ASHESI UNIVERSITY

B. Sc. ELECTRICAL AND ELECTRONICS ENGINEERING

WIRELESS MOTOR CONTROL

AMHOL JACOB DHIEU 87232020

2020

ASHESI UNIVERSITY

WIRELESS MOTOR CONTROL

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering,

Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Electrical and Electronics Engineering.

AMHOL DHIEU

2020

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

Amhol Jacob Dhieu

……

Date:

……

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on the supervision of the capstone laid down by Ashesi University.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

Acknowledgments

I would like to express my gratitude to everyone who helped me most throughout this project. Although this was an individual endeavor, I must acknowledge the contribution and support from all who made sure that I had the right resources and environment conducive to the progress of the project. Most especially, am grateful to my supervisor, Kofi Adu-Labi, for being a consistent supporter and mentor from the beginning of this project to this point.

Not to forget the contribution of others who committed their time and effort to make this project a success, a special thanks goes to my colleagues whom I could count on for their skills, brainstorming, and general ideas. I will always be indebted to you.

Table of Contents

Abstract

Today, both Direct Current (DC) and Alternating Current (AC) motors are utilized in many different applications in a variety of sectors. Such applications in the automobile and domestic sectors provide the energy required to generate mechanical power. The mechanical utilities such as winches, garbage compactors, auto walks, and elevator shafts use motor power to provide functional mobility at variable speed ranges adjusted as required [1] [2].

However, the mechanism to control motors is not 'smart,' as is the case in different electronic applications of today that are controlled by the mechanism of Application Programming Interfaces (APIs) [3] [4]. The listed examples still make use of analog functionalities, i.e., levers and pushbuttons. In the garbage compactor application, controlling the motor uses pushbuttons in the cabin, and, also the levers at the back of the truck. Operating the machine is tedious for the driver or the loader in that they must leave the cabin to operate the auxiliary machine each time. The process is labor-intensive with additional health hazards of having to deal with bad odor from the waste material. There is also an increased risk of physical injury, exposure to dangerous diseases from the dirty working environment; such as cholera and diarrhea as well as exposure to novel viruses not yet known in the medical field. Cases in point include exposure to Severe Acute Respiratory Syndrome (SARS) and Covid-19 [5].

In addition to the health hazards inherent in the manual application of the winch machine, the operator faces a further challenge of not being able to keep the back of the truck within the line of sight. The operator has to focus on the operating buttons in the cabin while at the same time trying to manage the auxiliary functionalities at the back.

This project, therefore, proposes to solve such problems that arise mainly from manual control of motors by implementing wireless controls by integrating functionalities like that with the APIs. The proposed method of motor control involves controlling the DC or AC motor using an external device such as remote control or mobile applications.

Chapter 1: Introduction

In the world today, electric machines have not widely incorporated a smart method of control. Moreover, speed control of AC and DC motors is one of the key aspects of automobile application. This project, therefore, aims to create a system that can vary speed, directions, start and stop controlled by an external device such as a mobile phone application or remote control. Doing so eases the labor requirement for motor control as well as preventing the pending possible hazards arising from controlling auxiliary waste machine at proximity.

1.1 Objectives of the project

The main objective of this project is to design an operating system of wireless AC and DC motor control that is feasible and can be used to solve the problems associated with an analog system of operation. The objectives are:

- to design a microcontroller-based controller unit that involves a module for sending and receiving commands.
- develop a necessary computer program that can be used for the microcontroller control unit and execution plans.
- Also, to provide hands-on experience that combines electronics, computing, and mechanical activities.

1.2 Motivation

- To be able to physically apply the knowledge of engineering into a problem in real life.
- The subject covers skills from computing, electronics, and mechanical fields. Therefore, the project will adapt hands-on experience in respective fields for real-life applications.

1.3 Scope of the work

This project proposes to build a system that will operate the dc motor system wirelessly within a limited range, of up to 100 meters, using Bluetooth communication module as a controller interfaced with the system; the motor.

The microcontroller implementation plays a major part in this project, in that, it is interfaced with the circuit. In using the Pulse-Width Modulation (PWM) technique, which changes the duty cycle of a signal transmitted from the microcontroller to the motor driver, different parameters can be influenced [6] [7]. The commands in the microcontroller are done in the C programming software and implemented in the device. The frequency pattern of the pulse width modulator sets the speed of the motor.

