بسم الله الرحمن الرحيم

SUDAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF GRADUATE STUDIES

Arduino Universal Serial Bus (USB) Interfacing with MATLAB Control Toolbox

الربط البيني للأردوينو مع مكتبة التحكم في الماتلاب بواسطة الناقل التسلسلي العالمي

A Thesis Submitted in Partial Fulfillment for the Requirements of the Degree of M.Sc. in Mechatronics Engineering

Prepared By:

Rufaidah Abdallah Ibrahim Mohammed

Supervised By:

Dr. Awadalla Taifour Ali Ismail

July 2018

الآيـة

قَالَ تَعَالَى:

﴿ فَبَدَأَ بِأَوْعِيَتِهِمْ قَبْلَ وِعَاءِ أَخِيهِ ثُمَّ اسْتَخْرَجَهَا مِنْ وِعَاءِ أَخِيهِ كَذَلِكَ كِدْنَا لِيُوسُفَ مَا كَانَ لِيَأْخُذَ أَخَاهُ فِي دِينِ الْمَلِكِ إِلَّا أَنْ يَشَاءَ اللَّهُ نَرْفَعُ دَرَجَاتٍ مَنْ نَشَاءُ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ﴾

سورة: يـوسف الآية(76)

Dedication

She gives me unconditional love, she support me, her love is the fuel for me to go and try

…................My mother.

He makes me feel like I can do anything

.... My father.

.... For my friends.

..................................For my teachers.

.......................For all everyone know me, I dedicate this research.

Acknowledgement

In the Name of Allah, the Beneficent, the Merciful first praise is to Allah, the Almighty, on whom ultimately we depend for sustenance and guidance.

I wish to express my deepest gratitude to my supervisor: **Dr. Awadalla Taifour Ali** for his excellent guidance, encouragement and support were really valuable throughout this research.

This research would never have been possible without devotion, cooperation and an endless will to help of **Eng. Yassir Mohammed Alkasim**, I would like to express my deep and sincere gratitude for his continuous support me, guidance and academic cooperation.

I also offer my regards to the faculty of engineering, Omdurman Islamic University, its leadership and the staff for providing me with an academic base, which has enabled me to take up this study. I deepest thanks and appreciation and gratitude to the head of Electrical and Electronic Department **Ust. Husam ELdin Alfadel**.

Finally, I so thankful to express the profound gratitude to my family, teachers and friends who supported me to complete the research successfully.

Abstract

Nowadays, the Arduino are widely used in daily applications because it is easy to be deal with. Besides, MATLAB software tool is proper tool be used as interface with hardware such as Arduino. The main aim of this research is to design a Direct Current (DC) motor speed controller circuit controlled and monitored by MATLAB program designed in SIMULINK and Graphical User Interface (GUI) environment.

This thesis focuses on the DC motor speed control by varying the duty cycle of Pulse Width Modulation (PWM) signal via Arduino, which is used to generate PWM signals assisted by MATLAB software. PWM speed control is desirable due to its high power efficiency compare and with another method of speed control like frequency control, current and voltage control. The motor averages the input duty cycle into a constant speed which is directly proportional to the percent duty cycle. The MATLAB sends PWM signal to Arduino through Universal Serial Bus (USB), Arduino will boost the PWM signal to the driver to control the motor. Both SIMULINK and GUI programs were able to control and monitor the motor speed which is most important feature in engineering control applications as data acquisition and research tools.

مستخـــــــــــــــلص

في الوقت الحاضر يستخدم أردوينو على نطاق واسع في التطبيقات اليومية لسهولة التعامل معه. الى جانب ذلك يعتبر الماتلاب من أهم التطبيقات الهندسية وأدوات البحث والبرمجيات المناسبة التي تستخدم للربط البيني مع الأجهزة الاخرى مثل أردوينو. الهدف الرئيسي من هذا البحث هو تصميم دائرة للتحكم في سرعة محرك التيار المستمر وعرض النتائج بإستخدام برنامج الماتلاب من خلال واجهة السميولينك وواجهة المستخدم الرسـوميــة Graphical User Interface (GUI) .

وتركز هذه الأطروحة على التحكم في سرعة المحرك عن طريق تغيير إشارة عرض النبضة (PWM) بإستخدام أردوينو والذي يقوم بتوليد إشارات PWM بمساعدة برنامج الماتلاب. التحكم في السرعة باستخدام تقنية PWM هي طريقة مرغوب فيها نظراً للكفاءة العالية مقارنة مع الطرق الأخرى المستخدمة للتحكم في سرعة المحركات مثل التحكم في التردد، والتحكم في التيار والجهد. يقوم الماتلاب بإرسال إشارة PWM إلى أردوينو من خلال وصلة (USB) للتحكم في سرعة المحرك (DC Motor) بواسطة ناقل الحركة Driver. وبواسطة كل من SIMULINK و GUI تم التحكم في سرعة المحرك وعرض النتائج وتعتبر هذه البرامج من أهم أدوات البحث في تطبيقات التحكم الهندسي مثل الحصول على البيانات DAQ.

Table of Contents

List of Figures

List of Tables

Abbreviation

Abbreviation	Meaning
DC	Direct Current
GUI	Graphical User Interface
PWM	Pulse Width Modulation
USB	Universal Serial Bus
DAQ	Data Acquisition
ODE	Ordinary Differential Equation
IDE	Integrated Development Environment
I/O	Input/output
LCD	Liquid Crystal Display
GPS	Global Position System
GSM	Global System for Mobile Communication
DSP	Digital Signal Processing
ASCII	American Standard Code for Information Interchange
API	Application Program Interface
IR	Infra-Red
RPM	Revolutions Per Minute
IC	Integrated Circuit
PC	Personal Computer
GUIDE	Graphical User Interface Development Environment
BLDC	Brushless Direct Current
PID	Proportional Integral Derivative
HIL	Hardware In the Loop
TTL	Transistor Transistor Logic
FRC	Flat Ribbon Cable
LPF	Low Pass Filter

Chapter One

Introduction

Chapter One
Introduction

1.1 General Review

Data Acquisition (DAQ) systems are used by most engineers and scientists for laboratory research, industrial control, test and measurement to input and output data to and from computer. DAQ system is a system which acquires the data i.e. input/output parameters from the field with the help of sensors associated with the system, it analyses the data and generate control action required for operating and also monitor the acquired data on computer. The DAQ system is basic need for controlling and monitoring of a Mega system in modern industries.

The field of data acquisition encompasses a very wide range of activities. At its simplest level, it involves reading electrical signals into a computer from some kind of sensor. These signals may represent the state of a physical process i.e. position and orientation of machine tools, furnace temperature, size and shape of a manufactured component etc. The acquired data may have to be stored, printed or displayed. Often the data have to be analyzed or processed in some way in order to generate further signals for controlling external equipment or for interfacing to other computers [1].

1.2 Problem Statement

Interfacing Arduino based microcontroller with DAQ software tool in MATLAB in order to have maximum usability and applying a control theories. The DAQ implementation in MATLAB /SIMULINK and GUI environments to make flexibility in algorithm usage. These DAQ tools as well as the hardware circuits based on Arduino microcontrollers as the challenges of this research, and the problem to be solved.

1.3 Objectives

- To develop a low cost DAQ system for controlling and monitoring a system using Arduino-UNO controller and MATLAB (SIMULINK and GUI).
- To interface between the computer and DC motor using Arduino.

- To investigate the communication between Arduino-UNO board and MATLAB (SIMULINK and GUI).
- To design control circuit to test for control operations such as control experiments in the speed of DC motors.

1.4 Methodology

Combined theoretical and practical methodologies were adopted in this research. A theoretical investigation of control theory and applied control, besides software implementation in both MATLAB and Arduino based microcontroller. Practical implementation of DAQ circuit and MATLAB software tools.

1.5 Research Layout

This thesis consists of five chapters including this chapter one. Chapter two presents an introduction of control systems, theoretical background, and focuses on the literature review. Chapter three explains and discusses the methodology that have been used in order to complete this study, there are two parts in this chapter which are the software development and hardware implementation. Chapter four discusses about the result obtained and limitations of the study, all discussion are concentrating on the result and performance of the developed system. Finally Chapter five states the conclusion of the development of the study and discusses the recommendations for future work.

Chapter Two

Literature Review

Chapter Two
Literature Review

2.1 Introduction

A control system is a collection of components working together under the direction of some machine intelligence. In most cases electronic circuits provide the intelligence and electromechanical components such as sensors and motors provide the interface to the physical world. A good example is the modern automobile various sensors supply the on-board computer with information about the engine’s condition. The computer then calculates the precise amount of fuel to be injected into the engine and adjusts the ignition timing. The mechanical parts of the system include the engine, transmission, wheels, and so on. To design diagnose, or repair these sophisticated systems, you must understand the electronics, the mechanics, and control system principles. In days past, so-called automatic machines or processes were controlled either by analog electronic circuits, or circuits using switches, relays, and timers. Since the advent of the inexpensive microprocessor, more and more devices and systems are being redesigned to incorporate a microprocessor controller. Examples include copying machines, soft-drink machines, robots, and industrial process controllers. Many of these machines are taking advantage of the increased processing power that comes with the microprocessor and, as a consequence, are becoming more sophisticated and are including new features. Taking again the modern automobile as an example, the original motivation for the on-board computer was to replace the mechanical and vacuum-driven subsystems used in the distributor and carburetor. Once a computer was in the design, however, making the system more sophisticated was relatively easy. For example, self-adjusting fuel/air ratio for changes in altitude. Also features such as computer-assisted engine diagnostics could be had without much additional cost. This trend toward computerized control will no doubt continue in the future [2].

The basic of control system consists of three components: input, logic operation and output or decision device. Input is the cause parameter on which the control system acts, the logic operation is the intended or desired operation to perform on the input for generating a new output state, and the output is drive parameter which actuates the end component to perform the desired task. These can be open loop or closed loop control system depends on output feedback. Block diagram reduction helps in analysis and simplification of control system.

2.1.1 Open loop and closed loop system

There are two common classes of control systems, open loop control systems and closed loop control systems. In open loop control systems output is generated based on inputs. An open-loop control system utilizes an actuating device to control the process directly without using feedback as shown in Figure 2.1.

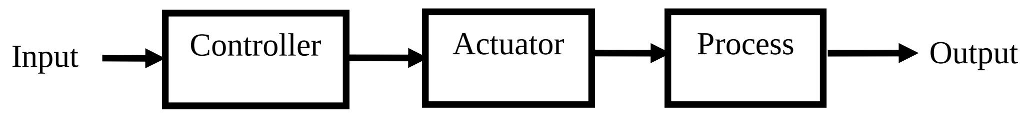


Figure 2.1: Open loop control system

In closed loop control systems current output is taken into consideration and corrections are made based on feedback. A closed loop system is also called a feedback control system. The human body is a classic example of feedback systems**.** A closed-loop control system uses a measurement of the output as feedback of this signal to compare it with the reference or command signal as illustrated in Figure 2.2.

