM1.

1.10 LED Blinking Using Arduino Nano

1.10.1 Components required

- 1 × Breadboard
- 1 × Arduino Nano v3
- 1 × LED
- 1 × 220Ω Resistor
- 2 × Jumper

1.10.2 Connections

Connect LED's positive end to the one end of resistor and other end to Nano's digital pin D13 and negative end to Nano's ground.

To power Nano board, you can use USB cable or you can also connect an external power supply (5v-12v) by connecting positive pin to VIN and negative to the ground [10].

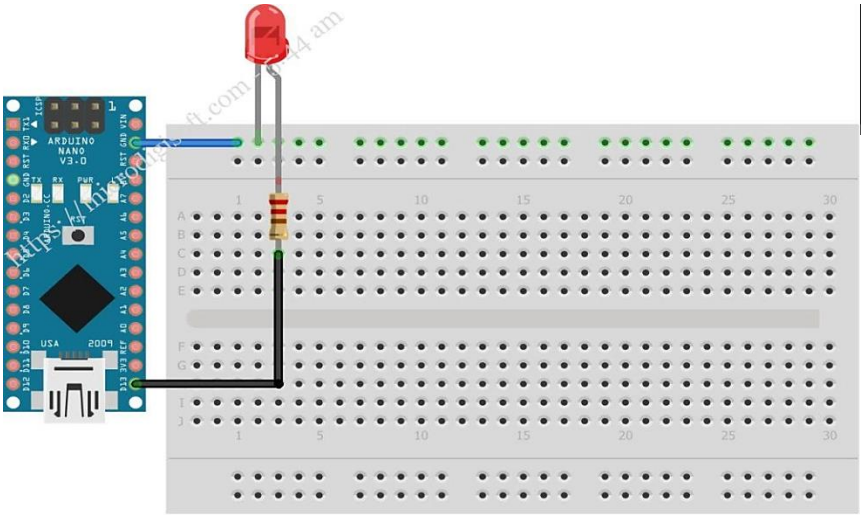


Fig. 10 - LED Blinking Using Arduino Nano

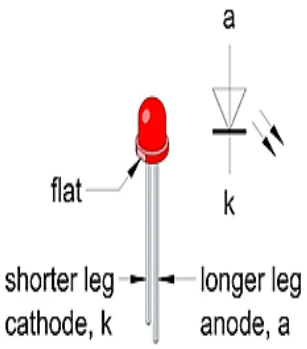


Fig. 11 - led Pinout

1.10.3 Arduino code

Table .2 -Arduino code for LED blinking with 1s interval

```
int led_pin=13;
void setup () {
pinMode(led_pin,OUTPUT);
}
void loop()
{
digitalWrite(led_pin,HIGH);
delay(1000);
digitalWrite(led_pin,LOW);
delay(1000);
}
```

1.10.4 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, and verified and uploaded the code to your board.

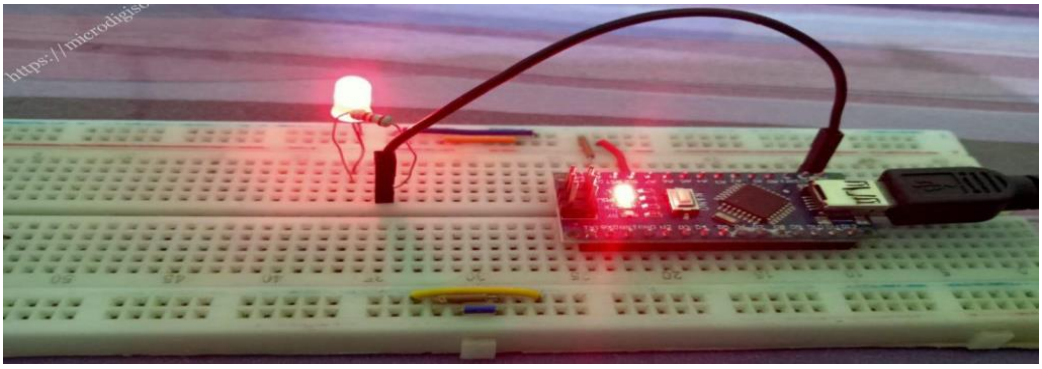


Fig. 12 - LED Blinking

1.11 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 in the following table [19].

Table. 3 -Difference between Arduino UNO and Arduino Nano

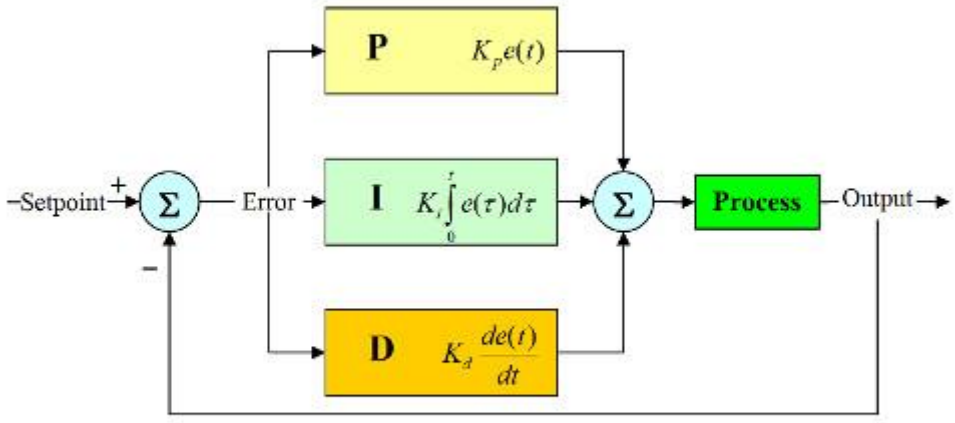
Projectiot123.com

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

1.12 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 [14] .

Chapter 2 Types of control



2-1 PID control

Proportional-Integral-Derivative (PID) is a cornerstone algorithm in control theory. The PID algorithm smoothly and precisely controls a system, such as temperature in an oven or the position of a control

surface on an airplane [20].

A PID controller works by calculating an amount of error based upon the difference between a set value and a feedback value, and provides an adjustment to the output to correct that error. The control and decision of the adjustment is done in math instead of pure logic control such as if...else statements.

PID controllers have many types of uses, including controlling robotics, temperature, speed, and positioning.

2-3 Design of PID Controller

PID controllers have three control coefficients, proportional, integral and derivative.

This control scheme is oftenly implemented in various industrial automation control systems

The PID control scheme achieves the real time speed very closed to reference speed in an efficient way at a faster rate and reduces the problem of steady state error [21];[22] .

PID controllers are generally used to different type of dynamic plants and manage the time domain behavior.

The architecture of PID controller is revealed in Fig.13 The consequence of variation of PID controller coefficients is represented using Tab.4 [23].

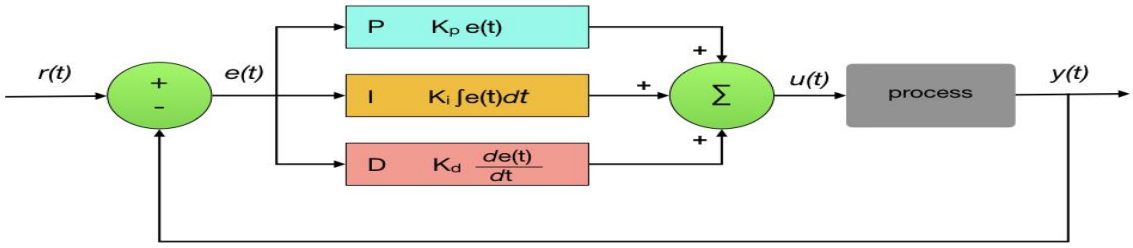


Fig.13- PID block diagram .

Table .4- PID Controller parameter characteristics .

Parameterincrease	Rise time	Overshoot	Settling time	Steady state error
Kp	Decreases	Increases	Small change	Decreases
Ki	Decreases	Increases	Increases	Highly reduced
Kd	Small change	Decreases	Decreases	Small change

2-4 Mathematical form

The error signal e (t) is the input to the PID controller

$$u = k_p e + k_i \int e\,dt + k_d \frac{de}{dt} \quad \text{.....(1)}$$

where,

Kp -→ Proportional Gain,

Ki -→ integral Gain,

Kd -→derivative Gain

PID controller functions with closed-loop system. Variable (e) specifies tracking error, difference among desired output (r) and actual output (y). Error signal (e) is provided to PID controller, and controller evaluates derivative and integral of error signal with time [24], then processed to calculate a new process input. This input will try to adjust the measured process value back to the desired set point. [25]

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.

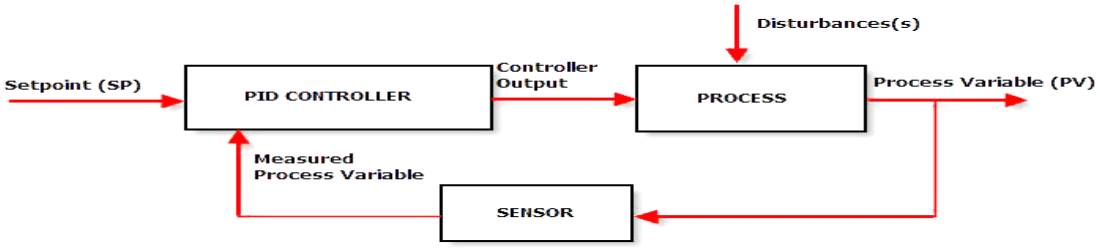


Fig.14- Closed Loop System with PID Controller .

The basic idea is that the controller reads the system state by a sensor then it subtracts the measurement from a desired reference to generate the error value. [26] The error will be managed in three ways, to:

1. Handle the present, through the proportional term
2. Recover from the past, using the integral term
3. Anticipate the future, through the derivative term.

2-5 Different PID Controllers

- Proportional-Integral-Derivative (PID) controller is the most common type of feedback control. The PID controller uses all the three components P, I, and D.
- A proportional (P) controller uses only proportional action to correct deviations from desired parameter value while acting on the error signal generated by the controller output.
- Proportional-Integral (PI) controller also uses the integral action to maintain closed-loop performance for extended time periods at steady-state conditions. As a result, it is very stable, but it may cause system overshoot due to its delayed corrective action.
- Proportional-Derivative(PD) controller uses the derivative action to correct deviations from desired parameter value measured by the rate of change of process variable, i.e., error signal generated by the controller output [27].

2-6 Manual tuning

- The most common way of tuning your PID controller is to use a slow manual approach. This means that you'll have to look at the results from your system and find a P gain, I gain, and D gain, which let it work as expected
- This process can be very time-consuming, making this approach less useful for more complex applications with high demands for control!
- A fast PID loop tuning usually overshoots slightly to reach the setpoint quickly; however, some systems cannot accept overshoot.

