

COPYRIGHT WARNING

This paper is protected by copyright. You are advised to print or download **ONE COPY** of this paper for your own private reference, study and research purposes. You are prohibited having acts infringing upon copyright as stipulated in Laws and Regulations of Intellectual Property, including, but not limited to, appropriating, impersonating, publishing, distributing, modifying, altering, mutilating, distorting, reproducing, duplicating, displaying, communicating, disseminating, making derivative work, commercializing and converting to other forms the paper and/or any part of the paper. The acts could be done in actual life and/or via communication networks and by digital means without permission of copyright holders.

The users shall acknowledge and strictly respect to the copyright. The recitation must be reasonable and properly. If the users do not agree to all of these terms, do not use this paper. The users shall be responsible for legal issues if they make any copyright infringements. Failure to comply with this warning may expose you to:

- Disciplinary action by the Vietnamese-German University.
- Legal action for copyright infringement.
- Heavy legal penalties and consequences shall be applied by the competent authorities.

The Vietnamese-German University and the authors reserve all their intellectual property rights.





RUHR-UNIVERSITÄT BOCHUM

MechEng
Mechanical Engineering



Vietnamese-German University

WIRELESS DC MOTOR CONTROL USING A MICROCONTROLLER AND A SUPPLEMENTAL RF COMMUNICATION MODULE

BACHELOR THESIS

PLACE 2023



Vietnamese-German University

Submitted by: VUONG HOANG THIEN TAM

RUB Student ID: 108018207532

VGU Student ID: 9938

Supervisor: Dr. Liu Wai Yip

**WIRELESS DC MOTOR CONTROL USING A
MICROCONTROLLER AND A SUPPLEMENTAL RF
COMMUNICATION MODULE**

A Thesis Presented

by

VUONG HOANG THIEN TAM

Submitted to the  Department of Mechanical Engineering of the
Vietnamese-German University
RUHR – UNIVERSITÄT BOCHUM and VIETNAMESE – GERMAN UNIVERSITY

in partial fulfillment

of the requirement for the degree of

BACHELOR IN MECHANICAL ENGINEERING

March 2023

VUONG HOANG THIEN TAM

MEN2016

 **WIREFLESS DC MOTOR CONTROL USING A
MICROCONTROLLER AND A SUPPLEMENTAL RF
COMMUNICATION MODULE**

Vietnamese-German University

Approved by:

Supervisor: Dr. LIU WAI YIP

(This page is intentionally left blank)



Vietnamese-German University

DISCLAIMER

I hereby declare that the work presented here is part of my Bachelor's thesis, and is the result of my own independent research conducted under the supervision of Dr. Louis W.Y. Liu, unless otherwise specified. All data, findings, and conclusions presented in this work are my original work, and have not been previously published or submitted for any academic or professional qualification. I take full responsibility for the content of this work. I confirm that all sources used in the preparation of this work have been appropriately cited and referenced. Any quotations or excerpts from published or unpublished works are clearly identified, and the sources of these materials have been acknowledged. Furthermore, the thoughts, opinions, and recommendations expressed in this work are solely mine and do not reflect the policies or viewpoints of the Vietnamese German University, or any other organization or institution. In conclusion, this work is a genuine and authentic representation of my own research, and I am willing to defend my research and findings if required.



Vietnamese-German University

Vuong Hoang Thien Tam

ABSTRACT

Many robotic control algorithms have been proposed, but hardly any of them focus on the antenna topology. The significance is no longer placed on how much torque or how fast a motor may perform, but has instead moved onto how accurately and conveniently a motor may be controlled. This work aims to implement a wireless control system for a DC motor that remotely controls its speed and other basic functionalities such as start, stop, accelerate, and decelerate.

The methodology is described as follows: The hardware comprises a microcontroller board - Arduino UNO R3 (chip ATmega328p), a DC motor (Hitachi), an H-Bridge driver, an RF24L01 module, as well as a monopole antenna for communicating at 2.4 GHz. The motor's direction was controlled by means of an Arduino microcontroller using Pulse Width Modulation (PWM) together with a PID (Proportional, Integration, Differentiation) algorithm. Experiments with both wired and wireless setup have been conducted at the Vietnamese-German University.

Results: The outcome of this experiment has proven beyond any doubt that, using the present monopole antenna topology, the basic functionalities of the DC motor were able to be remotely controlled at a maximum distance of 400 meters. In accordance with the theoretical prediction made by Friis' formula, we have also found that the transmission range was changeable by changing the power at the transmitting end. Conclusion: Overall, this work was successful. The results further suggest that a further increase in the transmission range is possible if the antenna gain at the transmitting end and/or the receiving end is increased.

AUTHORSHIP STATEMENT

Family Name, First Name: VUONG HOANG THIEN TAM

Matriculation Number: 9938

Title of Thesis: WIRELESS DC MOTOR CONTROL USING A MICROCONTROLLER AND A SUPPLEMENTAL RF COMMUNICATION MODULE


I hereby declare in lieu of an oath that I have produced the aforementioned thesis independently and without any other means except the aids listed. Any thoughts directly or indirectly taken from somebody else's sources are made discernible as such. To date, the thesis has not been submitted to any other board of examiners in the same or a similar format and has not been published yet.

Binh Duong, March 28th, 2023

Signature

TABLE OF CONTENT

CHAPTER 1 – INTRODUCTION	6
1.1. BACKGROUND	6
1.2. THESIS STRUCTURE	7
1.3. REPORT CONTENT	7
CHAPTER 2 – THEORETICAL REVIEW	9
2.1. DC MOTOR	9
2.1.1. DC MOTOR’S WORKING PRINCIPLE	9
2.1.2. TYPES OF DC MOTOR AND THEIR EQUIVALENT CIRCUITS	11
2.1.3. BRUSHLESS DC MOTOR (BLDC)	15
2.2. PULSE WIDTH MODULATION (PWM)	15
CHAPTER 3 – DESIGN OF A CLOSE-LOOP DC MOTOR CONTROL	18
3.1. PROPOSED SYSTEM	18
3.2. HARDWARE IMPLEMENTATION	19
3.2.1. DC MOTOR	20
3.2.2. OPTICAL ROTARY ENCODER	20
3.2.3. MICROCONTROLLER – ARDUINO UNO R3	24
3.2.4. LM2596 VOLTAGE REGULATOR	26
3.2.5. H-BRIDGE MOTOR DRIVE – BTS 7960	27
3.2.6. LCD 1602 KEYPAD SHIELD	31
3.3. HARDWARE DIAGRAM	33
3.4. WIRING DIAGRAM	33
3.5. SYSTEM LAYOUT DESIGN	35

3.6. PROGRAMMING	38
3.6.1. CODE STRUCTURE	38
3.6.2. PULSE WIDTH MODULATION IN ARDUINO	39
3.6.3. TIMER/COUNTER	40
3.6.4. EXTERNAL INTERRUPT / PIN CHANGE INTERRUPT	46
3.6.5. PID TUNING	46
3.6.6. PROGRAM FLOWCHART	48
CHAPTER 4 - IMPLEMENTING NRF24L01 2.4GHz RF MODULE FOR WIRELESS DC MOTOR CONTROL	49
4.1. INTRODUCTION nRF24L01	49
4.2. SYSTEM LAYOUT	51
4.3. EXPERIMENTAL SETUP	52
4.4. PROGRAMMING	53
 Vietnamese-German University	
CHAPTER 5 – EXPERIMENT RESULTS AND DISCUSSION	56
CHAPTER 6 - FEASIBILITY OF EXTENDING THE TRANSMISSION RANGE	60
6.1. RELATED FORMULAS	60
CHAPTER 7 – CONCLUSION	63
REFERENCE LIST	64
APPENDIX 1 – MAIN SOFTWARE FOR CLOSE-LOOP CONTROL	67
APPENDIX 2 – TRANSMITTER’S CODE – LCD & KEYPAD	75
APPENDIX 3 – RECEIVER’S CODE – DC MOTOR	79
APPENDIX 4 – CODE SECTION FOR OPEN-LOOP DC MOTOR CONTROL	84

LIST OF FIGURES

Figure 1.1: Thesis' target approach method	7
Figure 2.1: A simple DC motor	10
Figure 2.2: Equivalent circuit of a separately excited motor	11
Figure 2.3: Equivalent circuit for a shunt DC motor	11
Figure 2.4: Shunt DC motor illustration	12
Figure 2.5: Equivalent circuit for a series DC motor	12
Figure 2.6: Series DC motor illustration	13
Figure 2.7: Equivalent circuit for a compound DC motor	13
Figure 2.8: Speed – Torque characteristic	14
Figure 2.9: The averaging effect on the output voltage by PWM technique	16
Figure 3.1: Block diagram of speed control of DC motor	18
Figure 3.2: Optical encoder connected to the motor shaft to detect movements	21
Figure 3.3: A disc associated with an optical incremental encoder	22
Figure 3.4: Encoder working principle for indicating motor speed	23
Figure 3.5: Example on direction determination of rotary encoder	23
Figure 3.6: Arduino UNO R3 layout	24
Figure 3.7: Arduino UNO R3 layout	26
Figure 3.8: BTS 7960 H-Bridge IC driver	28
Figure 3.9: BTS 7960's heatsink	29
Figure 3.10: BTS 7960 Pin Configuration	29
Figure 3.11: Circuit of resistors designed for the button array	32
Figure 3.12: LCD 16x02 keypad shield in the development platform	32
Figure 3.13: Hardware diagram for controlling the DC motor	33

Figure 3.14: System wiring diagram for controlling the DC motor	33
Figure 3.15: System connection testing with a mini breadboard	34
Figure 3.16: System wiring diagram for controlling the DC motor	34
Figure 3.17: Wiring goes under the plastic perforated board	36
Figure 3.18: Fixture for motor	36
Figure 3.19: Brass hexagonal standoff used for several elements	37
Figure 3.20: Finished system layout design for DC motor controlling	37
Figure 3.21: System configuration served for programming structure	38
Figure 3.22: Timer/Counter block diagram	41
Figure 3.23: Example to explain the prescaler's importance in Timer/Counter	42
Figure 3.24: DC motor control program flowchart	48
Figure 4.1: nRF24L01 module + PA + LNA + detachable antenna	50
Figure 4.2: System Block Diagram	52
Figure 4.3: System's transceiver	52
Figure 4.4: System's receiver	53
Figure 5.1: Capacitor was added on (a) : supply power source	59
Figure 5.1: Capacitor was added on (b) : nRF24L01 modules	59
Figure 6.1. The proposed antenna topology for the transmitting end	62

LIST OF TABLES

Table 3.1: DC Motor Specifications	20
Table 3.2: LM2596 Module's features	27
Table 3.3: BTS 7960 Control Logic	30
Table 3.4: Control Method specifically designed for this pin configuration	31
Table 3.5: Available frequencies for the Arduino PWM pins	39
Table 3.6: Prescaler values and PWM frequencies for an 8-bit counter	43
Table 3.7: TCCR2A Timer 2 Control Register A	43
Table 5.1: Basic Functionality Tests	57
Table 5.2: Transmission Range Test	58

CHAPTER 1 – INTRODUCTION

1.1. BACKGROUND

DC motor has a fairly long historical development process and has been dominating the market in a vast range of applications since its first invention for commercial use in 1886 by American scientist Frank Julian Sprague [1], and still remains the preferred choice [2] even with the appearance of its "close cousin" – brushless DC motor (BLDC) one century later in the late 1980s when permanent magnet materials became readily available [3].

Here at Vietnamese-German University (VGU), a lot of expensive instruments have been purchased, such as robotic arms and DC motors, but very few of them are remotely controlled. Even if a small subset of equipment is remotely controllable, almost none of it is under the direct management or control of the members at VGU. For the first time, wireless control at VGU involving a special antenna topology became an undergraduate project.

Receiving such attention, there were numerous experiments that we, the researchers and students, set out to study and, more importantly, to remotely control DC motor performance. One of the key aspects of controlling DC motor is using wireless technology [4]. Being able to control automotive devices from a distance without any hardware linked in between would allow an unlimited amount of creativity in invention. This is the source of inspiration for this topic.

1.2. THESIS STRUCTURE

The project is broken down into two specific phases that would be followed in an orderly manner:

- *Phase 1:* Control and display DC motor speed and direction
- *Phase 2:* Implement RF module into system achieved from first phase for wireless control

Even when this topic is neither original nor complicated, as demonstrated in some related work [4] and [5], it deserves a proper approach structure.

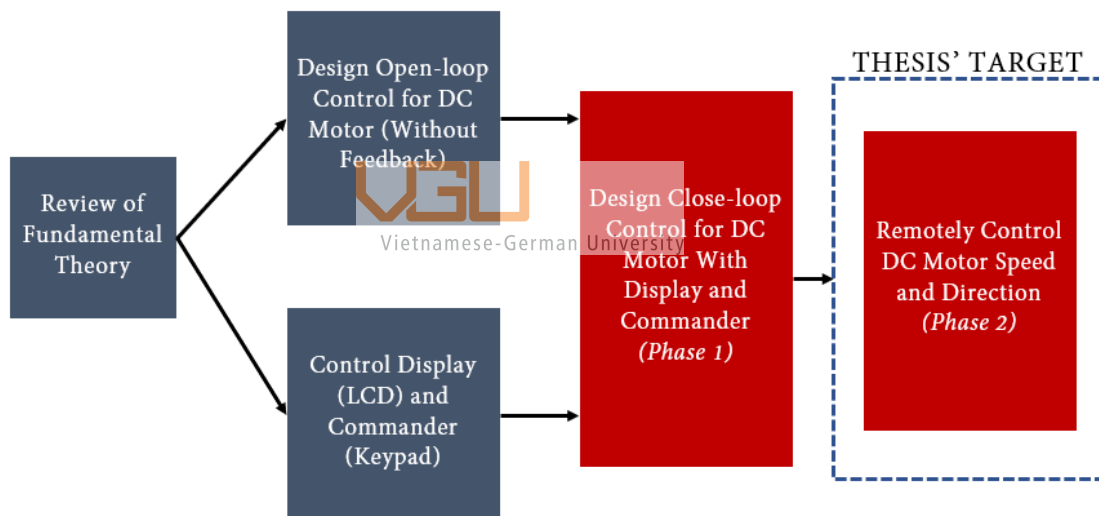


Figure 1.1: Thesis' target approach method

1.3. REPORT CONTENT

The thesis report comprises of following contents:

- **Chapter 1** is used for introduction, where the problem statement is defined along with a brief overview of the thesis structure.

- **Chapter 2** constructs a theoretical framework as foundation of the project, including operating principle of DC motor, related formulas; PWM and PID control methods' basic ideas.
- **Chapters 3 and 4** describe the implementation of the research, where the former pays attention to Phase 1 (Close-loop DC Motor Control) and the latter focuses on Phase 2 (Wireless DC Motor Control), which is also the main study goal of this thesis.
- **Chapters 5 and 6** discuss the achieved results while considering the feasibility of further improvement by enhancing the transmission range.
- **Chapter 7** concludes what has been done in this work.



CHAPTER 2 – THEORETICAL REVIEW

2.1. DC MOTOR

In present day, with the fast-paced of technological growth, the traditional DC motor is still widely used in many fields – from household appliances to industrial electrical equipment, especially in the lower power range.

2.1.1. DC MOTOR'S WORKING PRINCIPLE

DC motor, or Direct Current motor, works on a simple mechanism to converts electrical energy into mechanical energy, which is the magnetic effect of current. This effect can simply be described as:

Electric current passes through a coil, initiating a magnetic field. With magnets, similar magnetic poles repel while opposite magnetic poles attract each other. Thus when the field is excited and current is supplied to armature, these repulsive or attractive forces combine, tend to drive the coil of the motor to rotate [6].

Every DC motor has two main parts. The fixed part is called the stator (often a permanent magnet) and the rotating part is called the rotor (armature). The armature consists of a pair of commutator rings and brushes. These parts generates magnetic fields, which interact and make the motor spins.

As direct current flows from negative to positive through the wire coil, it creates an electromagnetic force going up and the coil spins towards the magnet. The commutator part that connects with the power source of opposite polarity acts as a switch for alternately flipping the poles. Without the commutator rings, the coil always has the same polarity and is unable to spin since it is just attracted to one of the magnets. Thus, motors are designed with a conductive ring – the commutator – around the motor shaft.

Current is transferred from the power source to the commutator through stationary brushes. They are made of soft conductive material that presses against the commutator. As the rotor turns, different segments of the commutator touch the brushes and change the direction of current through the coil as well as its polarity. This happens repeatedly and results in continuous rotation of the motor.

Manufactured motors are designed to enhance rotation performance by using multiple windings with separate commutator pair for each loop. This design overcomes the shortcoming of irregular motion observed at simple DC motor, which caused by the zero torque when the coil is nearly perpendicular to the magnetic flux. This arrangement allows multiple points of force that are attracting and repelling, causing the motor to spin more stable and smoother.

In a practical motor, especially a large one, an electromagnet is preferable for stator part than permanent magnet.

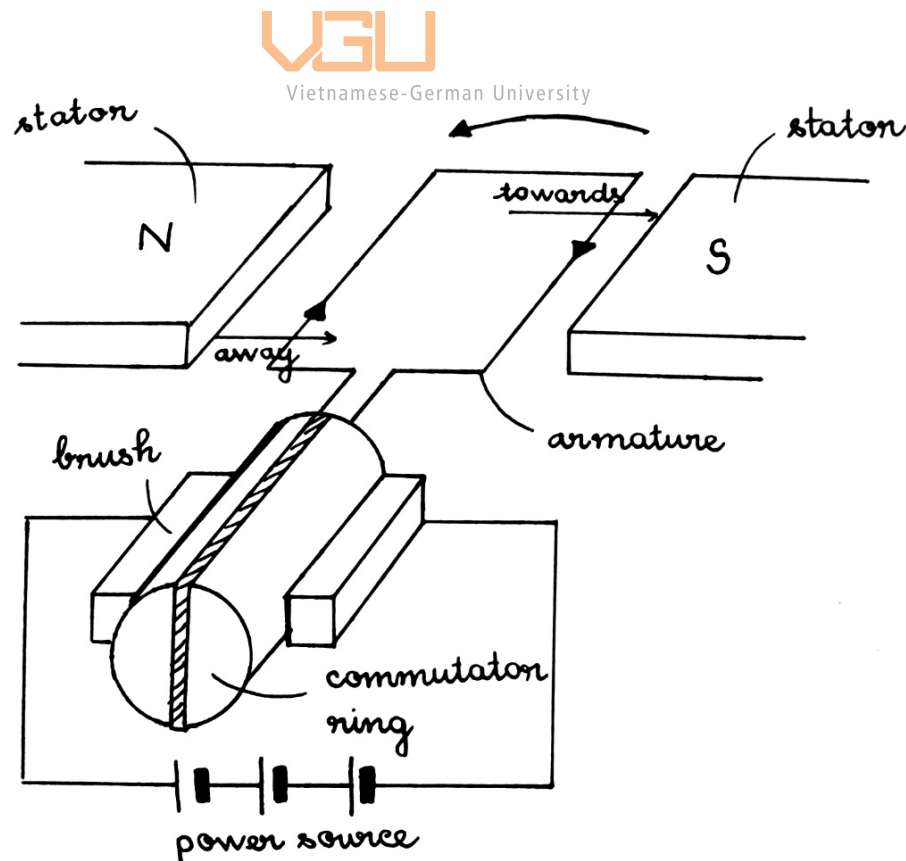


Figure 2.1: A simple DC motor (hand-drawn by the author)

2.1.2. TYPES OF DC MOTOR AND THEIR EQUIVALENT CIRCUITS

Direct current motor constructions can be classified into four groups based on the arrangement of their field windings [7], e.g., how the field coils is connected to the rotor windings:

1. **Separately excited motor:** the field windings and armature circuits are excited by separate sources [7]. Most often, the field circuit is a permanent magnet for small motors, thus the flux of the field cannot be adjusted.