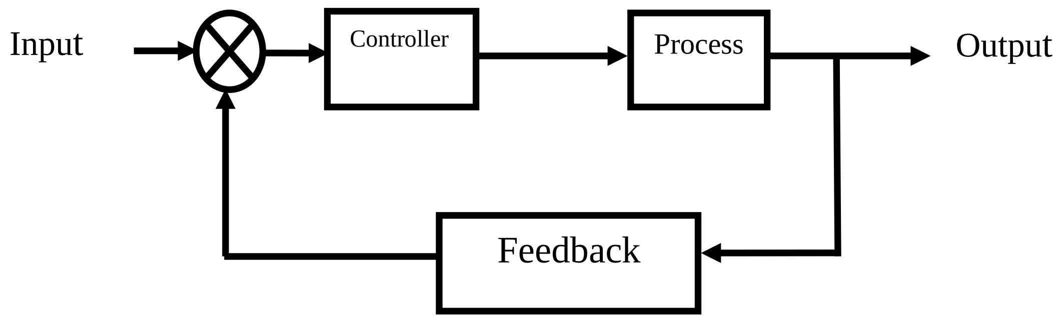


Figure 2.2: Closed-loop feedback system

2.1.2 Classical and modern control system

There are essentially two methods to approach the problem of designing a new control system: the classical approach, and the modern approach. Classical and modern control methodologies are named in a misleading way, because the groups of techniques called “classical” were actually developed later than the techniques labeled "modern". However, in terms of developing control systems, modern methods have been used to great effect more recently, while the classical methods have been gradually falling out of favor. Most recently, it has been shown that classical and modern methods can be combined to highlight their respective strengths and weaknesses. Classical methods are methods involving the Laplace transform domain. Physical systems are modeled in the so-called "time domain", where the response of a given system is a function of the various inputs, the previous system values, and time. As time progresses, the state of the system and its response change. However, time-domain models for systems are frequently modeled using high-order differential equations which can become impossibly difficult for humans to solve and some of which can even become impossible for modern computer systems to solve efficiently. To counteract this problem, integral transforms, such as the Laplace transform and the Fourier Transform, can be employed to change an Ordinary Differential Equation (ODE) in the time domain into a regular algebraic polynomial in the transform domain. Once a given system has been converted into the transform domain it can be manipulated with greater ease and analyzed quickly by humans and computers like. Modern control methods, instead of changing domains to avoid the complexities of time-domain ODE mathematics, converts the differential equations into a system of lower-order time domain equations called state equations, which can then be manipulated using techniques from linear algebra.

2.2 Direct Current Motor

DC motor is used in many industrial applications that require adjustable speeds. In uses requiring quick stops, a DC motor can minimize the size of mechanical brake or make it unnecessary. This is done by dynamic braking (motor generated energy fed to a resistor grid), or by regenerative braking (motor generated energy returned to the ac supply).

2.2.1 Advantages of DC motor

- Reasonably inexpensive
- Easy to control
- Adaptable

2.2.2 Disadvantages of DC motor

- Brushes eventually wear out
- Brushes create electrical interference

2.2.3 Classification of DC motor

DC motors are commonly constructed with wound rotors and either wound or permanent magnet stators. Wound-field motors use an electromagnet called the field winding to generate the magnetic field. The only other way to generate a magnetic field is with permanent magnets. DC motor can be classified according to the method of connecting the field windings with the armature windings so the main types of DC motors are:

- Series-wound DC motor
- Shunt-wound DC motor
- Compound-wound DC motor

2.2.4 Basic DC motor equation

The DC motor consists of the field and an armature, the field being excited by DC windings, which set up a magnetic flux ($\emptyset$) linking the armature as shown in Figure 2.3 part a. The function of the commutator and brushes is to ensure that the current in armature conductors under any pole is in a given direction irrespective of the speed of the armature. The commutator acts as mechanical frequency-changer to change the alternating current in a given armature conductor to the direct in the brush lead. The circuit diagram of a DC motor is shown in Figure 2.3 part b. When rotating, an internal voltage (E) or back E.M.F (E_b) is generated in the armature by virtue of its rotation in magnetic flux. The terminal armature voltage (V) will differ from E by the internal voltage drop. The load torque will give rise to a current in the armature, the current level being a function of the torque and magnetic flux values. The basic DC motor equations are:

$$V = E_b + I_a R_a \tag{2.1}$$

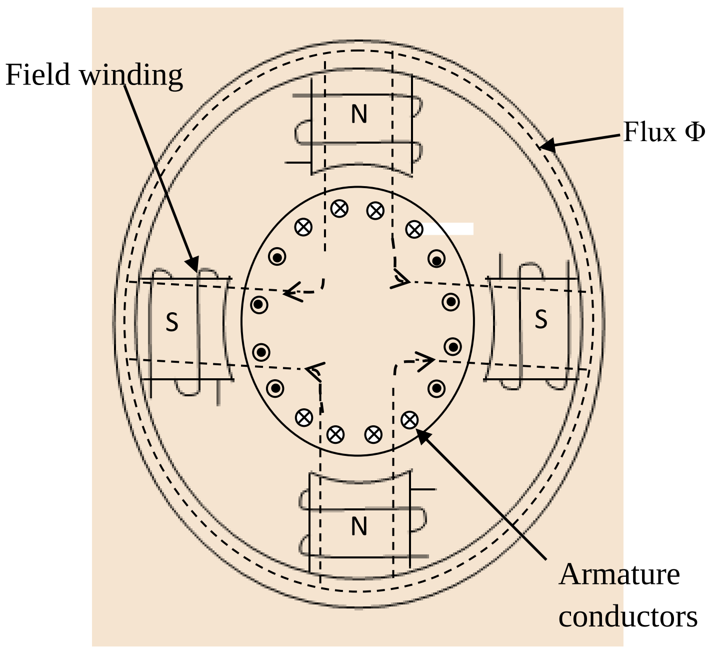
$$E_b = k_1 N \emptyset \tag{2.2}$$

$$T = k_2 I_a \emptyset \tag{2.3}$$

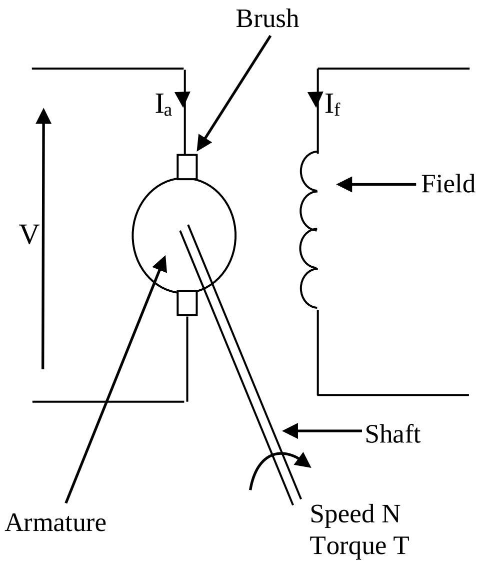
$$\emptyset = k_3 I_f \tag{2.4}$$

Cross mechanical power $= \mathrm{TN} = \mathrm{E_b I_a}$ (2.5)

Where T is the torque, $\mathrm{I_a}$ and $\mathrm{I_f}$ are the armature and field currents respectively, $\mathrm{R_a}$ is the armature resistance, N is speed (rad/s), and $\mathrm{k_1}, \mathrm{k_2}$ and $\mathrm{k_3}$ are constants of proportionality [3].



(a) Construction (4-poles)



(b) Circuit representation

Figure 2.3: The DC machine

2.2.5 Speed control of DC motor

Speed control of a motor refers to the intentional change of the motor speed to a value needed for performing the required work. The natural change in speed due to change in load on the shaft is not included in the concept of speed control. The speed of wound-field motors is controlled by varying the voltage to the armature or field windings. This particular series of motors is available from 0.25 to 1hp and at several standard rated speeds. The rated speed of a motor is the speed when it is supplying the rated horsepower when the motor is unloaded, it will go faster than the rated speed. These motors are designed to run at 90$\mathbf{V_{dc}}$ because 90V is about what a practical rectifier circuit can produce from standard 120 V_{ac}. The speed can be controlled by adjusting the rectified voltage. Based on the operating parameters, the speed of DC motors governed by the equation:

$$N = \frac{(V_a - I_a R_{ac})}{C_E \quad \emptyset} \tag{2.6}$$

On the right hand side of Equation (2.6) there are three operating parameter, namely, the voltage applied to the armature circuit (V_a), the voltage drop in the armature circuit (I_a R_{ac}) and the useful flux per pole (Φ). Accordingly the different methods of controlling the speed of DC motors are broadly classified as [3]:

(i) Flux control method by changing Φ.
(ii) Armature resistance control by changing I_a R_{ac} drop.
(iii) Armature voltage control by changing V_a.

2.3 Sensors

A sensor is a device that converts a physical phenomenon into an electrical signal. As such, sensors represent part of the interface between the physical world and the world of electrical devices, such as computers. The other part of this interface is represented by actuators, which convert electrical signals into physical phenomena. In recent years, enormous capability for information processing has been developed within the electronics industry. The most significant example of this capability is the personal computer. In addition, the availability of inexpensive microprocessors is having a tremendous impact on the design of embedded computing products ranging from automobiles to microwave ovens to toys. In recent years, versions of these products that use microprocessors for control of functionality are becoming widely available. In automobiles, such capability is necessary to achieve compliance with pollution restrictions. In other cases, such capability simply offers an inexpensive performance advantage. All of these microprocessors need electrical input voltages in order to receive instructions and information. So, along with the availability of inexpensive microprocessors has grown an opportunity for the use of sensors in a wide variety of products. In addition, since the output of the sensor is an electrical signal, sensors tend to be characterized in the same way as electronic devices. The data sheets for many sensors are formatted just like electronic product data sheets.

However, there are many formats in existence, and there is nothing close to an international standard for sensor specifications. The system designer will encounter a variety of interpretations of sensor performance parameters, and it can be confusing. It is important to realize that this confusion is not due to an inability to explain the meaning of the terms rather it is a result of the fact that different parts of the sensor community have grown comfortable using these terms differently [4].

2.3.1 Sensor performance characteristics

The following are some of the more important sensor characteristics:

- Transfer function
- Sensitivity
- Span or Dynamic Range
- Accuracy or Uncertainty
- Hysteresis
- Nonlinearity (often called linearity)
- Noise
- Resolution
- Bandwidth

2.3.2 Types of sensor

- Chemical sensors
- Capacitive and inductive displacement sensors
- Flow and level sensors
- Force, load and weight sensors

- Humidity sensors
- Machinery vibration monitoring sensors
- Optical and radiation sensors
- Position and motion sensors
- Pressure sensors

2.3.3 Photo sensor

Photo sensor are classed by the physical quantity that is affected by the light, and the main classes are photoresistors, photovoltaic materials and photomitters. Historically, photomitters have been more important in unravelling the theory of the effect of light on materials, but photovoltaic materials notably selenium, were in use for some considerable time before the use of photoemission became practicable. Since photoemission allows us to combine the description of a usable device with the quantum effect, we will consider this type of sensor first, which can also be used to a limited extent and transducer.

A photoelectric sensor is an electrical device that responds to a change in the intensity of the light falling upon it. The first photoelectric devices used for industrial presence and absence sensing applications took the shape of small metal barrels, with a collimating lens on one end and a cable exiting the opposite end the cable connected a photo resistive device to an external vacuum tube type amplifier. A small incandescent bulb, protected inside a matching metal barrel, was the opposing light source. These small, rugged incandescent sensors were the forerunners of today's industrial photoelectric sensors [5].

2.4 Arduino

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. Arduino is an open-source design for a microcontroller interface board, it is actually rather more than that, as it encompasses the software development tools that you need to program an Arduino board, as well as the board itself. There is a large community of construction, programming, electronics, and even art enthusiasts willing to share their expertise and experience on the internet. The first Arduino was introduced in 2005, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

2.4.1 Features of Arduino

- Designers and architects build interactive prototypes.
- Makers, of course, use it to build many of the projects example.
- Arduino is a key tool to learn new things.
- Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

2.4.2 Arduino characteristics

- Inexpensive

Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.