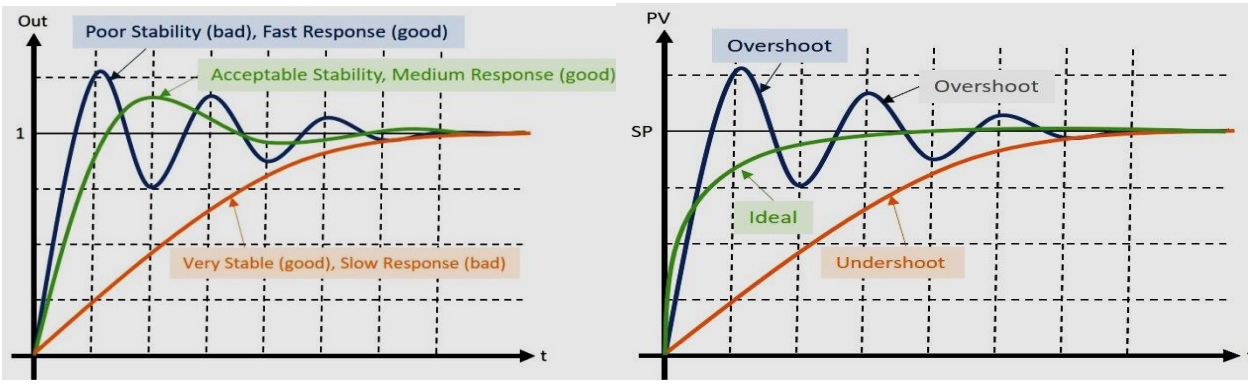


Fig.15- Behavior and comments in the response time of the PID control system

2-7 Manual tuning procedure

1. start with a low P gain, usually 1.
2. increase P until oscillations occur
3. the amplitude of the oscillations is proportional to the error in your system – so it's useful for finding out how good your controller is.
4. then the P should be set to approximately half of that value
5. increase I by a small amount and repeat step 2. Keep doing this until you can't find any more improvements
6. increase D by a small amount and repeat step 2. Keep doing this until you can't find any more improvements and the loop is acceptably quick to reach its reference after a load disturbance

2-8 Conclusion

This paper reports Performance Analysis of PID Controller Parameters, we concluded that for designing of a PID controller for any system, require some steps to obtain a desired response few things are to be added like proportional gain (Kp) to improve the rise time, a derivative gain (Kd) to improve the overshoot, internal gain (Ki) for eliminate the steady state error. Kp, Ki and Kd are being analyzed and optimized until an desired overall response is obtain.

At last also conclude that do not need to implement all three controller (proportional, derivative and integral) into a single system, if not necessary .

2-9 PWM theory

PWM stands for Pulse Width Modulation and it is a technique used in controlling the brightness of LED, speed control of DC motor, controlling a servo motor or where you have to get analog output with digital means.

The Arduino digital pins either gives us 5V (when turned HIGH) or 0V (when turned LOW) and the output is a square wave signal. So if we want to dim a LED, we cannot get the voltage between 0 and 5V from the digital pin but we can change the ON and OFF time of the signal. If we will change the ON and OFF time fast enough then the brightness of the led will be changed

Before going further, let's discuss some terms associated with PWM. [28].

TON (On Time): It is the time when the signal is high.

TOFF (Off Time): It is the time when the signal is low.

Period: It is the sum of on time and off time.

Duty Cycle: It is the percentage of time when the signal was high during the time of period.

$$\boldsymbol{duty\ cycle = \frac{Ton}{Ton+Toff} * 100 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)}$$

So at 50% duty cycle and 1Hz frequency, the led will be high for half a second and will be low for the other half second. If we increase the frequency to 50Hz (50 times ON and OFF per second), then the led will be seen glowing at half brightness by the human eye.

The power applied to the motor can be controlled by varying the width of these applied pulses and thereby varying the average DC voltage applied to the motors terminals. By changing or modulating the timing of these pulses the speed of the motor can be controlled, ie, the longer the pulse is "ON", the faster the motor will rotate and likewise, the shorter the pulse is "ON" the slower the motor will rotate.

When the duty cycle is 0%, the motor will stop completely because there is no voltage difference. When the duty cycle is 50%, the motor will rotate at half the speed of the maximum speed because the voltage is half the full voltage. When PWM is in 100% condition, the motor rotates with maximum speed because of the continuous output of PWM.

In other words, the wider the pulse width, the more average voltage applied to the motor terminals, the stronger the magnetic flux inside the armature windings and the faster the motor will rotate and this is shown below [29].

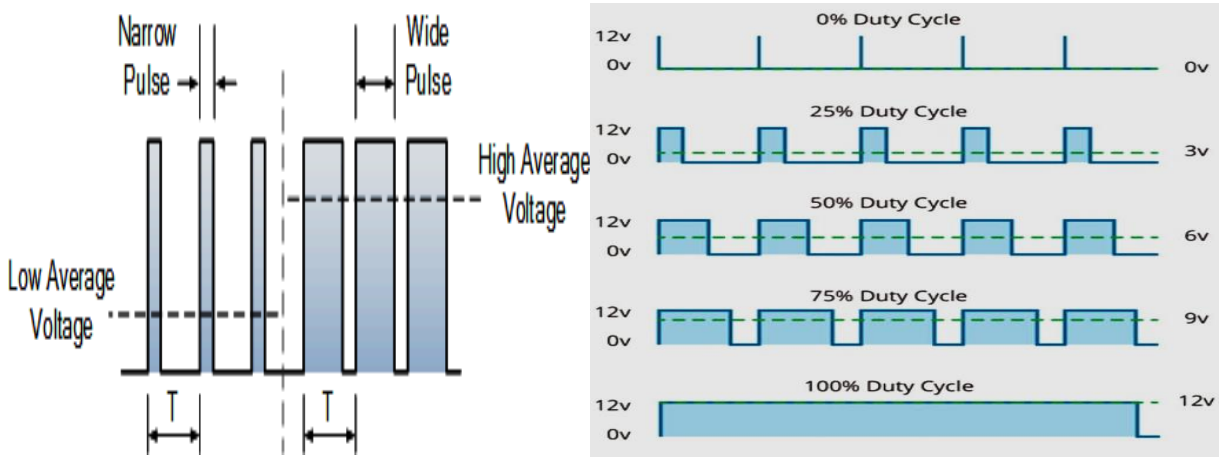


Fig. 16- Nomenclature for de_nition of PWM duty cycle

2-10 PWM Generation

In Arduino there are lots of pins where we can generate PWM's. They are indicated by '~' symbol. Now the PWM generated at pins 5,6 are of 980Hz. While other pins generate PWM frequency are 490Hz. The specifications are for Arduino Nano boards. Now the Arduino compiler have inbuilt function named as analogWrite(). To generate PWM using arduino two parameters are passed during this function call [30].

It is used like this analogWrite(pin, speed). analogWrite(10, 255). Generate PWM at 100% duty cycle at pin no 10.

$u = 5V \rightarrow analogWrite\ 255$

$u = xV \rightarrow analogWrite\ 51 * x$

Consider the below image:

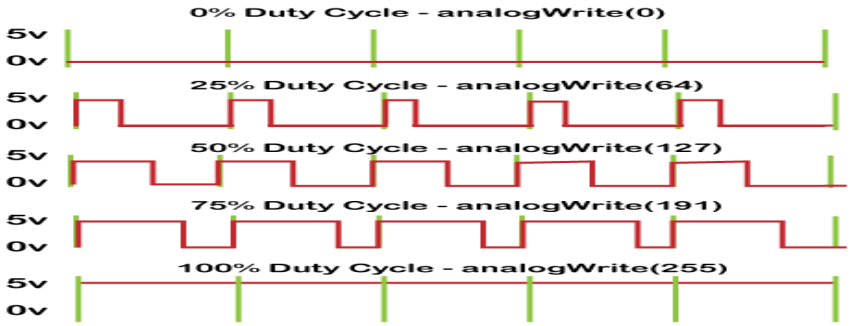
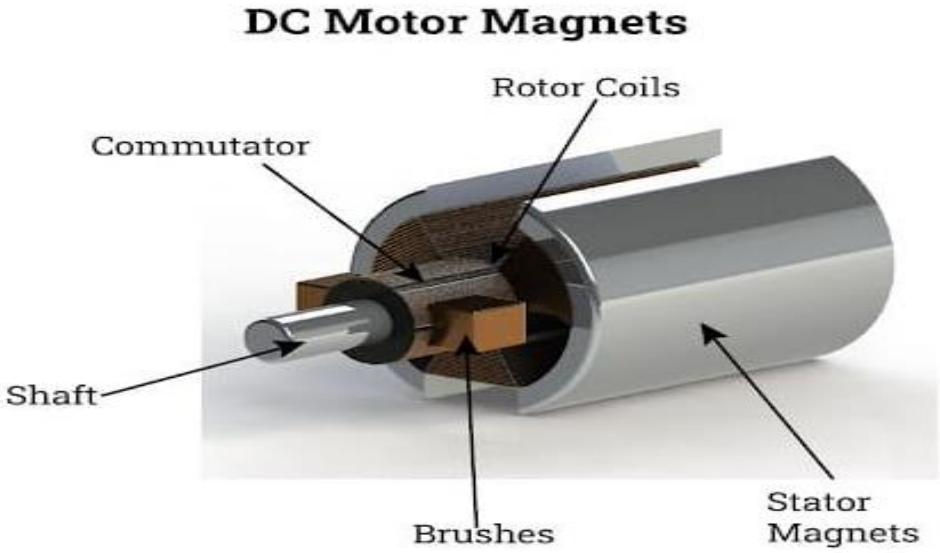


Fig.17- Pulse width Modulation

2.11 Conclusion

Switch mode controllers are favored in applications like audio power amplifiers, inverters and motors due to their high efficiency and low system design cost. The active components in a switching converter are controlled by PWM signals applied to their gates. The duration for which the switches remain on is determined by the duty cycle of the PWM signals, which in turn controls the energy delivered to the load [31]. The major share of the energy received by the load will be a function of the modulation only if the switching frequency is ensured to be much higher than the modulating signal frequency.

Chapter3 Methodology



3.1 Abstract

In this project we used the PID in closed-loop system, we need a feedback from DC motor. Therefore, to use PID control, DC motor need to has an encoder that will output the signal, which is used to calculated the speed (called feedback value).

Library on Arduino will perform adjustment based on the feedback value, desired value, Kp, Ki and Kd gain, and staling factor. after adjusting, Arduino send command along with PWM duty-cycle value to H-Bridge to control DC motor speed . This process is repeated in a infinite loop.

3.2 Hardware

The fig.18 shows the scheme. It is composed of the following components:

• DC motor: to operate the bench and to transform an input voltage into rotation speed

• Encoder: sensor required for speed, to close the control loop

• H-bridge: allows to electronically control both the speed and the direction of rotation of a DC motor

• Arduino: Hardware platform consisting of a series of elctronic boards equipped with a microcontroller.