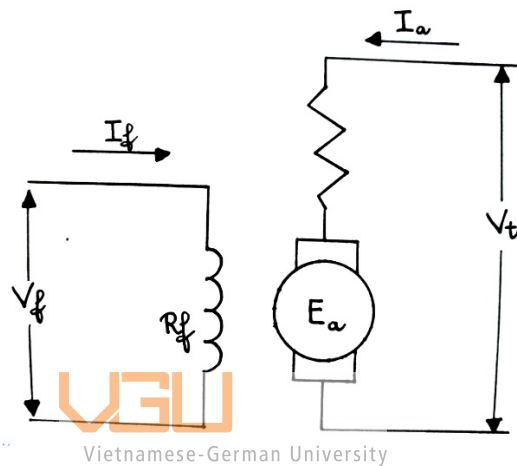


Figure 2.2: Equivalent circuit of a separately excited motor (hand-drawn by the author)

2. **Shunt motor:** The current I of the source is calculated by: $I = I_a + I_f$

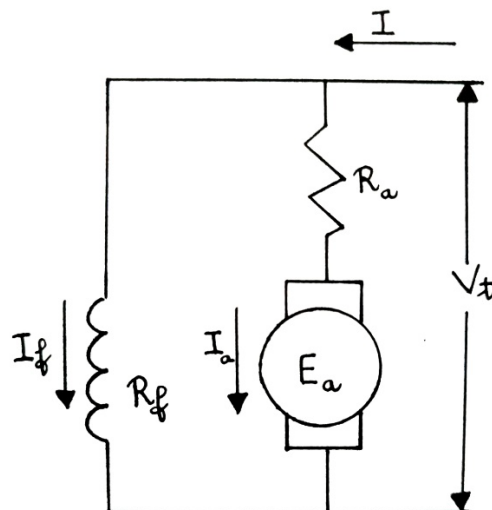


Figure 2.3: Equivalent circuit for a shunt DC motor (hand-drawn by the author)

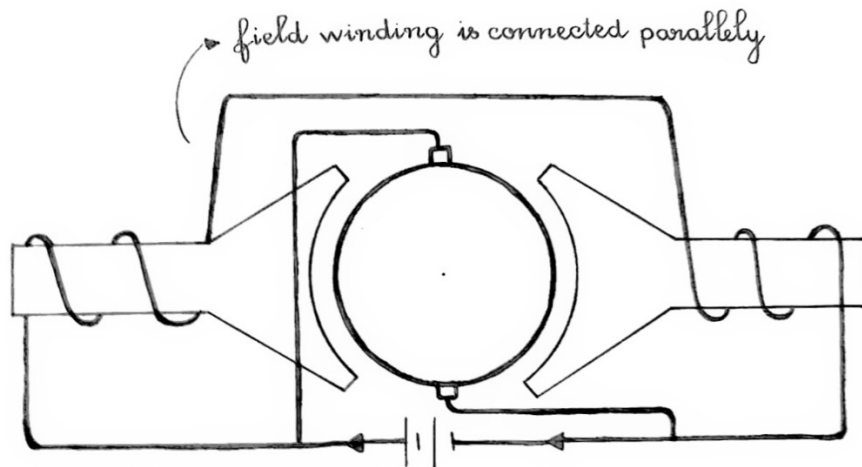


Figure 2.4: Shunt DC motor illustration (hand-drawn by the author)

3. **Series DC motor:** Series DC motor may carry a much larger current than the shunt field winding because the current of the series winding is equal to the armature current while the current of the shunt winding is: $I_f = \frac{V_t}{R_f}$, which is the supply voltage divided by the field resistance [7].

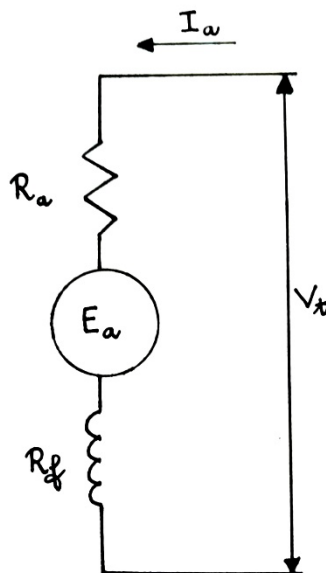


Figure 2.5: Equivalent circuit for a series DC motor (hand-drawn by the author)

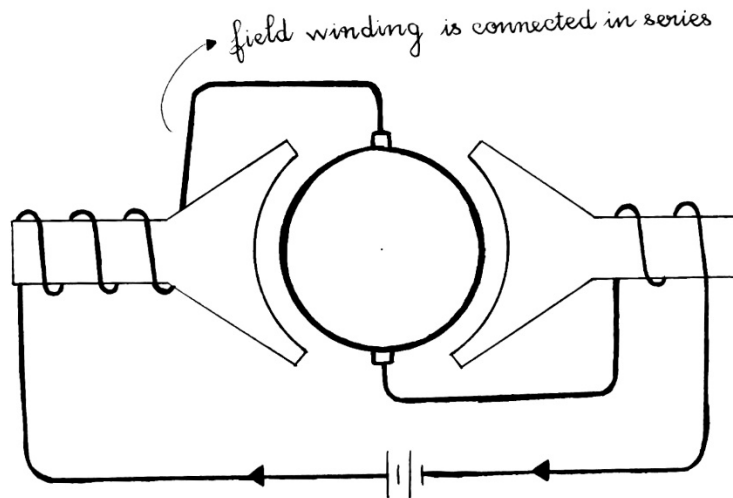


Figure 2.6: Series DC motor illustration (hand-drawn by the author)

4. **Compound DC motor:** as its name suggests, a compound DC motor is composed of both shunt and series windings in its design. The more widely used configuration is called the cumulative compound.

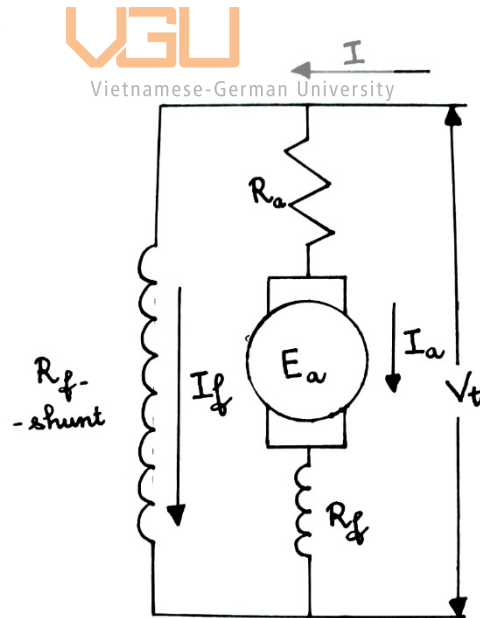


Figure 2.7: Equivalent circuit for a compound DC motor (hand-drawn by the author)

The performance of a DC motor is presented by the three important characteristics, given by the relation among the armature current, torque and speed. Those are torque

and armature current characteristic; speed and armature current characteristic; and speed and torque characteristic – this is also often referred as mechanical characteristic of a DC motor [8].

The mechanical characteristic of three DC motor types is plotted for comparative purpose:

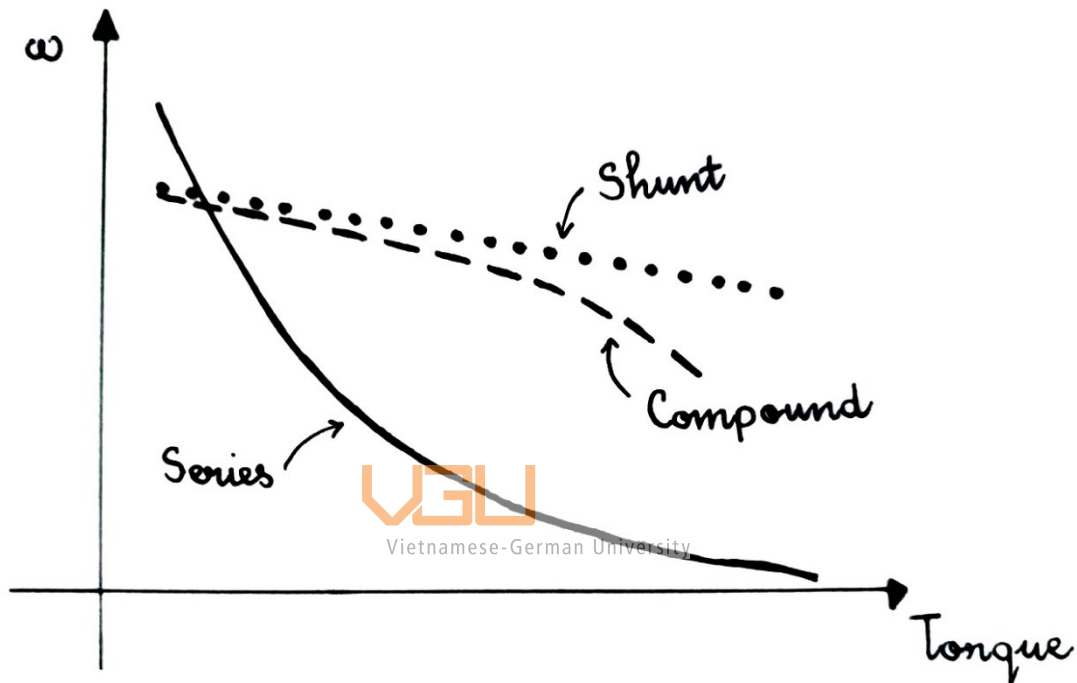


Figure 2.8: Speed – Torque characteristic of Shunt, Series and Compound DC motors (Measurements done by the author)

To conclude, the shunt DC motor has a low starting torque but it allows constant and stable speed irrespective of the load acting on the motor. The series DC motor has considerably high starting torque compared to the shunt DC motor. The speed of series DC motor when not connected to a mechanical load is excessively high that it may damage the motor due to excessive centrifugal forces exerted on the rotor. However, the speed drops rapidly with increasing load. For the compound DC motor, its characteristic lies in between of the shunt and series DC motor, e.g., its speed also decreases with load but not as drastically as series motor and it may develop large torque

nearly similar to series DC motor. At no-load condition, compound DC motor runs at more stable speed than series DC motor [8].

In short, depending on the purpose of use, field winding configuration will be decided. Series DC motor is good for high torque low speed while Shunt DC motor produces high speed low torque and Compound DC motor has a balanced performance that gives steady speed with acceptable torque.

2.1.3. BRUSHLESS DC MOTOR (BLDC)

Nowadays, along with the discovery of semiconductors and the invention of solid-state electronics came the development of brushless motors, brushed DC motors are being phased out and less popular than it used to be. However, they are still useful and can be found in a wide range of applications.

Brushed DC motors require low cost of construction. They are considerably simple and inexpensive to control. Brushed motors also benefit from high flexibility, e.g., they can be used in certain extreme environment since there is no concern for any electronics to malfunction. For brushed DC motors, the commutation is crucial to keep the rotor spinning by continuously switching the polarity of the current in the coil windings. This commutation process is done mechanically where the brushes come in contact with the commutator of the rotor. Due to this physical contact, the brushes wear out over thus make the motor performance less efficient [3], the brushes can still be maintained periodically or rebuildable to extend the lifetime of the motor, which is lower in cost compared to the brushless DC motors in certain cases.

2.2. PULSE WIDTH MODULATION (PWM)

Ideally, DC motor control of varying speed is done by varying the DC voltage supplied, however, that is not the case in reality. One common technique called Pulse Width

Modulation (PWM) may be implemented to adjust the voltage fed to the motor. Its target is to generate a waveform as trains of switched pulses to achieve an averaged output [9].

The idea behind this technique is to keep switching on and off the supplied voltage at a fast rate in order to achieve an average value of voltage. Duty cycle is the term used to describe the average power, which means the lower the duty cycle, the lower the power. Duty cycle is expressed in percent, with 0 percent corresponds to being fully off and 100 percent means on all the time [9]. Duty cycle can be expressed mathematically as follows:

$$\text{Duty Cycle} = \frac{T_{ON}}{T_{ON} + T_{OFF}} \times 100\% ; [9]$$

where T_{ON} is the time-on and T_{OFF} is the time-off of the electrical signal.

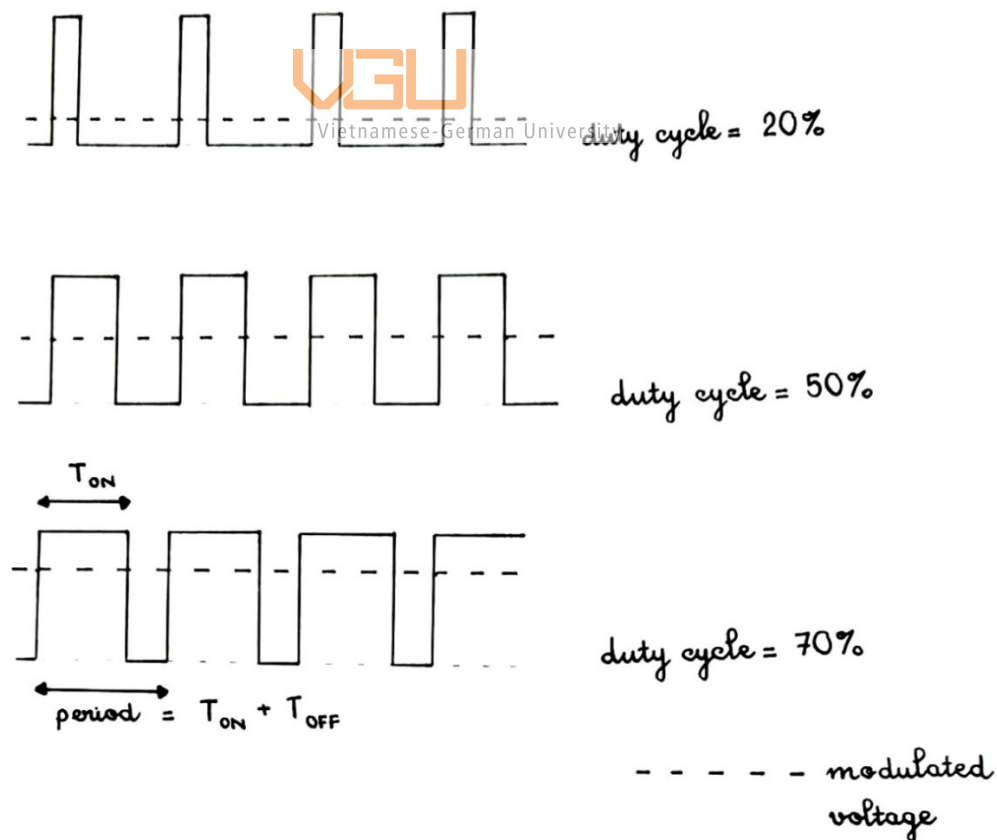


Figure 2.9: The averaging effect on the output voltage by PWM technique (hand-drawn by the author)

Most of the time, the PWM signal is a rectangular wave that repeats itself. PWM's result is not always an averaged voltage. Its output may be disturbed with noticeable ripples as the consequence of trying to follow the expected rectangular wave shape. This happens when the duration of $T_{ON} + T_{OFF}$ is considerably long. This will lead to irregular motion and the motor will alternately speed up and slow down. Solution given is to increase the switching frequency until motor performance is acceptably smooth. However, even with efficient frequency, ripples still occur due to the switching nature of PWM [10].

CHAPTER 3 – DESIGN OF A CLOSE-LOOP DC MOTOR CONTROL

3.1. PROPOSED SYSTEM

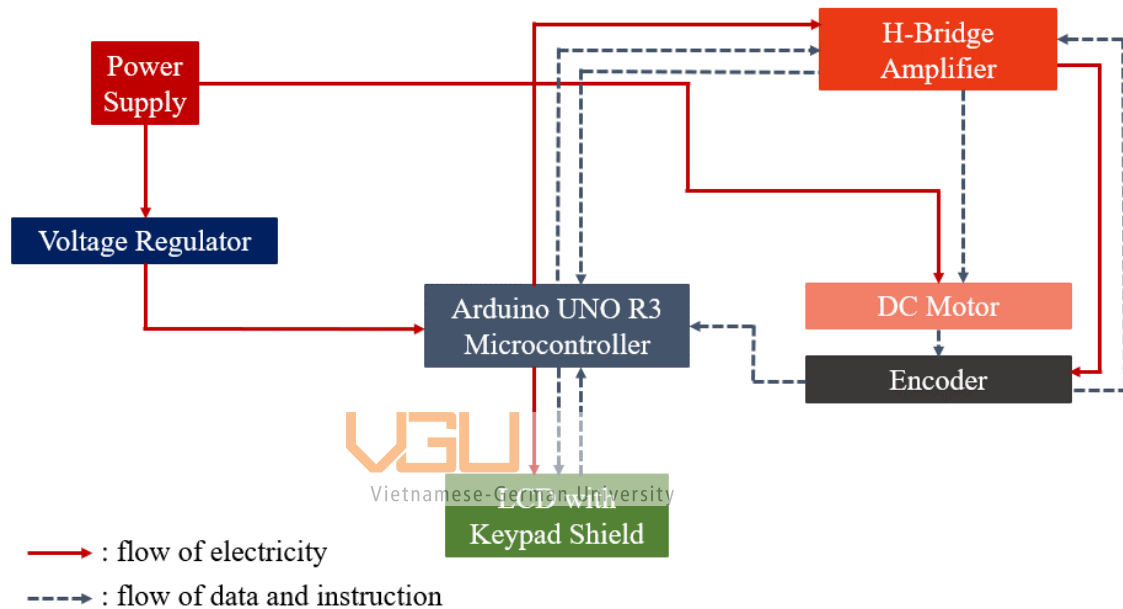


Figure 3.1: Block diagram of speed control of DC motor using Arduino UNO R3 (ATmega328P) (Created by the author at Vietnamese German University)

Figure 3.1 shows the block level representation of this system.

Since the Arduino board supply current is only about 40mA, which is not sufficient enough to power most motors directly, the microcontroller and the DC motor will share one mutual power supply of 24V. Thus, a step-down voltage regulator is used for the microcontroller. It takes input voltage of supply power source and reduce them to a lower fixed output voltage (5V for Arduino) that the board can consume without getting harm.

The Arduino UNO has in total three power pins for powering other components, one has a supply voltage of 3.3V and two pins provide 5V. These two 5V pins will supply the H-bridge amplifier and the LCD.

The DC motor is equipped with an optical rotary encoder. The actual motor speed is recorded thanks to the encoder in a form of binary values that the microcontroller interprets. The microcontroller will then execute a set of calculations to determine the Pulse Width Modulation (PWM). This PWM signal will either accelerate or decelerate the rotating speed through the H-bridge driver to reach or maintain the desired speed.

The system information (set speed and actual speed – positive value for clockwise direction and negative value for counter-clockwise direction) is displayed using a 16x02 LCD. The keypad provides a commander for simple system setting such as power ON/OFF, system reset, increase and decrease motor speed.

Following criteria are to be taken into consideration while designing the system:

- Effectively adjust the desired motor speed and direction
- Use keypad to command: Start/Stop, System reset, Speed increase, Speed decrease
- Accurately display system data

3.2. HARDWARE IMPLEMENTATION

This section discusses about hardware components that included in the project:

- Power supply: power supply 24V DC 5A
- Step-down voltage regulator module: LM2596
- Microcontroller: Arduino UNO R3 (ATmega328p)

- H-Bridge Amplifier: BTS 7960
- DC motor: Hitachi motor series, equipped with an optical rotary encoder
- LCD 16X02 keypad shield (connect with the microcontroller via screw shield)

The aim of this first phase of the project is to control the speed of the DC motor using Arduino UNO R3. As far as the hardware implementation is concerned, it is designed for close loop operation.