- Cross-platform

The Arduino software Integrated Development Environment (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment

The Arduino software IDE is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

- Open source and extensible software

The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware

The plans of the Arduino boards are published under a creative commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

2.4.3 Types of Arduino board

There are, in fact, several different designs of Arduino board, these are intended for different types of applications, they can all be programmed from the same Arduino development software, and in general, programs that work on one board will work on all. The original Arduino hardware is manufactured by the Italian company smart projects. Some Arduino-branded boards have been designed by the American company SparkFun Electronics. Sixteen versions of the Arduino hardware have been commercially produced to date listed below:

- Arduino Diecimila in Stoicheia
- Arduino Duemilanove (rev 2009b)
- Arduino UNO
- Arduino Leonardo
- Arduino Mega
- Arduino MEGA 2560 R3

- Arduino Nano
- Arduino Due (ARM-based)
- Lily Pad Arduino (rev 2007)

These are some of the various Arduino Input/ Output (I/O) board, clockwise from the top left is Uno (great starter board), Lilypad (normally used for clothing applications), a Pro (very small), a Mega (very powerful), and an Ethernet board as shown Figure 2.4. These boards all have analog and digital input/output. The Uno and Mega have the USB programming interface included - the other boards need an additional board to plug in (or solder in) to program.

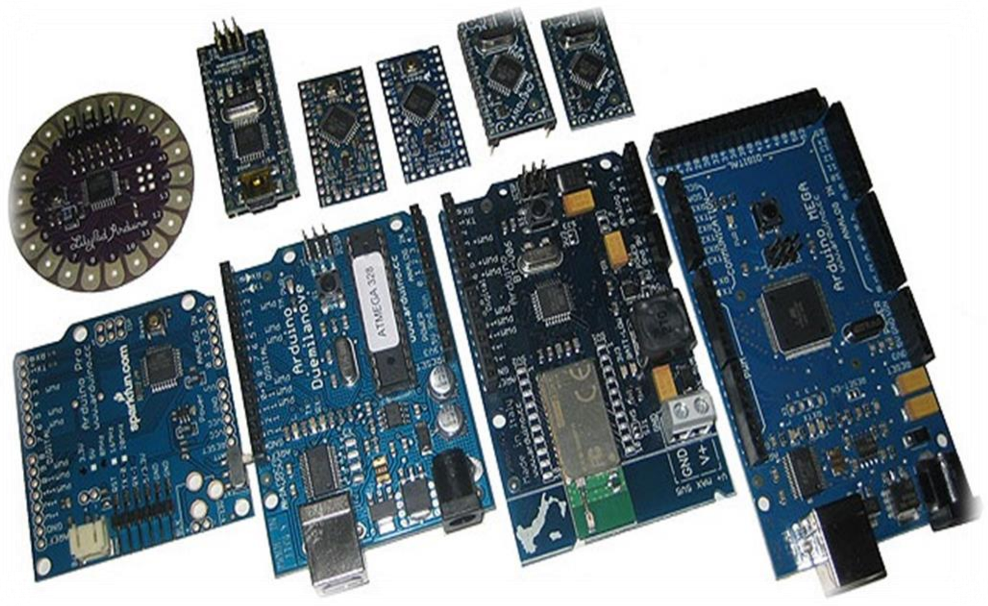


Figure 2.4: Arduino boards

2.4.4 Arduino hardware

There are two rows of connectors on the edges of the board, the row at the top of the diagram is mostly digital (on/off) pins, and although any marked with PWM can be used as analog outputs and the bottom row of connectors has useful power connections on the left and analog inputs on the right. These connectors are arranged to access shield boards. It is possible to buy ready-made shields for many different purposes, including:

- Connection to Ethernet networks
- Liquid Crystal Display (LCD) displays and touch screens
- XBee (wireless data communications)
- Sound
- Motor control
- Global Position System (GPS) tracking
- And many more

2.4.5 Arduino power supply

The power supply can be any voltage between 7 and 12 volts, so a small 9V battery will work just fine for portable applications. Typically, while you are making your project, you will probably power it from USB for convenience, when you are ready, disconnect the USB lead you will want to power the board independently, this may be with an external power adaptor or simply with 9V battery connected to a plug to fit the power socket.

2.4.6 Arduino programming

When you are making a project with an Arduino, you will need to download programs onto the board using a USB lead between your computer and the Arduino, this is one of the most convenient things about using an Arduino. Many microcontroller boards use separate programming hardware to get programs into the microcontroller. Serial communication with Arduino, it’s all contained on the board itself, this also has the advantage that you can use the USB connection to pass data back and forth between an Arduino board and your computer, For instance, you could connect a temperature sensor to the Arduino and have it repeatedly tell your computer the temperature.

2.4.7 Arduino board applications

- Wireless sensors networks
- Controlling robots
- Environment monitoring

- GPS tracking system
- Controlling motors
- Smart systems
- Global System for Mobile (GSM) communications
- Internet of thing applications
- Camera control and others

2.5 SIMULINK

SIMULINK is a software package for modeling, simulating, and analyzing dynamical systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multi rate, i.e., have different parts that are sampled or updated at different rates. For modeling, Simulink provides a GUI for building models as block diagrams, using click-and-drag mouse operations. With this interface, you can draw the models just as you would with pencil and paper (or as most textbooks depict them). This is a far cry from previous simulation packages that require you to formulate differential equations and difference equations in a language or program. Library of sinks, sources, linear and nonlinear components, and connectors. You can also customize and create your own blocks. For information on creating your own blocks, see the separate writing S-Functions guide. Models are hierarchical, so you can build models using both top-down and bottom-up approaches. You can view the system at a high level, then double-click on blocks to go down through the levels to see increasing levels of model detail. This approach provides insight into how a model is organized and how its parts interact. After you define a model, you can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window. The menus are particularly convenient for interactive work, while the command-line approach is very useful for running a batch of simulations (for example, if you are doing Monte Carlo simulations or want to sweep a parameter across a range of values). Using scopes and other display blocks, you can see the simulation results while the simulation is running. In addition, you can change parameters and immediately see what happens, for “what if” exploration. The simulation results can be put in the MATLAB workspace for post processing and visualization. Model analysis tools include linearization and trimming tools, which can be accessed from the MATLAB command line, plus the many tools in MATLAB and its application toolboxes. And because MATLAB and Simulink are integrated, you can simulate, analyze, and revise your models in either environment at any point.

2.5.1 Simulink Real-Time Workshop

The Simulink real-time workshop automatically generates C code directly from Simulink block diagrams. This allows the execution of continuous, discrete-time, and hybrid system models on a wide range of computer platforms, including real-time hardware. Simulink is required. The real-time workshop can be used for:

- Rapid prototyping. As a rapid prototyping tool, the real-time workshop enables you to implement your designs quickly without lengthy hand coding and debugging. Control, signal processing, and dynamic system algorithms can be implemented by developing graphical SIMULINK block diagrams and automatically generating C code.
- Embedded real-time control. Once a system has been designed with Simulink, code for real-time controllers or digital signal processors can be generated, cross-compiled, linked, and downloaded onto your selected target processor. The real-time workshop supports Digital Signal Processing (DSP) boards, embedded controllers, and a wide variety of custom and commercially available hardware.
- Real-time simulation. You can create and execute code for an entire system or specified subsystems for hardware-in-the-loop simulations. Typical applications include training simulators (pilot-in-the-loop), real-time model validation, and testing.
- Stand-alone simulation. Stand-alone simulations can be run directly on your host machine or transferred to other systems for remote execution. Because time histories are saved in MATLAB as binary or American Standard Code for Information Interchange (ASCII) files, they can be easily loaded into MATLAB for additional analysis or graphic display.

2.5.2 Features of real –time workshop

Real-time workshop provides a comprehensive set of features and capabilities that provide the flexibility to address a broad range of applications:

- Automatic code generation handles continuous-time, discrete-time, and hybrid systems.
- Optimized code guarantees fast execution.
- Control framework Application Program Interface (API) uses customizable make files to build and download object files to target hardware automatically.
- Portable code facilitates usage in a wide variety of environments.
- Concise, readable, and well-commented code provides ease of maintenance.
- Interactive parameter downloading from Simulink to external hardware allows system tuning on the fly.
- A menu-driven, graphical user interface makes the software easy to use.

2.6 Graphical User Interface

GUI is a particular case of user interface for interacting with a computer which employs graphical images and widgets in addition to text to represent the information and actions available to the user. Usually the actions are performed through direct manipulation of the graphical elements. The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Center in the 1970s, but it was not until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which until recently were prohibitively expensive. A GUI is a pictorial interface to a program, a good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. A true GUI includes standard formats for representing text and graphics. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on a pushbutton, the GUI should initiate the action described on the label of the button. Many DOS programs include some features of GUIs, such as menus, but are not graphics based. Such interfaces are sometimes called graphical character-based user interfaces to distinguish them from true GUIs. In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components. The MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces. These tools simplify the process of laying out and programming GUIs. The basic tasks in process of implementing a GUI is first, laying out a GUI where MATLAB implement GUIs as figure windows containing various styles of user interface objects. The second task is programming the GUI, where each object must be program to perform the intended action when activated by the user of GUI. Each component, and the GUI itself, is associated with one or more user written routines known as callbacks. The execution of each callback is triggered by a particular user action such as, mouse click, pushbuttons, toggle buttons, lists, menus, text boxes, selection of a menu item, or the cursor passing over a component and so forth. Clicking the button triggers the execution of a callback. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back. This kind of programming is often referred to as event-driven programming. The event in the example is a button click. In event-driven programming, callback execution is asynchronous, controlled by events external to the software. In the case of MATLAB GUIs, these events usually take the form of user interactions with the GUI. The writer of a callback has no control over the sequence of events that leads to its execution or, when the callback does execute, what other callbacks might be running simultaneously.

2.7 Literature Review

This study [1] presents data acquisition system is a system which acquires the data i.e. input/output parameters from the field with the help of sensors associated with the system, it analyses the data and generate control action required for operating and also monitor the acquired data on GUI. The DAQ system is basic need for controlling and monitoring of a Mega system in modern industries. So the main objective of this paper is to develop a low cost DAQ system for controlling and monitoring a system using Arduino-Uno controller and LabVIEW GUI. In other way if we want, we can also monitor the data on mobile phone with an Android application.

This study [6] describes a Simulink lab practice using Arduino as low cost hardware. A shell must be developed in order to adapt Arduino signals to the real plant, consist of a DC motor. With Arduino architecture and with open hardware a cheap data acquisition card has been build. Several tests have been done to validate de full system and a frequency study has been completed in order to know the possibilities of the proposed architecture in the control of new other plants.

This study [7] presents the DC motors are widely used for variable speed drive system in industrial applications such as industrial automation, electric traction, aircraft, military equipment, hard disk drives because of their high efficiency, silent operation, compact, reliability and low maintenance. Due to the advancement of wireless technology, there are several connections are introduced such as GSM, Wi-Fi, ZIGBEE and Bluetooth. Each of the connection has their own unique specifications and applications. Among these wireless connections, Bluetooth technology often implemented. The speed control was implemented using Bluetooth technology to provide communication access from smart phone. Communication plays a major role in day today’s life and can be used as a better tool in control system. It deals with wireless communication and voice recognition and is used to control the motor speed. On the other hand we have Arduino-Uno platform that we can use to quickly prototype electronic systems. It enables a person to work around independently using a touch screen and voice recognition applications which is interfaced with motors using Arduino-Uno microcontroller. This can also be controlled through simple voice commands. In addition to this Infra-Red (IR) sensor is used to sense the motor speed and in turn speed of the motor can be received via Bluetooth to the android mobile.