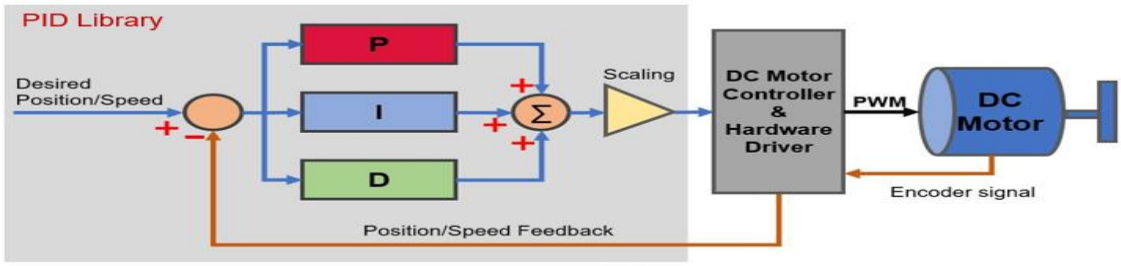


Fig.18-Test bench diagram

3.2.1 DC motor

3.2.1.1 What is a DC motor?

A DC motor is defined as a class of electrical motors that convert direct current electrical energy into mechanical energy. From the above definition, we can conclude that any electric motor that is operated using direct current is called a DC motor. We will understand the DC motor construction and how converts the supplied DC electrical energy into mechanical energy in the next few sections [32].

3.2.1.2 Different Parts of a DC motor

A DC motor is composed of the following main parts

- Armature or Rotor

The armature of a DC motor is a cylinder of magnetic laminations that are insulated from one another. The armature is perpendicular to the axis of the cylinder. The armature is a rotating part that rotates on its axis and is separated from the field coil by an air gap.

- Field Coil or Stator

A DC motor field coil is a non-moving part on which winding is wound to produce a magnetic field. This electro-magnet has a cylindrical cavity between its poles.

- Commutator

The commutator of a DC motor is a cylindrical structure that is made of copper segments stacked together but insulated from each other using mica. The primary function of a commutator is to supply electrical current to the armature winding.

- Brushes

The brushes of a DC motor are made with graphite and carbon structure. These brushes conduct electric current from the external circuit to the rotating commutator. Hence, we come to understand that the commutator and the brush unit are concerned with transmitting the power from the static electrical circuit to the mechanically rotating region or the rotor.

3.2.1.3 DC Motor Diagram

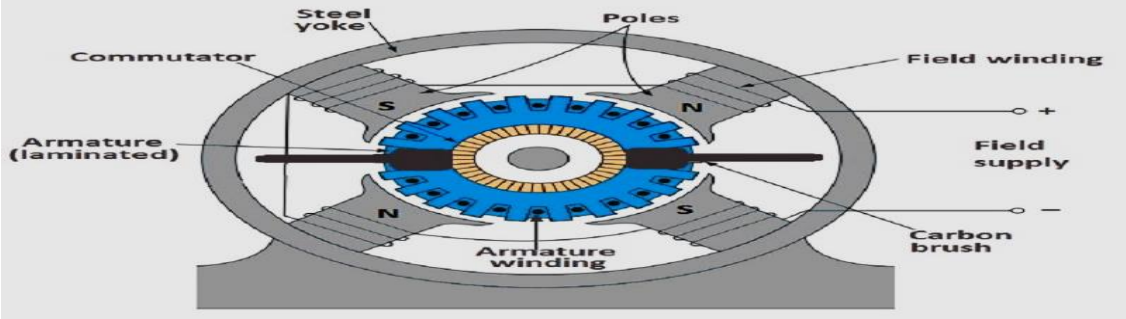


Fig. 19 - DC motor construction parts .

3.2.1.4 DC Motor Working

The basic principle of DC motor operation is that whenever a current-carrying conductor is brought into the magnetic field, it experiences a mechanical force.

Fleming's left-hand rule and its magnitude decide the direction of this force.

Fig.20 -Fleming's left hand rule

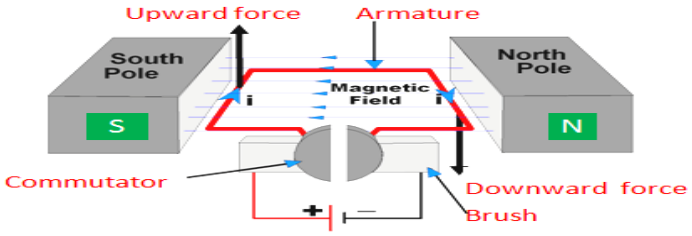


Fig.21- Production of torque in a DC motor

When armature winding is connected to a DC supply, an electric current sets up in the winding. Permanent magnets or field winding (electromagnetism) provides the magnetic field. In this case, current carrying armature conductors experience a force due to the magnetic field, according to the principle stated above. Fleming's left-hand rule decide the direction of this force.

The Commutator is made segmented to achieve unidirectional torque. Otherwise, the direction of force would have reversed every time when the direction of movement of the conductor is reversed in the magnetic field. [33]

In our project we used EG-530AD-6B (CCW) and it is shown in fig. 22

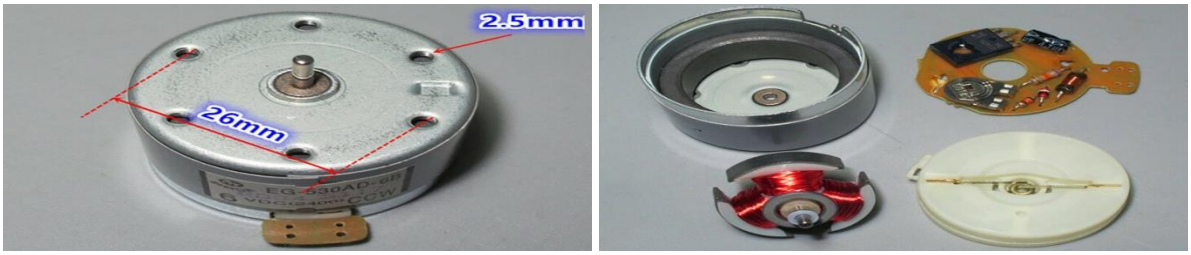


Fig . 22- EG-530AD-6B Spindle DC Motor .

3.2.1.5 Specifications [34]

Table .5- Specifications of the DC motor used in the experiments

1.Model		EG-530AD-6B
2.Rated voltage DC	6	V
3.Direction of rotation	CCW	
4.Speed adjustment	2400	RPM
5.Motor diameter	33	mm
6.Mounting holes	26	mm
7.Axial length	10	mm
8.Motor thickness	25	mm
9.The shaft diameter	2	mm
10. Current	0.132	Amps
11. Torque	8	g-cm

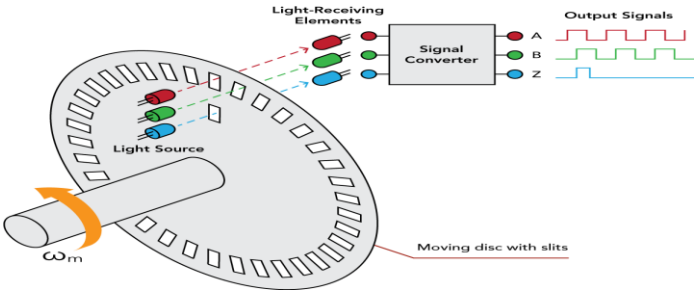
3.2.2 Encoder

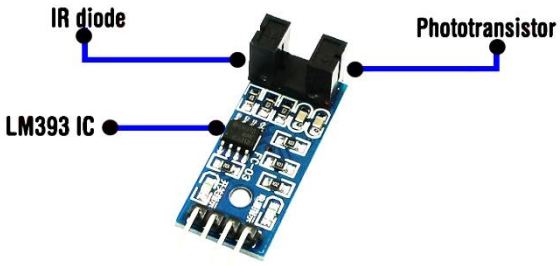
3.2.2.1 Encoder working principale

The encoder is an electromechanical device that can measure displacement**,** 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, 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.

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 (IR diode) 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 [35]





The rotor disc of the sensor The stator of the sensor

Fig.23- The principle of an incremental encoder

Commonly, the incremental encoder uses two output channels (A and B) to detect angular position. Using two code tracks with sectors positioned 90° out of phase, the two output channels of the quadrature encoder indicate both position and direction of rotation. If A leads B, for example, the disk is rotating in a clockwise direction (i.e. quadrature leading pulse). If B leads A, then the disk is rotating in a counter-clockwise direction (i.e. quadrature lagging pulse). Therefore, by monitoring both the number of pulses and the relative phase of signals A and B, you can track both the relative position and direction of rotation.[36]

The basic operation of this sensor is as follows; If anything is passed between the sensor slot, it creates a digital pulse on the D0 pin. This pulse goes from 0V to 5V and is a digital signal. Then with Arduino we can read this pulse.[37]

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.

Here are the different parts of the encoder:

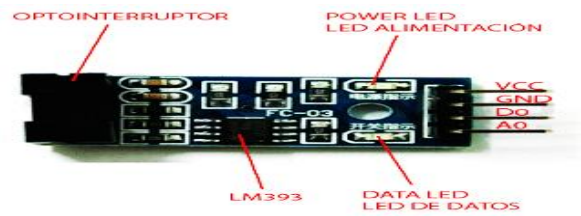


Fig .24 -Main parts of the encoder .

3.2.2.2 Connecting pins of the speed module

VCC: Module power supply from 3.3V to 12V.

GND: Ground.

D 0: Digital signal of the output pulses.

A 0: Analog signal of the output pulses. Output signal in real time. (Usually not used).

3.2.2.3 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 [38]

3.2.3 H-Bridge:

3.2.3.1 What Is an H-Bridge?

H-Bridge is one of the most important parts of DC motor control circuit. It is a device that enable you to change the direction of the motor spin by altering the direction of current flow in the motor. The device is simply a circuit that has 4 switches connected in series loop and a motor that is connected across the series, which forms an "H-Shaped" circuit , hence the name H-Bridge.

3.2.3.2 H-Bridge concept

The circuit shown here is a typical four transistor H Bridge. The diodes D1 to D4 provide a safer path for the back emf from the motor to dissipate and thus it protects the corresponding bipolar transistors from damage. Resistors R1 to R4 limit the base current of the corresponding transistors. Working of this circuit is very easy to understand. When terminal D is grounded and A is pulled to +Vcc, transistors Q1 and Q4 will be on and current passes through the motor from left to right. When terminal B is grounded and C is pulled to

+Vcc, transistors Q3 and Q2 will be on and current passes through the motor from right to right making the motor to rotate in the opposite direction.[39]

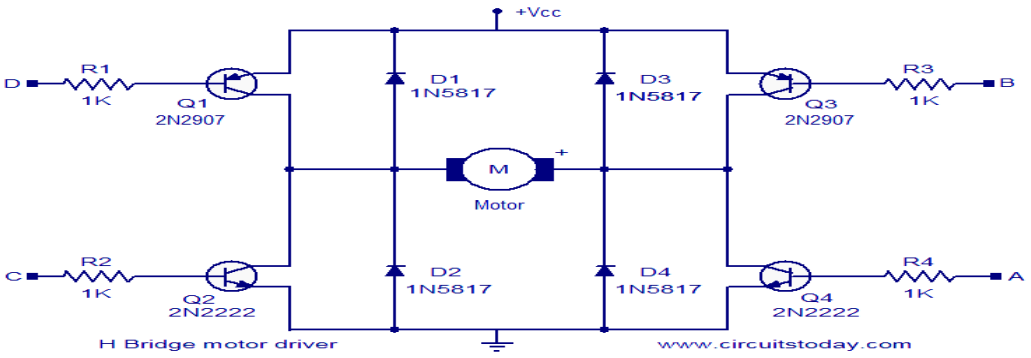


Fig.25-H-Bridge Circuit design using Transistors.

