CHAPTER ONE
INTRODUCTION

1.1 General Concept

Brushless DC (BLDC) motors are preferred as small horsepower control motors due to their high efficiency, silent operation, compact form, reliability, and low maintenance. However, the problems are encountered in these motor for variable speed operation over last decades continuing technology development in power semiconductors, microprocessors, adjustable speed drivers control schemes and permanent-magnet brushless electric motor production have been combined to enable reliable, cost-effective solution for a broad range of adjustable speed applications. Household appliances are expected to be one of fastest-growing end-product market for electronic motor drivers over the next five years. The major appliances include clothes washer’s room air conditioners, refrigerators, vacuum cleaners, freezers, etc. Household appliance have traditionally relied on historical classic electric motor technologies such as single phase AC induction, including split phase, capacitor-start, capacitor–run types, and universal motor. These classic motors typically are operated at constant-speed directly from main AC power without regarding the efficiency. Consumers now demand for lower energy costs, better performance, reduced acoustic noise, and more convenience features those traditional technologies cannot provide the solutions[1].

1.2 Problem Statement

The problem of this project is centric in required speed and actual speed of BLDC motor. Due to the remaining magnetism and error in manufactured, the actual speed doesn’t equal the required speed of BLDC motor. And this problem is causing a lot of errors in the application that’s needs a high efficiency and high accuracy in motion.

1.3 Objectives

The main objectives of this study are to:

i. Design the PID controller to control the speed of BLDC motor.

ii. Appraisal the performance of the proposed controller.

1.4 Methodology

- Study of all previous studies.
- Mathematical modeling of BLDC motor.
- Build ARDUINO microcontroller program to control in speed of BLDC with PID algorithm as software included in the controller.
- Design of PID controller using manual tuning.
- Use of MATLAB program to plot system response.

1.5 Layout

This project consists of five chapters: Chapter One gives an introduction, motivation and Objectives. Chapter Two discuss the theoretical background of BLDC motor, PID controller and microcontroller systems. Chapter Three presents the system modeling and system design. Chapter Four deals with the practical model of the system and the experimental results. Finally, Chapter Five provides the conclusions and recommendations.

CHAPTER TWO
THEORETICAL, BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

In this chapter would discuss brushless dc motor construction and theory of operation and microcontroller that has been developing through the year and proportional, Integral and Derivative (PID).and previous study of all.

2.2. Brushless DC Motor

A brushed DC motor is an internally commutated electric motor designed to be run from a direct current power source. Brushed motors were the first commercially important application of electric power to driving mechanical energy, and DC distribution systems were used for more than 100 years to operate motors in commercial and industrial buildings. A brushed DC motor is made up of 4 basic components: the stator, the rotor (or armature), brushes, and commutator. The stator generates a stationary magnetic field that surrounds the rotor and this magnetic field is generated by either permanent magnets or electromagnetic windings. The rotor is made up of one or more windings. When these windings are energized they produce a magnetic field. The magnetic poles of this rotor field will be attracted to the opposite poles generated by the stator, causing the rotor to turn. As the motor turns, the windings are constantly being energized in a different sequence so that the magnetic poles generated by the rotor do not overrun the poles generated in the stator. This switching of the field in the rotor windings is called Commutation. Brushed DC motors do not require a controller to switch current in the motor windings. Instead, it uses a mechanical commutation of the windings[2].

A copper sleeve (commutator), resides on the axle of the rotor. As the motor turns, carbon brushes slide over the commutator, coming in contact with different segments of the commutator. The segments are attached to different rotor windings; therefore, a dynamic magnetic field is generated inside the motor when a voltage is applied across the brushes of the motor. The brushes and commutator are the parts of a brushed DC motor that are most prone to wear. Thus, switching the electrical polarity of the rotor windings. This will create an attraction of the different polarities and keep the rotor rotating within the stator field. Brushed dc motors presents some disadvantages: less reliability, larger size than other types of motors, high maintenance requirements.

That all changed in the 1980s, when permanent magnet materials became readily available. The use of permanent magnets, combined with high voltage transistors, enabled brushless DC motors to generate as much power as the old brushed DC motors; the Brushless Direct Current (BLDC) motors are one of the motor types rapidly gaining popularity[1]. The brushless dc motor is conventionally defined as permanent magnet synchronous motor with a trapezoidal back elector motive force (EMF) waveform shape .This is a relatively new class of motors whose application have been increasing at a rapid rate each year, due both to declining costs as well as increasing functionality. BLDC motors have many advantages over brushed DC motors and induction motors. A few of these are:

• Better speed versus torque characteristics.

• High dynamic response.

• High efficiency.

• Long operating life.

• Noiseless operation.

• Higher speed ranges.

In addition, the ratio of torque delivered to the size of the motor is higher, making it useful in applications where space and weight are critical factors[3].

2.2.1 Construction of BLDC motor

Brushless permanent magnet motor operation relies on the conversion of energy from electrical to magnetic to mechanical. BLDC motors are a type of synchronous motor. This means the magnetic field generated by the stator, and the magnetic field generated by the rotor rotates at the same frequency. BLDC motors do not experience the “slip” that is normally seen in induction motors. BLDC motors come in single-phase, 2-phase and 3-phase configurations. Corresponding to its type, the stator has the same number of windings. Out of these, 3-phase motors are the most popular and widely used. This application note focuses on 3-phase motors. It is a rotating electric motor consisting of stator armature windings and rotor permanent magnets whereas in a conventional brushed DC motor the stator is made up of permanent magnets and rotor consists of armature windings. The conventional DC motor commutes itself feedback with the use of a mechanical commutator whereas brushless DC motors, the feedback is with some electronic feedback sensor such as magnetic Hall sensors, encoders or resolvers. They key point is that the windings are sequentially switched with the drive electronics in BLDC motors. Without the electronics, the motor cannot even run. Conversely since the BLDC motors have electronics any way, they can be used to accomplish adjustable speed control and many other useful functions. Typically BLDC motors have three phase windings that are wound in star or delta fashion and need a three phase inverter bridge for the electronic commutation[3].

i. Stator

Traditionally, the stator resembles that of an induction motor; however, the windings are distributed in a different manner. Most BLDC motors have three stator windings connected in star fashion. Each of these windings are

Constructed with numerous coils inter connected to form a winding. One or more coils a replaced in the slots and they are interconnected to make a winding. Each of these windings is distributed over the stator periphery to form an even numbers of poles[3]. There are two types of stator windings variants trapezoidal and sinusoidal motors. This differentiation is made on the basis of the interconnection of coils in the stator windings to give the different types of back electromotive Force (EMF). As their names indicate, the trapezoidal motor gives a back EMF in trapezoidal fashion and the sinusoidal motor's back EMF is sinusoidal[4]

ii. **Rotor**

The rotor of a typical BLDC motor is made out of permanent magnets. Depending upon the application requirements, the number of poles in the rotor may vary. Increasing the number of poles does give better torque but at the cost of reducing the maximum possible speed. Another rotor parameter that impacts the maximum torque is the material used for the construction of permanent magnet, the higher the flux density of the material, the higher the torque. Ferrite magnets are traditionally used to make permanent magnets. As the technology advances, rare earth alloy magnets are gaining popularity[3]. The ferrite magnets are less expensive but they have the disadvantage of low flux density for a given volume. In contrast, the alloy material has high magnetic density per volume and enables the rotor to compress further for the same torque. Also, these alloy magnets improve the size-to-weight ratio and give higher torque for the same size motor using ferrite magnets[5]. The rotor magnet comes in different cross section:

- Circular core with magnet on the periphery.
- Circular core with rectangular magnets embedded in the rotor.
- Circular core with rectangular magnets inserted into the rotor core.

iii. Hall sensors

Unlike a brushed DC motor, the commutation of a BLDC motor is controlled electronically. To rotate the BLDC motor, the stator windings should be energized in a sequence. It is important to know the rotor position in order to understand which winding will be energized following the energizing sequence. Rotor position is sensed using Hall Effect sensors embedded into the stator[1]. And the Hall Effect theory is that if an electric current carrying conductor is kept in a magnetic field, the magnetic field exerts a transverse force on the moving charge carriers which tends to push them to one side of the conductor. This is most evident in a thin flat conductor. A buildup of charge at the sides of the conductors will balance this magnetic influence, producing a measurable voltage between the two sides of the conductor. The presence of this measurable transverse voltage is called the Hall Effect after E. H. Hall who discovered it in 1879[3]. The underlying principles for the working of a BLDC motor are the same as for a brushed DC motor; i.e., internal shaft position feedback. In case of a brushed DC motor, feedback is implemented using a mechanical commutator and brushes. With an in BLDC motor, it is achieved using multiple feedback sensors. The most commonly used sensors are hall sensors and optical encoders. Most BLDC motors have three Hall sensors embedded into the stator on the non-driving end of the motor. Whenever the rotor magnetic poles pass near the Hall sensors, they give a high or low signal, indicating the N or S pole is passing near the sensors. Based on the combination of these three Hall sensor signals, the exact sequence of commutation can be determined. Hall sensors are embedded in the stator of a BLDC motor to determine the winding energizing sequence, Based on the physical position of the Hall sensors, there are two versions of output. The Hall sensors may be at 60° or 120° phase shift to each other. Based on this, the motor manufacturer defines the commutation sequence, which should be followed when controlling the motor[4].

2.2.2 Theory of operation

Each commutation sequence has one of the windings energized to positive power (current enters into the winding), the second winding is negative (current exits the winding) and the third is in a non-energized condition. Torque is produced because of the interaction between the magnetic field generated by the stator coils and the permanent magnets. Ideally, the peak torque occurs when these two fields are at 90° to each other and falls off as the fields move together. In order to keep the motor running, the magnetic field produced by the windings should shift position, as the rotor moves to catch up with the stator field. What is known as “Six-Step Commutation” defines the sequence of energizing the windings[1].

To understand these steps show the Figure 2.1, phase1, the green winding labeled "001" is energized as the North Pole and the blue winding labeled as "010" is energized as the South Pole. Because of this excitation, the south pole of the rotor aligns with the green winding and the North Pole aligns with the red winding labeled "100". In order to move the rotor, the red and blue Windings are energized in the direction shown in Figure 2.1, phase 2. This causes the red winding to become the North Pole and the blue winding to become the South Pole. This shifting of the magnetic field in the stator produces torque because of the development of repulsion (winding north-north alignment) and attraction forces (blue winding north-south alignment), which moves the rotor in the clockwise direction. This torque is at its maximum when the rotor starts to move, but it reduces as the two fields align to each other. Thus, to preserve the torque or to build up the rotation, the magnetic field generated by stator should keep switching. To catch up with the field generated by the stator, the rotor will keep rotating. Since the magnetic field of the stator and rotor both rotate at the same frequency, they come under the category of synchronous motor.