This study [8] presents a Precise, cheap control and monitoring the speed of DC motor is ever hot area of work. In the present paper have attempted to implement speed control and feedback monitoring for a 12 Volt/1000 Revolutions Per Minute (RPM) rated motor. Speed control is done with help of PWM pins on Arduino/AVR board and H-bridge Integrated Circuit (IC) L293D. Feedback speed monitoring is based on IR pair based interrupt monitoring. Hence Sensing and calculation part of process is handled by Arduino/AVR board. All of this is implemented with help of

Personal Computer (PC) based user interface developed in C#. In a typical user interface speed control is achieved with help of track bar and speed monitoring with help of text window, was we get direct readout of current speed in RPM.

This study [9] presents a graphical user interface of motor control through MATLAB Graphical User Interface Development Environment (GUIDE), interface the MATLAB GUI with hardware via communication port and control the DC motor through MATLAB GUI. By using MATLAB GUIDE, it provides a set of tools which simplify the process of laying out and programming GUIs and interface with PIC via serial communication port to control the DC motor. The PIC is used to control motor. As a result, the DC motor is able to be controlled through MATLAB GUI and interface the MATLAB GUI with PIC via serial communication port.

This study [10] presents a Brushless DC (BLDC) motors are the most widely used electrical drive in the industry. The development process of the drive is costly and time-consuming. However, various methods can be used to reduce the development time of the drive. This paper presents the model-based design technique of BLDC motor using MATLAB/Simulink with Arduino support block set. The model of BLDC motor is developed using black-box modeling approach; simulations are performed based on real-time data and processed using MATLAB system Identification tool box. The Proportional Integral Derivative (PID) controller is then designed and tuned within the simulations to attain the drive performance. For real-time application, the controller code is generated and uploaded into Arduino Mega embedded controller. The results obtained from simulation and experiment is discussed and compared. The performed works concludes that model-based design technique can be applied in any control design application using low cost controller such as Arduino embedded controller.

This study [11] presents a DC motor is widely uses for speed control and load characteristics, it's easy controllability provide effective and precise output so application of DC motor is large for commercial purpose. Speed control of DC motor is very crucial in application where required speed is precision and correcting signal representing and to operate motor at constant speed, so we used PWM method which are fulfill all requirements to speed control of DC motor. PWM based speed control system consists of electronic components (integrated circuits, sensors etc.).

This study [12] presents speed control mode of a DC motor without knowing the specific parameters of the motor is discussed. The approximate mathematical model of the control system is obtained by the system identification when the output of the system is measured by loading the specific input signal. The PID control algorithm is adopted and the P, I, and D parameters are obtained by auto tuning. Hardware In the Loop (HIL) experiments are carried out on MATLAB and Arduino platform, in which the experimental results demonstrate the feasibility of the proposal.

Chapter Three

System Description

Chapter Three

System Description

3.1 Description of System Components

The block diagram of system as shown in Figure 3.1. The system components are explained in detail.

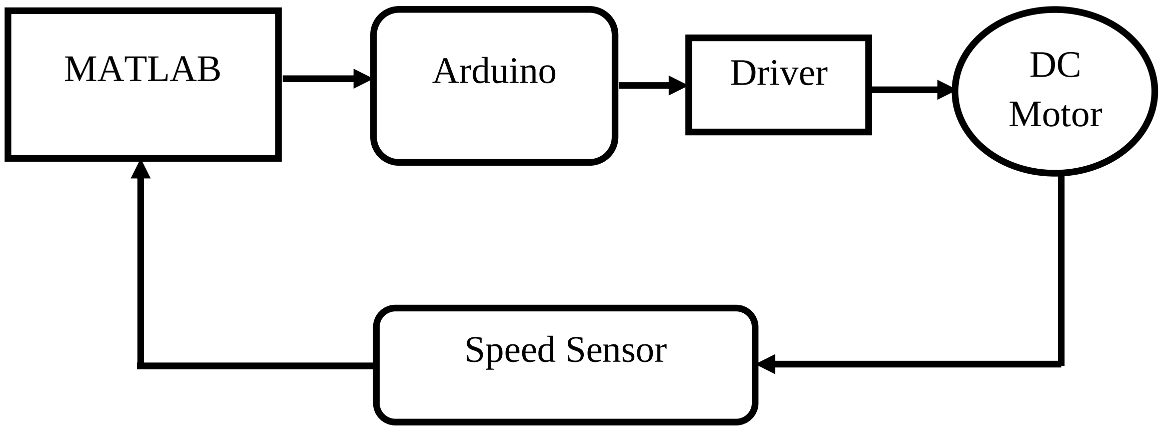


Figure 3.1: System block diagram

3.1.1 Arduino Uno

Arduino is an open hardware/software platform, The Arduino Uno is a microcontroller board based on the ATmega328 is shown in Figure 3.2. Arduino becomes a very fast development platform even for those how aren't into electronics' world. Arduino programming becomes much faster because the graphical open-source environment makes it easy to write code and upload it to the board.

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings. Arduino projects can be stand-alone, or they can communicate with software running on a computer. In this development, Arduino Uno is used as the main controller because it satisfies these conditions as Table 3.1.

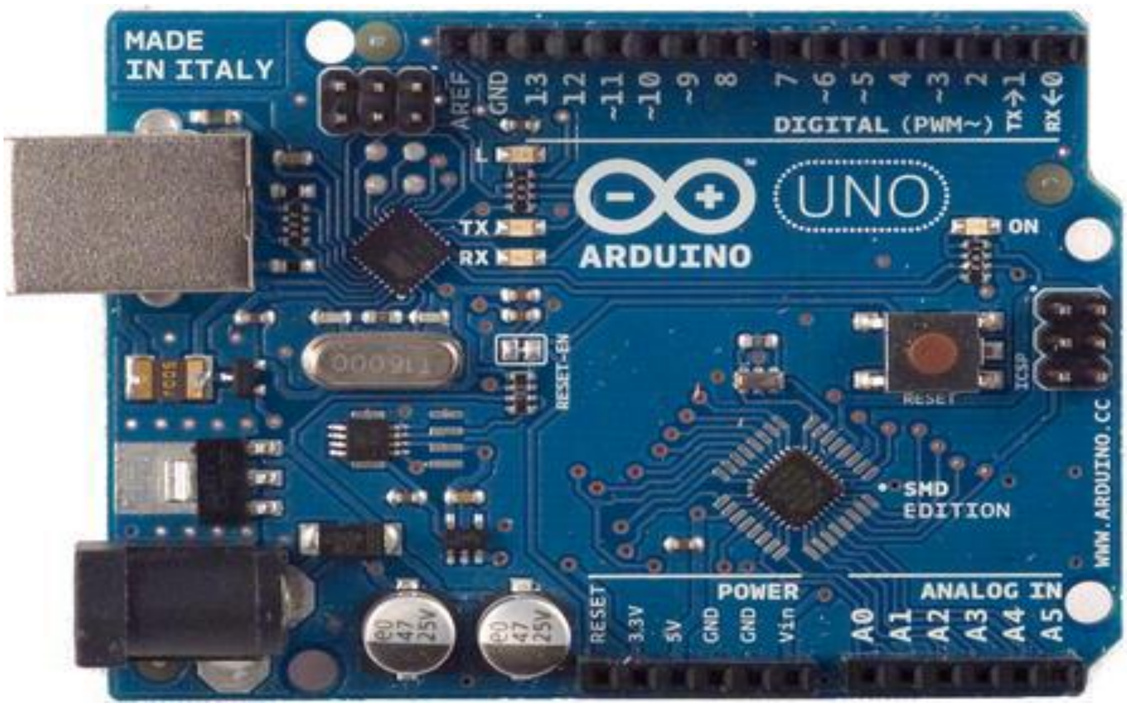


Figure 3.2: Arduino Uno board

Table 3.1: Specification of Arduino Uno board

Microcontroller	Specification
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5KB used by boot loader
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHZ

3.1.2 DC motor control using PWM

PWM signal consists of a train of voltages pulses such that the width of individual pulses controls the effective voltage level. Pulse width modulation is an entirely different approach to controlling the torque and speed of a DC motor. Power is supplied to the motor in a square wavelike signal of constant magnitude but varying pulse width or duty cycle. Duty cycle refers to the percentage of time the pulse is high (per cycle). Figure 3.3 shows the waveforms for four different speeds. For the slowest speed, the power is supplied for only one-quarter of the cycle time (duty cycle of 25%). The frequency of the pulses is set high enough to ensure that the mechanical inertia of the armature will smooth out the power bursts, and the motor simply turns at a constant velocity of about one-quarter speed. For a 50% duty cycle (power on one-half the time), the motor would turn at about half speed, and so on [2].

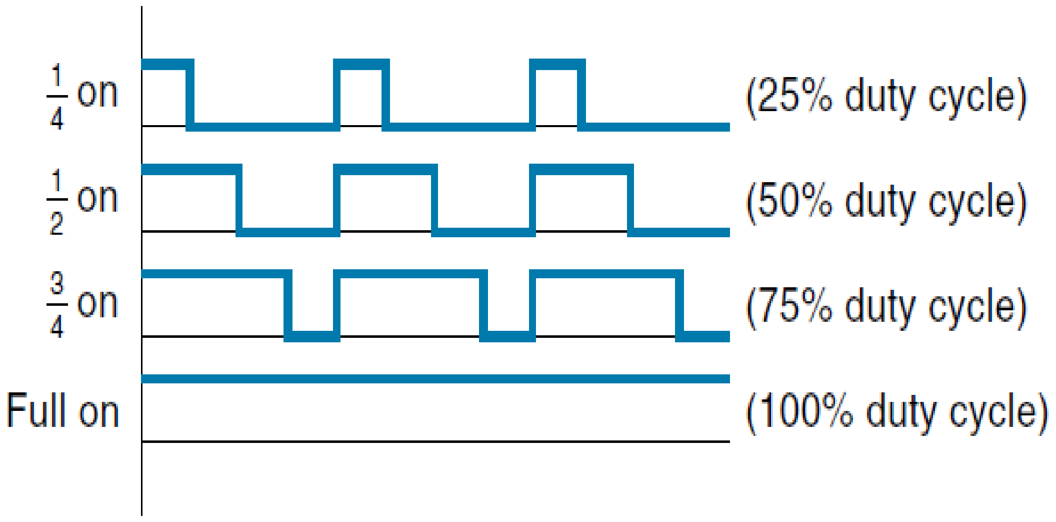


Figure 3.3: Pulse width modulation waveforms

PWM control is a very commonly used method for controlling the power across loads. This method is very easy to implement and has high efficiency. PWM signal is essentially a high frequency square wave (typically greater than 1KHz). The duty cycle of this square wave is varied in order to vary the power supplied to the load. Duty cycle is usually stated in percentage and it can be expressed using the equation:

$$\text{Duty Cycly} = \frac{T_{on}}{T_{on} + T_{off}} * 100\% \qquad (3.1)$$

Where $\mathbf{T_{on}}$ is the time for which the square wave is high and $\mathbf{T_{Off}}$ is the time for which the square wave is low. When duty cycle is increased the power dropped across the load increases and when duty cycle is reduced, power across the load decreases.