As you see we have used two types of transistor. The top two transistors are PNP transistors and the Bottom two transistors are NPN transistors. We know that when we applied current to the base terminal of the transistor it will conduct. So we can give supply to the base using Microcontroller to run the circuit.

Generally, the PWM signal is given to control the motor using H-Bridge circuit. [40]

3.2.4 Transistor

3.2.4.1 What is a Transistor?

Transistor is symmetrical to a vacuum triode and relatively very small in size. A transistor consists of three layers of semi-conductor material with two junctions and each layer is having the capability of transferring current to the other layers.

This transistor consisting of two n-type and one p-type layers of material or two p-type and one n-type layers of material. First type is called an NPN transistor, while the other is called a PNP transistor respectively.

Germanium and silicon are most preferable semiconductor materials which conducts electricity in semi energetic way. By the process of doping to the semi-conductor material, the result adds additional electrons to the material or produce holes in the material. [41]

Transistor comes with three terminals called emitter, base, and collector which are used for the external connection with electronic circuits.

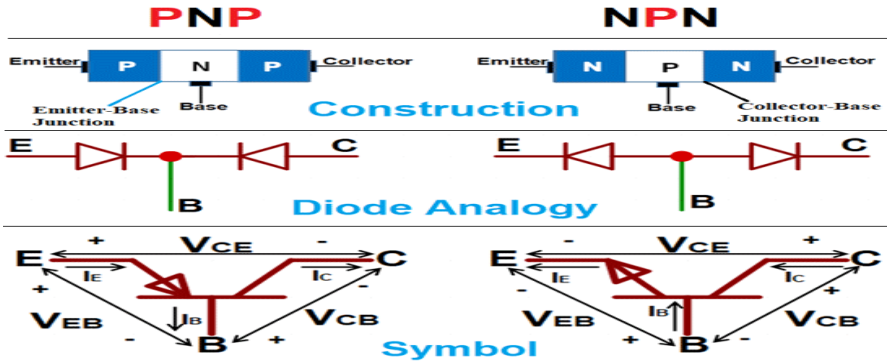


Fig.26-Transistor diagram and parts .

3.2.4.2 Transistor as a Switch

A transistor can be used for switching operation for opening or closing of a circuit. This type solid state switching offers significant reliability and lower cost when compared to conventional relays.

Both NPN and PNP transistors can be used as switches.

- NPN Transistor as a Switch

Based on the voltage applied at the base terminal of a transistor switching operation is performed. When a sufficient voltage (V_{IN} > 0.7 V) is applied between the base and emitter, collector to emitter voltage is approximately equal to 0. Therefore, the transistor acts as a short circuit.

The collector current V_{CC} / R_C flows through the transistor. Similarly, when no voltage is applied at the input, transistor operates in cutoff region and acts as an open circuit.

In this type of switching connection, load (LED) is connected to the switching output with a reference point. Thus, when the transistor is turned ON, current will flow from source to ground through the load.

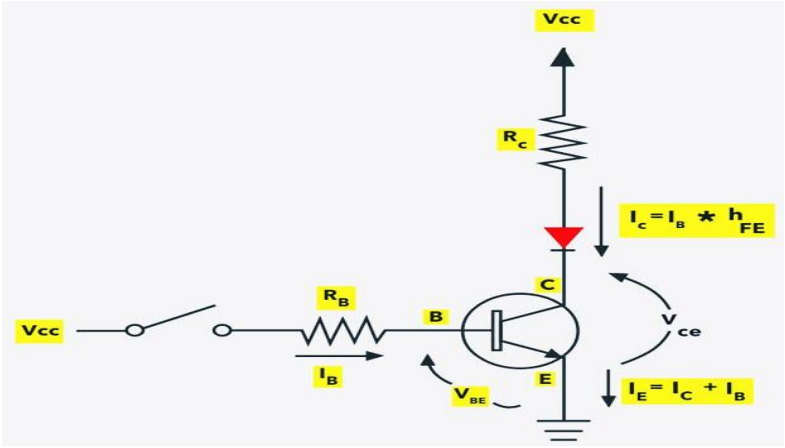


Fig.27- NPN transistor as switch .

- PNP Transistor as a Switch

PNP transistor works same as NPN for a switching operation, but the current flows from the base. This type of switching is used for negative ground configurations. For the PNP transistor, the base terminal is always negatively biased with respect to the emitter.

In this switching, base current flows when the base voltage is more negative. Simply, a low voltage or more negative voltage makes the transistor to short circuit otherwise, it will be open circuit .[41]

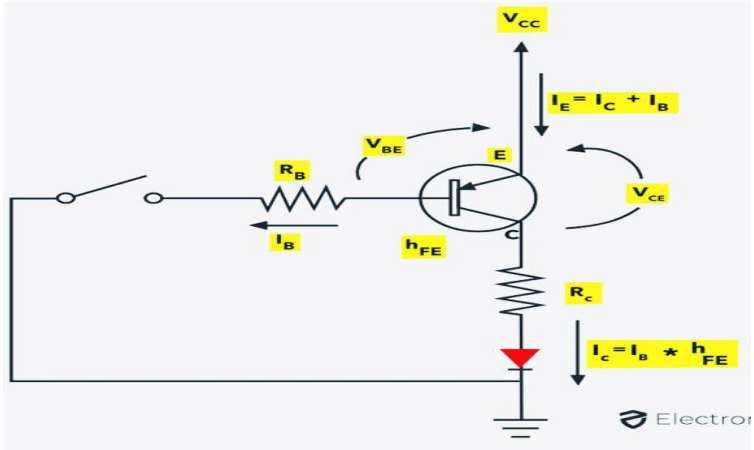


Fig.28- PNP transistor as switch .

3.2.4.3 BC557 PNP Transistor

BC557 is a transistor with general-purpose properties that is used as an amplifier or switching device in circuits that use electronic components. The hFE ratings for the transistor range between 125 and 800 make it ideal for being used as an amplifier in electronic circuits such as audio signal amplifying.

BC557 Pinout

Below is the BC557 Pinout:

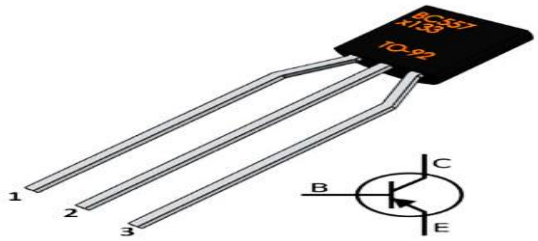


Fig.29-BC557 Pinout

It's a 3-pin silicon chip in which pin 1 serves as a collector where current flows in through. Pin 2 is the base that controls the biasing of the transistor. Pins 3 is the emitter where current drains out through.[42]

Table .6 - Features of BC557 Transistor . [43]

Type:	PNP
Collector-Emitter Voltage	-45 V
Collector-Base Voltage	-50 V
Emitter-Base Voltage	-5 V
Collector Current	-0.1 A
Collector Dissipation	0.5 W
DC Current Gain (hfe)	110 to 800
Transition Frequency	150 MHz
Noise Figure	2 dB
Operating and Storage Junction Temperature Range	-65 to +150 °C
Package	TO-92

3.2.4.4 BC547 NPN Transistor

BC547 is a general purpose BJT NPN transistor mostly used in electronics hobbyists and educational electronics projects. Besides these uses it can also be used in commercial circuits. It comes in TO-92 packaging and the maximum output current this transistor can handle is 100mA. The transistor is having very good DC current gain and low noise capabilities due to which it is ideal to use in signal amplification stages. The typical saturation voltage is only 90 millivolts which is also a good sign to use it as a switch.[44]

Pinout of BC547:

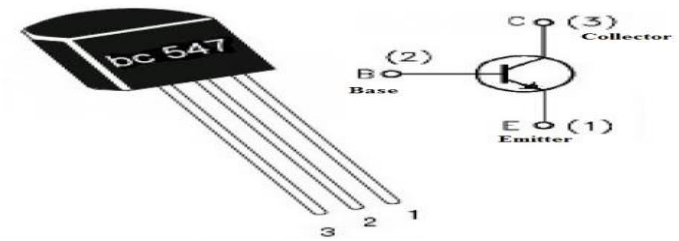


Fig .30 - Pinout of BC547 .

Table .7- Pinout of BC547.

Pin Number	Pin Name	Description
1	Collector	This pin act as an inlet as the current enters the transistor from here. The collector is denoted by 'C'.
2	Base	This pin controls the transistor biasing. The base is denoted by 'B'.
3	Emitter	This pin act as an outlet and the current comes out of the transistor from here. The emitter is denoted by ' E'.

Table.8- Features / technical specifications

Package Type	TO-92
Transistor Type	NPN
Max Collector Current(IC)	100mA
Max Collector-Emitter Voltage (VCE)	45V
Max Collector-Base Voltage (VCB)	50V
Max Emitter-Base Voltage (VEBO)	6V
Max Collector Dissipation (Pc)	500 miliWatt
Max Transition Frequency (fT)	300 MHz
Minimum & Maximum DC Current Gain (hFE)	110 – 800
Max Storage & Operating temperature Should Be	-65 to +150 Centigrade

3.2.5 I2C LCD display

3.2.5.1 What is an I2C LCD display?

A typical I2C LCD display consists of a HD44780 based character LCD display and an I2C LCD adapter. These LCDs are ideal for displaying text/characters only. A 16×2 character LCD, for example, has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row.

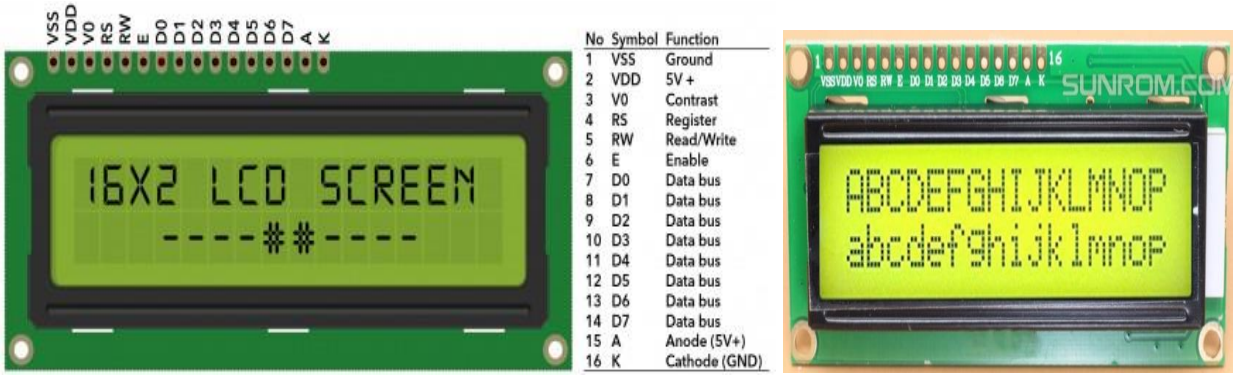


Fig.31-Green 16x2 LCD Display 5V .