3.2.1. DC MOTOR

The DC motor used in this project is a brushed series DC motor. Specific characteristics of this motor is shown in *Table 3.1*.

Attribute	Value
Type	Hitachi D06D401E DC Motor
Supply Voltage	30V DC
Output	42W
Current	2.3A
R.P.M.	2750

Table 3.1: DC Motor Specifications (Data taken from the datasheet with author's modification)

3.2.2. OPTICAL ROTARY ENCODER

In this system, the encoder is mechanically attached to the motor shaft so that its output signal will monitor and change as the rotor moves (*Figure 3.2*).

Rotary encoder is a device that reads and translates the rotational information in suitable binary representation that the microcontroller may interpret. There are multiple

technologies used for encoding the rotational displacement, for example, an optical encoder detects light and a magnetic encoder detects a magnetic field distribution, but the most famous and widely used encoders utilize optical means because its reliability, robustness and inexpensive cost surpasses the advantages offered by any other technology [11].

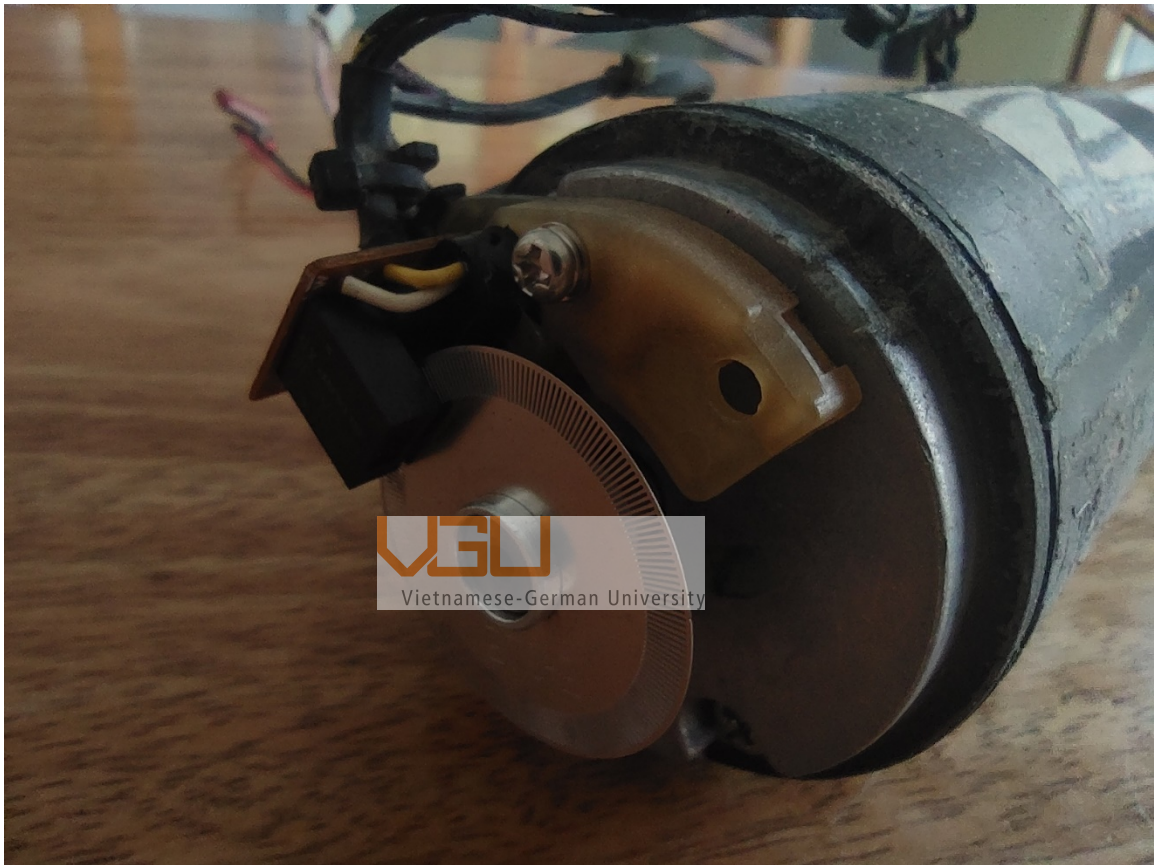


Figure 3.2: Optical encoder connected to the motor shaft to detect movements
(Photo taken for this thesis project)

Rotary optical encoders are classified into two types: absolute or incremental encoders, which describes the desired signal output for the encoders.

This particular encoder used in this system is an incremental one (*Figure 3.3*). It is mounted on a shaft which is the rotating part and thus will be able to generate incremental data in the form of digital pulses. It measures the instantaneous angular position relative to some arbitrary datum point but unable to give the absolute position

[12]. This means the microcontroller may calculate – based on the received signal – how much the shaft has rotated since the rotational data has been started recording.

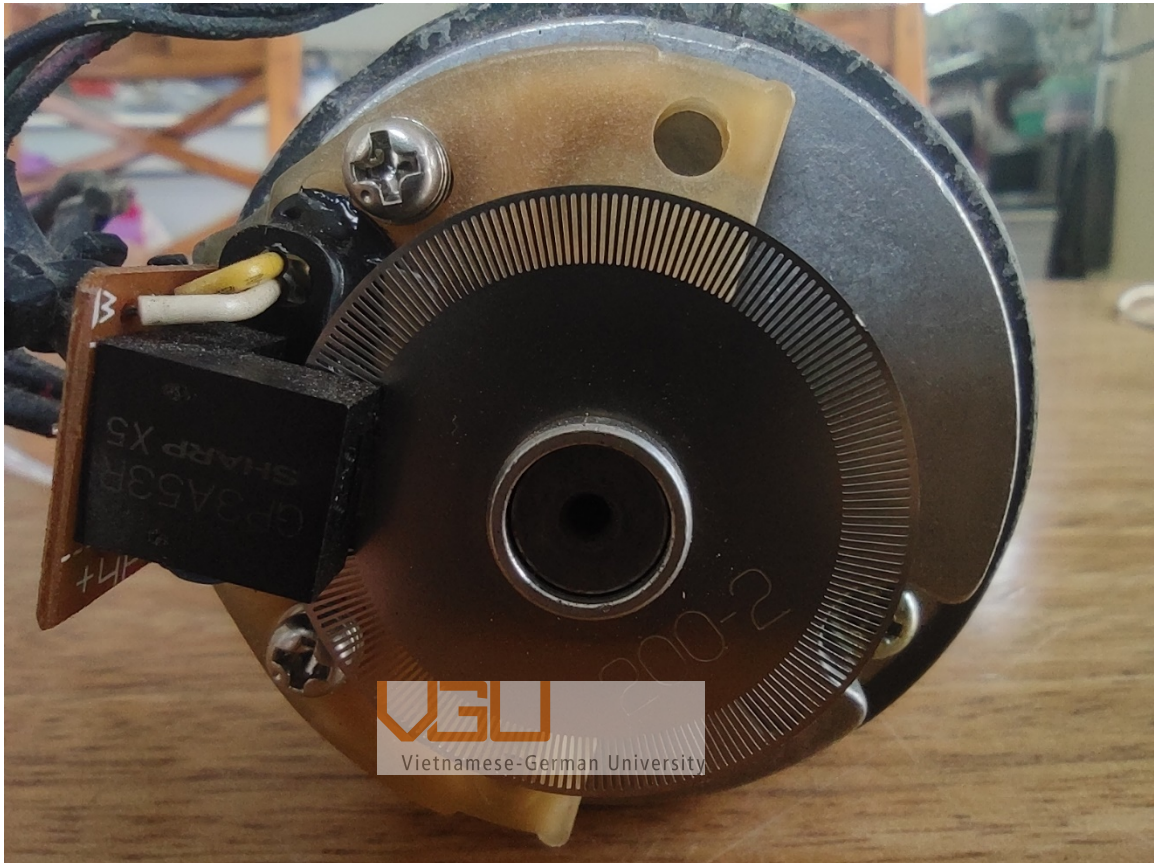


Figure 3.3: A disc associated with an optical incremental encoder (Photo taken for this thesis project)

The optical encoder consists of three elements: one is an emitter, which is a light source, generally infrared LED; one is a detector, which is a phototransistor; and a disc in a shape of a wheel with slits in a specific pattern. These slits are designed to be able to prevent or let light pass through.

The resolution of an optical encoder is based on the number of slits on the disc. Therefore, the higher the slits, the higher resolution achieved. Here the encoder resolution is 240 P/R (Pulse per Rotation).

Principle of operation: When the phototransistor detects appearance of light, it returns value of “HIGH” or 1. Otherwise, when there is no light passed through, the output goes “LOW” or 0. By counting these pulses, the microcontroller may determine the

displacement of the motor shaft and thus, indicate motor speed by timing the frequency of these pulses.

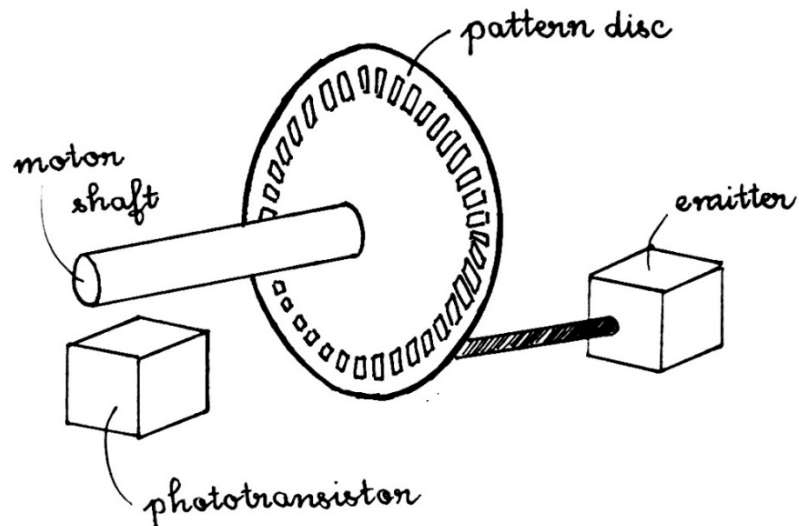


Figure 3.4: Encoder working principle for indicating motor speed (hand-sketches by the author)

Furthermore, the encoder is able to detect not only the speed but also the direction of rotation by installing two tracks that are coded 90 electrical degrees offset. These two tracks are usually denoted as Channel A (Phase A) and Channel B (Phase B). The direction is then determined based on the relationship between Phase A and Phase B.

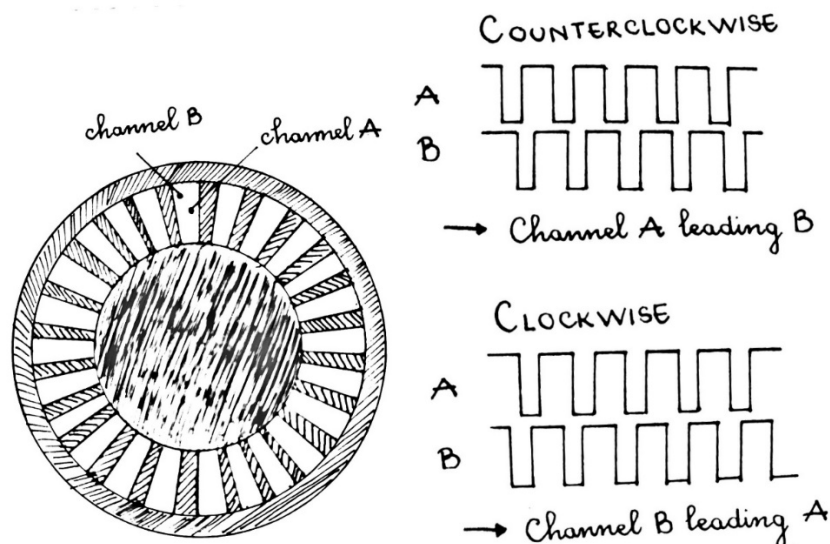


Figure 3.5: Example on direction determination of rotary encoder (hand-sketches by the author)

With the configuration like *Figure 3.5*, when the disc is rotating counterclockwise, the material will block light received by phase A first and phase B. Then, as a result, channel A reaches low state before channel B. Similar explanation takes place when the disc rotates clockwise, which means channel B is leading channel A. This type of configuration is known as quadrature encoder.

For higher resolution, it is possible to let the controller counter to take in account the number of rising and falling edges of pulses generated by one channel or even both of two channels.

3.2.3. MICROCONTROLLER – ARDUINO UNO R3

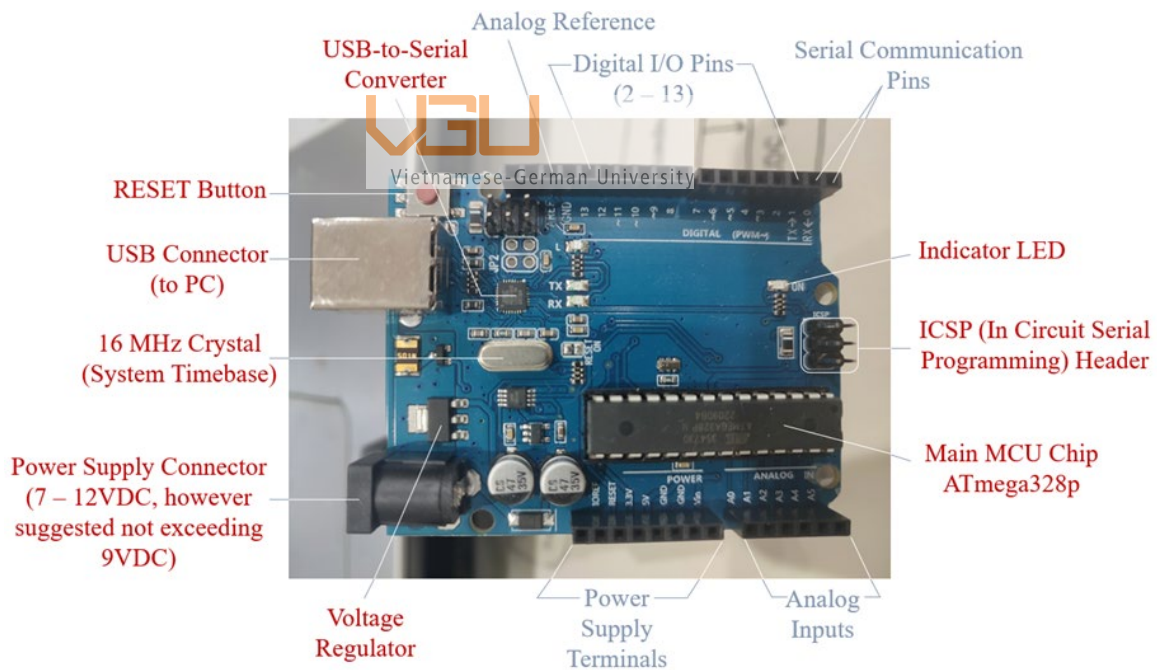


Figure 3.6: Arduino UNO R3 layout (Photo taken by the author for this thesis project at Vietnamese German University)

There are multiple Arduino board types. This project employs the UNO R3 version whose host processor is the Atmel Atmega328P.

It is a low-power, high performance microcontroller. The “Atmega328” implies that it is an 8-bit microcontroller with 32KB of integrated Flash memory, “P” simply denotes that it needs less power than its predecessors. It also has 2KB of SRAM (Static Random Access Memory) and 1KB of EEPROM (Electrically Erasable Programmable Read-Only Memory). This MCU operates due to a crystal of 16MHz on the UNO board [14]. When being activated, the current consumption is 0.2Ma.

The Arduino UNO has a resettable polyfuse to prevent the board from drawing too much power through the computer’s USB that may be resulted in shorts and overcurrent. Once that happens, the fuse will automatically break the connection until the case is solved.

Up to now, the UNO board is considered to be the most popular version due to its powerful performance, high flexibility (for example, built-in USB interface) and cost-effective solution, thus make it one of the easiest-to-use Arduino board. This statement can be proved due to the fact that all Arduino shields ever made are compatible with the UNO.



The processor is equipped with a wide variety of features [15]. This project utilizes and focuses on its memory system, port system, timer system, Analog-to-Digital converter (ADC), interrupt system, and the Serial communication system.

Arduino has its own development board allowing users to program and burn through Arduino IDE (Integrated Development Environment). The original Arduino programming language is based on C++. However, it is possible to code in Python or any other high-level programming language. Simply put, it can be sketched using any programming language as it is developed into binary code by the compiler. The code is initially saved as a piece of text and then converted into machine code and resulted in a single hex file [14].

3.2.4. LM2596 VOLTAGE REGULATOR

The power source in this design is a honeycomb mesh power supply of 24V, 5 amperes, which is connected to a voltage regulator module (or buck converter) LM2596. This LM2596 drops the power voltage to nearly 6V before passing it to the Arduino board. Its output voltage can be adjusted by turning the potentiometer on the module.

Besides voltage-step-down purpose, the module is also served for protection of the microcontroller. Leakage of AC voltages in Mv is enough to damage the microcontroller. In order to avoid this kind of situation, a smooth and stable DC voltage is required. This can be approached by implanting a DC-to-DC buck converter, in this project, the LM2596, to ensure a supply DC voltage with negligible ripples to the electronic circuit components, especially the Arduino board.

The LM2596 series operates at a 150 kHz switching frequency, thus it is able to filter components of smaller sizes than would be expected with lower frequency switching regulators [16].



Vietnamese-German University

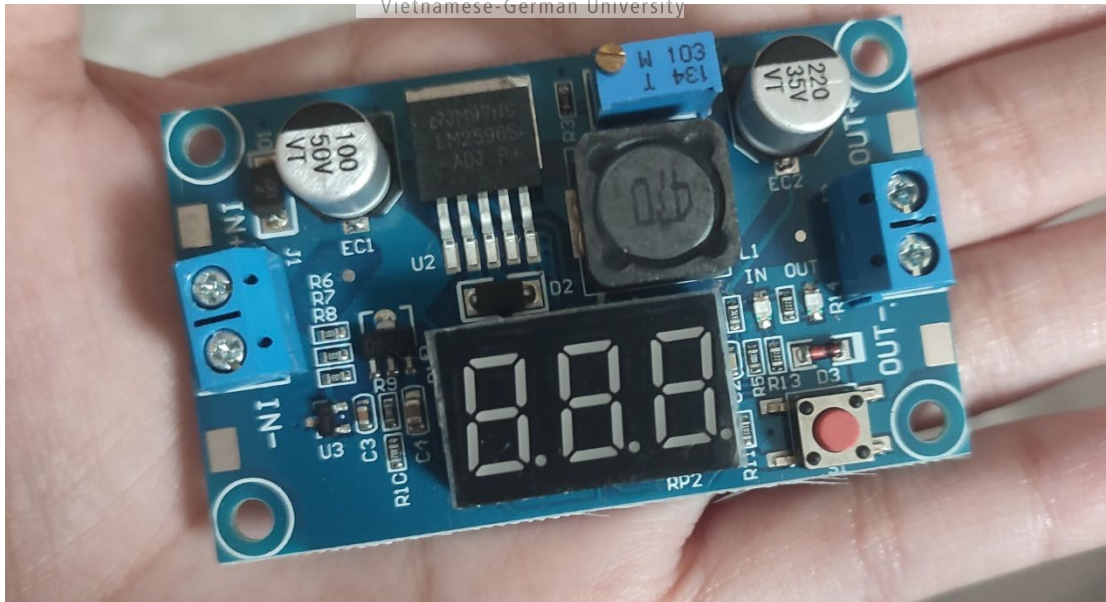


Figure 3.7: Arduino UNO R3 layout (Photo taken by the author for this thesis project)

Features:

-
- Wide input voltage range from 3V up to 30V
 - Adjustable output voltage range from 1.5V – 30V
 - Guaranteed 3.0A output load current
 - Wide input voltage range from 3V up to 30V
 - 150 kHz fixed frequency internal oscillator
-

Table 3.2: LM2596 Module's features

3.2.5. H-BRIDGE MOTOR DRIVE – BTS 7960

There is a wide range of options when considering components to drive the motor, it is often more convenient and easier to use an H-bridge driver in the form of IC (Integrated Circuit). Nowadays, many manufactures offer fully integrated H-bridge circuits that can control high-power motors without additional external components. Furthermore, it also prevents short circuit which could arise if switches are directly controlled.