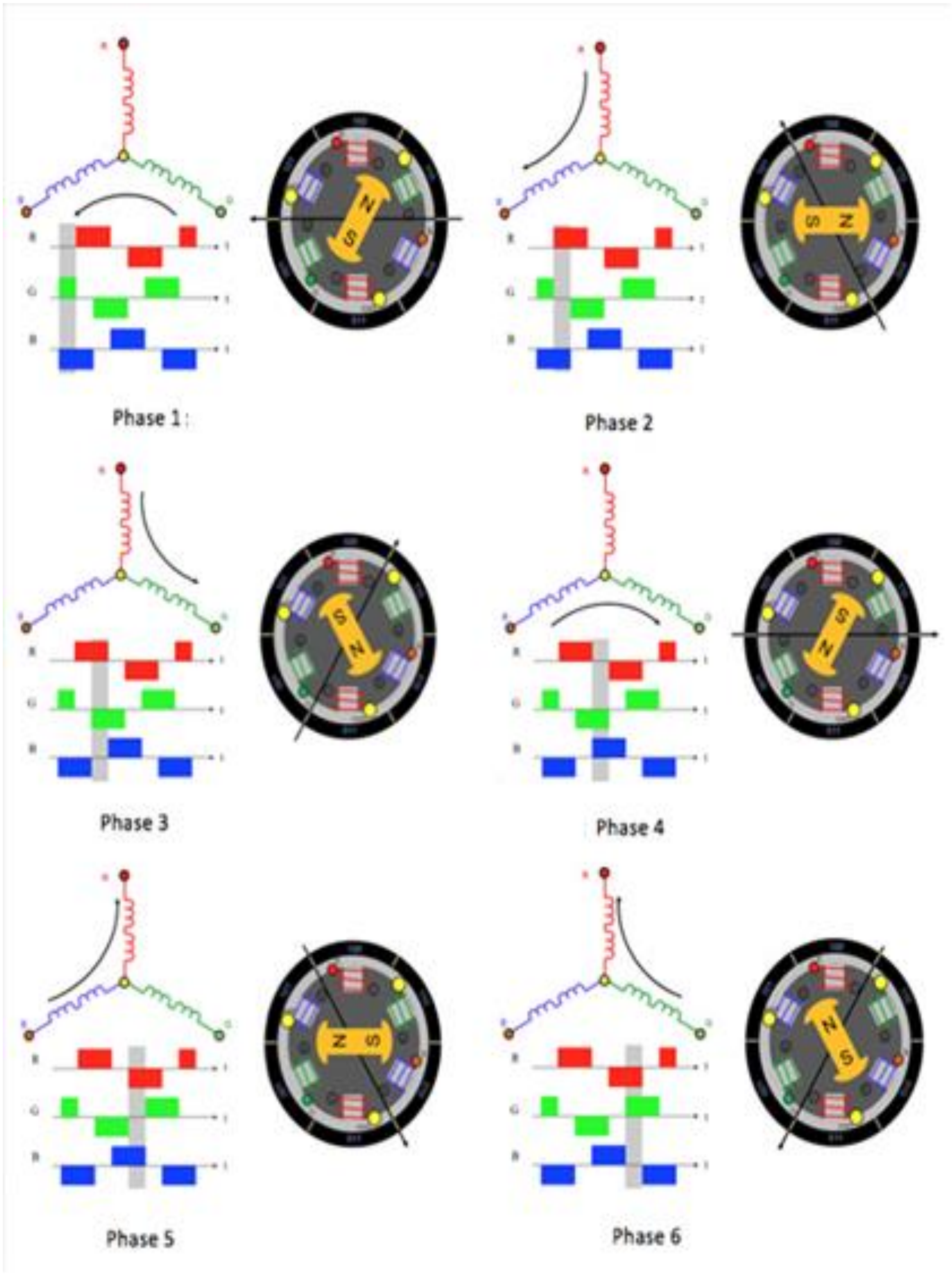

Figure 2.1: Theory of operation of BLDC motor

This switching of the stator to build up the rotation is known as commutation. For 3-phase windings, there are steps in the commutation, 6 unique combinations in which motor windings will be energized[3].

2.2.3 Torque and efficincy

By definition, torque is the tendency of force to rotate an object about its axis.

Torque (Newton- meters) = Force (Newton) ×Distance (meters) (2.1)

Thus, to increase the torque, either force has to be increased which requires stronger magnets or more current or distance must be increased for which bigger magnets will be required. Efficiency is critical for motor design because it determines the amount of power consumed. A higher efficiency motor will also require less material to generate the required torque.

$$\text{Efficiency} = \frac{\text{Output Power}}{Input\ Power}\ \% \qquad (2.2)$$

Where,

Output Power = Torque × Angular velocity. (2.3)

Input Power = Voltage × Current. (2.4)

As a conclusion:

- ❖ With an increase in speed, the torque reduces (considering the input power is constant).
- ❖ Maximum power can be delivered when the speed is half of the "no load" speed and torque is half of the stall torque[5].

2.2.4 Comparing BLDC motor to other motor type

Compared to brushed DC motors and induction motors, BLDC motors have many advantages and few disadvantages. Brushless motors require less maintenance, so they have a longer life compared with brushed DC motors.

BLDC motors produce more output power per frame size than brushed DC motors and induction motors. Because the rotor is made of permanent magnets, the rotor inertia is less, compared with other types of motors[3].

2.2.5 Applications of brushless DC motor

- **Single-speed**: For single-speed applications, induction motors are more suitable, but if the speed has to be maintained with the variation in load, then because of the flat speed-torque curve of BLDC motor, BLDC motors are a good fit for such applications.
- **Adjustable speed**: BLDC motors become a more suitable fit for such applications because variable speed induction motors will also need an additional controller, thus adding to system cost. Brushed DC motors will also be a more expensive solution because of regular maintenance.
- **Position control**: Precise control is not required applications like an induction cooker and because of low maintenance; BLDC motors are a winner here too. However, for such applications, BLDC motors use optical encoders, and complex controllers are required to monitor torque, speed, and position.
- **Low-noise applications**: Brushed DC motors are known for generating more EMI noise, so BLDC is a better fit but controlling requirements for BLDC motors also generate EMI and audible noise. This can, however, be addressed using Field-Oriented Control (FOC) sinusoidal BLDC motor control[1].

2.3 Arduino microcontroller

A microcontroller is a small, low-cost computer on-a-chip present in a single integrated circuit which is dedicated to perform one task and execute one specific application which usually includes:

- ➢ An 8 or 16 bit microprocessor (CPU).
- ➢ A small amount of RAM.

- Programmable ROM and/or flash memory.
- Parallel and/or serial I/O.
- Timers and signal generators.
- Analog to Digital (A/D) and/or Digital to Analog (D/A) conversion.

Often we use it to run dedicated code that controls one or more tasks in the operation of a device or a system. Also called embedded controllers, because the microcontroller and support circuits are often built into, or embedded in, the devices they control. Devices that utilize microcontrollers include car engines, consumer electronics (VCRs, microwaves, cameras, pagers, cell phones ...), computer peripherals (keyboards, printers, modems...), test/measurement equipment (signal generators, multi meters, oscilloscopes …).Microcontrollers usually must have low-power requirements (~. 05 - 1 Watt opposed to ~10 - 50 Watt for general purpose desktop CPUs) since many devices they control are battery-operated In addition to control applications. Such as the home monitoring system, microcontrollers are frequently found in embedded applications. Among the many uses you can find one or more microcontrollers: automotive applications, appliances (microwave oven, refrigerators, television and VCRs, stereos), automobiles (engine control, diagnostics, climate control), environmental control (greenhouse, factory, home), instrumentation, aerospace, and thousands of other uses[6]. Microcontrollers are used extensively in robotics. In this application, many specific tasks might be distributed among a large number of microcontrollers in one system. Communications between each microcontroller and a central, more powerful microcontroller (or microcomputer, or even large computer) would enable information to be processed by the central computer, or to be passed around to other microcontrollers in the system[7].

Arduino microcontroller, Arduino is a small microcontroller board with a USB plug to connect to your computer and a number of connection sockets that can be wired up to external electronics, such as motors, relays, light sensors, laser diodes, loudspeakers, microphones, etc. They can either be powered through the USB connection from the computer or from a 9V battery. They can be controlled from the computer or programmed by the computer and then disconnected and allowed to work independently. We can use Arduino for many different purposes. From teaching to home automation, from scientific purposes to commercially available devices, as well as to have fun (you can be surprised about the many ways in which people use Arduino). Arduino is very simple interface to I/O ports you can control many different devices, both digital and analogical. Arduino is then a very useful tool both to control measuring apparatus or as a device to take measurements by itself (for many purpose it can be accurate enough to replace professional, and expensive, instruments)[8].

The Arduino project was started in Italy to develop low cost hardware for interaction design. The Arduino board features an Atmel ATmega328 microcontroller operating at 5 V with 2 Kb of RAM, 32 Kb of flash memory for storing programs and 1 Kb of EEPROM for storing parameters. The clock speed is 16 MHz, which translates to about executing about 300,000 lines of C source code per second. The board has 14 digital I/O pins and 6 analog input pins. There is a USB connector for talking to the host computer and a DC power jack for connecting an external 6-20 V power source, for example a 9 V battery, when running a program while not connected to the host Computer. Headers are provided for interfacing to the I/O pins using 22 g solid wire or header connectors[9]. The board of Arduino can show in Figure 2.2.