3.2.5.2 I2C LCD Adapter

The main component of the adapter is an 8-Bit I/O Expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required by the LCD display.

The board also comes with a small trim-pot to make fine adjustments to the contrast of the display. In addition, there is a jumper on the board that supplies power to the backlight. To control the intensity of the backlight, you can remove the jumper and apply an external voltage to the header pin that is marked as LED

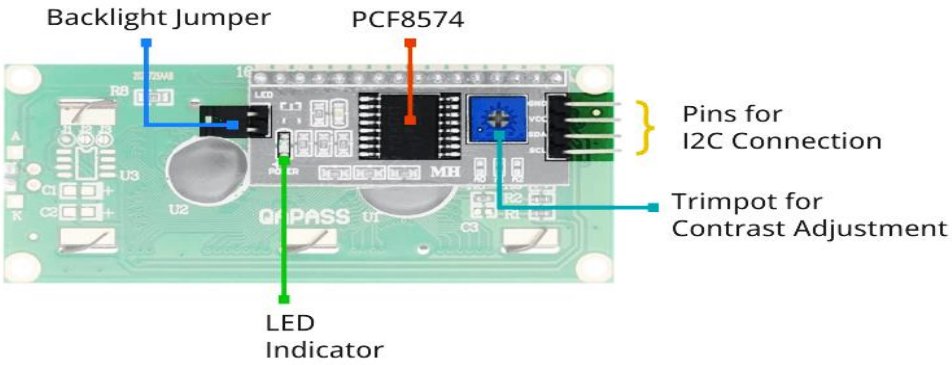


Fig.32-Serial I2C LCD Display Adapter

3.2.5.3 I2C LCD display Pinout

An I2C LCD has only 4 pins that interface it to the outside world. The connections are as follows:

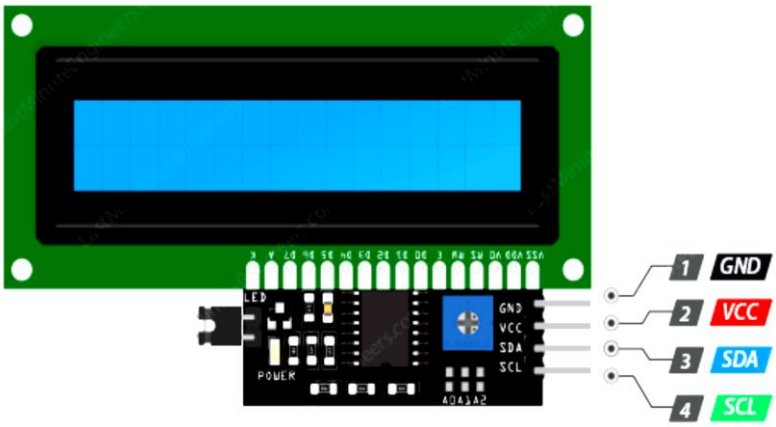


Fig.33-I2C LCD display Pinout .

GND is a ground pin and should be connected to the ground of Arduino.

VCC supplies power to the module and the LCD. Connect it to the 5V output of the Arduino or a separate power supply.

SDA is a Serial Data pin. This line is used for both transmit and receive. Connect to the analog pin A4 on the Arduino.

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

Chapter 4 Results and discussion

4 DC Motor Speed: System Modeling

4.1 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.

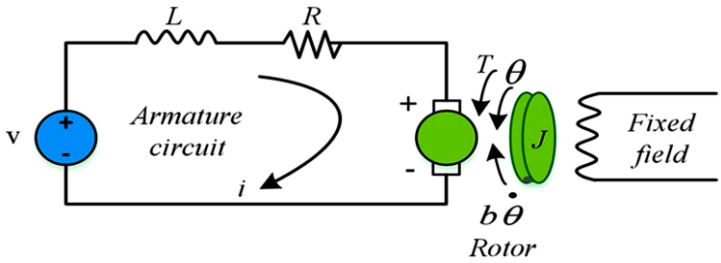


Fig.34-Schematic diagram of a DC Motor.

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 $\dot{\theta}$. 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.

Table .9- The physical parameters for our example

(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

4.2 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 torque is proportional to only the armature current ***i*** by a constant factor $\boldsymbol{k_t}$ as shown in the equation below. This is referred to as an armature-controlled motor.

$$T = k_t\, i \qquad \text{.....(3)}$$

The back emf e, is proportional to the angular velocity of the shaft by a constant factor k_e.

$$e = k_e \dot{\theta} \qquad \text{…… (4)}$$

In SI units, the motor torque and back emf constants are equal, that is, $\boldsymbol{k_t = k_e}$ therefore, we will use $\boldsymbol{k}$ 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's voltage law.

$J\ddot{\theta} + b\,\dot{\theta} = k\,i$ ……..*(5)*

$$L\,\frac{di}{dt} + Ri = V - k\dot{\theta} \quad \text{……(6)}$$

4.3 Transfer Function

Applying the Laplace transform, the above modeling equations can be expressed in terms of the Laplace variable *s*.

$s(J\,s + b)\,\theta(s) = K\,I(s)$ ……..*(7)*

$(Ls + R)I(s) = V(s) - Ks\,\boldsymbol{\theta}(s)$ ……..*(8)*

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.

$$P(s) = \frac{\dot{\theta}(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2} \quad \left[\frac{rad/sec}{V}\right] \quad \text{…….(9)}$$

4.4 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 chosen as the output.

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}\frac{-b}{J} & \frac{K}{J}\\ \frac{-K}{L} & \frac{-R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V \quad \text{……..(10)}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} \text{.....(11)}$$

4.5 Design requirements

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 as it 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 response with overshoot of less than 5%.

4.6 MATLAB representation

4.6.1. Transfer Function

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

```
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)
```

```
P motor=
          0.01
 ----------------------------
 0.005 s^2 + 0.06 s + 0.1001
```

4.6.2. 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 above state-space model can also be generated by converting your existing transfer function model into state-space form. This is again accomplished with the ss command as shown below

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

Now let's see how the original open-loop system performs. Add the following linearSystemAnalyzer command onto the end of the m-file and run it in the MATLAB command window [45] .

```
linearSystemAnalyzer('step', P_motor, 0:0.1:5);
```

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,

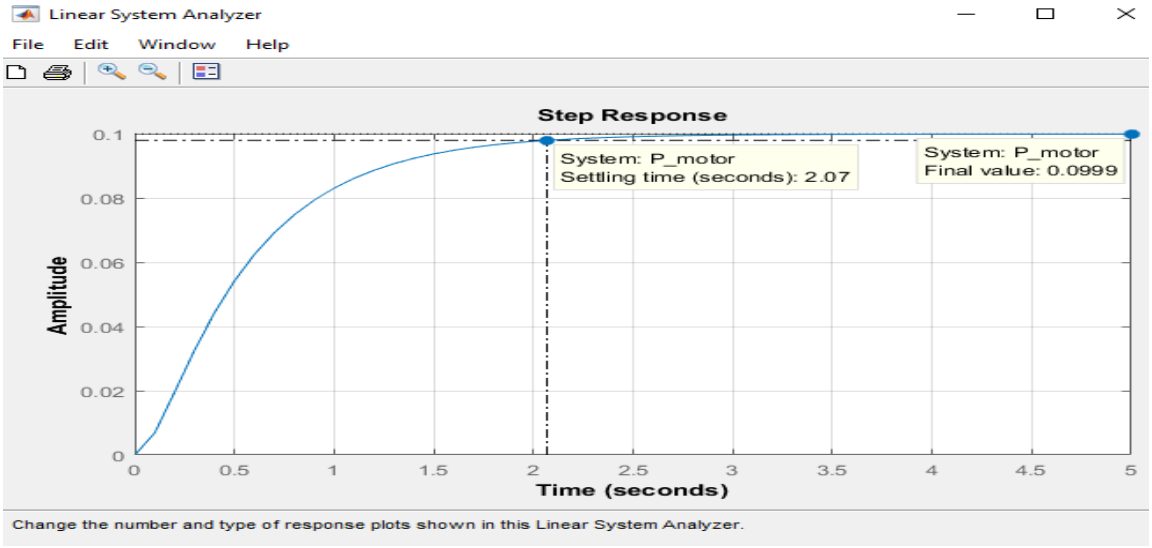


Fig.35-Open-loop response .

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.

4.6.3 PID Controller Design

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

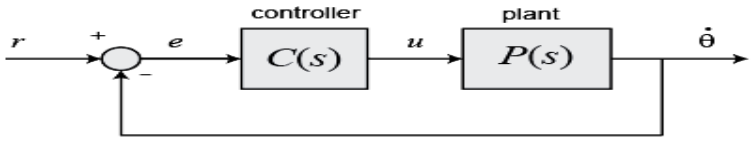


Fig.36-The structure of the control system.

Recall that the transfer function for a PID controller is:

$$C(s) = k_p + \frac{k_i}{s} + k_d\, s = \frac{k_d s^2 + k_p s + k_i}{s} \quad \text{.....(12)}$$

- 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.

```
Kp = 100;
C = pid(Kp);
sys_cl = feedback(C*P_motor,1);
t = 0:0.01:5;
    step(sys_cl,t)
    grid
    title('Step Response with Proportional Control')
```

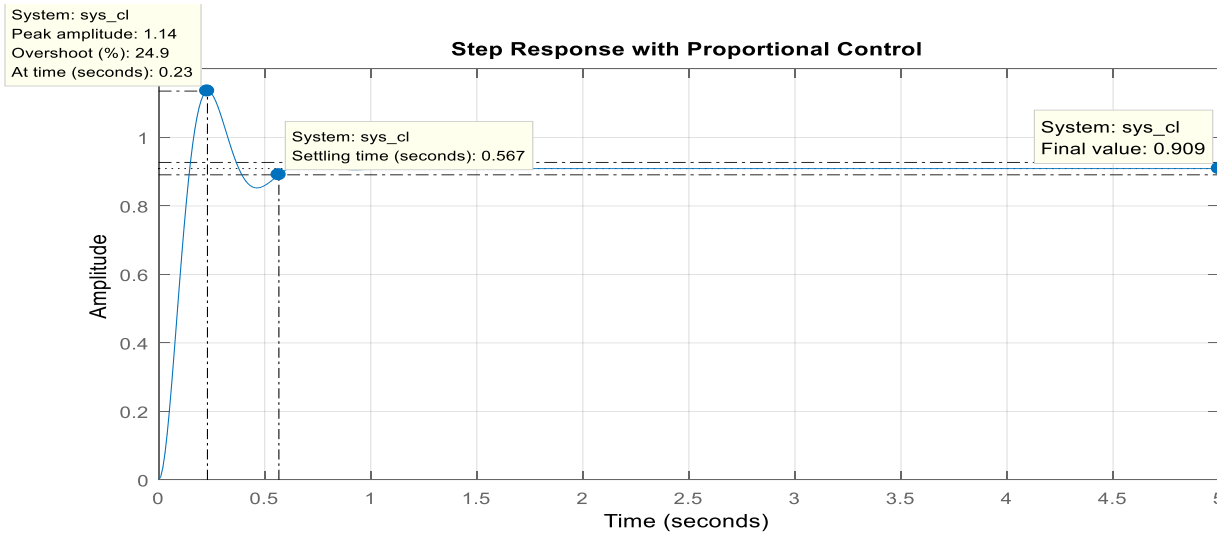


Fig.37-step response with propotional control .