3.1.3 Photoelectric sensor (LM 393 IR sensor)

Photoelectric sensors detect objects that pass between the two arms. Because the emitter and receiver are built into the same enclosure, optical axis alignment is not necessary for accurate detection. These U-shaped photoelectric sensors are suitable for positioning; cut-off mark and paper feed sensing needs. Photoelectric sensors are designed for optimum value and sensing performance in a wide range of manufacturing applications. They provide a variety of optical styles and electrical configurations in small, easy-to-use, economical packages. With high speed and the ability to work with many types of sensors, these devices are ideal for controlling batch size, material length-cutting, punch/drill machinery and gas/liquid flow applications. The LM393 sensor is a voltage comparators and photoelectric counting sensor chip, used for counting (motor speed etc.) with three pins, the ground pin, the voltage pin, and the output pin on board as shown in Figure 3.4 and specification as Table 3.2.

3.1.4 Motor driver unit

The motor driver unit L293D is available for providing user with ease and user friendly interfacing for embedded application. L293D motor driver is mounted on a good quality. The pins of L293D motor driver IC are connected to connectors for easy access to the driver IC's pin functions as shown in Figure 3.5 and specification of driver as Table 3.3. The L293D is a dual full bridge driver that can drive up to 1A per bridge. It can drive two DC motors, relays, solenoids, etc. The device is Transistor Transistor Logic (TTL) compatible. Two H bridges of L293D can be connected in parallel to increase its current capacity to 2A. L293D motor driver is easily compatible with any of the system and easy interfacing through Flat Ribbon Cable (FRC).

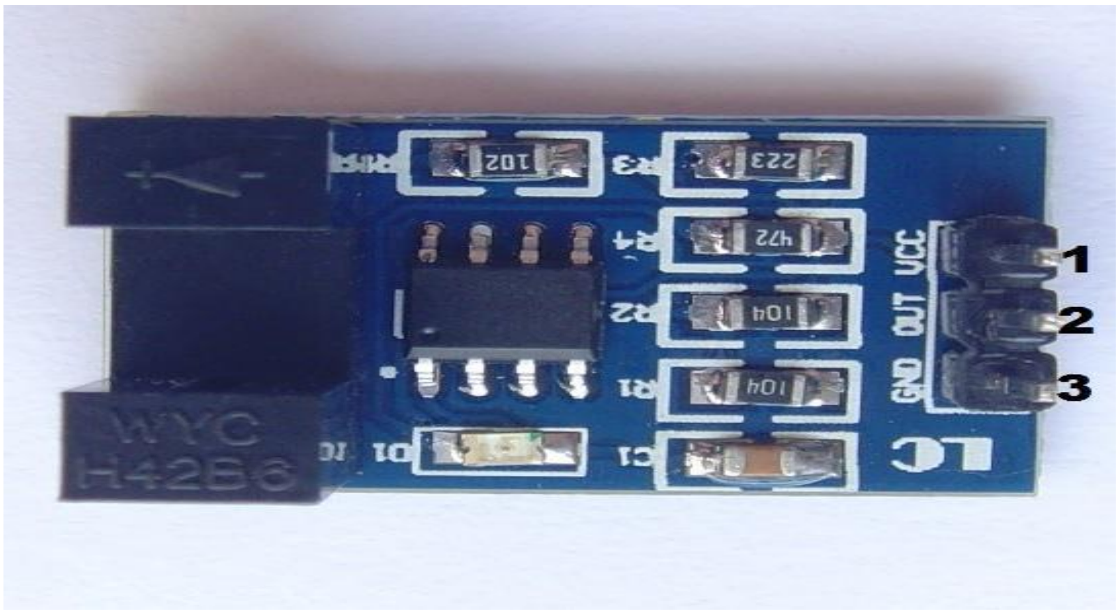


Figure 3.4: LM393 sensor board

Table 3.2: Specification of LM393 sensor

Specification	
Input Supply Voltage	5v / 3.3v
Detection Distances Range	5 - 50 mm
Repeatability	± 0.1 - 0.2 mm
Output Low	LED on (Nothing detected)
Output High	LED off (Obstruction between the emitter and receiver)
Size	PCB (24 x15 mm)
Options	Single and dual beam

Table 3.3: Specification of motor driver unit L293D

Specifications	
Internal Power Supply	5V DC
External Power Supply	9V to 24V DC
Output Current Capability per Channel	600 mA
Temperature Range	44mm x 37mm x14mm (l x b x h)
Dimensional Size	0°C to +70 °C

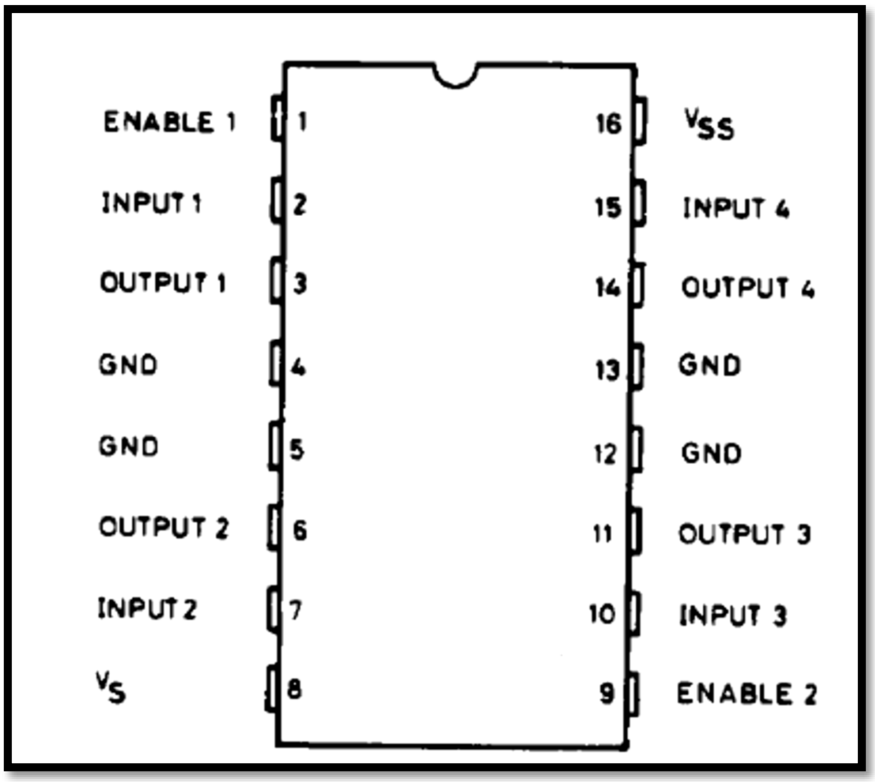


Figure 3.5: L293D pin configuration

3.2 System Design

The DC motor design model as shown in Figure 3.6. The control system designs generally consist of Arduino Uno controller development board, DC motor driver, DC motor, IR speed sensor and PC with MATLAB software. In system speed of the DC motor is controlled through the Arduino Uno using MATLAB.

3.2.1 Hardware design

The physical system is a DC motor connected to Arduino Uno board via a motor driver.

1. Arduino board

An Arduino board is used due to low cost, simplicity and flexibility.

2. Motor driver unit (IC L293D)

Pins 2 and 7 of L293D are connected to PWM pins of Arduino (pins 3 and 4 respectively). This pins are used to control speed of rotation. Values written in the range of 0-5 volt on this pin change the speed of motor from 0-max rated. Pin 8 is connected to V_{in} pin of Arduino (supply voltage).

3. IR sensor

IR sensor is used for speed measurement. VCC pin of module connected to 5V pin of Arduino, the GND pin connected to the GND pin of Arduino and the output pin connected to pin 2 of Arduino.

4. Motor Unit

Motor is 12 volt/1000 RPM rated DC motor. Motor is connected to pins 3 and 6 of L293D.

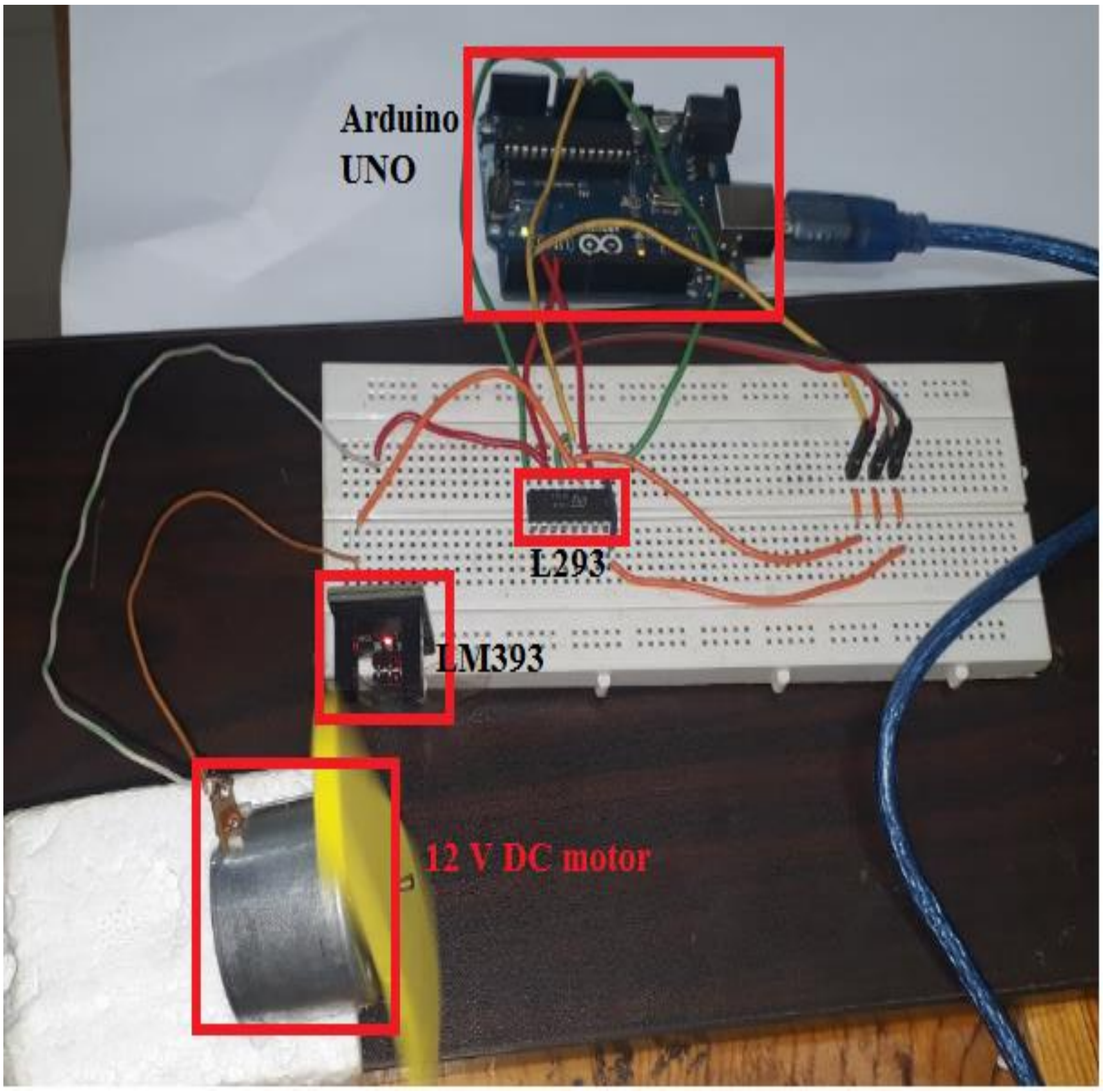


Figure 3.6: The DC motor design model

3.2.2 Software consideration

In system speed of the DC motor is controlled through the Arduino Uno using MATLAB. Figure 3.7 shows flow chart of the system.