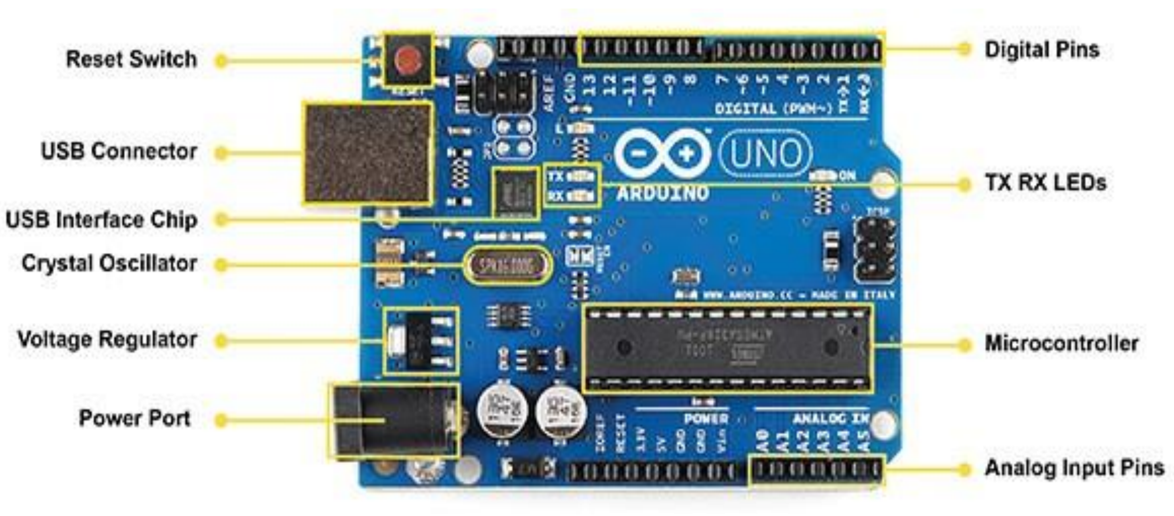

Figure 2.2: Content of Arduino UNO board

2.4 Proportional, Integral, Derivate Controller

Feedback control is a control mechanism that uses information from measurements. In a feedback control system, the output is sensed. There are two main types of feedback control systems is positive feedback and negative feedback. The positive feedback is used to increase the size of the input but in a negative feedback is used to decrease the size of the input. The negative systems are usually stable[10]. A Proportional– Integral–Derivative controller (PID controller) is widely used in feedback control of industrial processes on the market in 1939 and has remained the most widely used controller in process control until today. Thus, the PID controller can be understood as a controller that takes the present, the past, and the future of the error into consideration. After digital implementation was introduced, a certain change of the structure of the control system was proposed and has been adopted in many applications. But that change does not influence the essential part of the analysis and design of PID controllers. PID is a method of the control loop feedback[11].

2.4.1 PID effect

PID control is the most common control algorithm used in industry and has been universally accepted. The PID controller attributes are partly to their robust performance in a wide range of operating conditions and partly to their functionality simple, allowing engineers to operate them in a simple manner. As the name suggests, a PID algorithm consists of three basic coefficients: proportional, integral and derivative. These gains are varied to achieve an optimal system response the basic structure of a system with PID control implemented is illustrated above in Figure 2.3. The system output (also called the process variable) with a sensor and compared to the reading to the reference value (also called the set point). Reference and the measured output are compared and the result is an error value which is used in calculating proportional, integral, and derivative responses. Summing the three responses to obtain the output of the controller[11].

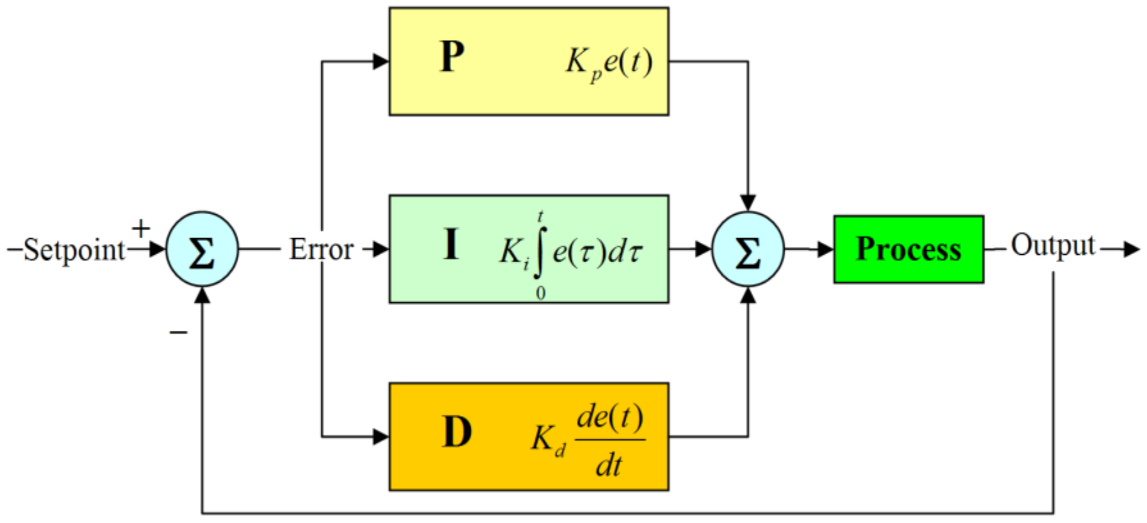

Figure 2.3: Proportional integral derivative

The output of the controller is used as an input to the system you wish to control, changing some aspect of the system. For example, if motor is controlled, the controller would provide more or less current. If controlling the flow of a fluid, the controller would cause a valve either to open or closed.

The output of the system gets measured and the process continues. Iteration of the control loop can be known as completion of the process in single time.
The ideal version of the PID controller is given by the formula:

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{d}{dt} e(t) \quad (2.5)$$

After Laplace transfer

$$\frac{u(s)}{e(s)} = k_p(1 + \frac{1}{t_i s} + t_d s) \quad (2.6)$$

Where

k_p= Proportional gain.

k_i= Integral gain.

k_d= Derivative gain.

e: Error=Set point –Process Value.

t: Instantaneous time.

t_i: Integral time.

t_d: Derivative time.

2.4.2 Proportional controller

The proportional term is produced, in which an output value that is proportional to the present error value. The proportional response is adjusted by multiplying the error with a constant k_p, called proportional gain constant. The proportional term is given as

$$u(t) = k_p e(t) \quad (2.7)$$

Or

$$\frac{u(s)}{e(s)} = k_p \quad (2.8)$$

A high proportional gain resulting in a large change in output for a given change of the error. If the proportional gain is very high, the system becomes unstable. Whereas, a small gain results in a small output response to a large input error and a low responsive or less sensitive controller. If the proportional gain is much less, the control action may be too small when responding to the system disturbances. Tuning theory and industrial practice implies that, the proportional term must contribute to the bulk of the output.

2.4.3 Integral controller

The contribution of the integral term is proportional to both the magnitude of the error and the duration of error. In a PID controller the integral is the sum of the instantaneous error over time and gives the accumulated offset that should have been previously corrected. Accumulated error gets multiplied by the integral gain (k_i) and added to the controller output. The integral term is given by:

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt \tag{2.9}$$

$$\frac{u(s)}{e(s)} = k_p(1 + \frac{1}{t_i s}) \tag{2.10}$$

The integral term accelerates the movement of the process towards the set point and eliminates the residual steady-state error which occurs with a pure Proportional controller. Integral term responds to accumulated errors of the past, it can result the present value to overshoot the set point value.

2.4.4 Proportional derivative

PD controller is used in the process where offset value is acceptable, where some anticipation can be considered and should have low noise. The equation of PD Controller is:

$$u(t) = k_p e(t) + k_d t_d \frac{de(t)}{dt} \tag{2.11}$$

The ideal transfer function of PD controller is given by:

$$G(s) = \frac{u(s)}{e(s)} = k_p(1 + t_d s) \tag{2.12}$$

It is clear from above discussions that a suitable combination of proportional, integral and derivative actions can provide all the desired performances of a closed loop system. The transfer function of a PID controller is given by:
The order of the controller is low, but this controller has universal applicability; it can be used in any type of SISO system, linear, nonlinear, time delay etc. Many of the MIMO systems are first decoupled into several SISO loops and PID controllers are designed for each loop. PID controllers have also been found to be robust, and that is the reason, it finds wide acceptability for industrial processes. However, for proper use, a controller has to be tuned for a particular process; i.e. selection of PID parameters is very important and process dependent. Unless the parameters are properly chosen, a controller may cause instability to the closed loop system. The method of tuning of PID parameters would be taken up in the next lesson. It is not always necessary that all the features of proportional, derivative and integral actions should be incorporated in the controller. In fact, in most of the cases, a simple PI structure will suffice. A general guideline for selection of Controller mode is given below[10].

2.4.5 Guideline for selection of controller mode

i. **Proportional controller:** It is simple regulating type; tuning is easy. But it normally introduces steady state error. It is recommended for process transfer functions having a pole at origin, or for transfer functions having a single dominating pole.

ii. **Integral control:** It does not exhibit steady state error, but is relatively slow responding. It is particularly effective for:

   - Very fast process, with high noise level.

- Process dominated by dead time.
- High order system with all-time constants of the same magnitude.

iii. **Proportional plus integral control:** It does not cause offset associated with proportional control. It also yields much faster response than integral action alone. It is widely used for process industries for controlling variables like level, flow, pressure, etc., those do not have large time constants.

iv. **Proportional plus derivative control:** It is effective for systems having large number of time constants. It results in a more rapid response and less offset than is possible by pure proportional control. But one must be careful while using derivative action in control of very fast processes, or if the measurement is noisy (e.g. flow measurement).

v. **Proportional plus integral plus derivative control:** It finds universal application. But proper tuning of the controller is difficult. It is particularly useful for controlling slow variables, like pH, temperature, etc. in process industries.

2.4.6 The characteristics of PID controllers

A proportional controller (k_p) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error.

An integral control (k_i) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (k_d) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response[11]. Effects of each of controllersk_p,k_i and $k_d$on a closed-loop system are summarized in the Table 2.1.

Table 2.1: Explain the characteristic of PID parameter.