Along with that, there are three DC motor specifications that need to be taken into account when selecting the suitable motor driver:

1 – Operating voltage: driver should be able to comfortably handle this voltage.

2 – Average current: the amount of current that the motor consumes under normal load condition.

3 – Stall current: motor driver needs to have a peak current that is capable of handling current which motor would draw when starting from rest or in such situation that its shaft is held in place and rotor is forced to stop.

BTS 7960 H-bridge IC driver, also known as the IBT2, is utilized in this project (*Figure 3.8*).

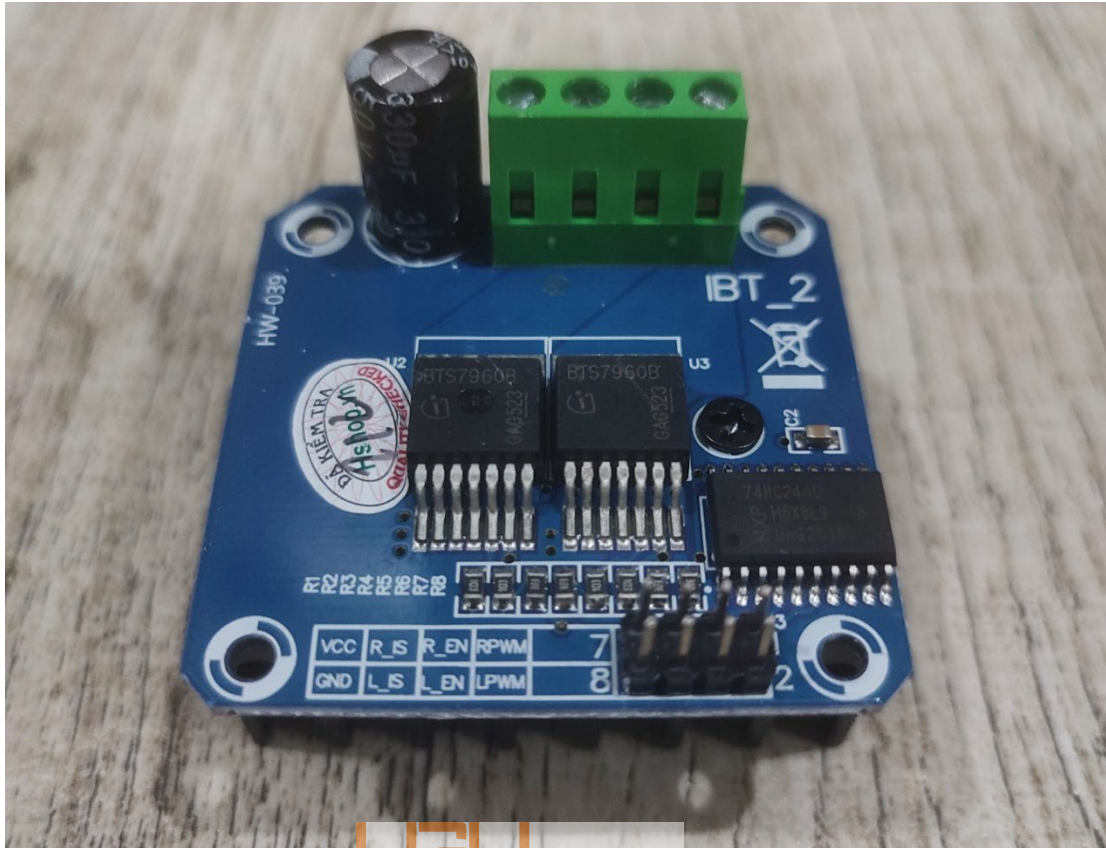


Figure 3.8: BTS 7960 H-Bridge IC driver (Photo taken by the author for this thesis Vietnamese-German University project).

It is a powerful fully integrated high-current H-bridge designed to provide bidirectional motion controlling. This board is composed of 2 half-bridges and supports a current up to 43A under a voltage between 5.5V – 27V. It is built using a multi-technology process which combines one p-channel highside MOSFET and one n-channel lowside MOSFET [17].

This IC is featured with protection such as [18]:

- **Overvoltage Lock Out:** if the supply voltage exceeds the permitted voltage level, the IC will shut the lowside MOSFET off and turn the highside MOSFET on, which will lead to freewheeling in highside during over voltage.
- **Undervoltage Shut Down:** to avoid uncontrolled motion of motor: if the supply voltage drops under 5.4V, the driver will switch off and unable to be back on until voltage reaches 5.5V.

- **Overtemperature Protection:** the BTS 7960 has an integrated temperature sensor that protects against overtemperature by shutting down of both output stages. Furthermore, it is also equipped with a heatsink to reduce dissipating heat due to its large current (*Figure 3.9*).
- **Short Circuit Protection:** is triggered by a combination of current limitation and overtemperature shut down.

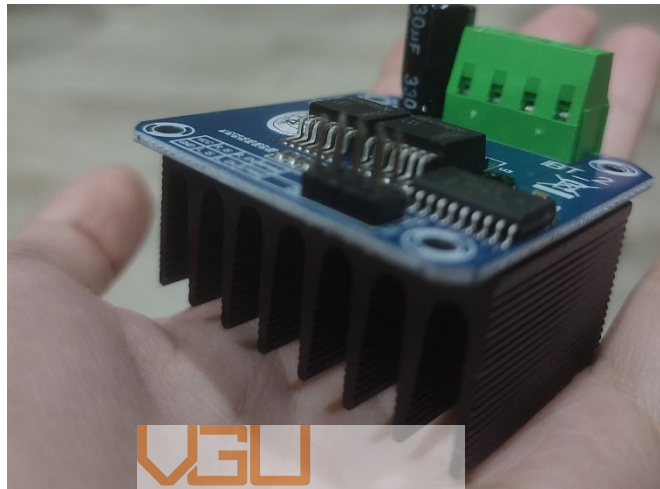


Figure 3.9: BTS 7960's Heatsink (Photo taken by the author at Vietnamese German University)

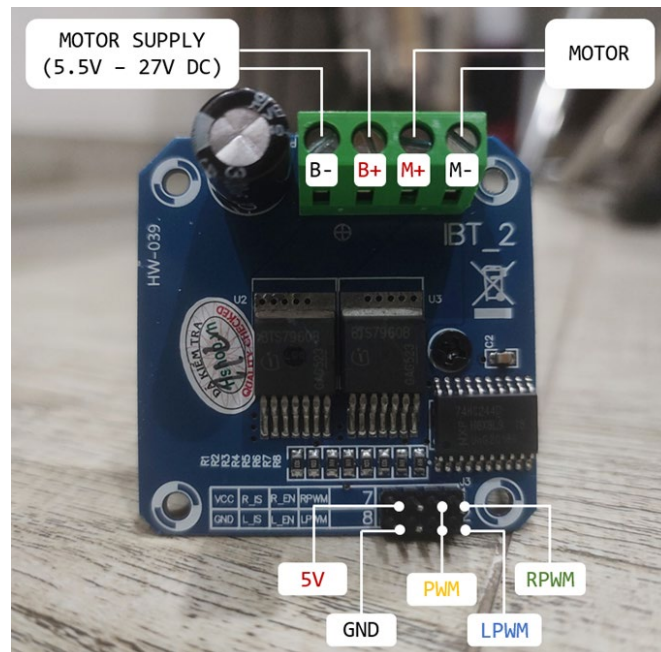


Figure 3.10: BTS 7960 Pin Configuration (Photo taken by the author at Vietnamese German University)

PWM	1 – 255	1 – 255	1 – 255	1 – 255
RPWM	1	0	0	1
LPWM	0	1	0	1
	Counter-clockwise	Clockwise	Lowside Braking	Highside Braking

Table 3.3: BTS 7960 Control Logic (Photo taken by the author at Vietnamese German University)

Table 3.3 shows the BTS 7960 control logic.

The lowside and highside braking output is realized by current limitation feature offered by the IC manufacturer to protect the module from short circuit [18]. Braking is a term used when the motor is rapidly slowing down by short-circuiting its two terminals, which make the motor much harder to turn thanks to an effect called electrodynamic braking [19].




There are four inputs to the module: LPWM; RPWM; L_EN; R_EN where the left and right enable pins refer to the left and right half bridges. This configuration shorts R_EN and L_EN together to avoid damaging the module by mistakenly set their values both HIGH. The reason for enabling one side is to fix the motor rotation direction. The combined enable pin is used as a HIGH/LOW signal or PWM signal to either enable or disable or control speed of the whole module and motor. Therefore, the LPWM and RPWM input pins directly control the direction state of outputs. The combined PWM pin is used for PWM control of the motor.

This IC can be interfaced directly to the microcontroller board [18].

ENABLE pin (R_EN + L_EN)	DIRECTION pin		MOTOR STATUS
	RPWM	LPWM	
0	-	-	OFF/STANDBY
1	0	0	Lowside Braking
1	0	1	Clockwise
1	1	0	Counterclockwise
1	1	1	Highside Braking

Table 3.4: Control Method specifically designed for this pin configuration (table taken from [18] with author’s modification)

3.2.6. LCD 1602 KEYPAD SHIELD

This module is a combination of a two-lines-six-teen-characters LCD display and an array button of five that shield on top of the Arduino board, which save space and make a neat development platform.  Vietnamese-German University

For programming, the LCD display element uses the same library for the “traditional” LCD 16x02 module, which is LiquidCrystal library available in Arduino IDE. The keypad on this shield includes a total of six keys where five of them is programmable. The right-most button labelled RST is mapped directly to the RST button on the Arduino, whose function is to reset the board. The other five buttons are mapped to analog pin A0 on Arduino. They are assigned different values ranging from 0 to 1023 and will function depending on which value is written, in other words, on which button is pressed. This is achieved by a circuit of resistors connected as illustrated (*Figure 3.11*); and pin A0 measures the potential value to call the function.

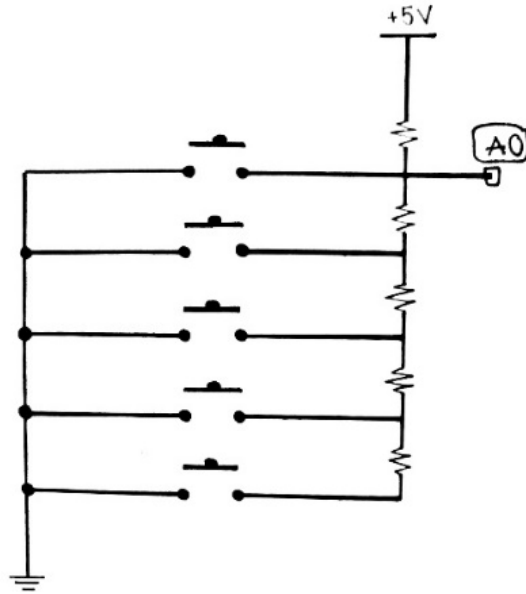


Figure 3.11: Circuit of resistors designed for the button array (diagram hand-drawn by the author at Vietnamese German University)

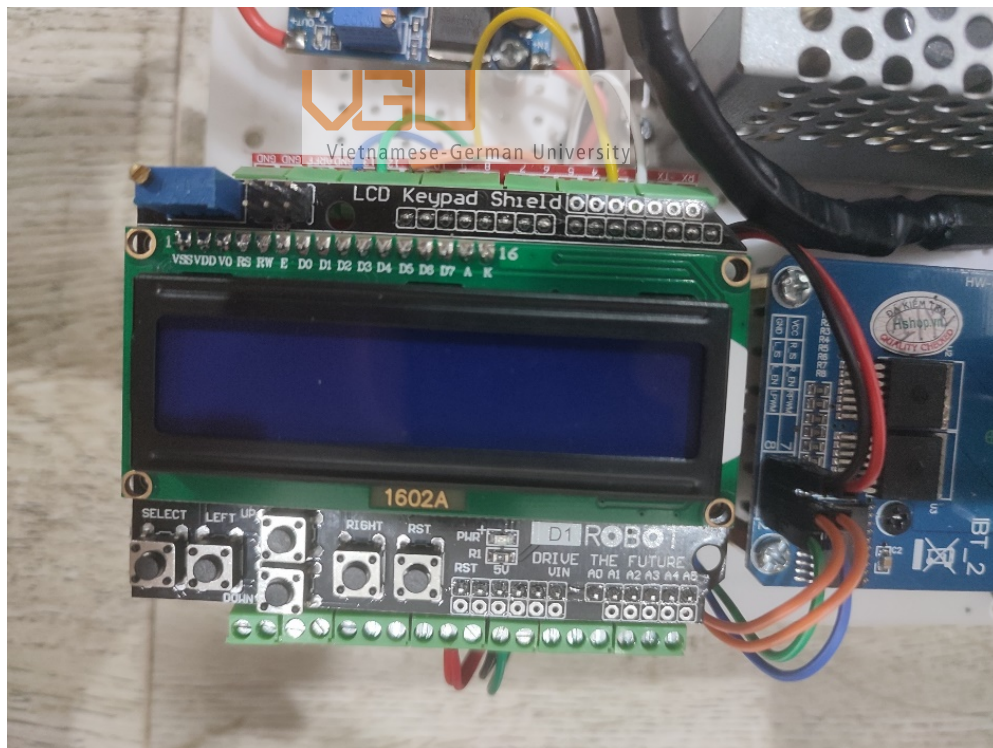


Figure 3.12: LCD 16x02 keypad shield in the development platform (Photo taken by the author at Vietnamese German University)

3.3. HARDWARE DIAGRAM

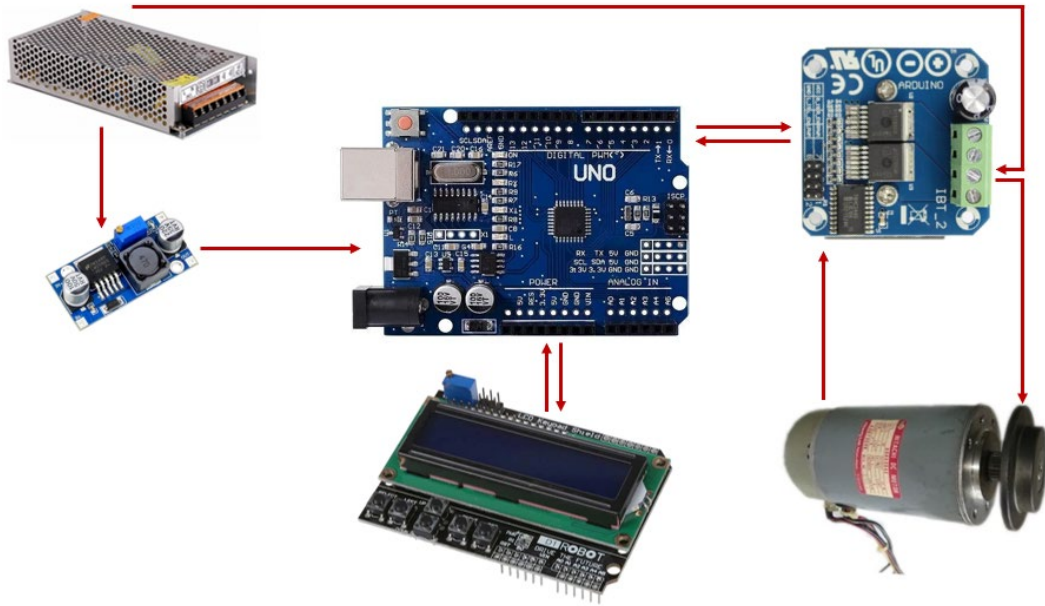


Figure 3.13: Hardware diagram for controlling the DC motor (Photo taken by the author at Vietnamese German University)



3.4. WIRING DIAGRAM

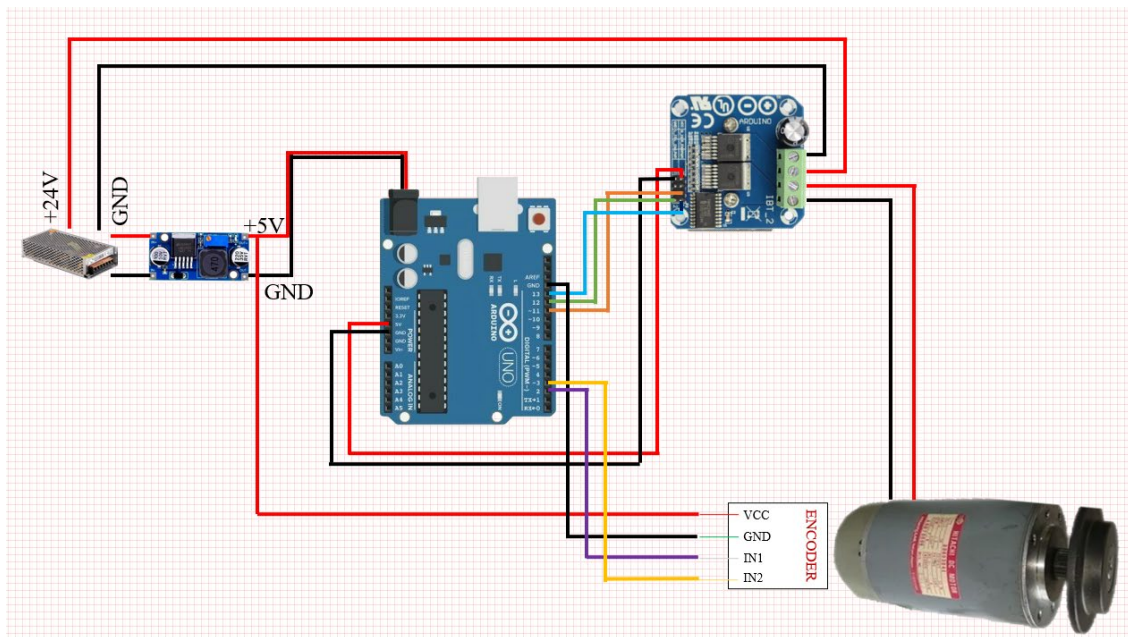


Figure 3.14: System wiring diagram for controlling the DC motor (Photo taken by the author at Vietnamese German University)