Increasing the proportional gain K_P will reduce the steady-state error. also increased overshoot, therefore, it appears that not all of the design requirements can be met with a simple proportional controller.

A proportional controller is insufficient; derivative and/or integral terms must be added to the controller.

PID control

Tuning the gains

In this case. This process can be sped up without overshoot by increasing the value of ***ki*** and ***kd*** ; Go back to your m-file and change ***ki ; kd*** as in the following. Rerun the file and you should get the plot shown below.

```
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
title('PID Control with Large Ki and Large Kd')
```

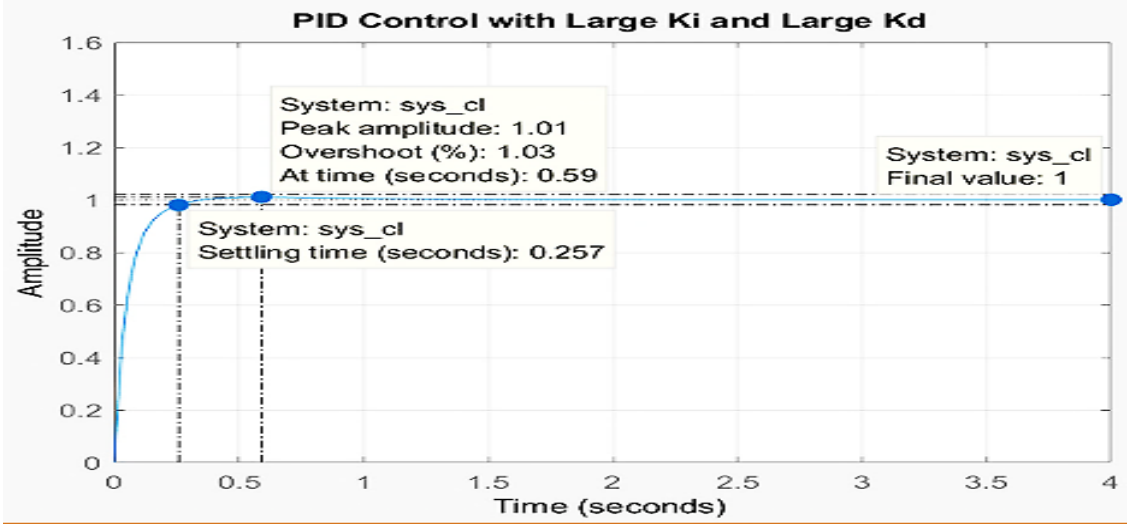


Fig.38-PID Control with Large Ki and Large Kd

As we had hoped, the increased ***kd*** reduced the resulting overshoot. Now we know that if we use a PID controller with ***kp***= 100, ***ki***= 200, and ***kd***= 10, all of our design requirements will be satisfied.

4.7 Results and Analysis of real implementation of the system

In this section we will walk through our project step by step to introduce obtained results. This instruction mainly introduces about making program in Arduino NANO , to control motor speed by PID algorithm

4.7.1 Step 1-Hardware and Software needed

To make this project, you will need following hardware:

1. DC Motor with encoder LM393
2. H-bridge

- 2 * BC557 or equivalent PNP transistors
- 2 * BC547 or equivalent NPN transistors
- 4 * 1K Ohm resistors
- 4 *1N4007 or equivalent diodes

3. Arduino NANO
4. Breadboard
5. Jumper Wires
6. LCD
7 a Capacitor 562J250V

A list of those complementary components is shown below.

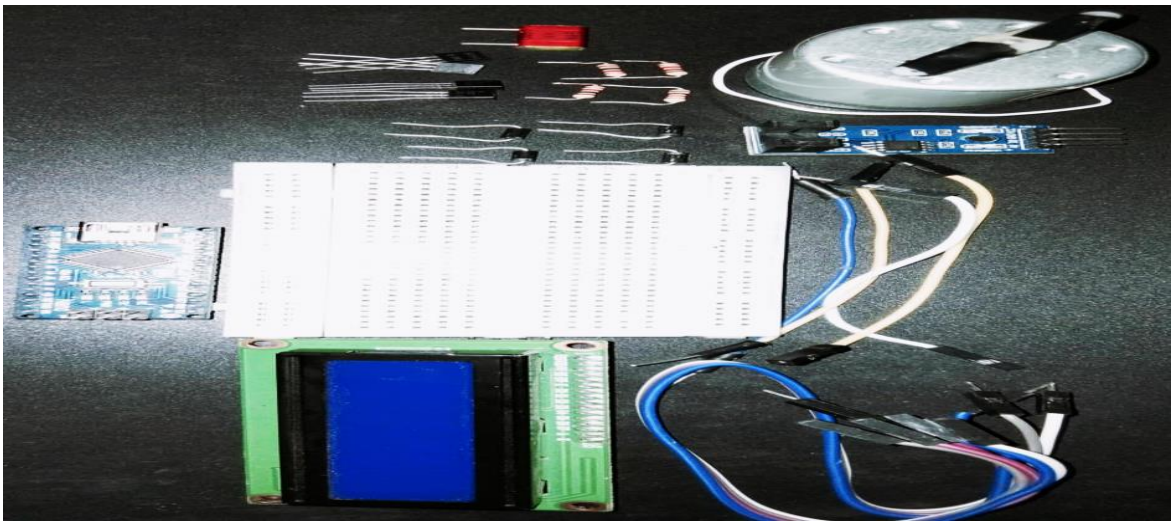


Fig.39 - Components Required to Build a PID Enabled Encoder Motor Controller

The motor with encoder, you can choose any DC motor but you need to know how many pulse of encoder per revolution. In case you don't know, you can make simple program with Arduino, then rotate motor shaft to know how many pulse of encoder per revolution. In my case, it is 2 pulses per revolution.

4.7.2 Step 2- Hardware connection

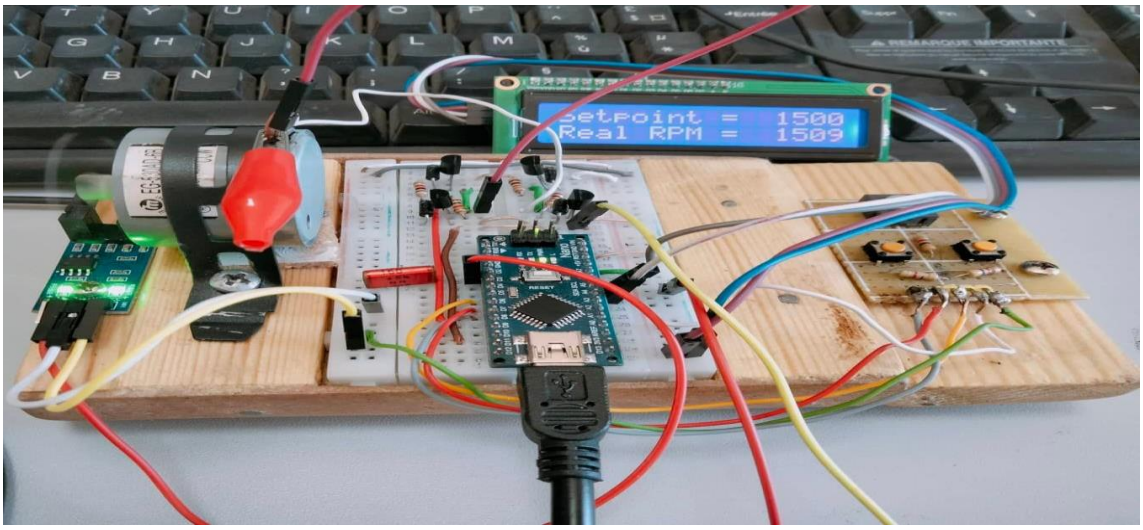


Fig.40- Experiment for DC motor speed control .

it's easy for us: encoder from motor will connect to pin 2; H-bridge is used to control motor speed, then pin 5, 6 and 7 will connect to H-bridge. And, output of H-bridge is connected to motor

This circuit has a DC motor attached with the disc . We have used IR sensor module and placed it in front of the disc attached to the DC motor. The IR sensor detects the change in intercepted Infrared values while the disc is rotating along with the dc motor and ultimately results in to an estimation (precise after each iteration) of the current RPM of the DC motor that we will display on LCD. After knowing the motor rpm Now we can manage the speed of the motor.

This RPM value send to the microcontroller as a feedback to compare with set point and treat the difference (error) by the PID ;Then Arduino gives a control signal (PWM) to the H-Bridge . Arduino Code is required to fix the speed because rpm of motor varies with respect to time set point (speed isnt smooth). If load on motor increased motor rpm will reduce, by measuring rpm, Arduino code will sense changed rpm. Accordingly it will send the signal(PWM) to make changes in power supplied to Dc motor it compensate the change RPM. Comparative thing happen if load is decreased

4.7.3 Step 3- Arduino Code

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

4.7.3 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

Fig.41 shows the comparison of desired rpm and actual rpm graph. According to this figure, the speed of the DC motor with encoder can be control via computer by employing the gains of PID.

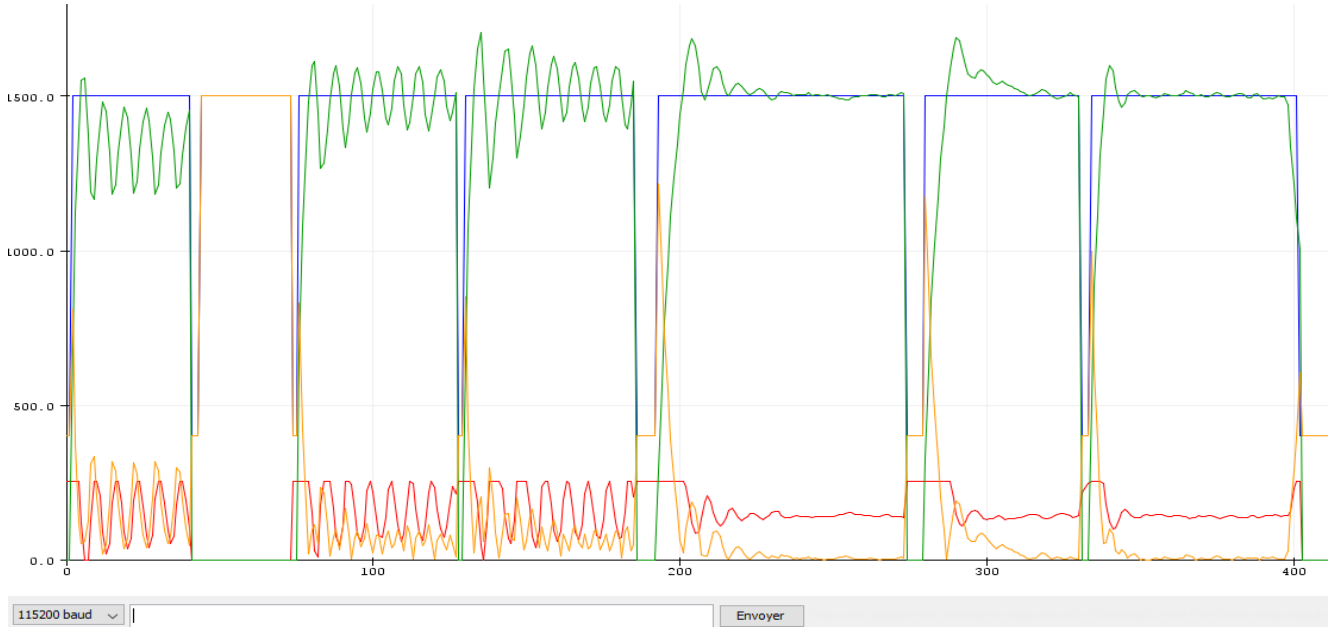


Fig.41- System's response with various state feedback control parameters.