CL response	Rise time	Overshoot	Settling time	S-S error
k_p	Decrease	Increase	Small Change	Decrease
k_i	Decrease	Increase	Increase	Eliminate
k_d	Small Change	Decrease	Decrease	Small Change

2.5 Literature review

Brushless DC motor actually represents the most recent result of a long evolution of motor technology. Before there were brushless DC motors there were brush DC motors. The brush DC motor was invented all the way back in 1856 by famed German inventor and industrialist Ernst Werner von Siemens. That Resulted in adjusting the output voltage of the DC generator, which in turn adjusted the motor speed. The Ward Leonard system remained in place all the way until 1960, when the Electronic Regulator Company's thyristor devices produced solid state controllers that could convert AC power to rectified DC power more directly. It supplanted the Ward Leonard system due to its simplicity and efficiency. In 1962 two engineers, T.G. Wilson and P.H. Tricky published a paper in which they described a "brushless DC" motor. Magnet and power switching device technology prevented this invention from becoming a practical general purpose drive technology until the late 1980s when power Tec Industrial Corporation started manufacturing general purpose brushless systems at prices competitive with brush DC systems. Remember that the key element of brushless DC motor is that it requires no physical commutator[12]. A KUSKO and S.M. PEERAN presented definitions of a brushless DC motor are inadequate to distinguish it from other types of brushless motors in the industry. A formal definition of a brushless

DC motor is described. The definition includes the components of the motor and the types of circuits to energize the stator windings. M.A.JABBAR, M.A.RAHMAN discussed the design considerations for permanent-magnet motors intended for brushless operation. Two rotor configurations are described the imprecated rotor and the segmented rotor. The segmented rotor is designed especially for high speed operation. A brushless DC drive system is also described and the performance of a neodymium-iron-boron excited p.m. motor with an imprecated rotor in a BLDC drive is presented[1].
The origin of the Arduino project started at the Interaction Design Institute IVREA (IDII) in IVREA, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2003 Hernando BARRAGAN created the development platform wiring as Master's thesis project at IDII, under the supervision of Massimo BANZI and Casey REAS, who are known for work on the Processing language. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller. In 2003, Massimo BANZI, with David MELLIS, another IDII student, and David CUARTILLES, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they copied the Wiring source code and renamed it as a separate project, called Arduino. The initial Arduino core team consisted of Massimo BANZI, David CUARTILLES, Tom IGOE, GIANLUCA Martino, and David MELLIS, but BARRAGAN was not invited to participate[8].
PID controllers have their origins in 19th century speed governor design. The theoretical basis for the operation of governors was first described by James Clerk Maxwell in 1868 in his seminal paper on governor. It was not until 1922 that PID controllers were first developed using a theoretical analysis,

by Russian American engineer Nicolas MINORSKY for automatic ship steering. MINORSKY was designing automatic steering systems for the US Navy and based his analysis on observations of a helmsman, noting the helmsman steered the ship based not only on the current course error, but also on past error, as well as the current rate of change this was then given a mathematical treatment by MINORSKY. His goal was stability, not general control The Navy ultimately did not adopt the system, due to resistance by personnel. Similar work was carried out and published by several others in the 1930s. One of the earliest examples of a PID-type controller was also developed by Elmer Sperry in 1911, though his work was intuitive rather than mathematically-based[13].

CHAPTER THREE
SYSTEM MODELING AND DESIGN

3.1 System Description

Speed control of BLDC motor by using PID system shown in Figure 3.1 consists of two parts, hardware and software design, the first one is the hardware, it consist of brushless DC motor, IR sensor, Arduino, Bluetooth module, smart phone and power supply. The second part consists of design of PID controller and Arduino code, the following block diagram is explain that.

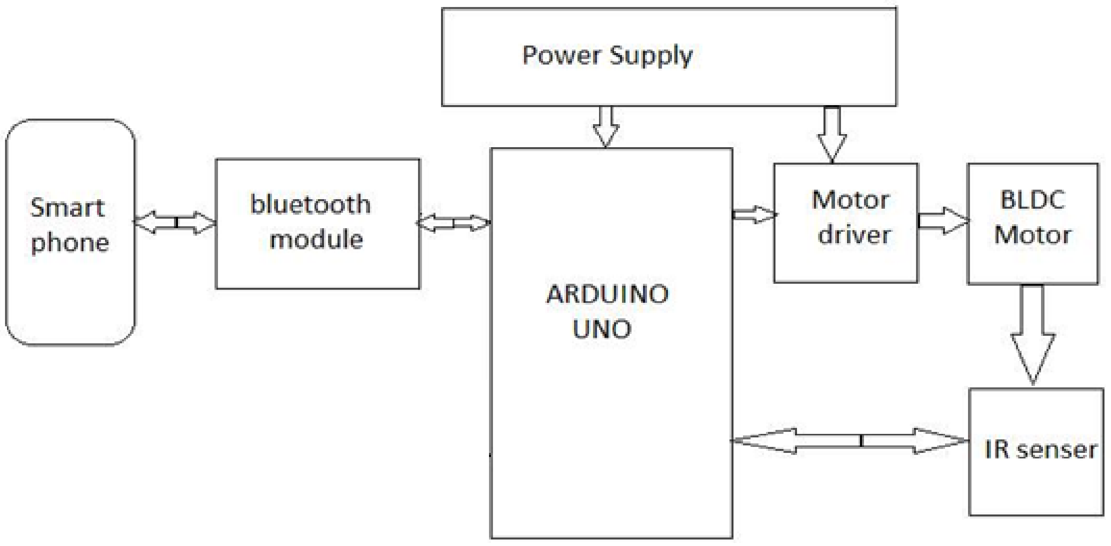

Figure 3.1: Block diagram of complete system

The block diagram is explains the arrangement of design of the system. First a smart phone send the required speed and PID parameters to Arduino through Bluetooth module, Arduino processes this signal, produces appropriate signal to the BLDC motor through the driver and motor runs, the speed which monitored by the IR sensor, sensor which feeds microcontroller with feedback signals ,and the result of the feedback is monitored on the smart phone.

3.2 The Brushless DC Motor Model

Brushless DC motors usually driven by balanced three phase waveforms. Equivalent circuit of each phase consists of a winding inductance, a resistance and induced back-EMF voltage due to the rotation of the rotor. Per phase equivalent circuit of a BLDC motor is shown in figure 3.2.

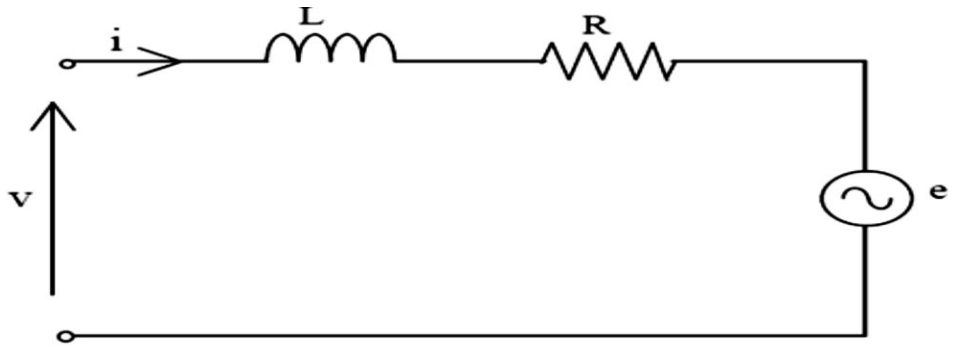

Figure 3.2: Per phase equivalent circuit of brushless DC motor.

The model of the armature winding of the BLDC motor is expressed as follows:

$$V_r = Ri_r + L\frac{di_r}{dt} + e_r \tag{3.1}$$

$$V_b = Ri_b + L\frac{di_b}{dt} + e_b \tag{3.2}$$

$$V_y = Ri_y + L\frac{di_y}{dt} + e_y \tag{3.3}$$

Where:

L is armature self – inductance [H].

R - Armature resistance [Ω].

V_r , V_b, V_y - terminal phase voltage [V].

i_r, i_b, i_y - motor input current [A].

e_r, e_b, e_y – motor back- EMF [V].

In the 3-phase BLDC motor, the back- EMF is related to a function of rotor position and the back-EMF of each phase has 120° phase angle difference so equation of each phase should be as follows:

$$e_r = k_w \times f(\theta_e)\omega \qquad (3.4)$$

$$e_b = k_w \times f(\theta_e - {}^{2\pi}/_{3})\omega \qquad (3.5)$$

$$e_y = k_w \times f(\theta_e + {}^{2\pi}/_{3})\omega \qquad (3.6)$$

Where:

k_w- is back EMF constant of one phase [V/ rad.s-1].

θ_e- is electrical rotor angle [°el.].

ω - Rotor speed [rad.s-1].

The electrical rotor angle is equal to the mechanical rotor angle multiplied by the number of pole pairs P.

$$\theta_e = \frac{P}{2} \times \theta_m \qquad (3.7)$$

Where:

θ_m – Mechanical rotor angle [rad].

Total torque output can be represented as summation of that of each phase. Next equation represents the total torque output:

$$T_e = \frac{e_r i_r + e_b i_b + e_y i_y}{\omega} \qquad (3.8)$$

3.2.1The transfer function of BLDC motor

The transfer function of BLDC motor show in equation

$$V = R_a i + L_a \frac{di}{dt} \qquad (3.9)$$

By using Laplace transfer:

$$V(s) = R_a I_{(s)} + L_a s I_{(s)} \qquad (3.10)$$

$$I_{(s)} = \frac{V_{(s)}}{R_{a\,(s)} + L_a(s) * s} \qquad (3.11)$$

$$G(s) = \frac{I_{(s)}}{V_{(s)}} = \frac{1}{L_a s + R_a} \tag{3.12}$$

The parameters of the motors used in Simulink are as show follows in Table3.1.

Table 3.1: The BLDC parameter used in Simulink

Symbol	Value
R_a	$50\ \Omega$
L_a	20 H

So it becomes:

$$G(s) = \frac{0.05}{s+2.5} \tag{3.13}$$

3.3 System Flow Chart

The flow chart shown in Figure 3.3 explained the flow of the system operation, at start input the required speed and PID parameter then compare the input parameter with the ideal parameter if any one of parameter is not ideal the system will return the inputted value of this parameter and if all parameter is ideal the BLDC motor will run, the IR sensor will sense the actual speed and monitor it.