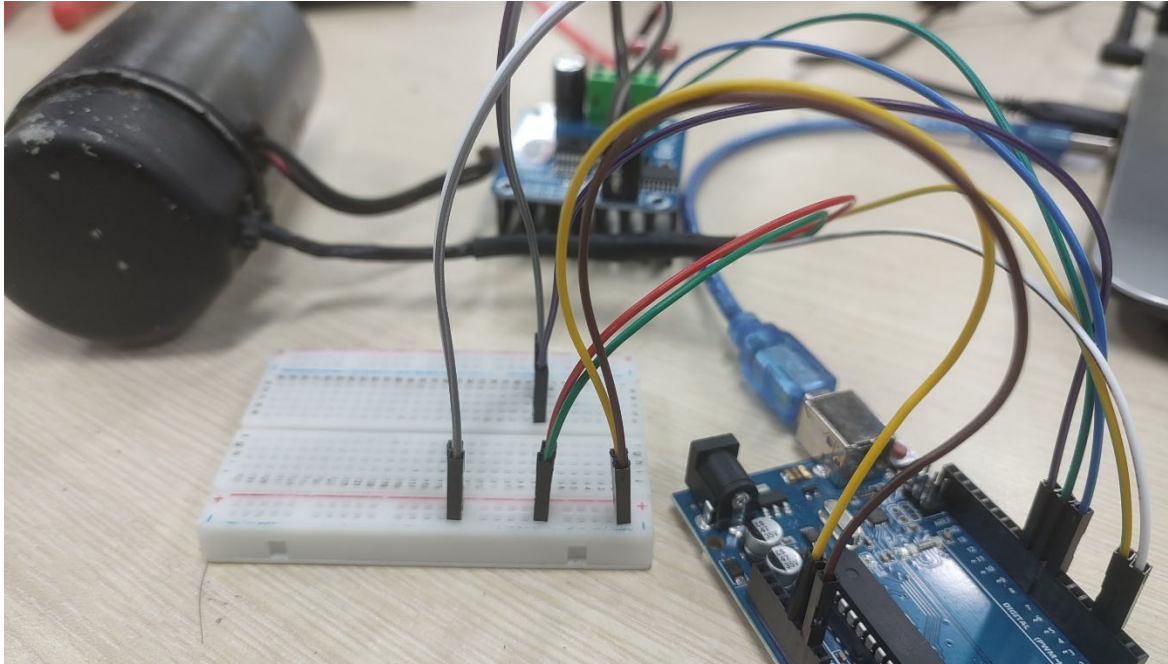


Figure 3.15: System connection testing with a mini breadboard (Photo taken by the author at Vietnamese German University)

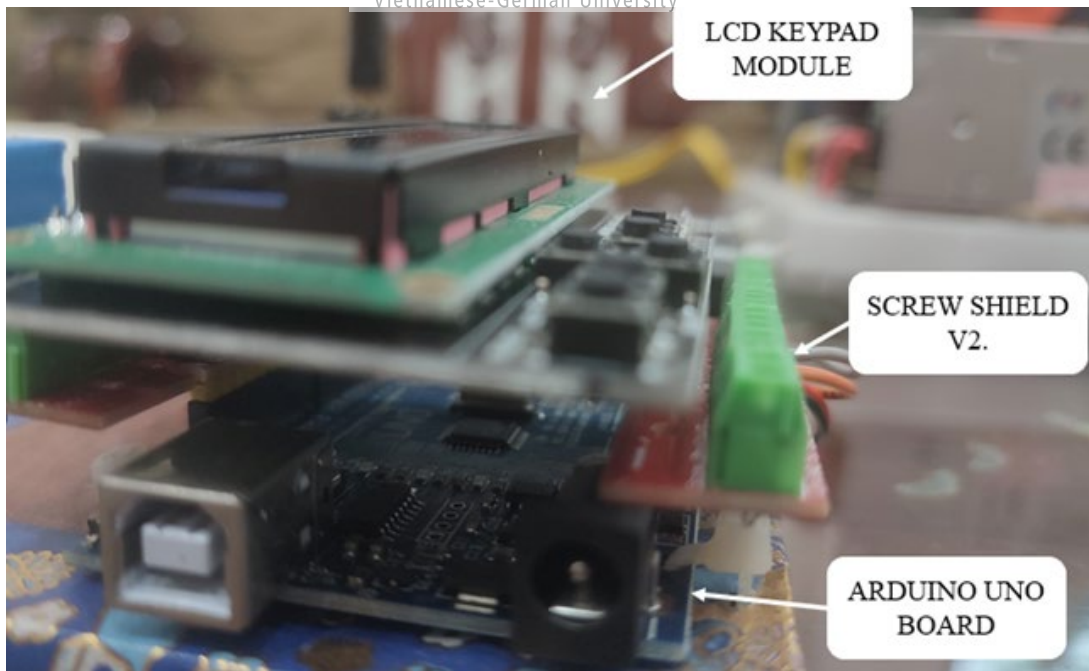


Figure 3.16: System wiring diagram for controlling the DC motor (Photo taken by the author at Vietnamese German University)

The screw shield extends the amount of pins of the Arduino by providing additional soldering pins. The term “shield” in Arduino convention refers to adding external hardware modules using the daughter card concept. This allows the Arduino board to act like a motherboard, providing electrical connection to the extra circuitry. This screw shield makes it convenient to stack the LCD Keypad on the Arduino board and is resulted in a less wiring work required.

3.5. SYSTEM LAYOUT DESIGN

Following criteria are to be taken into consideration when designing the layout for controlling the DC motor:

- Safely secure all connection by soldering where possible, thus utilization of breadboard is eliminated due to its instability.
- Make the model readily portable.
- Create the motor body fixture.
- Shock absorber for the motor when operating.
- Enhance neatness and aesthetics.

Most often, electronic project is suggested to be assembled on a PCB (Printed Circuit Board). However, PCB has its disadvantages as impeding circuit alterations, and the motor cannot be constructed on the board. Therefore, this project employs the plastic perforated sheet, which is inexpensive and easy to manipulate. It is made of polypropylene – a soft material that is flexible with a relatively low melting point, thus make it effortless to create holes onto for wiring and no need to drill through. This sheet permits rapid assembly and capability to alter the circuit as the wiring may be done underneath it. It also eliminates the chances of wires get tangled when transporting.

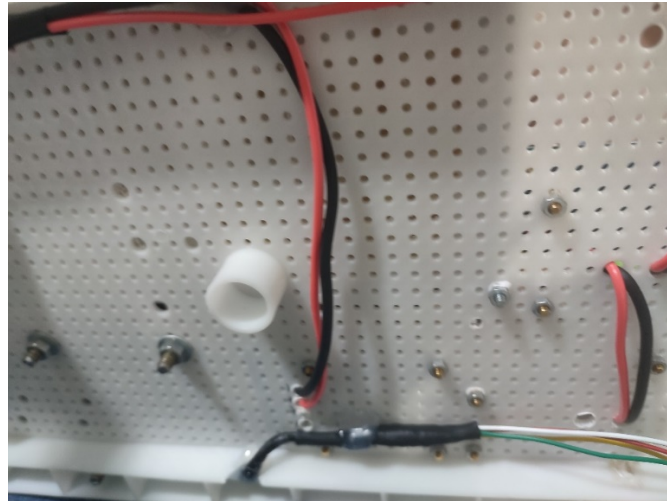


Figure 3.17: Wiring goes under the plastic perforated board (Photo taken by the author at Vietnamese German University)

Motor is fixed using nylon cable tie and five eye bolt fasteners, three on one side and two for other sides. Also, there are three layers of leather cloth placed beneath the motor for shock absorbing.



Figure 3.18: Fixture for motor (Photo taken by the author at Vietnamese German University)

These hexagonal-shaped threaded standoffs (*Figure 3.19*) are used to set distance between components as well as to elevate the component from the plastic perforated sheet, especially component like the BTS 7960 driver with its heatsink. The purpose is to increase the space for airflow around these hardware modules because when operating, they may become hot.

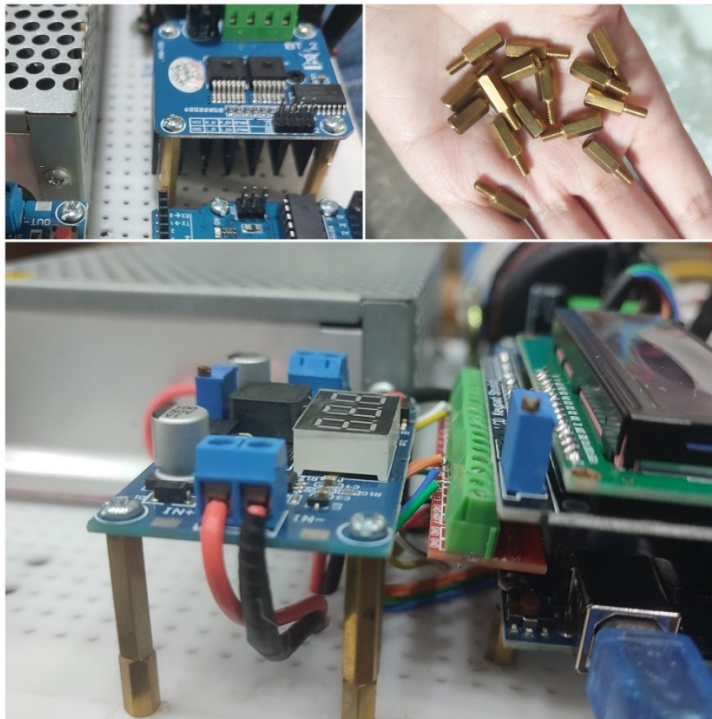


Figure 3.19: Brass hexagonal standoff used for several elements (Photo taken by the author at Vietnamese German University)



Vietnamese-German University

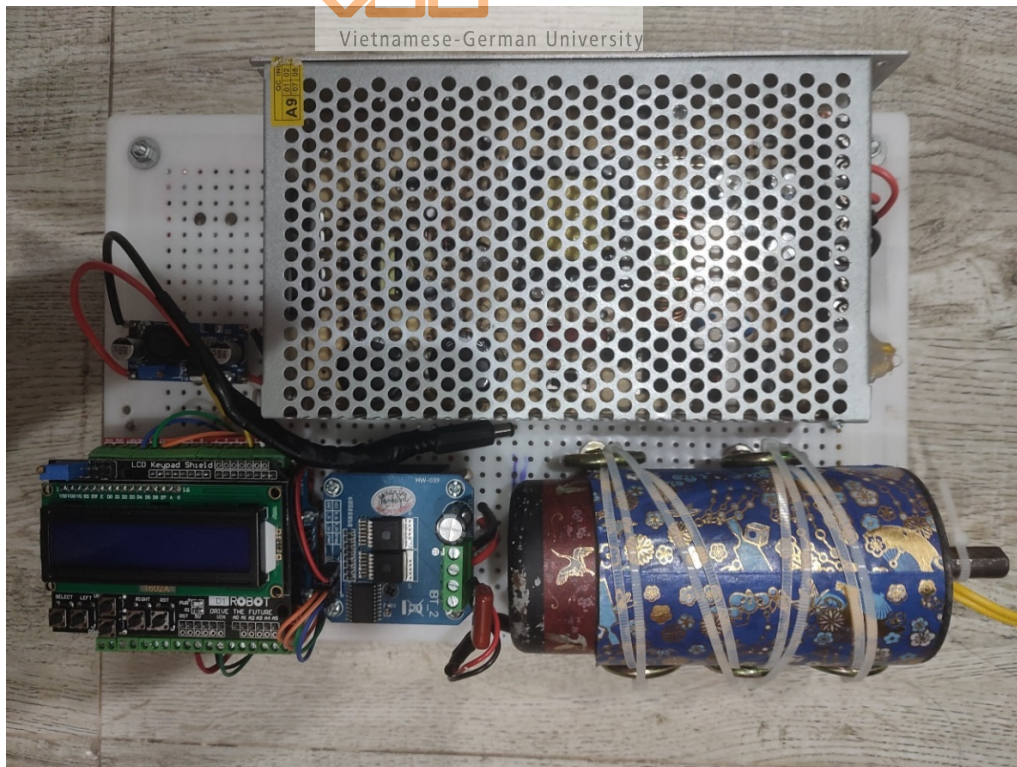


Figure 3.20: Finished system layout design for DC motor controlling (Photo taken by the author at Vietnamese German University)

3.6. PROGRAMMING

3.6.1. CODE STRUCTURE

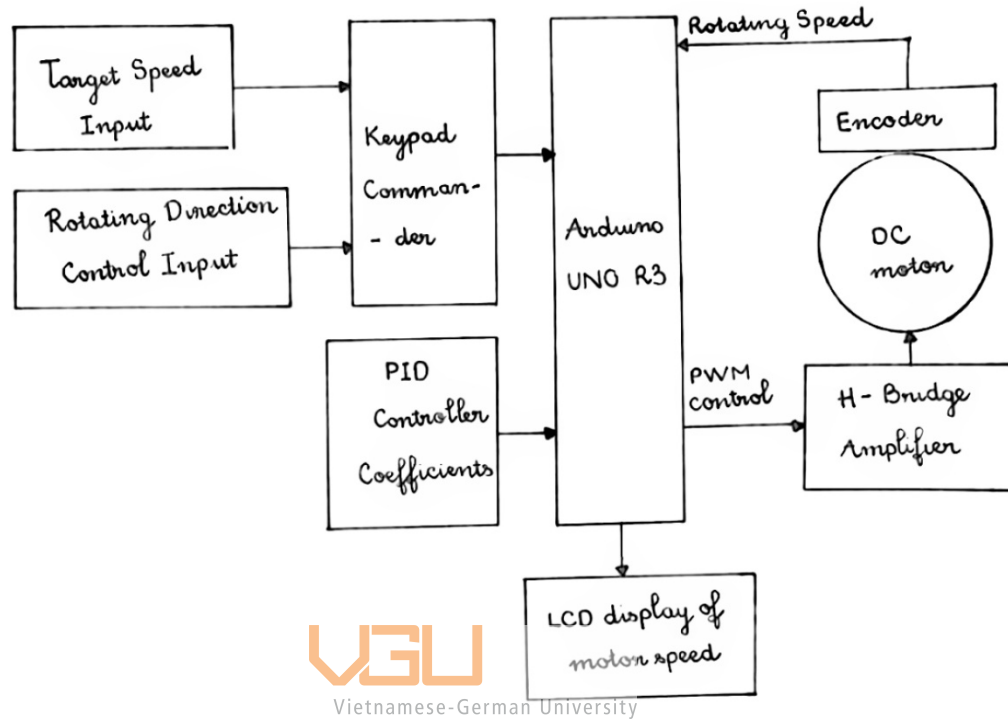


Figure 3.21: System configuration served for programming structure (diagram hand sketched by the author at Vietnamese German University)

The specific goal of this program is to create a close-loop (or feedback-control) system where the Arduino controller continuously calculates the PWM value to adjust the rotating speed based on the error between encoder feedback and target speed, then sends signal to the H-Bridge amplifier to control the motor as desired. Therefore, the programming is featured with:

- Pulse Width Modulation (PWM) Control
- Proportional, Integral, Derivative (PID) Algorithm
- Timer Interrupt
- External Interrupt (Pin Change Interrupt)

3.6.2. PULSE WIDTH MODULATION IN ARDUINO

Referring to Arduino UNO R3, Pulse Width Modulation (PWM) is a technique for converting from digital sources to analog output. PWM – a signal switched between on and off – is produced by using digital control. This on-off pattern can simulate voltages ranging from the full voltage of the Arduino board (5V) to off state (0V) by changing the duty cycle. With a frequency of 1kHz, there are 1000 PWM cycles per second which makes each cycle last for 1 millisecond. During each of these one-millisecond-cycle, the signal can be high part of time and low the rest of the cycle time.

The duty cycle is requested by an **analogWrite(PWM_pin, dutyCycle)** call, where **dutyCycle** is a value ranging from 0 to 255, with 0 mapping to an off state, 127 requesting a 50% duty cycle and 255 being always on; and **PWM_pin** is one of the PWM pins – pins denoted with a sign (~). These PWM pins on Arduino UNO R3 are pin 3, 5, 6, 9, 10 and 11. If values above 255 or below 0 is written to a PWM pin, erratic and unwanted behavior may occur [20].



PWM Frequency (Hz)	Switching Cycle Time	Arduino PWM Pins
30Hz	32 milliseconds	9 & 10; 11 & 3
61Hz	16 milliseconds	5 & 6
122Hz	8 milliseconds	9 & 10; 11 & 3
244Hz	4 milliseconds	5 & 6; 11 & 3
488Hz	2 milliseconds	5 & 6; 11 & 3
976Hz ~ 1kHz	1 milliseconds = 1000 μ s	5 & 6; 11 & 3
3906Hz ~ 4kHz	256 microseconds	9 & 10; 11 & 3
7812Hz ~ 8kHz	128 microseconds	5 & 6
31250Hz ~ 32kHz	32 microseconds	9 & 10; 11 & 3
62500Hz ~ 62kHz	16 microseconds	5 & 6

Table 3.5: Available frequencies for the Arduino PWM pins (taken from datasheet with the author's modification)

The most common and easiest way to adjust PWM is varying the duty cycle and holding frequency at default set value. One of the reason why this method is preferred for controlling the motor speed is that no heat is wasted during the switching process.

However, the frequency can be manually changed from 30Hz up to 62kHz by timer prescaler utilization (*Table 3.5*). The set frequency must be carefully chosen since too low the frequency will result in a rough performance and a “whine” noise from motor but too high the frequency will lead to unwanted side effects. As the frequency increases, the length of the switching cycle is subsequently decreased. For a short cycle, the output does not have enough time to completely reach high or low state, instead, it is stuck in somewhere between on and off state, which will generate excess heat in the system. This is called a shoot-through state [21].

The program listing of a function generated for an open-loop control system that implementing PWM technique is shown in Appendix 4.



Vietnamese-German University

3.6.3. TIMER/COUNTER

The **analogWrite()** command is the simplest method when using PWM technique but it is limited by a set frequencies for each PWM pins. According to the ATmega328p’s datasheet, the Arduino is set up with two fixed PWM frequencies, the frequency of the PWM signal on most pins is approximately 490Hz and 980Hz on pins 5 and 6. By manipulating the chip’s timer register directly, more control may be gained than the **analogWrite()** command provides.

Basically, timer is a binary counter, works by incrementing a counter variable known as a counter register. The counter can be configured to count clock pulses (internal/external) until it reaches the maximum value, at which point the counter overflows, generates the interrupt if enable then resets back to zero and started counting from the beginning again.

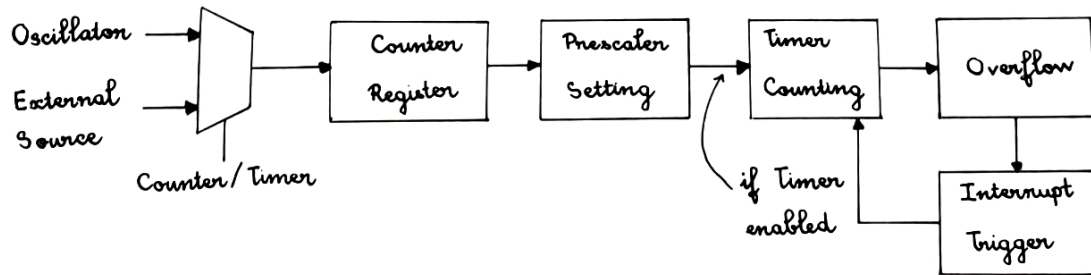


Figure 3.22: Timer/Counter block diagram

The interrupt can be triggered by defining an Interrupt Service Routine (ISR).

The timer counts thanks to the clock source that each timer's increment takes one time pulse as its unit. If a 1MHz clock signal is provided to a timer, timer resolution is calculated as follow:

$$T = \frac{1}{f} = \frac{1}{1\text{MHz}} = \frac{1}{1 \times 10^6 \text{Hz}} = 1 \times 10^{-6} \text{s}$$

Where T is increment unit (time it's take for one "tick") and f is clock frequency.

The ATmega328p chip has three PWM timers, controlling 6 PWM outputs, named TIMER 0, TIMER 1, TIMER 2 [14].