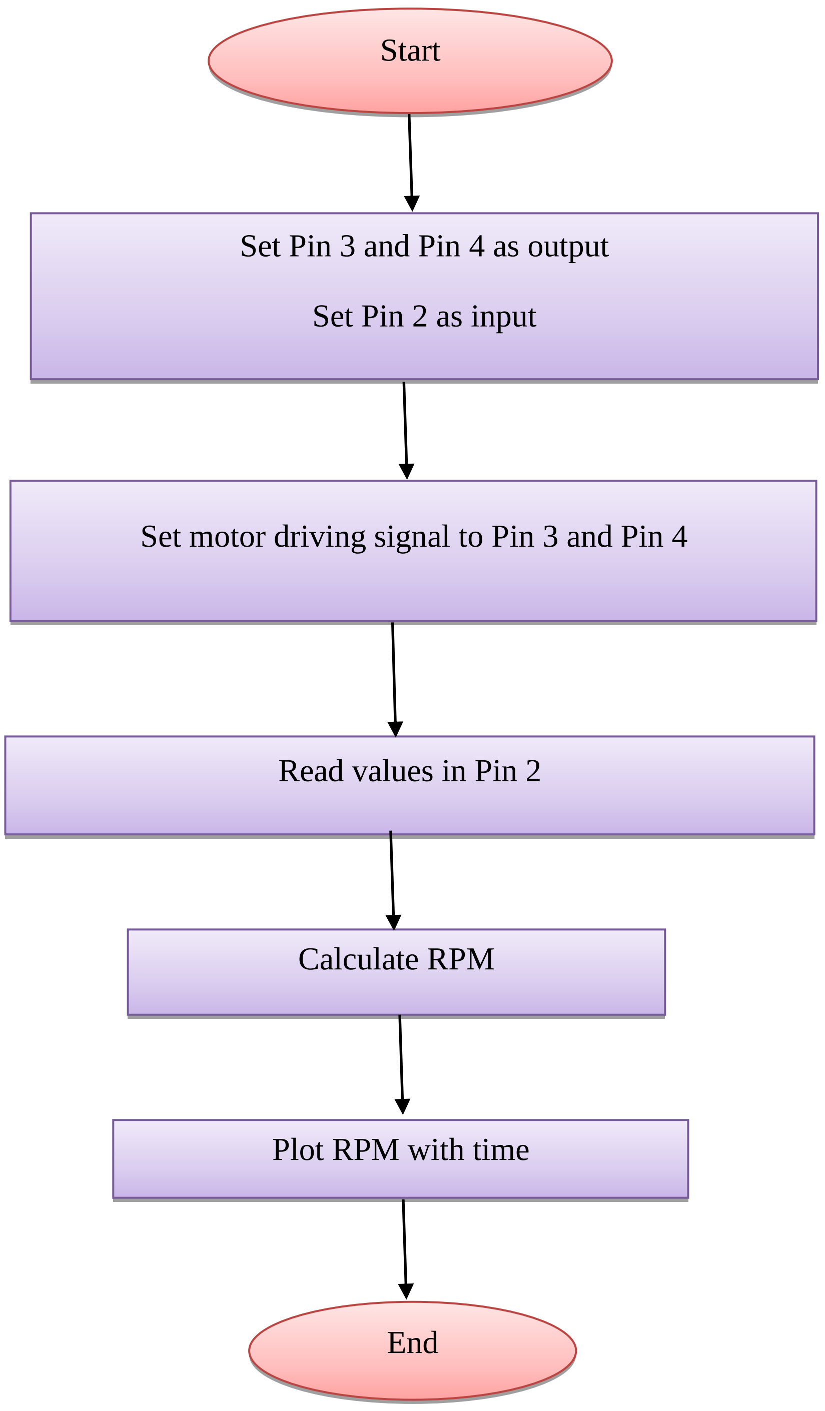


Figure 3.7: System flow chart

A. Simulink model

Figure 3.8 shows the Simulink model. This model explains how to use Simulink blocks to read analog signals from real world and how to use PWM output signals to control of the DC motor. Arduino boards are usually programed by writing C/C++ code in Arduino IDE window, but in this system it will be programmed using MATLAB/SIMULINK package for Arduino. Simulink is a block diagram based environment for simulating mathematical models and it can be used to program some of the Arduino boards. To use Simulink with Arduino a support package is needed. The Arduino can easily interface with a host PC through Simulink with the Arduino IO package provided by Mathworks. The Arduino IO package allows any blocks developed in Simulink to quickly interface with an ATMega328 running an Arduino IO server on the Arduino Uno. With the use of the real-time pacer block to match the simulation clock to real time, models can run in real time with the physical system. This simple block diagram provides real time communication between the board and Simulink. The interface for the integration of control designs created in Simulink are made fast and simple.

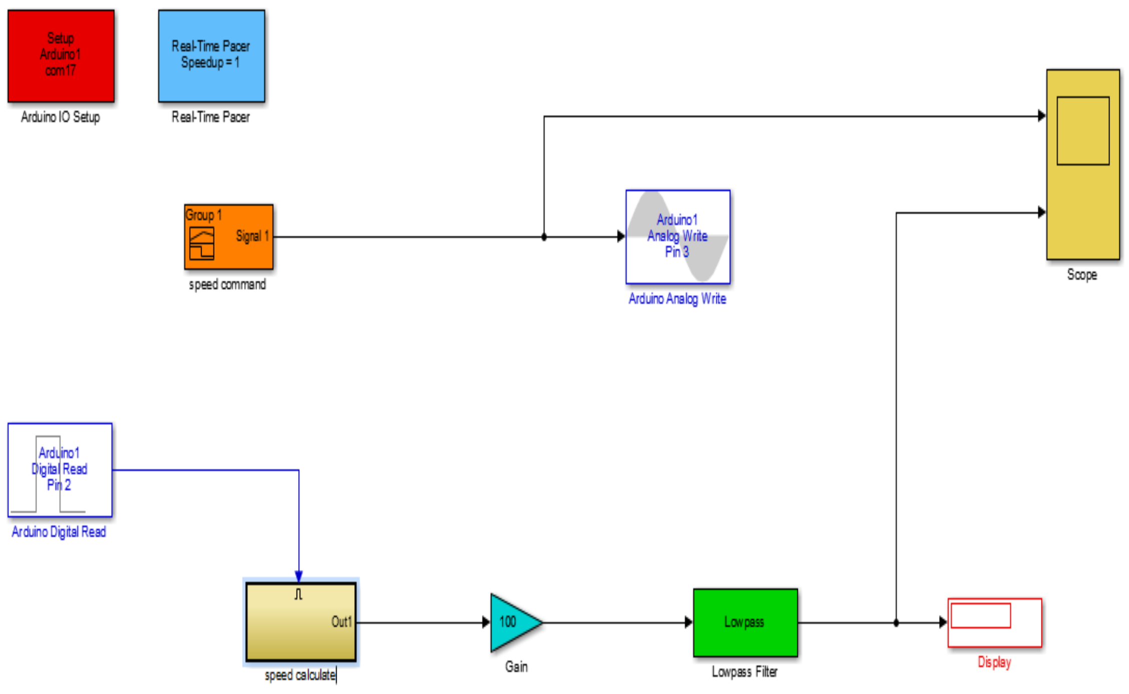


Figure 3.8: The Simulink model

B. Graphical User Interface

GUIDE stands for graphical user interface development environment. It provides a set of tools for creating GUI. GUI is useful for letting users input data graphically and view results graphically, all in the same figure window. MATLAB GUIDE, it provides a set of tools which simplify the process of laying out and programming GUIs and interface with Arduino to control the DC motor.

The system present user can control speed of DC motor with help of a friendly user interface as shown in Figure 3.9 and the m-file as shown in Appendix contains the code to load the figure and call backs for each GUI element. User can select direction of rotation viz. Clockwise/Anticlockwise at click of a button during any time of operation. Can select speed of rotation with help of slider. Stop motor with button click. And also monitor current speed (speed RPM) of speed sensor in text window. The speed slider can be moved to the left or right of the center (zero) point depending on the direction one wish the motor to rotate. Negative speed (left of the slider) means the motor will rotate reverse clockwise, while positive speed (right of the slider) means the motor will rotate clockwise. The motor can be stopped returning the slider to zero. In addition, it can be stopped by pushing the Stop button.

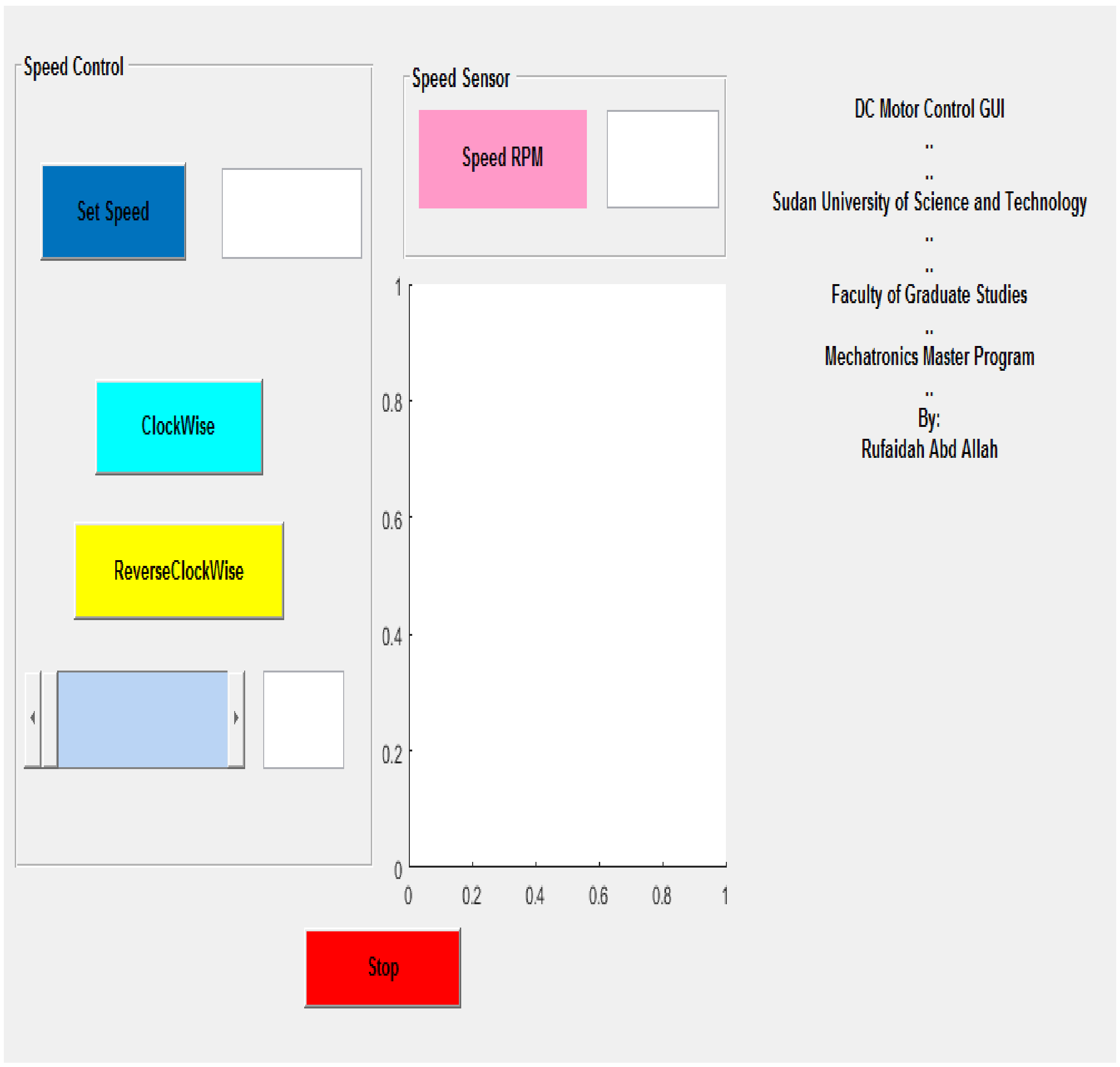


Figure 3.9: GUI for the DC motor speed control

Chapter Four

Experimental Results and Discussions

Chapter Four

Experimental Results and Discussions

4.1 Results

Regarding DAQ using SIMULINK model, where output to the Arduino and motor has been driven using SIMULINK model as shown in Figure 3.7. Driving signal (red colored) and feedback signal (blue colored) measuring RPM of the motor without and with filter has been shown in Figures 4.1 and Figure 4.2 respectively.