- Set point
- Real speed RPM
- error
- pwm

In this closed loop system analysis, the PID parameter is tuning manually in sketch. The purpose of tuning the parameter is to find the optimum parameter for the PID. The good system response should be decreasing in time rising, settling time, overshoot (%) and steady state error.

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

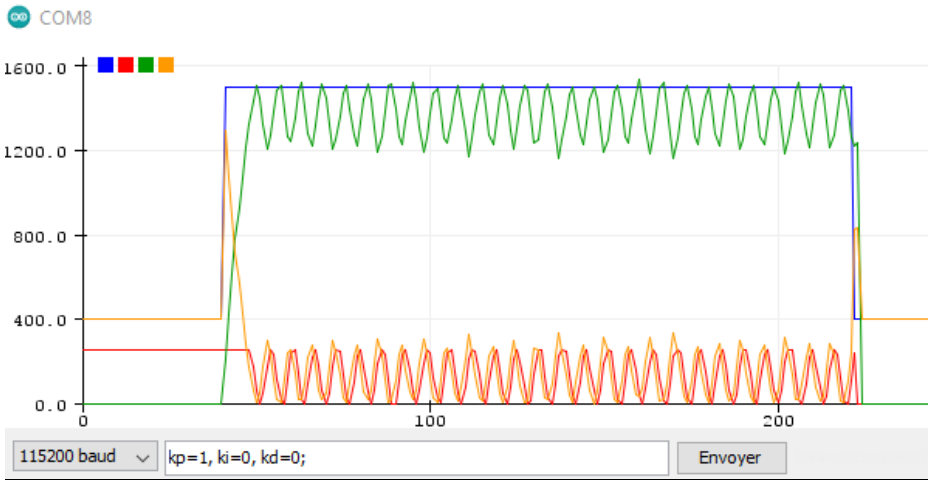


Fig.42- closed loop response with kp=1 .

The controller's response parameters are discussed below:

- The Steady State Error SSE increase for this mode of controller ,SEE=300rpm
- The system becomes unstable with oscillations

We try now to increase the gain Ki to 0.7 with KP=0.2, the results are shown in Fig.43

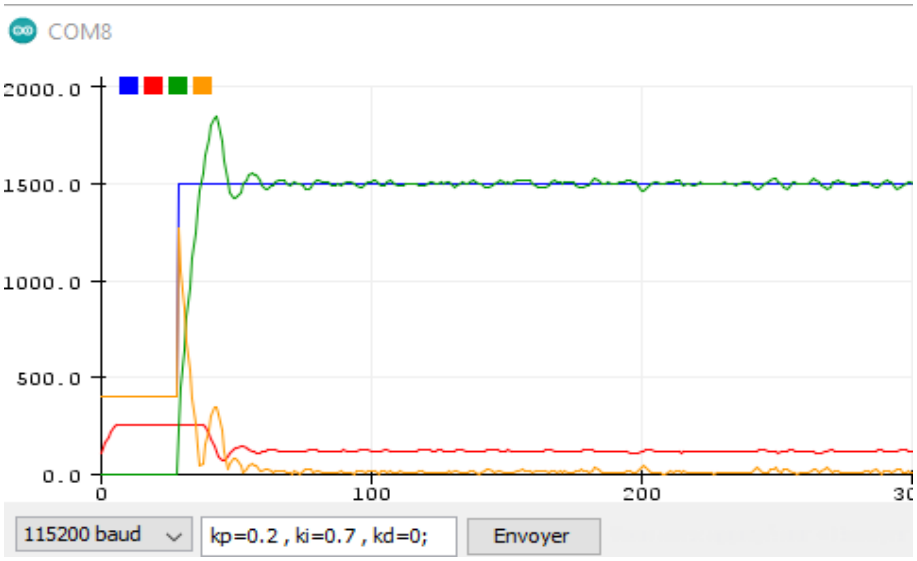


Fig .43- closed loop response with PI Controller .

- the Steady State Error SSE is start to decrease ,it's becomes very small and eliminated (SSE=0)
- but there are a high overshoot came to the system.
- The settling time is 60second very slow
- The system is start to stable.

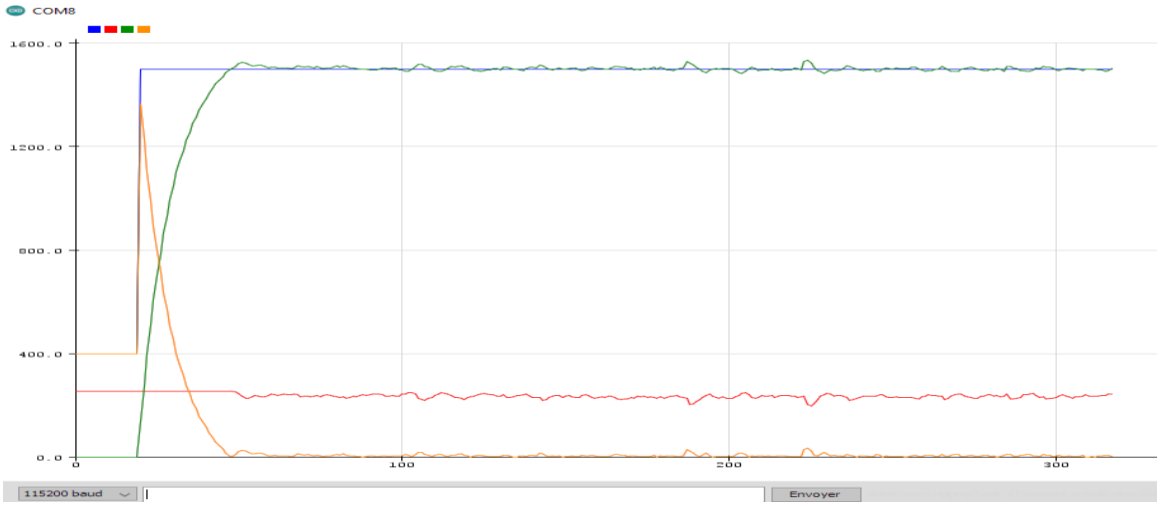


Fig .44-System's response with Kp =0.1 and Ki= 0.3'

After testing several experiments for tuning the PID controller, the best result for Kp=0.1and Ki=0.3 it show

- The response is much faster than before
- there are no overshoot
- the Steady State Error SSE =0
- But the performance still occurs a delay time between the desired and actual speeds

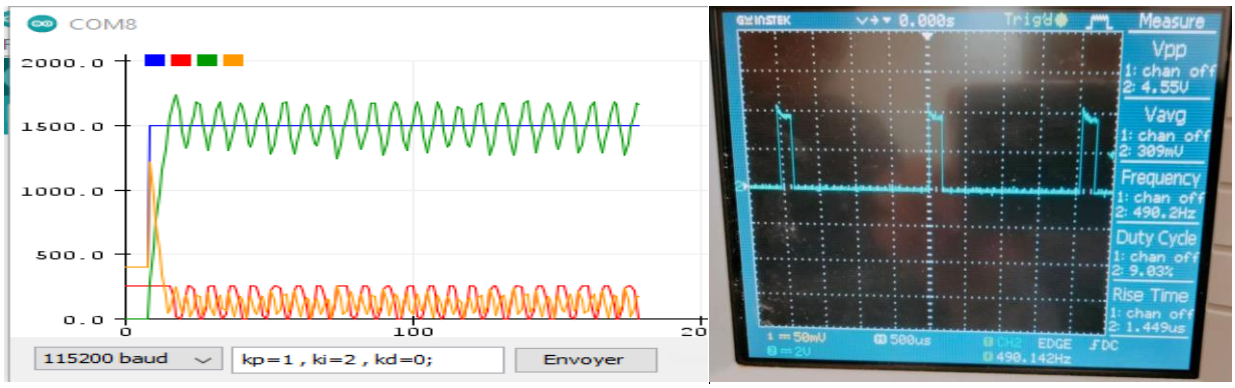


Fig.45- Effects of PWM signal on the speed, rpm=1600

When the speed increases than set point (1500 RPM) at the instant (t = 180s), the duty cycle of PWM decreases to 9.03% in order to compensate the real disturbance and then immediately returns (Fig.45).

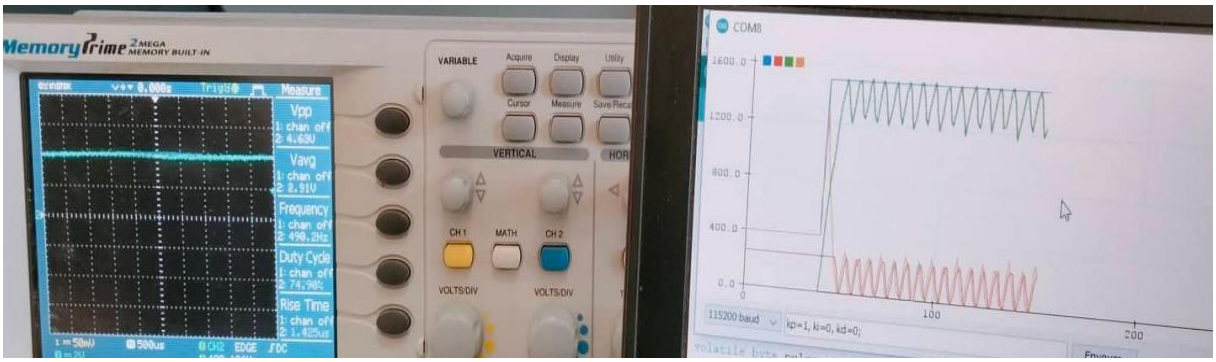


Fig .46- Effects of PWM signal on the speed ,rpm=1200

When the speed decreases than the setpoint at the instant (t = 150s), the duty cycle of PWM increases 100% in order to compensate the real disturbance (Fig.46) and then speed immediately increases