TIMER 0

Timer 0 is an 8-bit timer, makes its counter register capable of counting up to 255. Timer 0 is also the clock source used by basic Arduino functions such as **delay()** and **millis()**. It is suggested to not use advanced PWM control technique on this Timer because it may interfere other irrelevant functions and resulted in unexpected consequences.

TIMER 1

Timer 1 is a 16-bit timer, thus its maximum counter register value is 65535.

TIMER 2

Timer 2 is an 8-bit timer which is fairly similar to Timer 0 but has a different set of prescale values from the other timers.

Each timer has two outputs, when the timer reaches the compare register value, the corresponding output is toggled [22].

Prescaler is an important key element involved when considering using Timer system.

Figure 3.23 illustrates why prescaler is often needed.

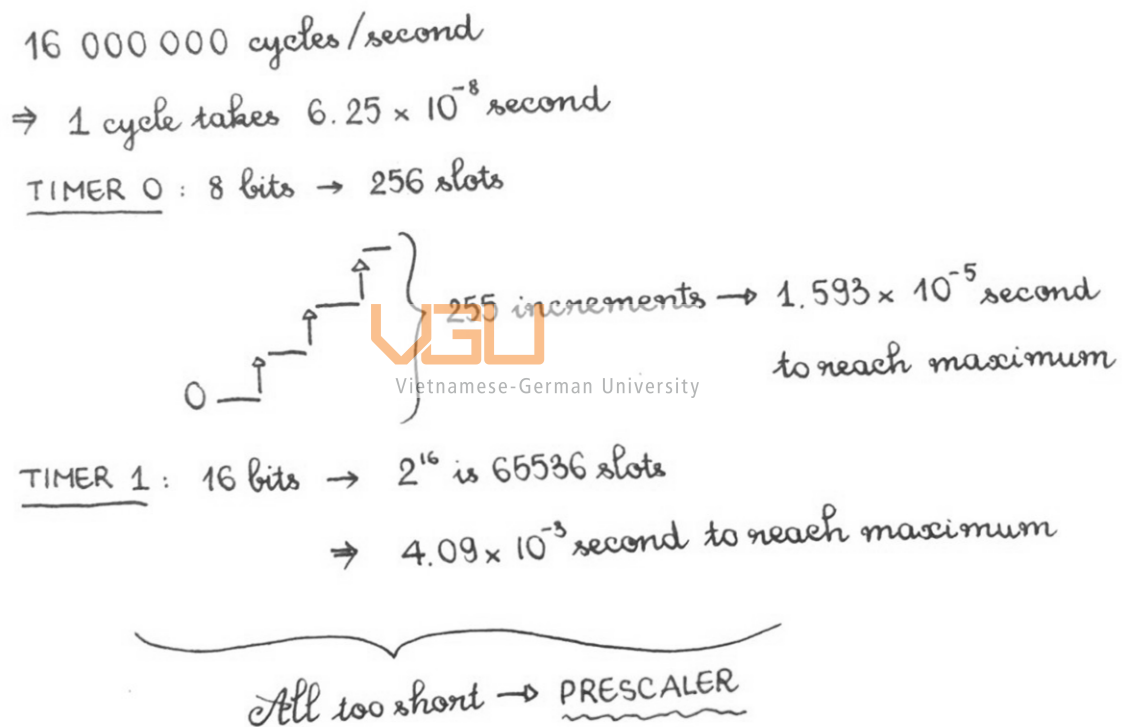


Figure 3.23: Example to explain the prescaler's importance in Timer/Counter
(Diagram hand sketched by the author at Vietnamese German University)

Therefore, with presence of prescaler, instead of incrementing every cycle, now it increments every 8 cycles (or 64 cycles, 256 cycles, 1024 cycles, dependent on prescaler mode registered).

Prescaler	Fast PWM Frequency	Fast PWM Period (in microsecond)
1	62.5kHz	16
8	7.8kHz	128
32	1.9kHz	512
64	976Hz	1024
128	488Hz	2048
256	244Hz	4096
1024	61Hz	16384

Table 3.6: Prescaler values and PWM frequencies for an 8-bit counter (taken from the datasheet with the author's modification)

This system focuses on Timer 2, which allows generating Output Compare Match and Overflow interrupts. For Output Compare Match enablement, OCIE2B and OCIEA2 bits are used. For Overflow interrupt enablement, TOIE2 bit is used [14].

Vietnamese-German University

For timer configuring, there are built-in registers available on the AVR chip. Two of these registers store setup values. They are TCCRxA and TCCRxB, where x is the timer number, for example, TCCR2A and TCCR2B. TCCR stands for Timer/Counter Control Register. All the bits in TCCR2A register are set to 0 [14].

Bit	7	6	5	4	3	2	1	0
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Table 3.7: TCCR2A Timer 2 Control Register A (taken from the datasheet with the author's modification).

For TCCR2B, the first three bits are used to set prescaler value, those are CS20, CS21 and CS22.

Bit	7	6	5	4	3	2	1	0
(0xB1)	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Table 3.8: TCCR2B Timer 2 Control Register B (taken from the datasheet with the author's modification)

Table 3.9 shows the bits of CS22, CS21 and CS20 registration for a specific prescaler value [14]:

CS22	CS21	CS20	Description
0	0	0	Timer Stop
0	0	1	Prescaler 1
0	1	0	Prescaler 8
0	1	1	Prescaler 32
1	0	0	Prescaler 64
1	0	1	Prescaler 128
1	1	0	Prescaler 256
1	1	1	Prescaler 1024

Table 3.9: TCCR2B Timer 2 Control Register B (taken from the datasheet with the author's modification).

There are various ways to select the prescaler. Below (Figure 3.24) is an example for setting a 256 prescaler with Timer 1.

CS12	CS11	CS10
1	0	0

First Step: Set CS12 to 1. $TCCR1B |= (1 \ll CS12)$

$\left\{ \begin{array}{l} TCCR1B \text{ is } xxxxxxxx \\ 1 \ll CS12 \rightarrow 00000001 \ll CS12 \rightarrow 00000100 \\ \Rightarrow TCCR1B \text{ OR } (1 \ll CS12) = xxxxxx1xx \end{array} \right.$

Second Step: Set CS11 to 0. $TCCR1B \&= \sim(1 \ll CS11)$

$1 \ll CS11 = 00000010$

Reverse (\sim) yields: 11111101

And ($\&$) yields:

$xxxxx1xx \& 11111101 = xxxxxx10x$

Third Step: Similarly, set CS10 to 0. $TCCR1B \&= \sim(1 \ll CS10)$

Similarly, result is: xxxxx100

Figure 3.24: Register for a 256 prescaler of Timer 1 (Data compiled by the author)

Values to be loaded into the reload timer is set based on the designing goal. For Timer 2, the amount of pulses generated by the optical rotary encoder is designed to be counted every *1 milliseconds*. Assuming prescaler of 1024 is used.

Following steps are taken:

Calculate the period of the timer clock using:

$$T_{clock} = \frac{1}{f_{timer}}$$

Where f_{timer} is the frequency of the clock used for the timer, with a prescaler of 1024,

f_{timer} is:

$$f_{timer} = \frac{16MHz}{1024} = \frac{16 \times 10^6 Hz}{1024} = \mathbf{15625Hz}$$

Thus,

$$T_{clock} = \frac{1}{f_{timer}} = \frac{1}{15625Hz} = 6.4 \times 10^{-5}s$$

Determine number of needed clocks by dividing the desired time interval by T_{clock} .

$$\text{Number of needed clocks} = \frac{1ms}{T_{clock}} = \frac{0.001s}{6.4 \times 10^{-5}s} = 15.625 \approx \mathbf{16} \text{ (whole number)}$$

Perform $256 - n$, where n is the decimal value achieved in Step 2.

$$256 - 16 = \mathbf{240}$$

Convert the result of Step 3 to hex, this hex value is the reload timer value that will be used in the program.

240 in hex is: **0xF0**

Thus, OCR2A bit will be set to 0xF0.



Vietnamese-German University

3.6.4. EXTERNAL INTERRUPT / PIN CHANGE INTERRUPT

The ATmega328P microcontroller has only two pins that enable external interrupt, which are pins 2 and 3. Therefore, the encoder pins are defined by these pins. The amount of pulses recorded by the encoder is read using a pin change interrupt. For every time channel A gets a rising edge, the program trigger a function to increment the position, that function is attached in the form of an interrupt. To determine the direction, the position-recording variable will be increment positively by 1 if output B is HIGH (by the time the interrupt being triggered), otherwise, that variable will be subtracted by 1 if output B is LOW.

3.6.5. PID TUNING

The gains of P, I, and D need to be adjusted differently to each system because there is no conventional set of specific values for this tuning. Manual tuning of the gains is the

simplest method with considerably effective results. This method is utilized in this project, following a general procedure.

- First set K_i and K_d values to zero.
- Increase the K_p until the motor output oscillates, then set K_p to approximately half of the oscillating value.
- Increase K_i until the motor reaches target speed in desired time. If motor get instable, decrease K_i for a small amount because too much K_i will cause instability.
- Finally, increase K_d until the motor speed is acceptably quick to reach its set target. However, too much K_d will cause excessive response and overshoot, thus it needs to be adjusted slowly.

The P, I, D gains achieved after the manual tuning are:

$$K_p = 0.1;$$

 $K_i = 0.08;$

Vietnamese German University

 $K_d = 0.0005$

3.6.6. PROGRAM FLOWCHART

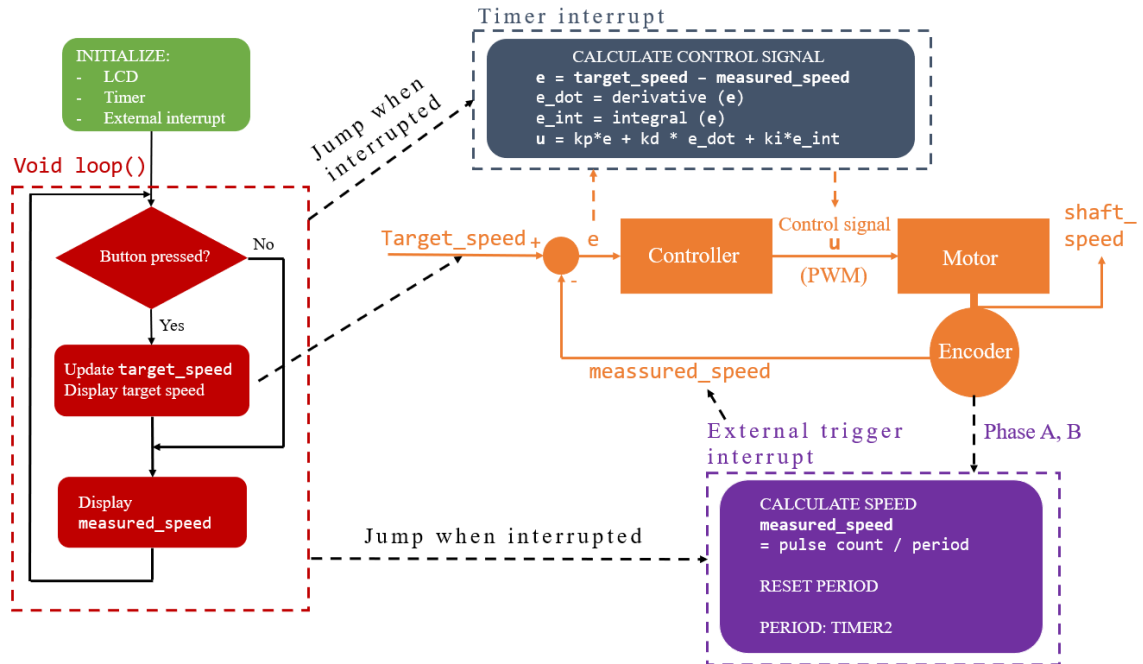


Figure 3.24: DC motor control program flowchart

CHAPTER 4 - IMPLEMENTING NRF24L01 2.4GHz RF MODULE FOR WIRELESS DC MOTOR CONTROL

4.1. INTRODUCTION nRF24L01

The concept of wireless transmission in technology refers to transmission of signals between two or more devices in space without any physical link established between them. Wireless signals are spread in waves form and are characterized by their amplitude and frequency. Signals are sent either by using radio waves or by beams of infrared light.

Radio waves have many advantages over beam of infrared light. While infrared light cannot penetrate through solid objects under any circumstances, radio waves can travel through obstacles at low frequency. In nature, radio waves can travel in every direction and infrared waves are unidirectional. However, radio waves still have disadvantages as: incapability of transmitting a lot of data simultaneously due to low frequency (3kHz to 1GHz) and it offers a poor security. But in general, they do meet the needs for an Arduino project.

Principle of transmitting radio signals: At the beginning, the data is converted, or modulated, into its equivalent electrical signal. This signal, called a carrier wave, contains the complete information that needs to be transmitted. The transmitting antenna receives this signal and radiates it into open space. (At this point, any receiver whose frequency matched within this range will be able to access the data. This explains why radio waves transmitting is poorly secured.) On the receiving end, the data carried by the wave is demodulated, or extracted into original information.

This system uses two nRF24L01 modules to communicate between the transmitter – LCD and Keypad and the receiver – DC motor in order to wirelessly control DC motor speed and receive motor information simultaneously.

nRF24L01 is a wireless transceiver module manufactured by Nordic Semiconductors. They are available in several types, varies in configuration of antenna– built-in type or with external antenna. The version used in this project is the variation with SMA (Sub-Miniature version A) connector that allows an attachment of a duck antenna for better transmission range – in an ideal condition, transmission range may be up to 1000 meters and averaged in 700 meters, depending on location, atmosphere and presence of obstacles.

In addition to the detachable antenna, it is packed with PA (Power Amplifier) and LNA (Low-Noise Amplifier). The PA enhances the signal power transmitted while the LNA extends weak signals from the transmitter to a better standard.



Figure 4.1: nRF24L01 module + PA + LNA + detachable antenna (Photo taken by the author at Vietnamese German University)

It operates on 2.4 GHz band which is allowed for unlicensed use in almost all regions [23].

Features:

Frequency range: 2.4 GHz

Operating voltage: 3.3V DC

Logic input: 5V tolerant

Maximum operating current: 45mA

Communication interface: SPI (Serial Peripheral Interface). This is a standard bus used to communicate between Arduino microcontroller and many other peripheral ICs (Integrated Circuit). The SPI uses the concept of Master and Slave where, most often, the Arduino is the Master and the nRF24L01 is the Slave. Communication between Master and Slave is bidirectional that means they can both transmit and receive at the same time.

Data rates: 250 kbps/ 1 Mbps / 2 Mbps

Communication range: 1000 meters (ideal condition)

4.2. SYSTEM LAYOUT

The project includes two Arduino microcontrollers and two nRF24L01 modules to build up a system of one transceiver (controller) and one receiver (DC motor).

Illustrated below is a simple block diagram designed for this system (*Figure 4.2*).

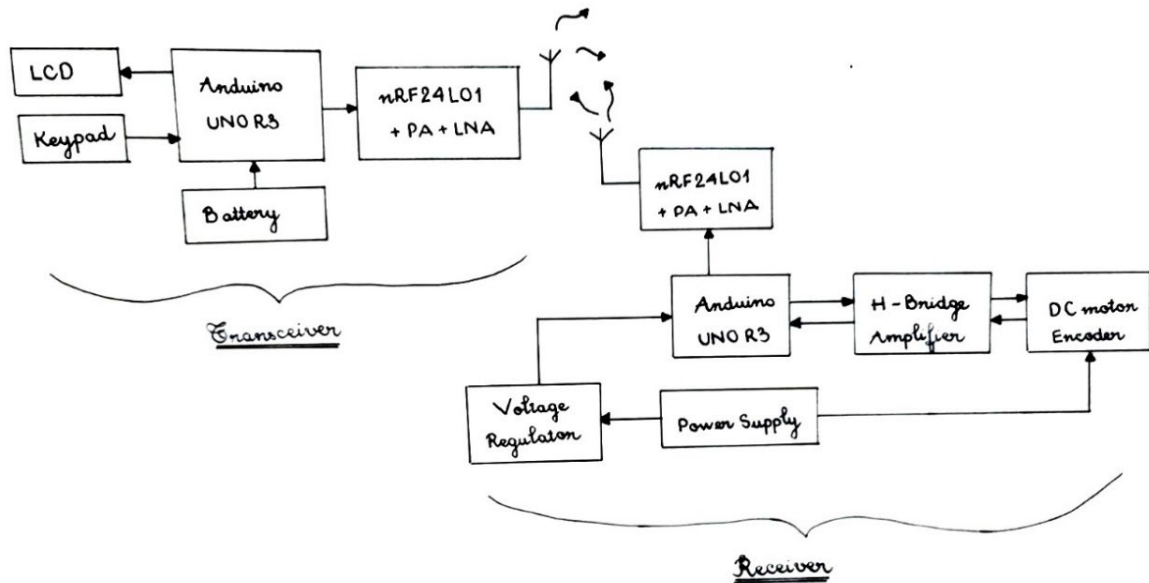


Figure 4.2: System Block Diagram

4.3. EXPERIMENTAL SETUP

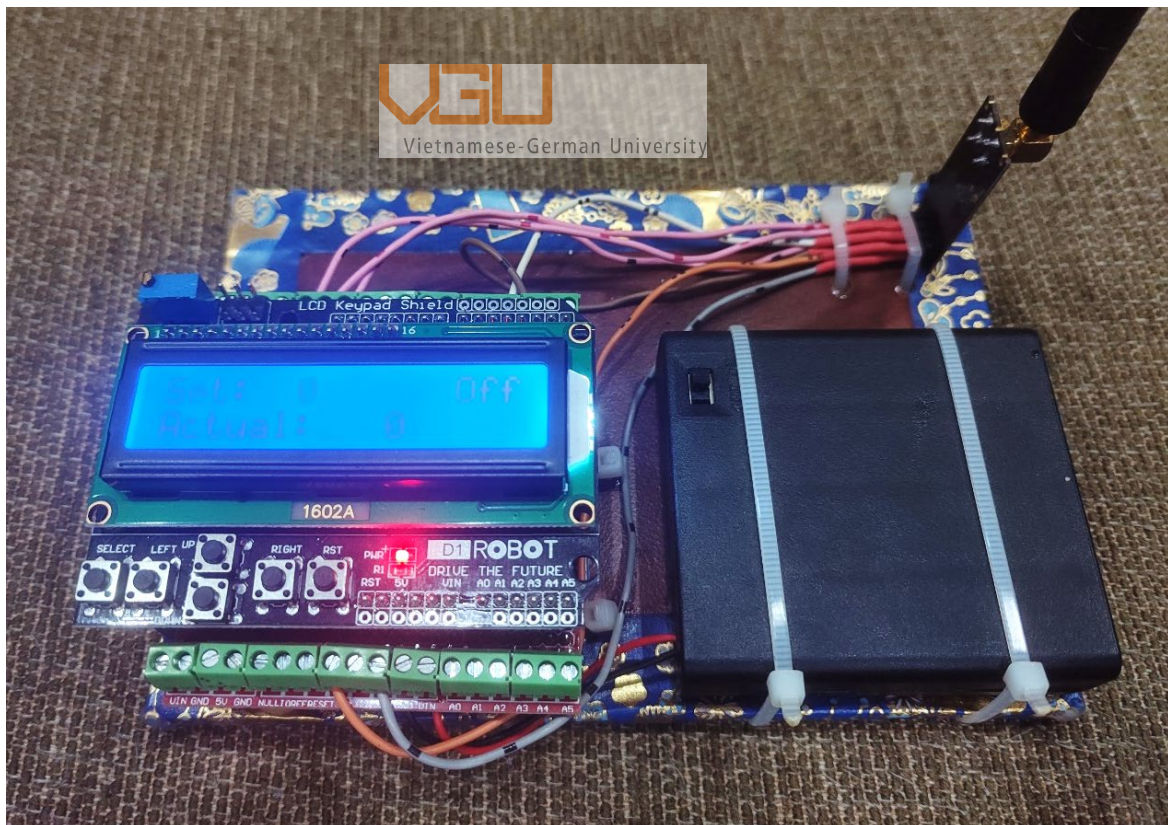


Figure 4.3: System's transceiver (Photo taken by the author at Vietnamese German University)