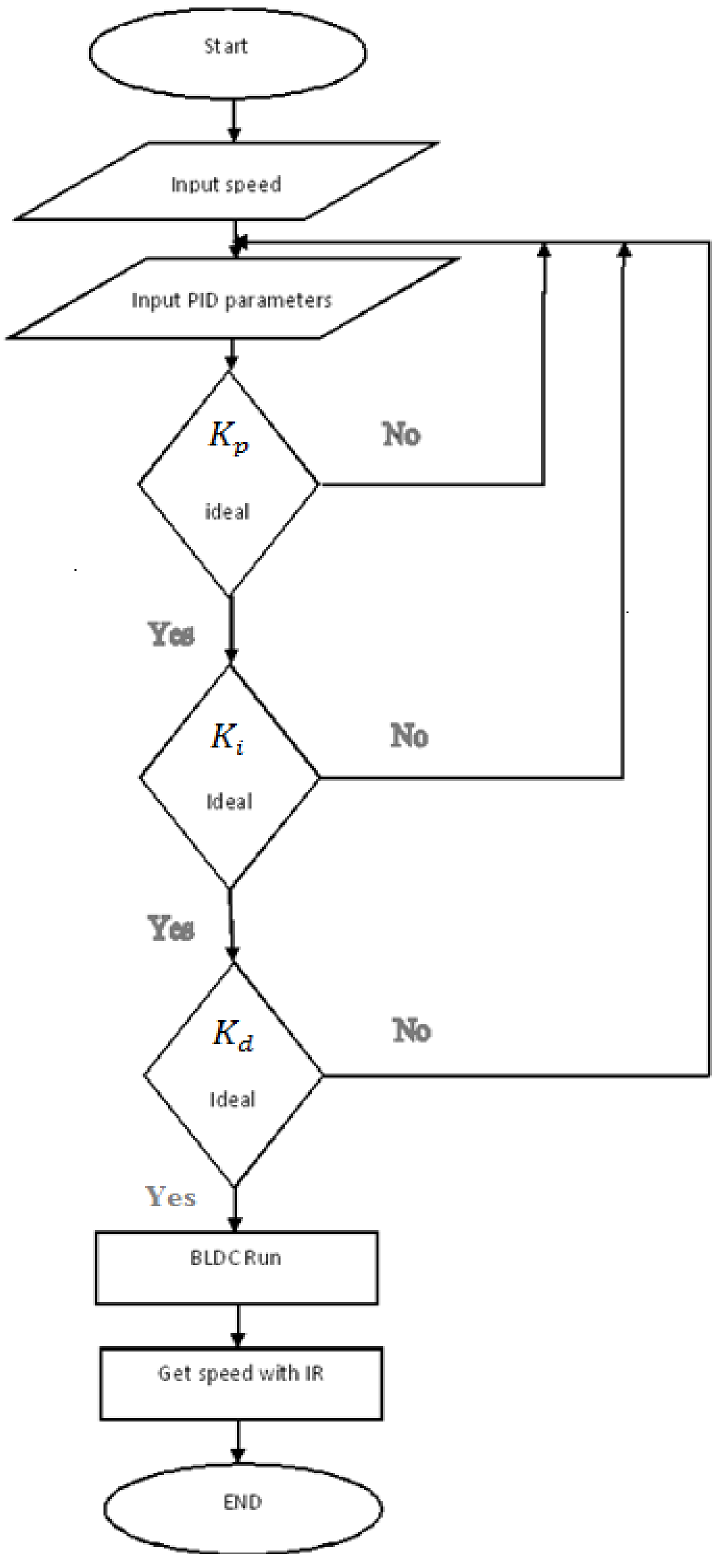

Figure 3.3: System flow chart

3.4 System with PID Controller Design

Control system model consist of the PID controller designed in Arduino and the parameter of BLDC motor. The figure below is Simulink of control system3.4. PID controller is the most widely used in closed-loop control system. The algorithm works on the error generated from the difference between the reference speed and the actual speed. PID parameters Proportional gain k_p, Integral Gain k_i, and Derivative Gain k_d affects system's overall performance. Hence choosing right parameters for a system is a difficult process and can be done by using several tuning methods which includes manual tuning, Ziegler-Nicholas tuning and Cohen-coon tuning.

Normally a mathematical model of the system is designed along with PID controller and the system performance is observed with applied set of values of PID parameters to finalize the best suited values[11] The block diagram of PID as show in figure 3.5.

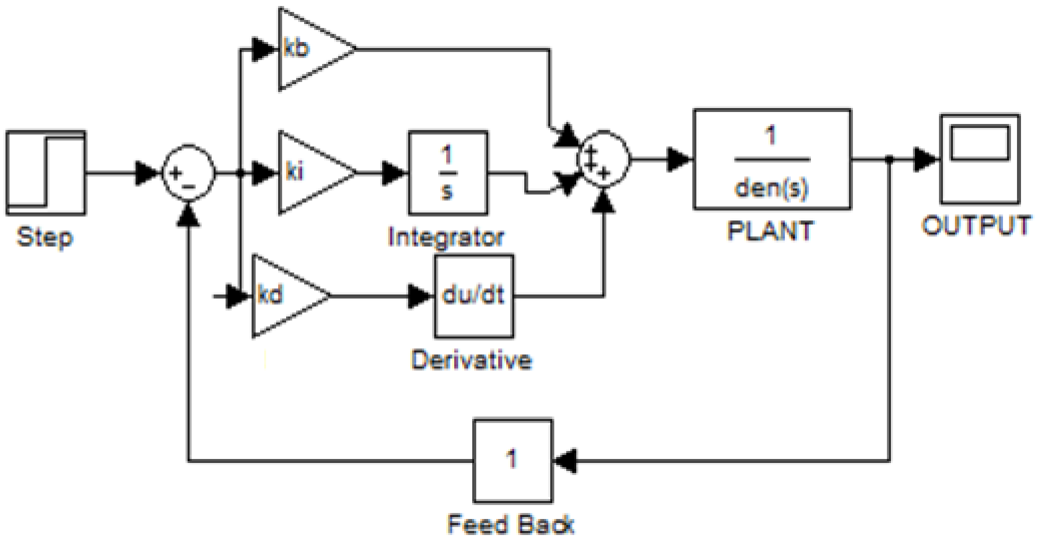

Figure 3.4: Simulink of control system.

Following are the effects of PID parameters (k_p ,k_i, and k_d):

- System Rise time will be reduced by k_p, it provides faster response in variable load condition.

- Steady state error will be reduced by k_i ; hence the motor speed is pushed near to reference speed.
- Settling time and overshoot will be reduced by k_d, hence provides faster response.

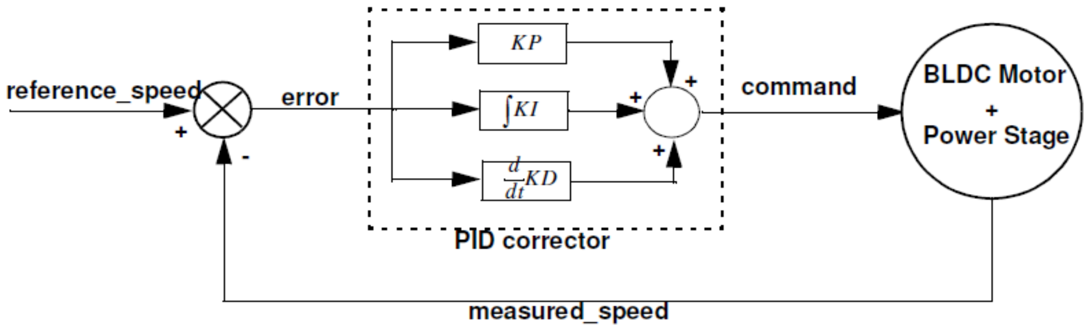

Figure 3.5: Block diagram of PID controller

Command (t) =k_p ×error (t) + k_i ×∫ error (t) d (t) + k_d ×$\frac{d}{d(t)}$ error (t) (3.14)

Error (t) = reference_ speed (t) - measured speed (t) (3.15)

Manual tuning of the gain settings is the simplest method for setting the PID controls. However, this procedure is done actively (the PID controller turned on and properly attached to the system) and requires some amount of experience to fully integrate. To tune PID controller manually, first input the transfer function in MATLAB and use the function (PID tools) for plot response of time with the transfer function, this way give the approximation values of k_p, k_i, k_d. Second Increase or decrease the proportional gain until observing oscillation in the output. After the proportional gain is set, the derivative gain can then be increased or decreased. Derivative gain will reduce overshoot and damp the system quickly to the set point value or near it, once the derivative gain is set, increase or decrease the integral gain until any offset is corrected for on a time scale appropriate for the system.

If increases values of $k_p$and $k_d$too much system well be unstable and causes big error in speed[10]. The best performance of parameter depends on designer specifications. All the parameters had got from manual tuning method are shown in Table 3.2. So the best response at number 6, it is best quickly response.

Table3.2: Experimentally PID parameters.

Number of experiment	k_p	k_i	k_d
1	9.6473	13.0625	0
2	90	55	10
3	290	380	70
4	250	350	50
5	260	360	60
6	260	500	15
7	350	400	100

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

4.1 System Components

Selection of appropriate material for a mechanical part is an essential element of all engineering projects. The main mechanical parts of the system are the base support.

4.1.1 BLDC motor

XXD-A2212 as shown in Figure 4.1 is a brushless DC motor with max efficiency 80%, its dimensions is 27.8*27 mm and shaft diameter: 3.17 mm. It can deliver a max efficiency current from 4 up to 10 AMP (>75%) and able to work in 10V.

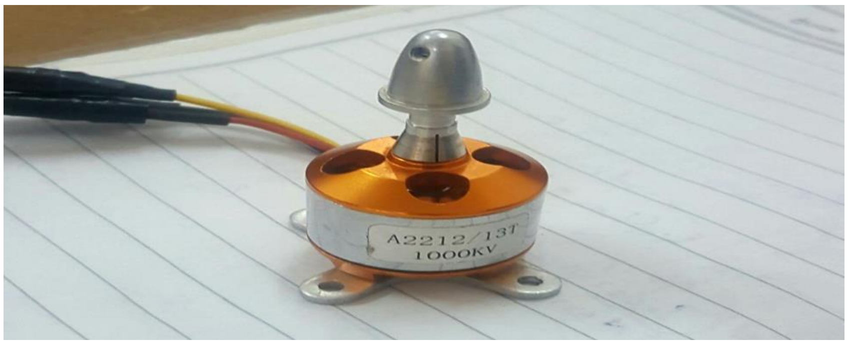

Figure 4.1: BLDC motor

4.1.2 IR sensor

The IR sensor as shown in Figure 4.2 is used to produce IR waves, IR sensor consist of IR transmitter and IR receiver the frequency of IR sensor varies depending upon its cost by using LED light at specified wavelength as required by the sensor we can look at intensity of the received light. When any object cut the light emitted by LED the light bounces back from the object to the light sensor this results a large change in the intensity which is detected by receiver of IR sensor[14].