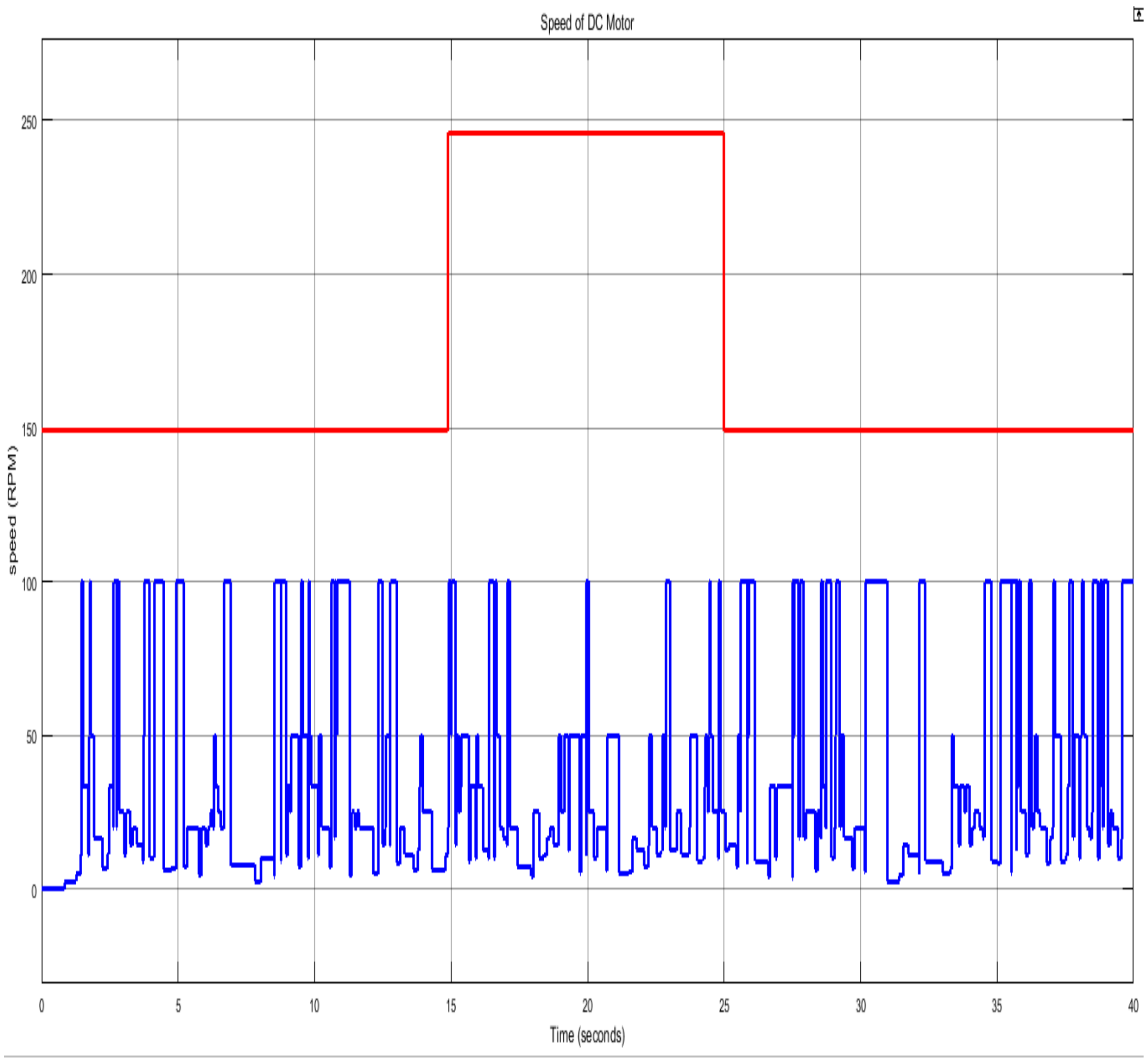


Figure 4.1: Motor driving signal and RPM feedback sensor measures (without filter)

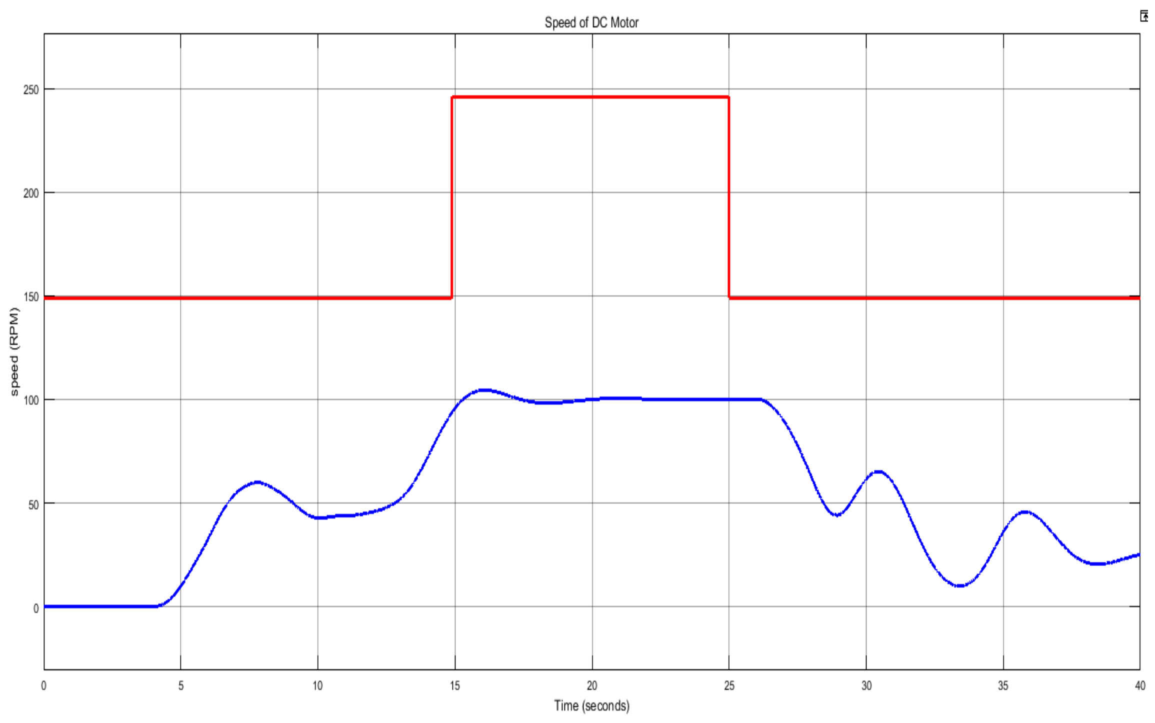


Figure 4.2: Motor driving signal and RPM feedback sensor measures (with filter)

When using GUI, the feedback reading are plotted is same window in the DAQ tool, as shown in Figures 4.3 and Figure 4.4.