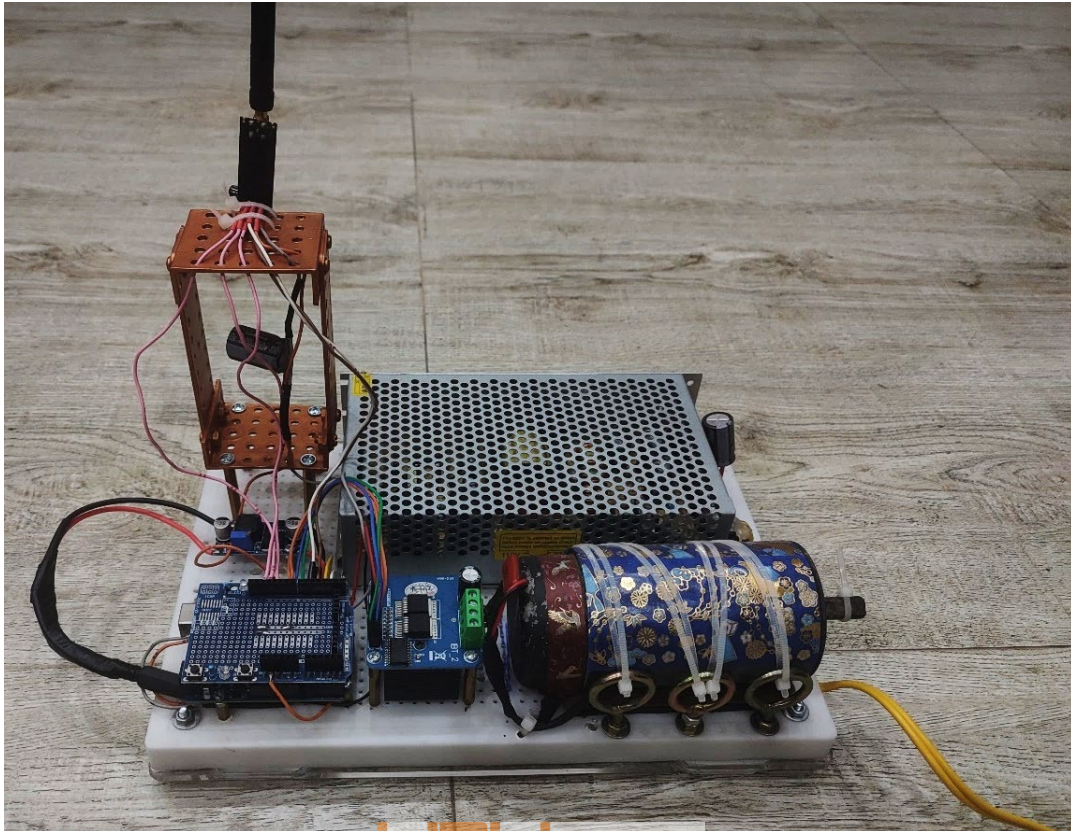


Figure 4.4: System's receiver (Photo taken by the author at Vietnamese German University)

Vietnamese-German University

4.4. PROGRAMMING

This sections focuses on programming the nRF2401 modules for the system's transceiver and receiver.

TRANSCEIVER - COMMANDER

```
//Pin Configuration
//SCK 13
//MISO 12
//MOSI 11
//CE 2
//SCN 3
```

RECEIVER - DC MOTOR

```
//Pin Configuration
//SCK 13
//MISO 12
//MOSI 11
//CE 4
//SCN 5
```

Pin functions:

SCK – Serial clock: provides clock pulses so that SPI communication may work properly, connects to pin 13 on Arduino by default.

MISO – Master in Slave Out: allows nRF module to send data from the microcontroller, connects to pin 12 on Arduino by default.

MOSI – Master Out Slave In: allows nRF module to receive data from the microcontroller, connects to pin 11 on Arduino by default.

CE – Chip Enable: enables SPI communication.

SCN – Ship Select Not: set to LOW to disable SPI communication, therefore, to operate nRF, this pin should be kept HIGH.

IRQ – Interrupt: in used when interrupt is required.

nRF24L01 employs Master and Slave control model. Only one slave can communicate at one time, thus, the nRF24L01 module has an interrupt pin (IRQ) to notify the master when one specific slave needs to communicate. However, the nRF library used in this project does not support this function, hence this pin is not exploited.

TRANSCIEVER – COMMANDER

```
#include <SPI.h>
```

```
#include <Nrf24l01.h>
```

```
#include <RF24.h>
```

RECEIVER – DC MOTOR

```
#include <SPI.h>
```

```
#include <Nrf24l01.h>
```

```
#include <RF24.h>
```

Library SPI is used for communicating interface with the modem. Library Nrf24l01 serves for this particular module. Library RF24 permits module controlling.

RECEIVER – DC MOTOR

TRANSCIEVER - COMMANDER

```
//Define pin CE, CSN           //Define pin CE, CSN
RF24 radio(2,3);               RF24 radio(4,5);
//Define address                //Define address
const byte addresses[] [6] =  const byte addresses[] [6] =
{"00001", "00002"};           {"00001", "00002"};
```

There are two addresses for both transceiver and receiver because this system is two-way communicating.

TRANSCIEVER - COMMANDER

```
radio.begin();
radio.openWritingPipe(address
es[1]); //00002
radio.openReadingPipe(1,adres
ses[0]); //00001
radio.setPALevel(RF24_PA_MIN)
;
```

RECEIVER - DC MOTOR

```
radio.begin();
radio.openWritingPipe(address
ses[0]); //00001
radio.openReadingPipe(1,adre
sses[1]); //00002
radio.setPALevel(RF24_PA_MIN
);
```

radio.setPALevel() is to define the power level, ranging between values of PA as MIN, LOW, HIGH, MAX.

CHAPTER 5 – EXPERIMENT RESULTS AND DISCUSSION

The experiment is brought out to confirm the model’s adaptability to several targets as originally set. To satisfy the objective of this work, we have conducted altogether two main experiments, including the basic functionality tests and the transmission range test. In the basic functionality tests, the implemented prototype has been tested for the basic functions, including the basic movement under the wired and wireless conditions, the basic speed control, the control for the reverse and forward directions and the functionality of the PID controller. *Table 5.1* summarizes the results of the basic functionality tests.

	OBJECTIVES	ACHIEVED	DISCUSSION AND COMMENTS
1	Wirelessly control DC motor speed and direction	Yes	Communication range varies from 100 meters to 400 meters, depending on the location of experiment.
2	Target speed and actual speed are displayed on LCD screen (pulses/second)	Yes	The actual speed display gets updated simultaneously as the encoder feedbacks, therefore the characters gets blurred. Power is supplied through a battery (for portable purpose) and not sufficient enough that the screen contrast is not as

			visible as when connecting using computer supply power.
3	Motor direction is changed when being controlled	Yes	
4	Up Button requests increasing speed by small step	Yes	Controller gets unstable when small step is larger than 200 (delay, motor's speed does not meet set speed).
5	Down Button requests decreasing speed by small step	Yes	Controller gets unstable when small step is larger than 200 (delay, motor's speed does not meet set speed).
6	Up Button requests increasing speed by large step	Yes	Controller gets unstable when large step is larger than 500 (delay, motor's speed does not meet set speed).
7	Down Button requests decreasing speed by large step	Yes	Controller gets unstable when large step is larger than 500 (delay, motor's speed does not meet set speed).
8	Motor gets ON/OFF properly when commanded	Yes	

Table 5.1: Basic Functionality Tests

In addition to the basic functionality check, a highly comprehensive experiment is conducted to explore the performance related to the transmission efficiency. One of the tests was the range test. In this test, the maximum distance between the transmitting end and the receiving end was measured while some of the parameters were changed. The

purpose of this test was to explore not only the communication range but also the enhancement of the radio. The results of this test are summarized in *Table 5.2*.

	METHOD	RESULT	COMMENT
1	Soldering decoupling capacitors of 1000 μ F to Nrf24L01 modules.	Worked	These RF modules are very sensitive to insufficient power, the capacitor evens out power spikes, thus provides a more stable voltage while operating . Also, the 3.3V pin on Arduino (which is used to power the RF module) is not really 3.3V, it is tested to be varied from 2.9V to 3.3V. Therefore, the capacitor functions as an external power source when needed. (<i>Figures 5.1</i>)
2	Set power level by defining <code>radio.setPALevel()</code> with different values: LOW, HIGH, MAX.	Partially working	The communication range is improved, however, motor's performance does not adapt controlling commands sufficiently. Because the longer the range, the slower the data transmission speed. The model therefore accepts PA level as MIN.

Table 5.2: Transmission Range Test

The following figures show setup done for testing first method.

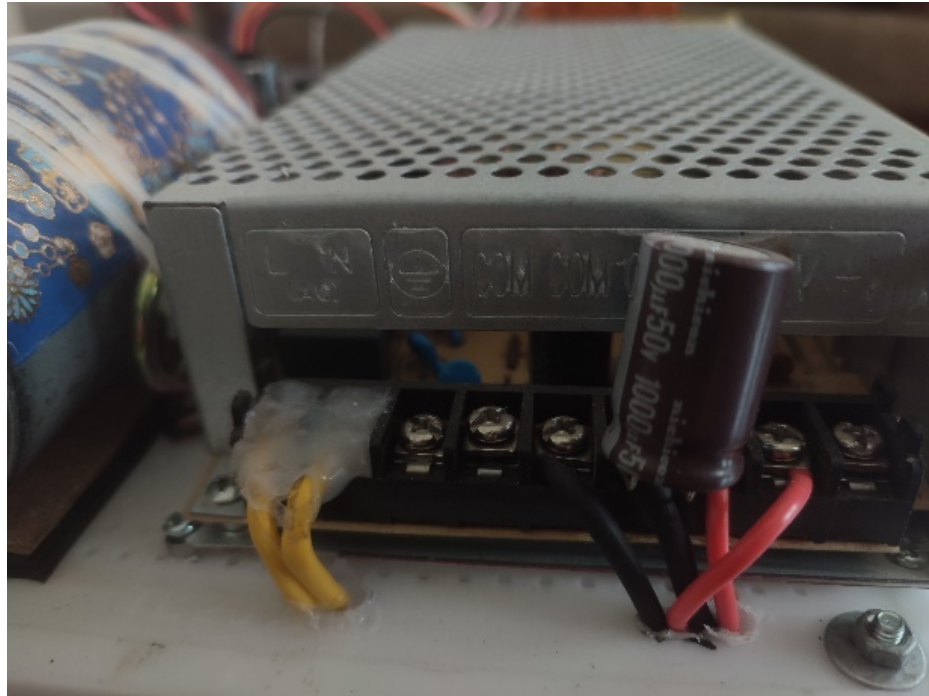


Figure 5.1: Capacitor was added on (a): supply power source

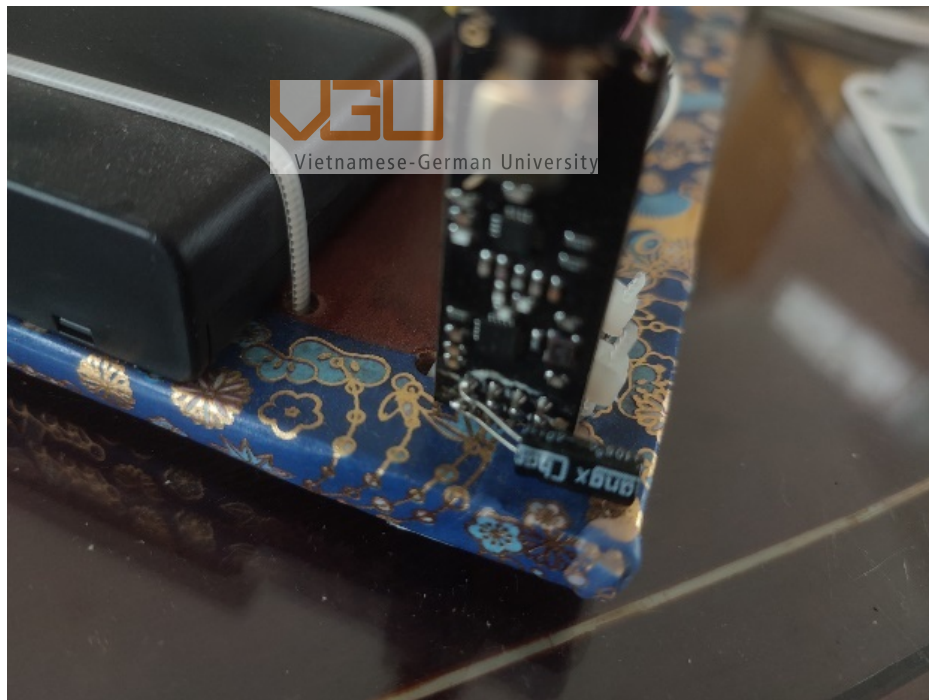


Figure 5.1: Capacitor was added on (b): nRF24L01 modules

CHAPTER 6 - FEASIBILITY OF EXTENDING THE TRANSMISSION RANGE

6.1. RELATED FORMULAS

The maximum range of remote control that was achieved for this aforementioned experimental setup was 400 meters. Therefore, a process for further extending this range was carried out. This range can be explained by the well known Friis' formula. According to Friis's theorem, the range of communication can be derived as follows:

The directivity is:

$$D = \frac{U}{U_i} \tag{1}$$



Multiplying the top and bottom of (1) with the radius r , we obtain:

$$D = \frac{r^2 U}{r^2 U_i} \tag{2}$$

The power received by antenna 2 is given by:

$$P_R = S_T A_{eR} = \frac{P_T D_T}{4\pi R^2} A_{eR} \tag{3}$$

Since the effective aperture for any antenna can also be expressed as:

$$A = \frac{\lambda^2}{4\pi} D$$

Equation (3) can be rewritten as:

$$P_R = \frac{P_T D_T}{4\pi R^2} \frac{\lambda^2}{4\pi} D_R \quad (4)$$

Further rearranging this equation yields:

$$P_R = \frac{P_T D_T D_R}{(4\pi R)^2} \lambda^2 \quad (5)$$

If all the antennas are 100% efficient, then the power receivable at the receiving end is:

$$P_R = P_T \frac{G_T G_R}{(4\pi R)^2} \lambda^2 \quad (6)$$

The final formula of (6) is known as Friis' formula. This formula clearly suggests that the increase in the transmitting power P_T will lead to an increase in the power receivable at the receiving end. The power at the receiving end as predicted by the Friis' formula is consistent with the experimental outcome as summarized in *Table 5.2*.

Vietnamese-German University

In addition to the power at the transmitting end, it is also possible to increase the antenna gain at the receiving end and/or the antenna gain at the transmitting end to further increase the power receivable at the receiving end. In so doing, we can increase the overall transmission range of the implemented prototype.

At the moment, the antennas we had in this project are monopole antennas with very limited gain. The reason is because the monopole antenna is omni-directional. The electromagnetic waves being sent out by a monopole antenna is two dimensionally spread to all directions, not just the direction of the receiving end. As such, much power has been lost to other unwanted direction during the transmission process. To boost the gain, we can change the antenna topology of the transmitting antenna. The simplest and most direct topology is the Yagi-Uda antenna topology. Figure 6.1 shows the proposed antenna topology.

It can be seen from *Figure 6.1* that enhancement of the gain at the transmitting end can be easily realized by adding a director and a reflector, both of them can be of the same material as the monopole at the middle of *Figure 6.1*. The reflector reflects the back lobe so that the power from the transmitting end can be radiated towards the right side only, assuming the right side is where the receiving end is. The director serves to focus the power and sharpen the beam towards the receiving end.

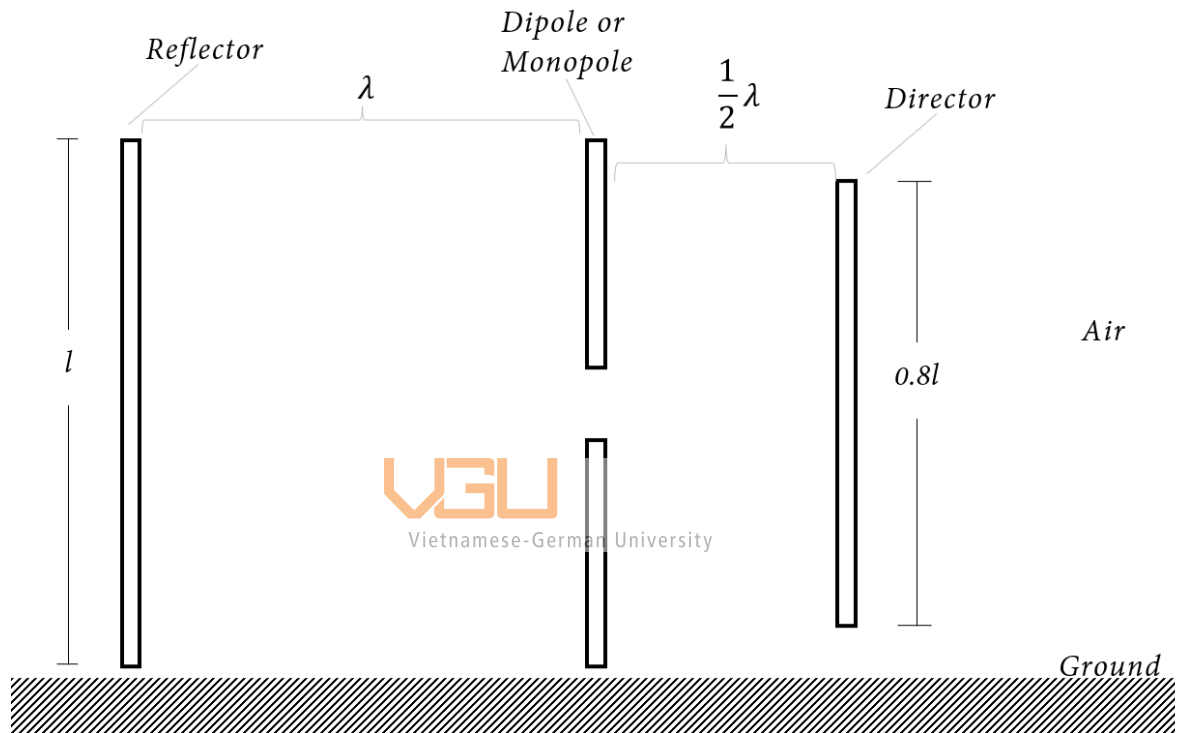


Figure 6.1. The proposed antenna topology for the transmitting end of our experimental setup.

CHAPTER 7 – CONCLUSION

In this work, we have realized the hardware for remote control of a DC motor as well as the software that controls the hardware using the built-in PWM library. The hardware comprises microcontroller board - Arduino UNO R3, a DC motor (Hitachi), H-Bridge driver, a monopole antenna for communicating at 2.4 GHz, and an nRF24L01 module. The outcome of this experiment has proven beyond any doubt that, using the present monopole antenna topology, the basic functionalities of the DC motor were able to be remotely controlled at a maximum distance of 400 meters. In accordance with the theoretical prediction made by Friis' formula, we have also found that the transmission range was changeable by changing the power at the transmitting end. The results of our analysis suggest that a further increase in the transmission range is possible if the antenna gain at the transmitting end and/or the receiving end is increased.