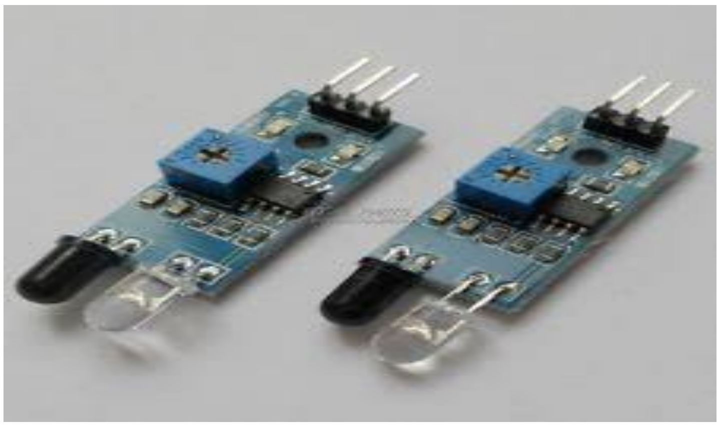

Figure 4.2: IR sensor

4.1.3 Power supply

We have used a 12V, and 5V DC power supply, the 12V is given to driver circuit and the Arduino is required 5V DC supply.

4.1.4 ARDUINO Uno

Arduino Uno as shown in Figure 4.3 is a microcontroller board based on ATmega328P microcontroller. It has 14 digital input/output pins (of which 6 can be used as Pulse Width Modulation (PWM) outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, header and a reset button. Using ID library in Arduino Uno code (See Appendix A) as a PID controller has many benefits such as:

- There are many ways to write the PID algorithm. A lot of time was spent making the algorithm in this library as solid as any found in industry.
- When using the library all the PID code is self-contained. This makes your code easier to understand. It also lets you do more complex stuff, like having 8 PIDs in the same program[8].

Figure 4.3: Arduino UNO

4.1.5 Bluetooth module

Bluetooth module as shown in Figure 4.4 is a wireless standard for data exchange over short distances it creates Personal Area Networks (PANs) with high security level which is an essential factor in this system, Bluetooth operates in the range of 2400-2483.5 MHZ (includes guard bands)[15].

Figure 4.4: Bluetooth module

4.1.6 ESC driver:

An Electronic Speed Controller (ESC) for brushless DC Motors as shown in Figure 4.5. This ESC can drive motors which consume up to 30A current. It works on a power supply. It can be used to control our brushless motors with a power supply (make sure motor doesn't draw more than 30A).

Specifications

- Running current: 30A.
- Output: Continuous 30A.
- Input Voltage: DC 6-12.6V.
- Has Low Voltage Protection Mode.

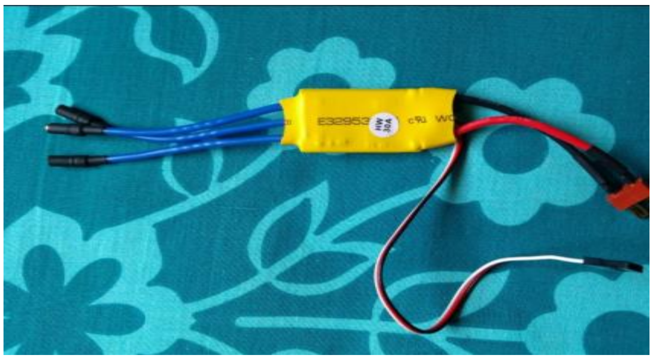
Figure 4.5: ESC driver

4.2 Software Type

First we define the PID parameters (K_p,K_i , K_d), and select a random quantities by it proper response which give us a small error rating and a proper speed (a speed that is near to the exact speed) with tuning PID. We will enter these parameters via smart phone into MATLAB application, the code will control our motor by using MATLAB application, this will happen when we enter the PID parameters and the speed of the motor that we measured by the sensor, the motor will run according to the speed input and some of these inputs will return as a closed loop feedback to Arduino.

4.3 System Implementations

The Bluetooth module connected to pin 1, 2(TX, RX) respectively, ESC driver connected to pin 3 and connected to power supply with 12V output, 18 AMP, IR sensor connected pin 10,and all of that is connected to 5V, and ground, as shown in Figure 4.6.

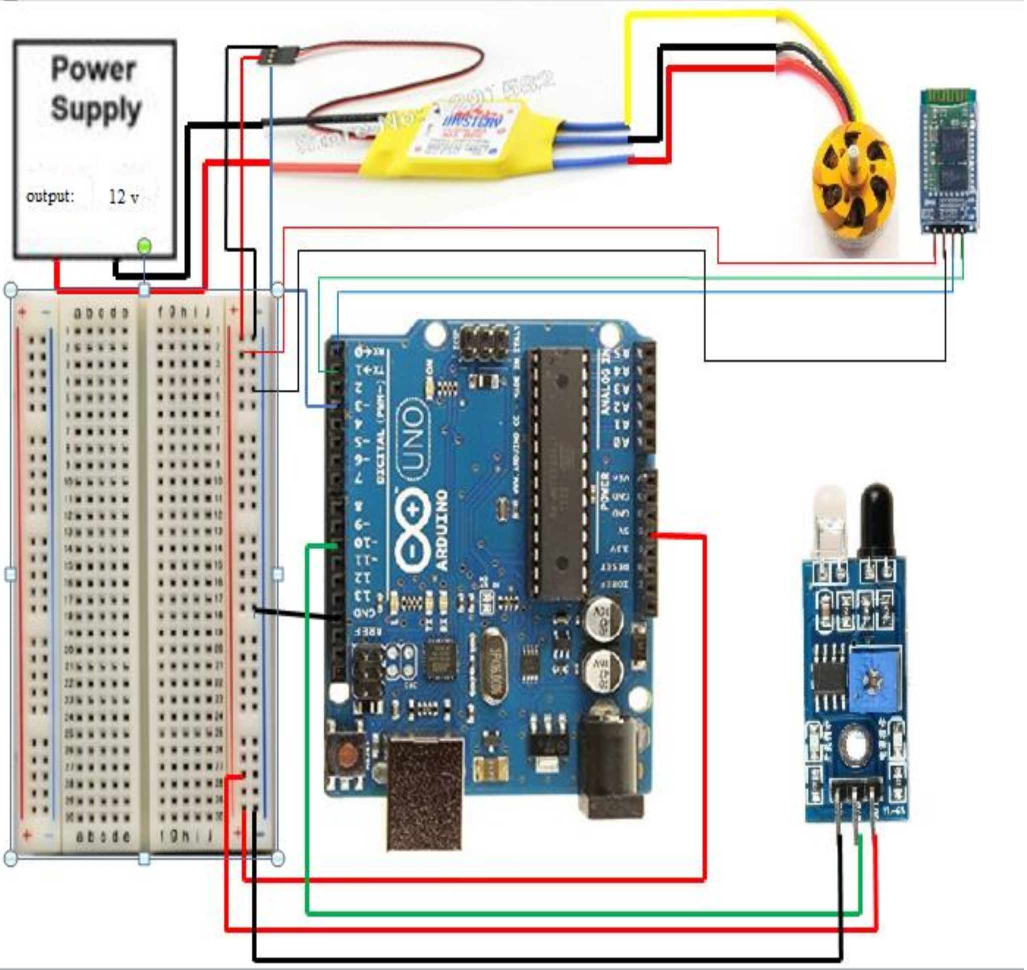

Figure 4.6: The electrical circuit connections

4.4 System Experimental Results

This section demonstrates the results of a real time with amplitude plotting using MATLAB to monitor the control of BLDC motor using design of PID controller with manual tuning method. By adjusting a proper value of PID parameters, with K_p=9.6473, K_i=13.0625, and K_d=0. The system rises to amplitude equal 1.04 with small overshooting and oscillates, the setting time period is long. This system has a large delay time and large rise time. Figure 4.7 shows the system characteristic.

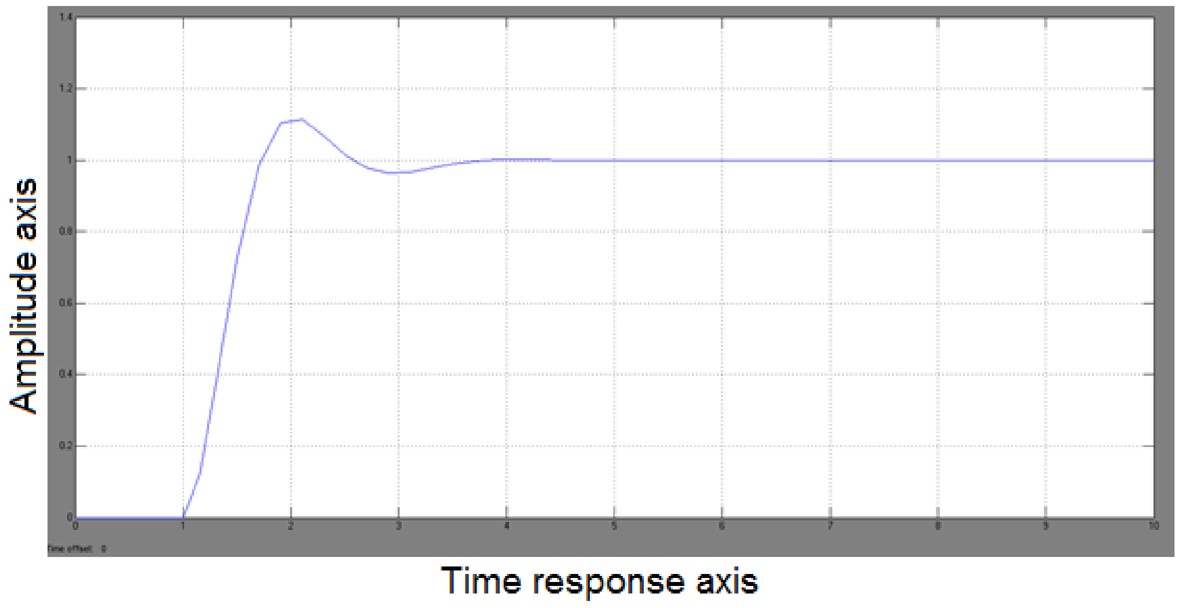

Figure 4.7: System time response with K_p=9.6473, K_i=13.0625, K_d=0

By substituting the PID parameter with K_p=260, K_i=360, K_d=60, the system rises to amplitude equal 1.13 with large overshooting oscillates in small setting time period. This system has a small delay time and small rise time and has a better setting time and has a worse overshoot when it compared with the first one Figure 4.8 shows the system characteristic.