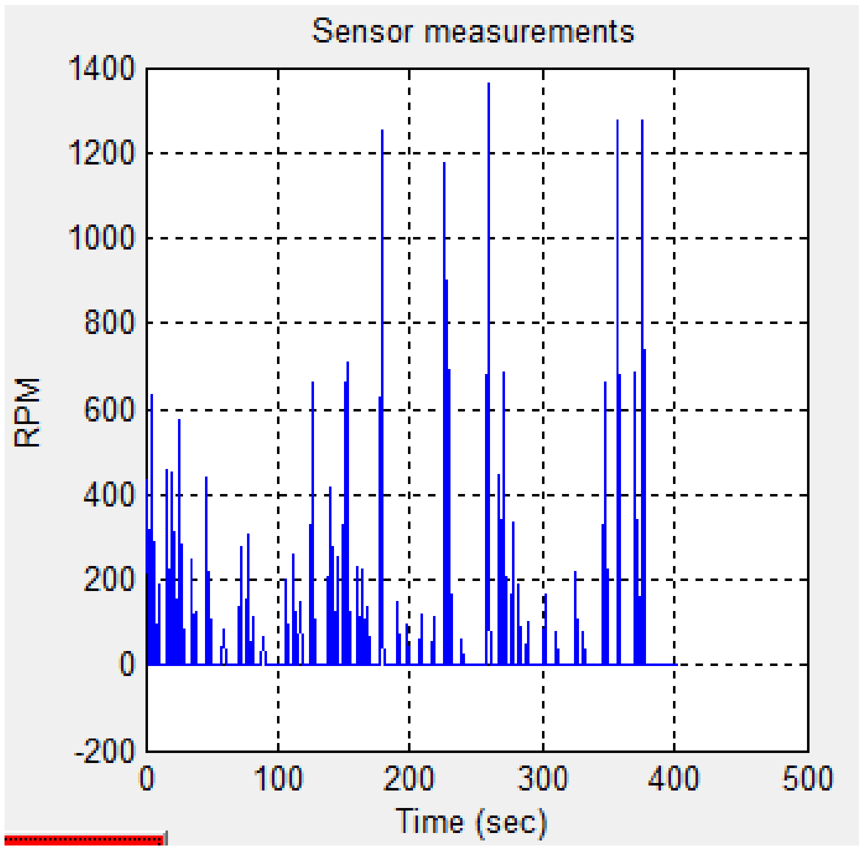


Figure 4.3: RPM sensor measurements

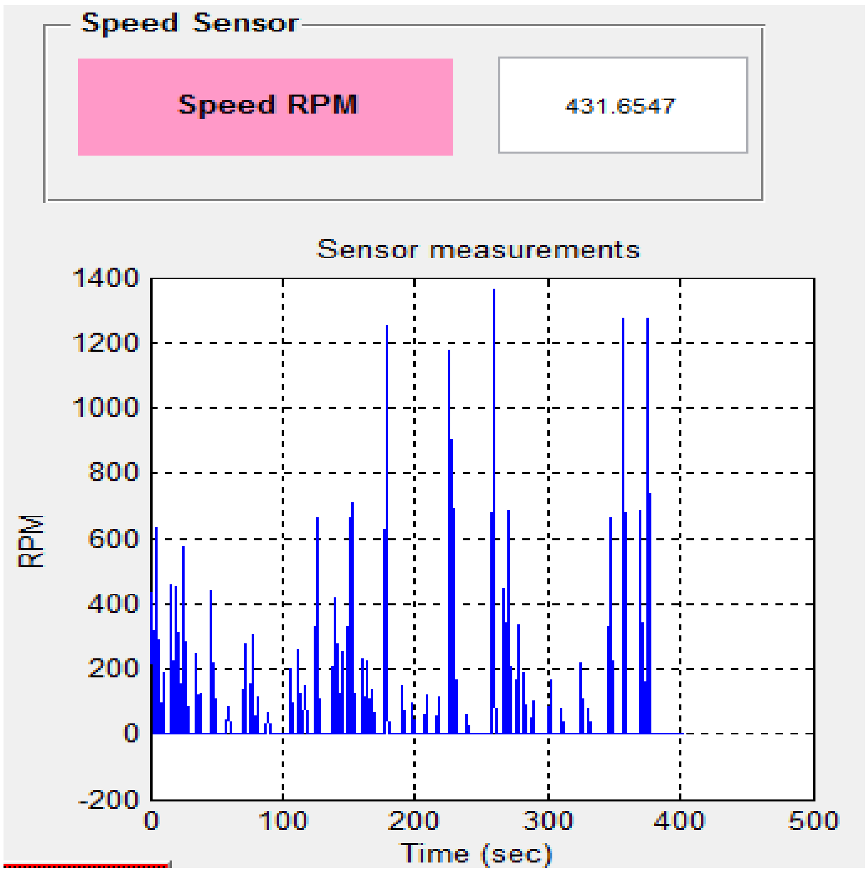


Figure 4.4: RPM sensor measurements and numeric value

4.2 Discussions

Data acquisition software tool developed in this research, has a task that driving a DC motor with a certain speed or signal waveform and measure the response of the motor using IR speed sensor. The tool which developed in SIMULINK environment has results which shown in Figure 4.1, shows that the motor responded for the input signal, besides, the RPM measurements need be averaged over the time, response of RPM signal showing follow up for driving signal up to 30 seconds, some sample after this time are strange values of readings. Which has been shown in Figure 4.2, the sensor values has passed through Low Pass Filter (LPF), where lag of response due to the filter, and some peaks appeared following up driving signal. For GUI DAQ, the motor derived using buttons in a GUI as shown Figure 3.9. The measured reading from IR sensor shown in Figure 4.3, which is reading versus time. Although a numerical value of the RPM written in text box as shown in Figure 4.4. Regarding Figure 4.3 the measurements were averaged through time.

Chapter Five

Conclusion and Recommendations

Chapter Five

Conclusion and Recommendations

5.1 Conclusion

This work proposed a low cost DAQ system using Arduino-UNO controller and MATLAB (SIMULINK AND GUI). Using SIMULINK and Arduino is one of the simplest ways to make satisfying controller for many purposes in variety of technical systems. Tests in this research show that Arduino can be very flexible for programming a control algorithms using MATLAB/SIMULINK. One purpose of this kind of controller is using it in laboratory exercises where students can easily modify controller’s parameters and see what is going on with output value of the system that they are studying. Sample time of this controller can be changed and adapted to the system requirements. The advantages of using Arduino controller with Simulink Arduino target is an inexpensive, open source microcontroller board and allows the creation of applications in the Arduino platform based on a visual programming environment with block diagrams.

The GUI can be extended to other purposes also and not only learning about the DC motor. It can be used for learning induction motors, alternators, image processing and other. The use of a GUI through MATLAB is quite extensive as it can be coupled with other toolboxes in MATLAB quite effectively making it an appealing prospective for students to learn about a new topic or area of interest.

5.2 Recommendations

1. Using more accurate RPM sensor such as incremental encoder sensors.
2. Enhance the averaging ability of the averaging filter, or design more optimal filter, in order to increase the accuracy of RPM values.
3. Add PID tuning option for both GUI and SIMULINK.

References

[1] Nidhi Kanani and Manish Thakker, ‘‘Low Cost Data Acquisition System Using LABVIEW’’, International Journal of Research and Scientific Innovation (IJRSI), Vol. 3, Issue 1, December 2015.

[2] Kilian Delemar, "Modern Control Technology: Components and Systems", 2nd Ed., Cengage, 2005.

[3] K. Murugesh Kumar, "DC Machines and Transformers", 2nd Ed., New Delhi: Vikas Pvt Ltd, 2003.

[4] Jon S. Wilison, "Sensor Technology Handbook", London: Elsevier Ltd, 2005.

[5] Lan R. Sinclair, "Sensor and Transducer", 3rd Ed. London: Butterworth-Heinemann, 2001.

[6] R. Barber and M. Horra. J. Crespo, ‘‘Control Practices using Simulink with Arduino as Low Cost Hardware’’, International Federation of Automatic Control (IFAC), Vol. 47, Issue 17, 2013.

[7] B.Gokul1, K. Karthi, A. Thiyagaseelan and V. Santhosh Kumar, ‘‘Android Based Closed Loop Speed Control of DC Motor Using Voice Recognition via Bluetooth’’, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE), Vol. 5, Issue 3, March 2016.

[8] Prithviraj R. Shetti1 and Ashok G. Mangave, ‘‘DC Motor Speed Control with Feedback Monitor Based ON C# Application’’, International Journal of Research in Engineering and Technology (IJRET), Vol. 3, Issue 3, March 2014.

[9] Arpita Srivatava and Anil Kumar Chaudhary, ‘‘DC Motor Speed Control Using ATMEGA89S52 and MATLAB GUI Application’’, International Journal of Science and Research (IJSR), Vol. 5, Issue 12, December 2016.

[10] M. K. Hat, B. S. K. K. Ibrahim, T. A. T. Mohd and M. K. Hassan, ‘‘Model Based Design of PID Controller for BLDC Motor with Implementation of Embedded Arduino MEGA Controller’’, ARPN Journal of Engineering and Applied Sciences, Vol. 10, Issue 19, October 2015.

[11] Khan Masoom Raza, Mohd. Kamil and Pushpendra Kumar, “Speed Control of DC Motor by using PWM” , International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 5, Issue 4, April 2016.

[12] Wei-Jie Tang, Zhen-Tao Liu and Qian Wang, “DC Motor Speed Control Based on System Identification and PID Auto Tuning”, Dalian, China, 2017.

Appendix

M.file Code

```
function varargout = DC_Motor (varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',mfilename, ...
                   'gui_Singleton',gui_Singleton, ...
                   'gui_OpeningFcn', @DC_Motor_OpeningFcn, ...
                   'gui_OutputFcn',  @DC_Motor_OutputFcn, ...
                   'gui_LayoutFcn', [], ...
                   'gui_Callback',   []);

if nargin && ischar (varargin{1})
    gui_State.gui_Callback = str2func (varargin{1});
end

if nargout
    [varargout {1:nargout}] = gui_mainfcn (gui_State, varargin{:});
else
    gui_mainfcn (gui_State, varargin{:});
end

function DC_Motor_OpeningFcn (hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata (hObject, handles);

function varargout = DC_Motor_OutputFcn (hObject, eventdata, handles)
varargout {1} = handles.output;
clc
clear all
global a;
global rpm;
global speed;
```

```
rpm = 0;
a = arduino
function ClockWise_Callback(hObject, eventdata, handles)
global a;
global rpm;
i = 0;
previous_time = 0;
keeplooping = true;
writeDigitalPin (a, 3,1);
writeDigitalPin (a, 4,0);
plot (rpm);grid;
title ('Sensor measurements')
xlabel ('Time (sec)')
ylabe l ('RPM')
while keeplooping
   val = readDigitalPin(a,2);
   rpm = [rpm i];
   i = rpm;
   plot (rpm);grid;
   title ('Sensor measurements')
   xlabel ('Time (sec)')
   ylabe l('RPM')
   if (val==1)
      t = clock;
      current_time = t(6);
      pulse_time = current_time - previous_time;
      rpm = (1/pulse_time)*60;
      previous_time = current_time;
      set (handles.rpm_value,'string',num2str(rpm));
   else
      rpm = 0;
   end
```

```
end
function ReverseClockWise_Callback(hObject, eventdata, handles)
global a;
global rpm;
i = 0;
previous_time = 0;
keeplooping = true;
writeDigitalPin (a, 3,0);
writeDigitalPin (a, 4,1);
while keeplooping
   val = readDigitalPin (a,2);
   rpm = [rpm i];
   i = rpm;
   plot (rpm);grid;
   title ('Sensor measurements')
   xlabel ('Time (sec)')
   ylabel ('RPM')
   if(val==1)
      t = clock;
      current_time = t (6);
      pulse_time = current_time - previous_time;
      rpm = (1/pulse_time)*60;
      previous_time = current_time;
      set(handles.rpm_value,'string',num2str(rpm));
   else
      rpm = 0;
   end
end

function Stop_Callback(hObject, eventdata, handles)
global a;
writeDigitalPin (a, 3,0);
writeDigitalPin (a, 4,0);
```

```
function slider1_Callback (hObject, eventdata, handles)
global a;
global rpm;
i = 0;
previous_time = 0;
keeplooping = true;
slider = get (hObject,'Value');
slider_1 = slider*20;
set(handles.SpeedValue, 'string', num2str(slider_1));
writePWMVoltage (a,3,slider);
guidata (hObject,handles);
while keeplooping
   val = readDigitalPin(a,2);
   rpm = [rpm i];
   i = rpm;
   plot (rpm);grid;
   title ('Sensor measurements')
   xlabel ('Time (sec)')
   ylabel ('RPM')
   if (val==1)
      t = clock;
      current_time = t (6);
      pulse_time = current_time - previous_time;
      rpm = (1/pulse_time)*60;
      previous_time = current_time;
      set (handles.rpm_value,'string',num2str(rpm));
   else
      rpm = 0;
   end
end

function slider1_CreateFcn (hObject, eventdata, handles)
```

```
if isequal (get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
 set (hObject,'BackgroundColor',[.9 .9 .9]);
end
function SpeedValue_Callback (hObject, eventdata, handles)
function SpeedValue_CreateFcn (hObject, eventdata, handles)
if ispc && isequal (get (hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set (hObject,'BackgroundColor','white');
end

function rpm_value_Callback(hObject, eventdata, handles)
function rpm_value_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function set_speed_Callback(hObject, eventdata, handles)
global speed;
speed = get (hObject,'string');
speed = str2double (speed);
function set_speed_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set (hObject,'BackgroundColor','white');
end

function set_speed_button_Callback (hObject, eventdata, handles)
global a;
global rpm;
global speed;
i = 0;
previous_time = 0;
```

```
keeplooping = true;
writePWMVoltage (a,3,speed/40);
while keeplooping
   val = readDigitalPin (a,2);
   rpm = [rpm i];
   i = rpm;
   plot (rpm);grid;
   title ('Sensor measurements')
   xlabel ('Time (sec)')
   ylabel ('RPM')
   if (val==1)
      t = clock;
      current_time = t (6);
      pulse_time = current_time - previous_time;
      rpm = (1/pulse_time)*60;
      previous_time = current_time;
      set (handles.rpm_value,'string',num2str(rpm));
   else
      rpm = 0;
   end
end
```