REFERENCE LIST

- [1] Glover, J.D. and Sarma, M.S. and Overbye, T. (2011). *Power System Analysis and Design*. [Online]. Available: <https://books.google.com.vn/books?id=uQcJAAAAQBAJ>
- [2] Neacșu, D.O. (2020). *Automotive Power Systems*. [Online]. Available: https://books.google.com.vn/books?id=nw_2DwAAQBAJ
- [3] Xia, C. (2012). *Permanent Magnet Brushless DC Motor Drives and Controls*. [Online]. Available: <https://books.google.com.vn/books?id=FkRYP7DWO9cC>
- [4] Sarb, D; Bogdan, R. (2016). *Wireless Motor Control in Automotive Industry*. [Online]. doi: 10.1109/TELFOR.2016.7818790
- [5] Vinothkanna, R. “Design and analysis of motor control system for wireless automation”, *Journal of Electronics*, vol.02, no.03, pp. 162-167. [Online]. doi: <https://doi.org/10.36548/jei.2020.3.002>
- [6] Theraja, BL. (2005). *A Textbook of Electrical Technology - Volume II*. [Online]. Available: https://books.google.com.vn/books?id=_RyjAsxFbdEC
- [7] El-Sharkawi, M. (2019). *Fundamentals of Electric Drives*. [Online]. Available: <https://books.google.com.vn/books?id=Qd3JswEACAAJ>



Vietnamese-German University

- [8] Drury, B.; Drury, W.H. (2001). *Control Techniques Drives and Controls Handbook*. [Online]. Available:
<https://books.google.com.vn/books?id=vDQHzeEmSfUC>
- [9] Peddapelli, S.K. (2016). *Pulse Width Modulation: Analysis and Performance in Multilevel Inverters*. [Online]. Available:
<https://books.google.com.vn/books?id=4cvJDQAAQBAJ>
- [10] Hnatek, E.R. (1989). *Design of Solid-State Power Supplies*. [Online]. Available: <https://books.google.com.vn/books?id=3jgfAQAAIAAJ>
- [11] Sclater, N. (2011). *Mechanisms and Mechanical Devices Sourcebook*. 5th ed. [Online]. Available: <https://books.google.com.vn/books?id=waSobYG567MC>
- [12] Morris, A.S. and Langari, R. (2020). *Measurement and Instrumentation: Theory and Application*. [Online]. Available:
<https://books.google.com.vn/books?id=z7vbDwAAQBAJ>
- [13] Tong, W. (2022). *Mechanical Design and Manufacturing of Electric Motors*. [Online]. Available: <https://books.google.com.vn/books?id=-7poEAAAQBAJ>
- [14] *Datasheet for Atmega328P and Variants*. Accessed: Mar. 09, 2023.
[Online]. Available: www.microchip.com/wwwproducts/en/ATmega328p

- [15] Barrett, S.F. (2013). *Arduino Microcontroller Processing for Everyone!*. 3rd ed. [Online] Available:
<https://books.google.com.vn/books?id=TbldAQAAQBAJ>
- [16] Kalya, S. and Kulkarni, M. and Shivaprakasha, K.S. (2021). *Advances in VLSI, Signal Processing, Power Electronics, IoT, Communication and Embedded Systems: Select Proceedings of VSPICE 2020*. [Online]. Available: <https://books.google.com.vn/books?id=7JQoEAAAQBAJ>



APPENDIX 1 – MAIN SOFTWARE FOR CLOSE-LOOP CONTROL

```
1  #define ENCA 2
2  #define ENCB 3
3  #define PWM 11
4  #define IN2 12
5  #define IN1 13
6  //Pins from D4 to D10 are saved for LCD Keypad pins
7
8  #include <LCDKeypadShield.h>
9  LCDKeypadShield shield;
10
11 unsigned int timer_reload = 0xF0;
12 //Use Timer 2, prescaler 1024, 1 millisecond per cycle
13 int pos = 0;
14 int pos_reset = 0;
15 float speed_prev = 0;
16 float speed_curr = 0;
17 int speed_target = 0;
18 float eprev = 0;
19 float eintegral = 0;
20
21 //PID CONSTANT
22 float kp = 0.1;
23 float kd = 0.0005;
24 float ki = 0.08;
25
26 long prevT = 0;
27 const float cycle_time = 0.001;
28 const int speedmax = 5000;
```



Vietnamese-German University

```

29  const int speedstep_large = 500;
30  const int speedstep_small = 100;
31  boolean controller_stat = LOW;
32
33  //DISPLAY SETTING
34  const int stat_col = 13;
35  const int stat_row = 0;
36  const int act_col = 10;
37  const int act_row = 1;
38  const int set_col = 6;
39  const int set_row = 0;
40  const int disp_count_over = 19;
41  int disp_count = 10;
42  int speed_disp = 0;
43
44  void setup() {
45      pinMode(ENCA, INPUT);
46      pinMode(ENCB, INPUT);
47
48      attachInterrupt(digitalPinToInterrupt(ENCA), readEncoder
49                      ,RISING);
50
51      //Configuring Timer 2
52      cli(); //disable interrupts
53      OCR2A = timer_reload;
54      TCCR2A = 1<<WGM21;
55      TCCR2B = (1<<CS22)|(1<<CS21)|(1<<CS20);
56      TIMSK2 = (1<<OCIE2A);
57      sei(); //enable interrupts

```

```

57 //Setup LCD
58 shield.setCursor(0,0);
59 shield.print(F("Set:          ")); //blank spaces to
    override previous printed line
60 shield.setCursor(0,1);
61 shield.print(F("Actual:          "));
62 }
63
64 void loop() {
65     switch(shield.getButtons()) {
66         case ButtonRight:
67             speed_target=speed_target+speedstep_large;
68             if (speed_target>speedmax) {
69                 speed_target=speedmax;
70             }
71             disp_set_speed();
72             break;
73         case ButtonLeft:
74             speed_target=speed_target-speedstep_large;
75             if (speed_target<-speedmax) {
76                 speed_target=-speedmax;
77             }
78             disp_set_speed();
79             break;
80         case ButtonUp:
81             speed_target=speed_target+speedstep_small;
82             if (speed_target>speedmax) {
83                 speed_target=speedmax;
84             }
85             disp_set_speed();

```

```

86         break;
87     case ButtonDown:
88         speed_target=speed_target-speedstep_small;
89         if (speed_target<-speedmax) {
90             speed_target=-speedmax;
91         }
92         disp_set_speed();
93         break;
94     case ButtonSelect:
95         controller_stat=!controller_stat;
96         disp_stat();
97         break;
98     case ButtonNone:
99         disp_set_speed();
100        break;
101    default:
102        break;
103    }
104    if (disp_count>disp_count_over) {
105        disp_count=0;
106        disp_actual_speed();
107    }
108 }
109
110 ISR(TIMER2_COMPA_vect) {
111     OCR2A=timer_reload;
112     if(pos_reset==0) {
113         speed_curr=0;
114         speed_curr=((speed_prev*2)+speed_curr)/3;
115         speed_prev=speed_curr;

```



Vietnamese-German University

```

116     }
117     pos_reset=0;
118
119     //Error Calculation
120     float e = (float)speed_target-speed_curr;
121
122     //Derivative Formula
123     float dedt = (e-eprev)/cycle_time;
124
125     //Integral Formula
126     eintegral = eintegral+e*cycle_time;
127
128     //Control Signal
129     float u=kp*e+kd*dedt+ki*eintegral;
130     float power=0;
131     int dir=0;
132
133     //Control Motor Power and Direction
134     if(controller_stat==HIGH) {
135         power=fabs(u);
136         if(power>255) {
137             power=255;
138         }
139         dir=1;
140         if(u<0) {
141             dir=-1;
142         }
143     }
144     else {
145         e=0;

```



Vietnamese-German University

```

146         eintegral=0;
147     }
148
149     //Define the Motor
150     setMotor(dir,power,PWM,IN1,IN2);
151
152     //Store error value for updating
153     eprev=e;
154
155     //Update Display Speed
156     disp_count++;
157     speed_disp=(speed_disp*2 + (int)speed_curr)/3;
158 }
159
160 void disp_set_speed() {
161     shield.setCursor(set_col,set_row);
162     shield.print(F(""));
163     shield.setCursor(set_col,set_row);
164     shield.print(speed_target);
165 }
166 void disp_actual_speed() {
167     shield.setCursor(act_col,act_row);
168     shield.print(F(""));
169     shield.setCursor(act_col,act_row);
170     shield.print(speed_disp);
171 }
172
173 void disp_stat() {
174     shield.setCursor(stat_col,stat_row);
175     if(controller_stat==HIGH) {

```



```

176         shield.print(F("On "));
177     } else {
178         shield.print(F("Off"));
179     }
180 }
181
182 void setMotor(int dir, int pwmVal, int pwm, int in1, int
    in2) {
183     analogWrite(pwm,pwmVal);
184     if(dir==-1) {
185         digitalWrite(in1,HIGH);
186         digitalWrite(in2,LOW);
187     }
188     else if(dir==1) {
189         digitalWrite(in1,LOW);
190         digitalWrite(in2,HIGH);
191     }
192     else {
193         digitalWrite(in1,LOW);
194         digitalWrite(in2,LOW);
195     }
196
197 void readEncoder() {
198     int b = digitalRead(ENCB);
199     if(b>0) {
200         pos++;
201         pos_reset++;
202     }
203     else {
204         pos--;

```

```

205         pos_reset--;
206     }
207
208     //Calculate time interval between two pulses
209     long currT=micros();
210     float deltaT=((float)(currT-prevT))/(1.0e6);
211     prevT=currT;
212     speed_curr=pos/deltaT;
213
214     //Filter to avoid jittering
215     speed_curr=((speed_prev*2)+speed_curr)/3;
216     speed_prev=speed_curr;
217     pos=0;
218 }
219 /*****END OF PROGRAM*****/

```



Vietnamese-German University

APPENDIX 2 – TRANSMITTER’S CODE – LCD & KEYPAD

```
1 //LCD
2 // Pin config:
3 //SCK 13
4 //MISO 12
5 //MOSI 11
6 //CE 2
7 //SCN 3
8 //All pins D4 - D10 are used for LCD
9
10 #include <LCDKeypadShield.h>
11 LCDKeypadShield shield;
12
13 #include <SPI.h>
14 #include <nRF24L01.h>
15 #include <RF24.h>
16
17 RF24 radio(2, 3); // CE, CSN
18 const byte addresses[][6] = {"00001", "00002"};
19
20 int pwr = 0;
21 int speed_dis = 0;
22 int speed_target = 0;
23
24 const int speedmax = 3500;
25 const int speedstep_large = 100;
26 const int speedstep_small = 20;
27 boolean controller_stat = LOW;
28
29 int send_val = 0;
30 int receive_val = 0;
31
32 //Display var or const
33 const int stat_col = 13;
34 const int stat_row = 0;
35 const int act_col = 10;
36 const int act_row = 1;
37 const int set_col = 6;
38 const int set_row = 0;
39
40 void setup() {
41     shield.setCursor(0, 0);
42     shield.print(F("Set:      "));
```



```

43 shield.setCursor(0, 1);
44 shield.print(F("Actual:          "));
45 disp_set_speed();
46 disp_actual_speed();
47 disp_stat();
48
49 radio.begin();
50 radio.openWritingPipe(addresses[1]); // 00002
51 radio.openReadingPipe(1, addresses[0]); // 00001
52 radio.setPALevel(RF24_PA_MIN);
53 }

54
55 void loop() {
56   delay(5);
57   radio.stopListening();
58
59   switch (shield.getButtons()) {
60     case ButtonRight:
61       speed_target=speed_target+speedstep_large;
62       if (speed_target>speedmax) {
63         speed_target=speedmax;
64       }
65       disp_set_speed();
66       send_val = speed_target;
67       break;
68     case ButtonLeft:
69       speed_target=speed_target-speedstep_large;
70       if (speed_target<-speedmax){
71         speed_target=-speedmax;
72       }
73       disp_set_speed();
74       send_val = speed_target;
75       break;
76     case ButtonUp:
77       speed_target=speed_target+speedstep_small;
78       if (speed_target>speedmax){
79         speed_target=speedmax;
80       }
81       disp_set_speed();
82       send_val = speed_target;
83       break;
84
85     case ButtonDown:
86       speed_target=speed_target-speedstep_small;

```

```

86         if (speed_target < -speedmax){
87             speed_target = -speedmax;
88         }
89         disp_set_speed();
90     send_val = speed_target;
91     break;
92     case ButtonSelect:
93     controller_stat = !controller_stat;
94     disp_stat();
95     if (controller_stat == HIGH){
96     send_val = speedmax+1000;
97     }
98     else {
99     send_val = -speedmax-1000;
100    }
101
102    break;
103    case ButtonNone:
104    disp_set_speed();
105    break;
106    default:
107    break;
108    }
109    radio.write(&send_val, sizeof(send_val));
110    delay(5);
111    radio.startListening();
112    //while (!radio.available());
113
114    for (int i=0; i<5; i++) {
115        if ( radio.available() ) {
116            while (radio.available() ) {
117                radio.read(&receive_val, sizeof(receive_val));
118                speed_dis =receive_val;
119            }
120            break;
121        }
122        else{
123            delay(1);
124        }
125    }
126
127    disp_actual_speed();
128 }
129

```

```

130 void disp_set_speed(){
131     shield.setCursor(set_col, set_row);
132     shield.print(F("      "));
133     shield.setCursor(set_col, set_row);
134     shield.print(speed_target);
135 }
136
137 void disp_actual_speed(){
138     shield.setCursor(act_col, act_row);
139     shield.print(F("      "));
140
141     shield.setCursor(act_col, act_row);
142     shield.print(speed_dis);
143 }
144 void disp_stat(){
145     shield.setCursor(stat_col, stat_row);
146     if (controller_stat==HIGH){
147         shield.print(F("On "));
148     } else{
149         shield.print(F("Off"));
150     }
151 }

```



Vietnamese-German University

APPENDIX 3 – RECEIVER’S CODE – DC MOTOR

```
1 //Motor
2 #define ENCA 2
3 #define ENCB 3
4
5 #define PWM 9
6 #define IN2 7
7 #define IN1 8
8
9 // Pin config:
10 //SCK 13
11 //MISO 12
12 //MOSI 11
13 //CE 4
14 //SCN 5
15
16 #include <SPI.h>
17 #include <nRF24L01.h>
18 #include <RF24.h>
19
20 RF24 radio(4, 5); // CE, CSN
21 const byte addresses[][6] = {"00001", "00002"};
22
23 int pwr = 0;
24 int speed = 0;
25 int send_val = 0;
26 int receive_val = 0;
27
28 byte timer_reload = 0x4e;
29 int pos = 0;
30 int pos_reset=0;
31
32 float speed_prev = 0;
33 float speed_curr = 0;
34 int speed_target = 0;
35
36 float eprev=0;
37 float eintegral=0;
38
39 //PID constant
40 float kp = 0.03;
41 float kd = 0;
42 float ki = 0.2;
```

```

43
44 long prevT = 0;
45
46 const float cycle_time = 0.005;
47 const int speedmax = 3500;
48
49 boolean controller_stat = LOW;
50
51 void setup() {
52     pinMode(ENCA,INPUT);
53     pinMode(ENCB,INPUT);
54     attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING);
55
56
57     cli();
58     OCR2A = timer_reload;
59     TCCR2A = 1<<WGM21;
60     TCCR2B = (1<<CS22) | (1<<CS21) | (1<<CS20);
61     TIMSK2 = (1<<OCIE2A);
62     sei();
63
64     radio.begin();
65     radio.openWritingPipe(addresses[0]); // 00001
66     radio.openReadingPipe(1, addresses[1]); // 00002
67     radio.setPALevel(RF24_PA_MIN);
68 }
69
70 void loop() {
71     delay(5);
72     radio.startListening();
73     if ( radio.available()) {
74         while (radio.available()) {
75             radio.read(&receive_val, sizeof(receive_val));
76         }
77         delay(5);
78         radio.stopListening();
79         send_val = (int)speed_curr;
80         radio.write(&send_val, sizeof(send_val));
81     }
82     if (abs(receive_val) < (speedmax+900)){
83         speed_target = receive_val;
84     }
85     else {

```



```

86         if (receive_val > (speedmax+900)) {
87             controller_stat = HIGH;
88         }
89         else {
90             controller_stat = LOW;
91         }
92     }
93 }
94
95 ISR(TIMER2_COMPA_vect)
96 {
97     OCR2A = timer_reload;
98     if (pos_reset ==0){
99         speed_curr = 0;
100        speed_curr = ((speed_prev*2) + speed_curr)/3;
101        speed_prev = speed_curr;
102    }
103    pos_reset=0;
104
105    // error
106    float e = (float)speed_target - speed_curr;
107
108    // derivative
109    float dedt = (e-e_prev)/cycle_time;
110
111    // integral
112    eintegral = eintegral + e*cycle_time;
113
114    // control signal
115
116    float u = kp*e + kd*dedt + ki*eintegral;
117    float pwr = 0;
118    int dir = 0;
119    // motor power
120    if (controller_stat==HIGH){
121        pwr = fabs(u);
122        if( pwr > 255 ){
123            pwr = 255;
124        }
125
126        // motor direction
127        dir = 1;
128        if(u<0){
129            dir = -1;
130        }

```

```

130     }
131     else {
132         e = 0;
133         eintegral =0;
134     }
135
136     // signal the motor
137     setMotor(dir,pwr,PWM,IN1,IN2);
138
139     // store previous error
140     eprev = e;
141
142     //update display speed
143     //disp_count++;
144
145     //speed_disp = (speed_disp*2 + (int)speed_curr)/3;
146 }
147
148 void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
149     analogWrite(pwm,pwmVal);
150     if(dir == -1){
151         digitalWrite(in1,HIGH);
152         digitalWrite(in2,LOW);
153     }
154     else if(dir == 1){
155         digitalWrite(in1,LOW);
156         digitalWrite(in2,HIGH);
157     }
158     else{
159         digitalWrite(in1,LOW);
160         digitalWrite(in2,LOW);
161     }
162 }
163
164 void readEncoder(){
165     int b = digitalRead(ENCB);
166     if(b > 0){
167         pos++;
168         pos_reset++;
169     }
170     else{
171         pos--;
172         pos_reset--;
173     }

```

```
173 // time difference
174 long currT = micros();
175 float deltaT = ((float) (currT - prevT))/( 1.0e6 );
176 prevT = currT;
177 speed_curr=pos/deltaT;
178 //filter
179 speed_curr = ((speed_prev*2) + speed_curr)/3;
180 speed_prev = speed_curr;
181 pos=0;
182 }
```



Vietnamese-German University

APPENDIX 4 – CODE SECTION FOR OPEN-LOOP DC MOTOR CONTROL

```
void setMotor(int dir, int pwmVal, int pwm, int in1, int in2) {  
    if (pwmVal>255) { //filter restrict pwmVal exceed 255  
        pwm=255;  
    }  
    if (pwmVal<0) { //filter restrict pwmVal go below 0  
        pwm=0;  
    }  
    analogWrite(pwm,pwmVal); //write pwmVal to pwm  
    //motor direction control  
    if (dir == -1) { //comparative if statement must use ==  
        digitalWrite(in1,HIGH); //counterclockwise direction  
        digitalWrite(in2,LOW);  
    }  
    else if (dir == 1) { //clockwise direction setting  
        digitalWrite(in1,LOW);  
        digitalWrite(in2,HIGH);  
    }  
    else { //no register of direction, turn the motor off  
        digitalWrite(in1,LOW);  
        digitalWrite(in2,LOW);  
    }  
}
```



Vietnamese-German University