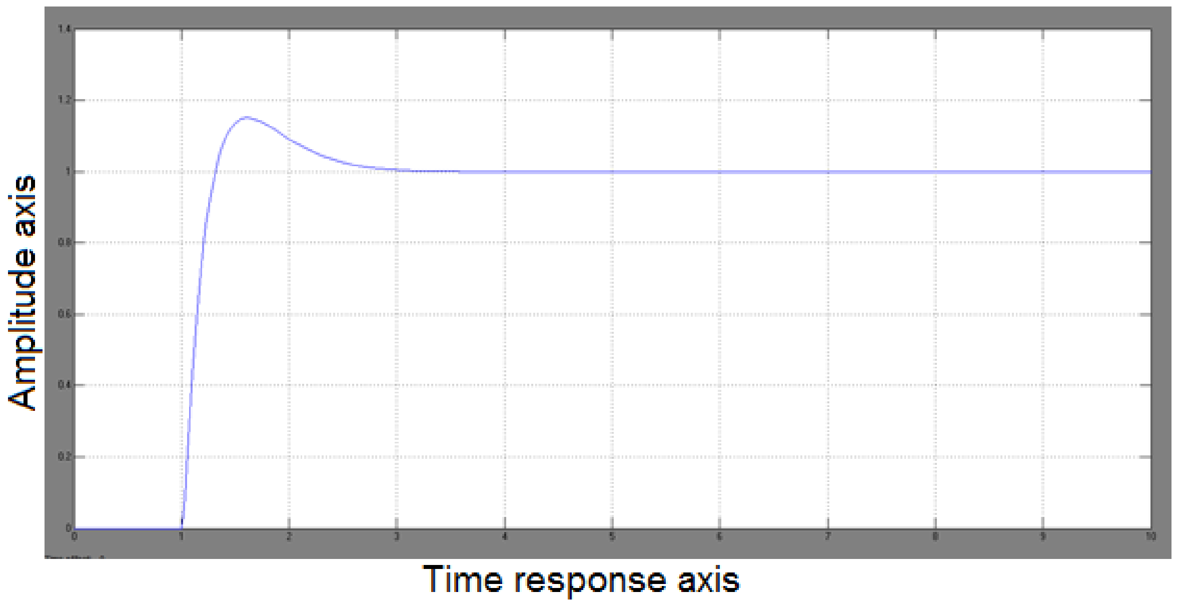

Figure 4.8: System time response with K_p=260, K_i=360, K_d=60

By substituting the PID parameter with K_p=260, K_i=500, K_d=15, the system rises to amplitude equal 1.117 with largest overshooting and oscillates in small setting time period. This system has the smallest delay time and the smallest rise time and has the best setting time and this when it compared to all of them. This is the chosen parameter because it has the best setting and quickly response Figure 4.9 shows the system characteristic.

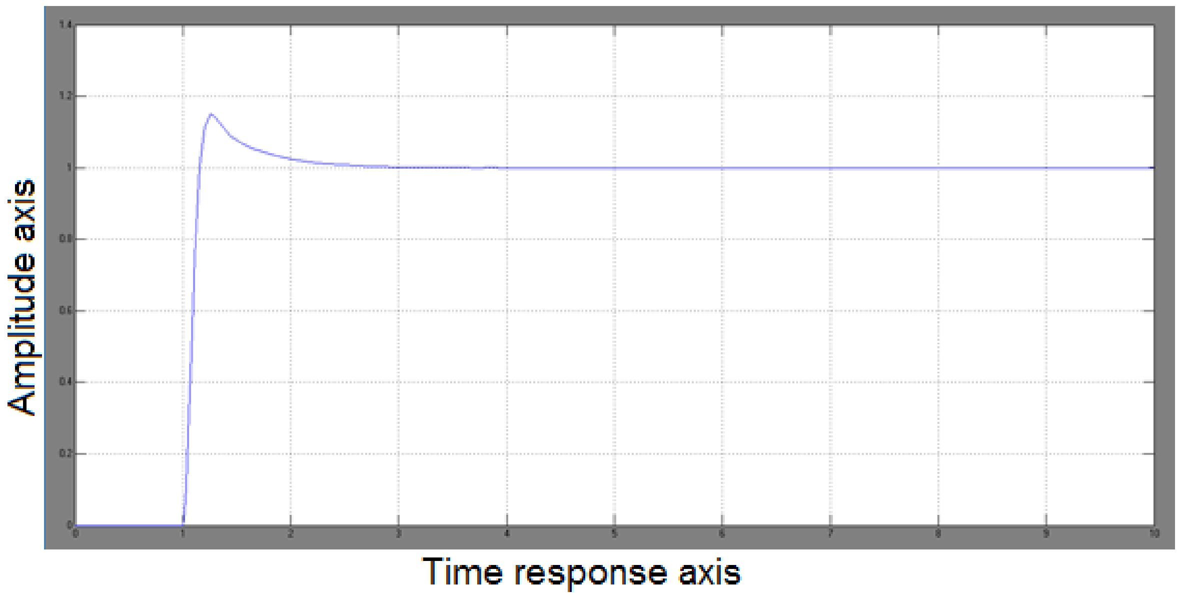

Figure 4.9: System time response with K_p=260, K_i=500, K_d=15

By substituting the PID parameter with K_p=230, K_i=200, K_d=0 as shown in Figure 4.10, and K_p=80, K_i=50, K_d=0 as shown in Figure 4.11 these systems rise to amplitude above 1.4 and above to 1.2 respectively with very long overshooting oscillates in small setting time period. These systems have the smallest delay time and the smallest rise time.

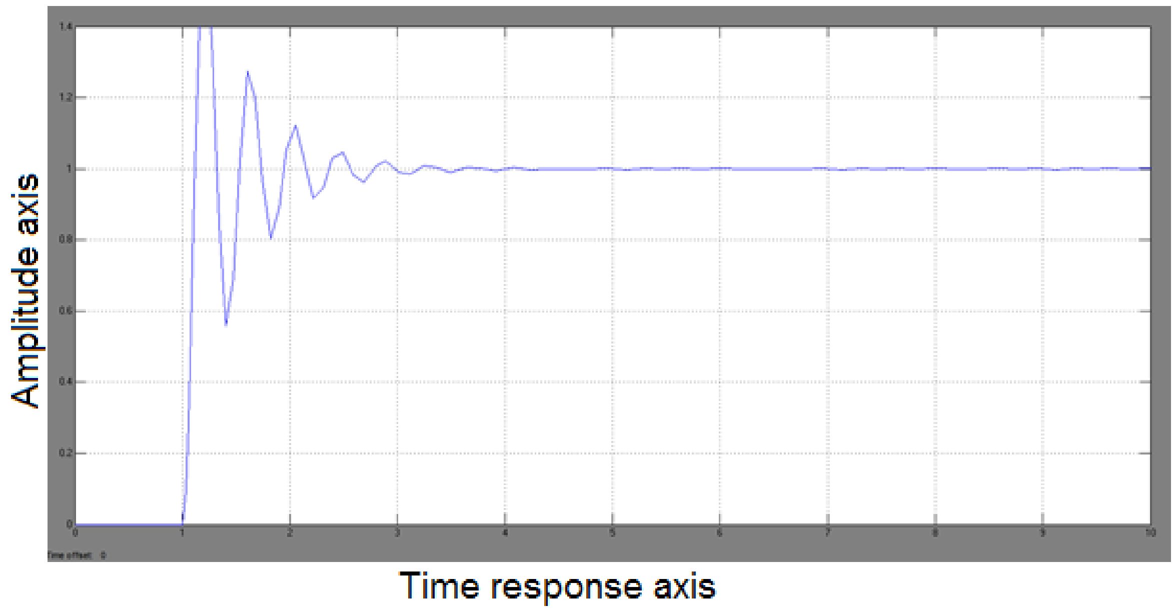

Figure 4.10: System time response with K_p=230, K_i=200, K_d=0

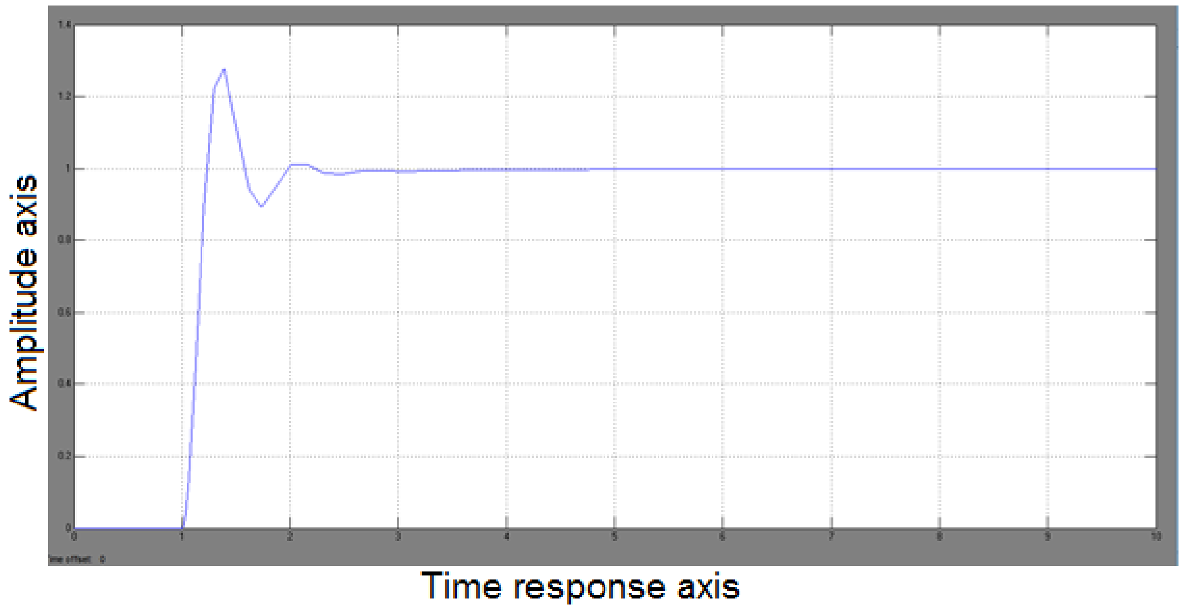

Figure 4.11: System time response with K_p=80, K_i=50, K_d=0

CHAPTER FIVE
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

A mathematical model of speed and control of BLDC motor was developed using physical and electrical laws. A simplified mathematical model was derived using system parameters. The controller parameters values (k_p, $k_i$andk_d) were obtained using manual tuning method from practical model. From experimental results, it is found that the best controller parameters given the best response of the system are: k_p=260, k_i=500 and k_d=15.

