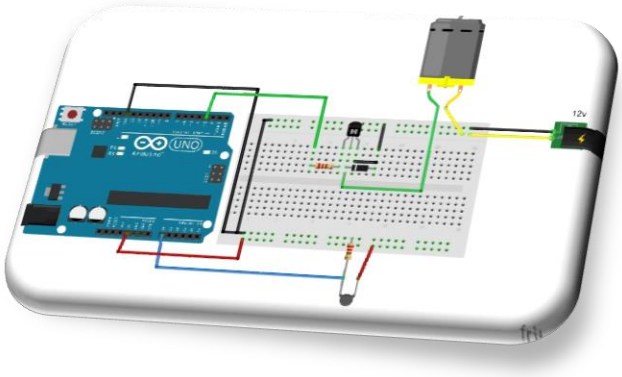
Abstract

This report “Controlling a motor using PID controller” aims to discuss how the project is made, the tools and components we used, the concept behind making this project. The project is about controlling the speed of a DC motor using Arduino to achieve the PID theory and use the PID controller.

PID control methods are widely used in the industry. In this project we will be using PID for speed control of a DC motor. For this purpose, encoder feedback is used to compare the target speed and actual speed and the difference between these two value (error) is used to drive the motor.



Contents

Table of figures

1. Introduction

In this modern industrial age, there is hardly any industrial application in which DC motors are not being used. This is so because of ease of control, low cost maintenance especially of brushless DC motor type, low price, and ruggedness of DC motor over a wide range of applications. Some industrial applications, which are worth mentioning, in which DC motors are being used widely are machine tools, paper mills, textile industry, electric traction, and robotics. In these applications, the purpose is to track the speed command by keeping output speed at desired level and to achieve desire speed or position control in minimum time without having large overshoots and settling times.

PID controller is one of the earliest and best understood controllers which is incorporated in almost every industrial control application due to its efficiency and ease of implementation [9]. Although there are many classical techniques for designing and tuning PID controller parameters (Kp,Ti,Td) which are widely understood and easily applied, one of the main disadvantages of these classical techniques is that, for tuning PID controller through these techniques, expertise and experience are required. This is so because these methods provide a starting point and achieving desired performance fine tuning of parameters through hit-and-trial method is required.

1.1. Problem statement

How to use PID controller through code and Arduino (controller chip) to control the speed of a DC motor automatically.

1.2. Meta behind the project

- How PID controller is used in earl and modern controllers and industries.
- Know how to control PID parameters and know each parameter use.
- Know how to tune PID parameters to achieve the desired input.

2. PID theory

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller.

Equation 1 - PID output eq.

$$\mathrm{u} = \underbrace{K_p e}_{\text{Proportional Term}} + \underbrace{K_i \int_0^t e\,dt}_{\text{Integral Term}} + \underbrace{K_d \frac{d}{dt} e}_{\text{Differential Term}}$$

K_p is the proportional gain, a tuning parameter,

K_i is the integral gain, a tuning parameter,

K_d is the derivative gain, a tuning parameter,

$e(t) = \mathrm{SP} - \mathrm{PV}(t)$ is the error (SP is the setpoint, and PV(t) is the process variable),

t is the time or instantaneous time (the present),

τ is the variable of integration (takes on values from time 0 to the present t).

2.1. How does PID work

GOAL of PID speed control: Make the **actual motor speed match the desired motor speed.**

When activated, the PID algorithm will use a motor's built-in rotation sensors to monitor its actual speed. The actual speed is compared to the desired speed, and the PID algorithm will calculate necessary power changes to get the actual speed equal to the desired speed.

The algorithm then starts over again by comparing the new actual speed to the desired speed. Based on the improvement (or lack thereof) that it sees, it will make further refinements to the motor's power. This creates a cycle where the motor's speed is constantly being checked against the desired speed, and the power level is always set based on what is needed to achieve the right result.

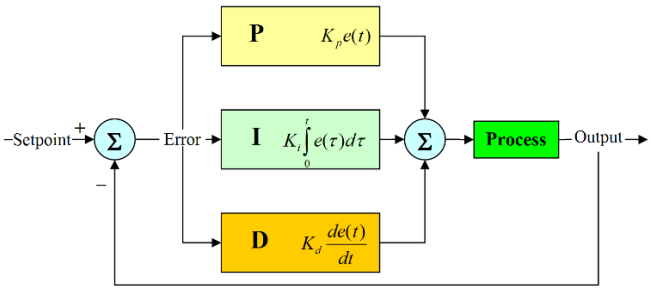


Figure 1 - PID diagram

2.2. Parameters

- P accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.
- I accounts for past values of the error. For example, if the current output is not sufficiently strong, the integral of the error will accumulate over time, and the controller will respond by applying a stronger action.
- D accounts for possible future trends of the error, based on its current rate of change.