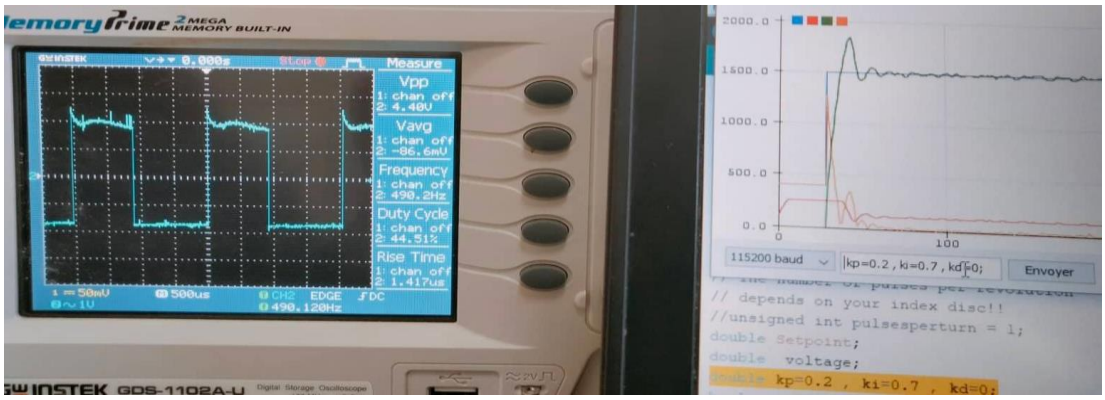


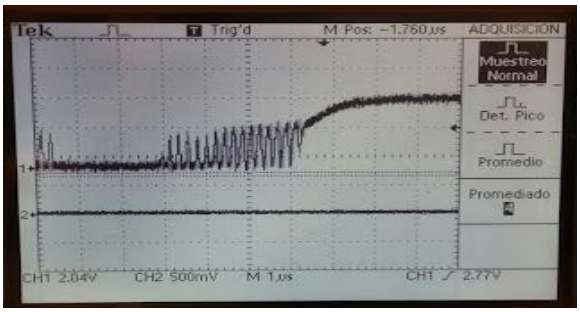
Fig .47- Effects of PWM command signal on the stable speed ,rpm=1500

When the speed achieves the setpoint and the system be stable , the duty cycle of PWM also is stable about 45% (Fig. 47) .

This method takes data from the PWM frequency with PWM Frequency = 490HZ .

4.8 Bad pulses

I have also had problems with this encoder, when reading the digital pulses generated by the comparator LM-393. Arduino Nano read more pulses than the encoder actually generate. The boarts read 4 times more pulses than the encoder actually generate, this sensor is very sensitive to the interference that can be introduced in the VCC and GND pins. If we feed the sensor from the Arduino itself to 3.3V or 5V, The voltage regulator of the Arduino itself can introduce stray currents into the sensor. Producing that this does not work properly.Using an oscilloscope connected between the pins D0 and GND and analyzed the pulses that are generated in the encoder. See the following pictures:



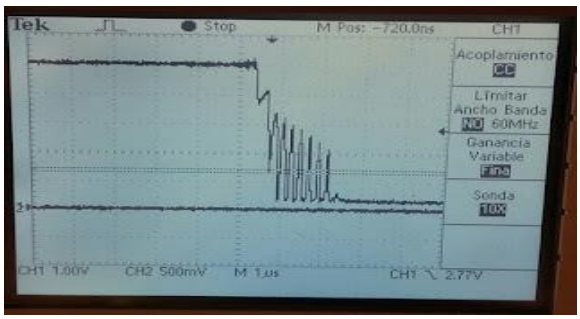


Fig.48- Initial rebound of the signal. **Fig.49 -Final rebound of the signal**

we can see that the pulse is not square. As we can see in the following photo, the square digital signal that generates the encoder FC-03 has rebounds at the beginning and the end of the pulse . Arduino is very sensitive and reads these rebounds as good pulses and really these rebounds are not correct.

To solve this problem, I have designed two solutions:

- The first solution is to make a program, an "Arduino" sketch that does not read the rebounds and does not read false signals.
- The second solution is to place a capacitor between pin D0 and pin GND to eliminate rebounds.

The capacitor that has given me the best result is the capacitor (562J 250v). This capacitor almost does not deform the digital pulse generated by the comparator LM393

This solution is very good, because false signals are not sent to Arduino and the program does not have to waste time checking if the signal is good or bad. In this way, only interruption of the Arduino is activated when the signal is correct. [46] See the following photo

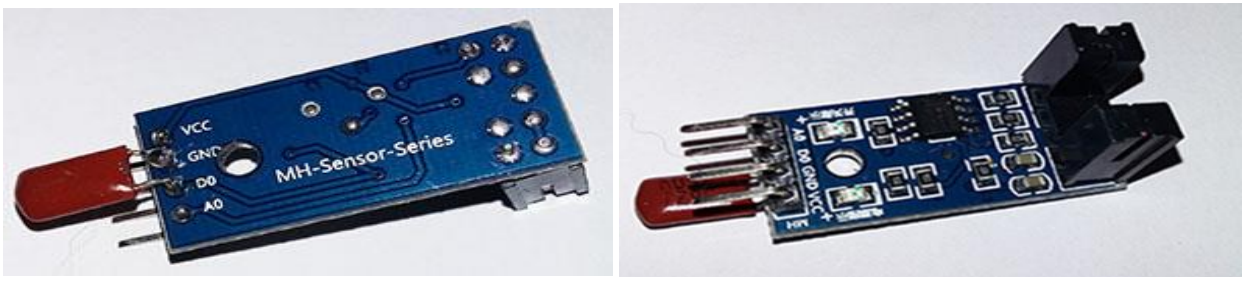


Fig.50- View of capacitor welded to two pins .

4.9 Conclusion

From the results, the PI-controller solve the problems of large disturbance and noise during operation processer by eradicating enforced oscillations and steady state error subsequent in operation of PI Controller.

The use of pulse width modulation to control a small motor has the advantage in that the power loss in the switching transistor is small because the transistor is either fully “ON” or fully “OFF”. As a result the switching transistor has a much reduced power dissipation giving it a linear type of control which results in better speed stability.

The simulation results are approximately similar to that obtained from the practical measurements.

The research is about controlling DC Motor using integral state feedback. The research was done by simulation and hardware implementation. In the simulation and hardware implementation result, the integral state feedback gave a good performance while reaching the set point. From the tracking control result with different setpoints, integral state feedback presented similar performance: the augmented system performed with fast rising time and settling time with small overshoot.

Compared with the PID controller, the integral state feedback had a better system response in tracking control at some setpoints.

Future works of the research are widely open in many areas. The tuning parameter was still done using trial and error. Thus it will need a method to determine the parameter controller. An experiment with uncertainty and disturbance has not been done yet. Another challenging problem in applying the integral state feedback is that all states must be known. Hence, observers, such as minimum or full order observers, can be applied to overcome this issue in the future. The observer makes the augmented system will not need all states to be known. Another possible future research is to apply the Kalman filter to minimize the oscillation and noises in the output sensor. From a hardware perspective, it is also possible to conduct future research on controlling the DC motor's angular speed using the current sensor since it has more stability while reading the measurement than using the encoder sensor[47] .

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

[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 Aeronautical Engineering, Quthbullapur, Telangana-43, India

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

[4] *International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 4, April 2016*

[5] https://www.elprocus.com/an-overview-of-arduino-Nano -board/

[6] https://www.arduino.cc/en/Guide/Introduction

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

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

[9] https://www.c-sharpcorner.com/UploadFile/167ad2/introducing-arduino-Nano /

[10] https://microdigisoft.com/introduction-of-arduino-Nano -board/

[11] https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-Nano .html

[12] http://www.farnell.com/datasheets/1682238.pdf

[13] https://components101.com/misc/breadboard-connections-uses-guide

[14] https://www.rs-online.com/designspark/what-is-arduino-Nano -a-getting-started-guide

[15] https://www.circuito.io/blog/arduino-code/

[16] https://www.makerspaces.com/simple-arduino-projects-beginners/

[17] https://stemify.weebly.com/programming-structure.html

[18] https://www.codeproject.com/Articles/1247684/Getting-Started-With-Arduino-Using-the-Small-Inexp

[19] https://projectiot123.com/2019/04/08/arduino-Nano -for-beginners/

[20] Pro Arduino by Released June 2013 Publisher(s): Apress ISBN: 9781430239390 chapter 7 pid controllers

[21] W. K. Ho, C. C. Hang, and L. S. Cao, “Tuning of PID contollers based on gain and phase margin specifications,” Automatica, vol. 31, pp. 497–502, 1995.

[22] F. Radke, R. Isermann, A parameter-adaptive PID controller with stepwise parameter optimization, Automatica, 23 Issue 4, pp. 449 – 457, 1987.

[23] Transactions on Control SystemsTechnology, vol. 13, pp. 559-576, 2005.

[24] 2020 International Conference on Computing and Information Technology, University of Tabuk, Kingdom of Saudi Arabia. Volume: 01, Issue: ICCIT- 1441, Page No.: 358 - 363, 9th & 10th Sep. 2020.

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

[26]. Ibrahim kaya, “IMC based automatic tuning method for PID controllers in a smith predictor configuration" , Sciencedirect, 2004

[27]https://plcynergy.com/pid-controller/

[28]https://create.arduino.cc/projecthub/muhammad-aqib/arduino-pwm-tutorial-ae9d71

[29]https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html

[30]https://elexfocus.com/speed-control-of-dc-motor-using-arduino-applying-pwm

[31] 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.

[32] https://byjus.com/physics/dc-motor/#what-is-a-dc-motor

[33] https://www.electrorules.com/dc-motor-definition-construction-and-working-principle/

[34] https://www.ebay.com/itm/252786292619

[35] https://instrumentationtools.com/encoder-working-principle/

[36] https://www.futek.com/Incremental-Encoder-Signal-Converter

[37] http://androminarobot-english.blogspot.com/2017/03/encoder-and-arduinotutorial-about-ir.html

[38] https://123dok.org/document/myjlv6zl-lm-motor-speed-measuring-sensor-module-for-arduino.html

[39] https://www.circuitstoday.com/h-bridge-motor-driver-circuit

[40] https://www.etechnog.com/2019/03/h-bridge-circuit-working-application-advantage.html

[41] https://www.electronicshub.org/introduction-to-transistors/#What_is_a_Transistor

[42]https://www.easybom.com/blog/a/bc557-pnp-transistor-what-is-the-difference-between-bc547-and-bc557

[43] https://enviremental b.com/bc557-transistor-pinout/

[44] https://www.componentsinfo.com/bc547-pinout-equivalent/

[45] https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling

[46] http://androminarobot-english.blogspot.com/2017/03/encoder-and-arduinotutorial-about-ir.html

[47] Journal of Robotics and Control (JRC) Volume 2, Issue 5, September 2021 ISSN: 2715-5072 DOI: 10.18196/jrc.2512

شهادة مشاركة

الطبعة الأولى

يشهد السيد مدير جامعة الشهيد زيان عاشور الجلفة بأن السيد(ة):

نايل البشير، مهلول رباب،لعربي نصيرة،شراب تركية ، ديربالي زهرة

قد شارك(ت) ضمن فعاليات الصالون الأول

للأفكار والأعمال المنجزة والمبتكرة بجامعة الجلفة يوم 19 ماي 2022

بفكرة/عمل بعنوان:

جهاز تحكم في سرعة و إتجاه محرك التيار مستمر

Real-time speed and direction control of DC motor based on ATMEGA328P µc

مدير الجامعة

مديـر جامعـة الجلفة

الأستاذ: عـلاء الـديـن