5.2 Recommendations

- Use of fuzzy logic control instead of PID controller. Because the PID controller still has low robust ability compared with the robust controller when the system encounters to multiple challenges from the operating environment of the system, such as temperature, weather, power surge.
- Investigation of IR sensor sensitivity, and exchange by any sensor have best frequency.

REFERENCES

[1] M. Mubeen, "Brushless DC Motor Primer," *Motion Tech Trends,* 2008.

[2] R. Condit, "Brushed DC Motor Fundamentals," *Microchip Technology Inc, http://ww1. microchip. com/downloads/en/AppNotes/00905a. pdf,* 2004.

[3] P. Yedamale, "Brushless DC (BLDC) motor fundamentals," *Microchip Technology Inc,* vol. 20, pp. 3-15, 2003.

[4] N. Rethinam and P. Abhishek, "Techniques for controlling a brushless DC (BLDC) electric motor," ed: Google Patents, 2017.

[5] K. Hameyer and R. J. Belmans, "Permanent magnet excited brushed DC motors," *IEEE Transactions on industrial electronics,* vol. 43, pp. 247-255, 1996.

[6] G. Gridling and B. Weiss, "Introduction to microcontrollers," *Vienna University of Technology Institute of Computer Engineering Embedded Computing Systems Group,* 2007.

[7] B. L. Toll, J. A. Miller, and M. A. Fetterman, "Method and apparatus for combining micro-operations to process immediate data," ed: Google Patents, 2011.

[8] A. G. Smith, "Introduction to Arduino," *Reference book, September,* vol. 30, pp. 1-172, 2011.

[9] M. Schmidt, *Arduino*: Pragmatic Bookshelf, 2011.

[10] H. Panagopoulos, K. Astrom, and T. Hagglund, "Design of PID controllers based on constrained optimisation," *IEE Proceedings-Control Theory and Applications,* vol. 149, pp. 32-40, 2002.

[11] K. J. Åström and T. Hägglund, *Advanced PID control*: ISA-The Instrumentation, Systems and Automation Society, 2006.

[12] "https://en.wikipedia.org/wiki/Brushless_DC_electric_motor."

[13] https://en.wikipedia.org/wiki/PID_controller.

[14] M. Matson and J. Dozier, "Identification of subresolution high temperature sources using a thermal IR sensor," *Photogrammetric Engineering and Remote Sensing,* vol. 47, pp. 1311-1318, 1981.

[15] R. Pahuja and N. Kumar, "Android Mobile Phone Controlled Bluetooth Robot Using 8051 Microcontroller," *International Journal of Scientific Engineering and Research,* vol. 2, 2014.

APPENDIX A

Arduino C code for speed control of BLDC motor

```
////// BLDC Project ///////

#include <PID_v1.h>

#include <EEPROM.h>

#include <Servo.h>

Servo esc_signal;

#include <boarddefs.h>

#include <IRremote.h>

#include <IRremoteInt.h>

#include <ir_Lego_PF_BitStreamEncoder.h>

#include <IRremote.h>

#include <Wire.h>

//#define PIN_INPUT 0

#define PIN_OUTPUT 3

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

//Define Variables we'll be connecting to

double Setpoint, input, Output;

//////////////////
```

```
unsigned long speed_bldc;

unsigned char  form1[500];

int frame_bit;

int Factor;

int x;

int ir_value;

unsigned long speed_response, data1, data2, data3, data4;

int counter_speed =0;

int counter_speed2 =0;

int counter_speed3 =0;

//Specify the links and initial tuning parameters

double  SPEED, kp, ki, kd;

PID myPID(&input, &Output, &Setpoint, kp, ki, kd, DIRECT);

void Send(){

  delay(1000);

  Serial.print('*');

  Serial.print(data1);

   Serial.print(',');

    Serial.print(data2);

     Serial.print(',');

      Serial.print(data3);
```

```
       Serial.print(',');
        Serial.print(data4);
         Serial.println('#');
}
/////////////////////////read eeprom//////
void read_eeprom(){
  for(int j=1;j>=60;j++){
  form1[j]= EEPROM.read(0x09+j);
  }
}
//////////////////////////////////read frame///////
void read_fram(){
  while(Serial.available()) {
          frame_bit = Serial.read();
  if(frame_bit=='*'){
   int  i=1;
    do{
          if (Serial.available()) {
 frame_bit = Serial.read();
                      form1[i]=frame_bit;
                     EEPROM.write(0x09+i,form1[i]);
```

```
                i++;
      }
    }while(frame_bit!='#');}
            delay(1);
          frame_bit=0;
}}
//////
void IR(){
    if (irrecv.decode(&results)) {
  Serial.println(results.value/42949667, DEC);
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
void speed_config(unsigned long SPEED, unsigned long kp, unsigned long
ki, unsigned long kd){
  if(kp >= 0 && kp <= 50){
    if(ki >= 0 && ki <= 40){
      if(kd >= 0 && kd <= 5){
        counter_speed++;
        if(counter_speed >= 30){
```

```
      counter_speed = 30; }
   if(counter_speed <= 10){
    Send();
IR();
speed_response = SPEED + 15;
esc_signal.write(speed_response);
Serial.println(speed_response);
 Serial.println(counter_speed);
IR();
delay(20);
speed_response = SPEED - 17;
esc_signal.write(speed_response);
Serial.println(speed_response);
IR();
delay(20);
speed_response = SPEED + 12;
esc_signal.write(speed_response);
Serial.println(speed_response);
IR(); }
   if(counter_speed > 10){
    Send();
```

```
       speed_response = SPEED;

       esc_signal.write(speed_response);

       Serial.println(speed_response);

            results.value = SPEED + 2;

Serial.println(results.value/42949667, DEC);

    }}}}

if(kp >= 50 && kp <= 150){

  if(ki >= 40 && ki <= 100){

    if(kd >= 0 && kd <= 5){

      counter_speed2++;

      if(counter_speed2 > 30){

        counter_speed2 = 30;

      }

      if(counter_speed2 <=7){

        Send();

  IR();

  speed_response = SPEED + 7;

  esc_signal.write(speed_response);

  Serial.println(speed_response);

  IR();

  delay(20);
  speed_response = SPEED - 8;

  esc_signal.write(speed_response);

  Serial.println(speed_response);

  IR();

  delay(20);

  speed_response = SPEED - 3;

 esc_signal.write(speed_response);

 Serial.println(speed_response);

 IR();

}

    if(counter_speed2 > 7){

      Send();

     speed_response = SPEED;

 esc_signal.write(speed_response);

 Serial.println(speed_response);

     results.value = SPEED + 2;

Serial.println(results.value/42949667, DEC);}} } }

  if(kp > 0 && kp <= 100){

  if(ki >= 20 && kd>= 20){

    if(ki = kd){

      counter_speed3++;
    if(counter_speed3 >= 30){
      counter_speed3 = 30;
    }
if(counter_speed3 <= 2){
Send();
IR();
speed_response = SPEED + 2;
esc_signal.write(speed_response);
Serial.println(speed_response);
IR();
delay(20);
speed_response = SPEED -1;
esc_signal.write(speed_response);
Serial.println(speed_response);
IR();
    }
    if(counter_speed3 > 2){
 Send();
 speed_response = SPEED;
esc_signal.write(speed_response);
Serial.println(speed_response);
```

```
  Serial.println(results.value/42949667, DEC); } } } }
void setup()
{
  Serial.begin(9600);
  //initialize the variables we're linked to
//  Input = analogRead(PIN_INPUT);
//  Setpoint = 100;
esc_signal.attach(3);
esc_signal.write(30);    //ESC arm command. ESCs won't start unless input
speed is less during initialization.
  delay(3000);  //ESC initialization delay.
  //turn the PID on
  myPID.SetMode(AUTOMATIC);
}
void loop()
{
//  Input = analogRead(PIN_INPUT);
  myPID.Compute();
//  analogWrite(PIN_OUTPUT, Output);
  ////////
read_eeprom();
```

```
read_fram();

data1 = ((form1[1]-48)*100 + (form1[2]-48)*10 + (form1[3]-48));

data2 = ((form1[5]-48)*100 + (form1[6]-48)*10 + (form1[7]-48));

data3 = ((form1[9]-48)*100 + (form1[10]-48)*10 + (form1[11]-48));

data4 = ((form1[13]-48)*100 + (form1[14]-48)*10 + (form1[15]-48));

speed_config(data1,data2,data3,data4);

delay(15);

//Send();

//x= results.value/42949667;

//Factor = x-33;

//Serial.println(Factor,DEC); }
```

APPENDIX B

MATLAB m.file for PID tune

```
s=tf('s')

s =
   s

Continuous-time transfer function.

sys = 0.05/(s+2.5)

sys =

   0.05
  -------
  s + 2.5

Continuous-time transfer function.

pidtool(sys)
t= 0:0.05:5;
step(sys,t)
grid
pidtool(sys)
pidtool(sys)
sys1=1/(s^2+(4.75*s)+5.5)
 sys1 =
         1
  ------------------
  s^2 + 4.75 s + 5.5
Continuous-time transfer function.
step(sys1,t)
grid
pidtool(sys1)
num=[0 0.05]
num =  0    0.0500
den=[1 2.5]
 den =
    1.0000    2.5000
numc=[0 0 1]
numc =
     0     0     1
denc=[1 4.75 5.5]
denc =

    1.0000    4.7500    5.5000

[c1,x1,t]=step(num,den,t);
[c2,x2,t]=step(numc,denc,t);
```