2.3. Steady state error

Because a non-zero error is required to drive it, a proportional controller generally operates with a so-called steady-state error.[a] Steady-state error (SSE) is proportional to the process gain and inversely proportional to proportional gain. SSE may be mitigated by adding a compensating bias term to the setpoint or output, or corrected dynamically by adding an integral term.

3. Code

```
//PID program
 if (motor_start) {
   e_speed = set_speed - pv_speed;
   pwm_pulse = e_speed * kp + e_speed_sum * ki + (e_speed - e_speed_pre) * kd;
   e_speed_pre = e_speed;  //save last (previous) error
   e_speed_sum += e_speed; //sum of error
   if (e_speed_sum > 4000) e_speed_sum = 4000;
   if (e_speed_sum < -4000) e_speed_sum = -4000;
 }
 else {
   e_speed = 0;
   e_speed_pre = 0;
   e_speed_sum = 0;
   pwm_pulse = 0;
 }


 //update new speed
 if (pwm_pulse > 255)
   analogWrite(pin_pwm, 255);
 else if (pwm_pulse < 0)
   analogWrite(pin_pwm, 0);
 else
   analogWrite(pin_pwm, pwm_pulse);

}
void serialEvent() {
 while (Serial.available()) {
   char inChar = (char)Serial.read();
   if (inChar != '\n')
     mySt += inChar;
   else
```

```
    stringComplete = true;
  }
}
```

3.1. Explaining the code

Using Arduino to achieve PID with code we enter the set-point and PID parameters.

PROPORTIONAL CONTROL:

The proportional part of PID examines the magnitude of the error and it reacts proportionally. A large error receives a large response

INTEGRAL CONTROL:

To address the first issue with the proportional control, integral control attempts to correct small error (offset). Integral is the sum of all results over certain time

DERIVATIVE CONTROL:

The derivative part of the control output attempts to look at the rate of change in the error signal. Derivative will cause a greater system response to a rapid rate of change than to a small rate of change. It tests the change by comparing the new and present change to predict what will happen to the curve.

Visual studio is used to make the interface we enter the parameters in and see the response of our inputs (how and when the desired input is achieved).

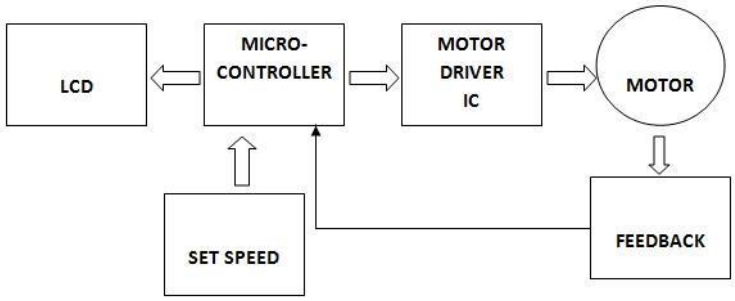


Figure 2 - project diagram

4. Components

We used simple components, Arduino and computer (to assign the inputs) to achieve the project.

4.1. Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, 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. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

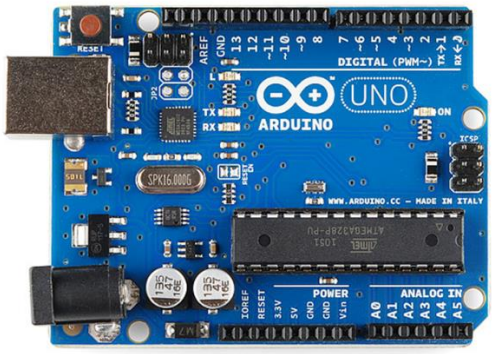


Figure 3 - Arduino

```
const byte pin_a = 2;   //for encoder pulse A
const byte pin_pwm = 6; // motor speed
```

to assign input and output pins

```
void setup() {
  Serial.begin(9600);
  pinMode(pin_a, INPUT);
  pinMode(pin_pwm, OUTPUT);
```

the measurement from the encoder comes as the PV and the error is measured, the parameters is assigned and the output (M) goes to the motor as pulses (Pulse Width Modulation) with different width as desired.

4.2. Encoder (RKI-2554)

Encoder is used to measure the speed of the dc motor. The Encoder consist of an infrared emitting diode and an NPN silicon phototransistor, encased side-by-side on converging optical axis in a black thermoplastic housing. The phototransistor receives radiation from the IRED only. But when an object is in between, phototransistor could not receive the radiation

Encoder is attached to the Arduino and receives its power from it.

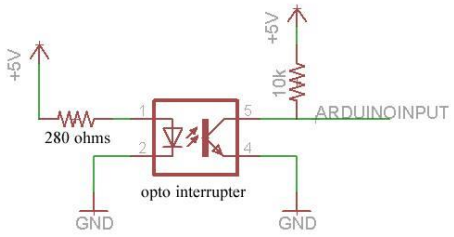


Figure 4 - encoder biasing

4.3. DC motor

A DC motor is any of a class of rotary electrical machines that converts direct current electrical power into mechanical power.

Figure 5 - motor

We attached a fan to it to act as the object to intersect the light from the encoder so the speed can be measured.

4.4. Transistors TIP

A transistor is a semiconductor device used to amplify or switch electronic signals and electrical power. It is composed of semiconductor material usually with at least three terminals for connection to an external circuit. A voltage or current applied to one pair of the transistor's terminals controls the current through another pair of terminals. TIP produces high current up to 6 ampere.

Figure 6 – TIP

4.5. Diode

A diode is a specialized electronic component with two electrodes called the anode and the cathode. The fundamental property of a diode is its tendency to conduct electric current in only one direction. We used it to protect the circuit and the motor from the reverse current.

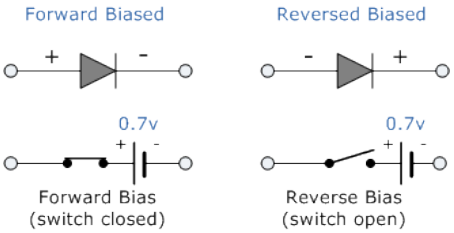


Figure 7 - Diode operation

5. Execution

We enter the setpoint and the PID parameters as follow (visual studio is used as the interface and the inputs go to the Arduino code). The output represents in the output curve.

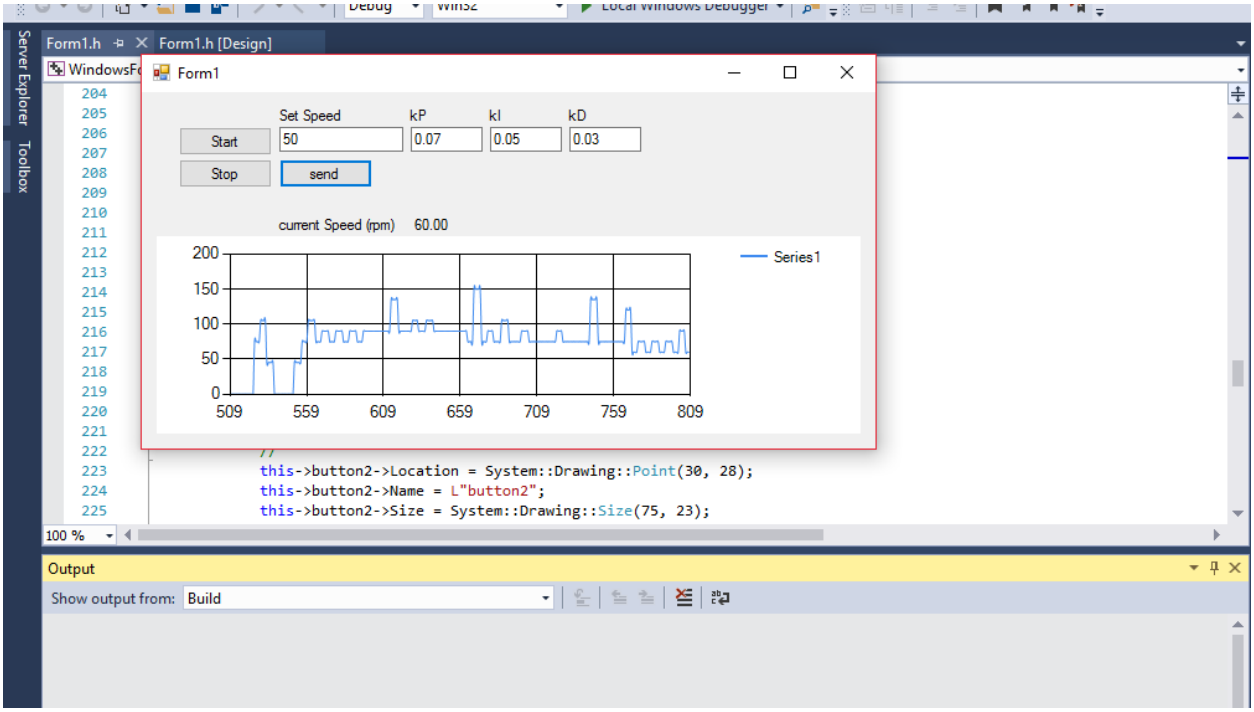


Figure 8 - VS execution

6. References

http://www.robotshop.com/letsmakerobots/arduino-pid-motor-controller

https://hackaday.io/project/11382-dc-motor-speed-control-with-pid

http://robokits.co.in/sensors/encoders/encoder-sensor-with-5mm-slit

http://www.education.rec.ri.cmu.edu/previews/robot_c_products/teaching_rc_lego_v2_preview/reference/hp_PID.pdf

https://en.wikipedia.org/wiki/PID_controller#PID_controller_theory

https://www.hindawi.com/journals/aans/2014/126317/

https://create.arduino.cc/projecthub/whitebank/arduino-motor-pid-speed-control-5f4632