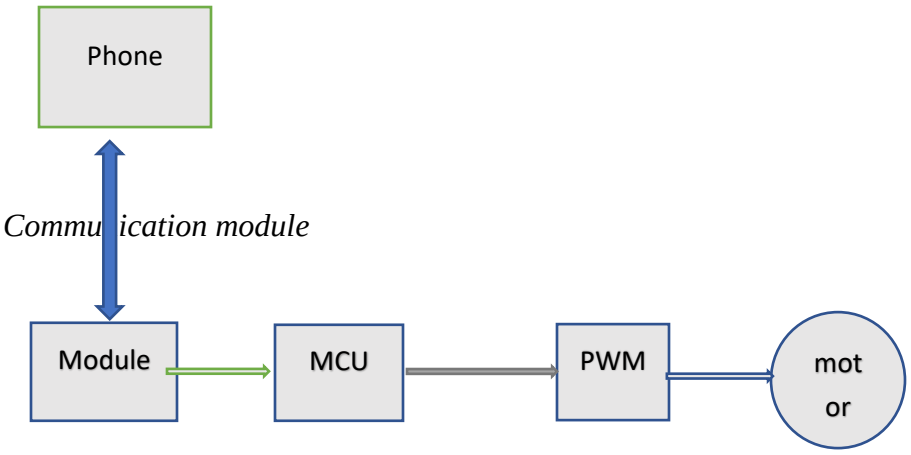

Figure 1.1 Block diagrams of the system

Expected Outcomes of the Project work

The user should be able to control the motor wirelessly. That involves start, stop, speed control, and direction.

The predetermined design of the motor application should be able to operate on the wireless control.

1.3.1 Research Methodology Used

Literature review.

Practical/experimentation.

1.3.2 Requirements

The wireless interface – the Bluetooth module- for controlling the motor operation. The receiver microcontroller senses the state of various parameters and responds appropriately as per the instructions provided. The system characteristics are mainly:

- Cost-effective
- Provides safety and relief (in terms of labor requirement)
- Reduces working time

Chapter 2: Literature Review

Several projects have been done on the related field; that is, wireless control of motors. Literature regarding the project is reviewed here. [8] Daniel Sarb and Razvan Bogdan have proposed a solution that uses a microcontroller-based radiofrequency dependent system. The proposed solution is built on a low-cost development board, the Freescale Freedom KL26Z, capable of enabling rapid embedded application development. The design incorporates the Proportional Integral Derivative (PID) controller that derives an accurate speed output of the motor. This is with the help of the Pulse-Width Modulation (PWM) technique process to give a clear cycle of the signal transmitted from the microcontroller to the motor driver. [9] Kouki, Boe, Vantroys, & Bouani proposed an efficient Network Predictive Control (NPC) firmware for IoT used to remotely control DC motor through radio frequency links. The application is based on a microcontroller STM 32 to design and improve the effectiveness and performance of NPC strategy in an RF communication system.

In [10], a project by K. S. Ravi Kumar., et al (2015) implemented the control of brushless DC motor using the Bluetooth module that is connected to Arduino Uno ATmega328 microcontroller. The motor speed is controlled with the help of the Pulse-Width Modulation technique and also includes the control of direction. [11] Jia Yunfei, Liu Jiping, and Jiang Feng designed a wireless controller and motor driving module based on the low-power microcontroller chip MSP430, an ultra-low-power 16-bit microcontroller. They also make use of the means of Pulse Width Modulation for speed control as well as an armature voltage control, giving the motor an accurate precision, fast response, and wide speed range.

[12] Mrs. A. D. Kadage developed a wireless control system motor for agricultural application. The system is implemented on the microcontroller AT 89C52 that can allow the operation on

either 900 MHz or 1800 MHz frequency band. Therefore, it is possible to control the agricultural motors using the GSM technology which performs the controlling action using Short Messaging Services of a cell phone, hence allowing easy control. They designed a self-aware system that can be controlled and give information via short messaging services (SMS). The system can check conditions, such as overload, temperatures, dry runs, etc. through the designed implementation on the microcontroller interfaced with the short messaging portal for the communication interface.

Chapter 3: Design

The design specification of the required device has considered both remote control and mobile application and narrowed down the scope to overall functionality rather than specific interfaces of each alternative. As such, the project requirements concerning the wireless control of AC and DC motors are listed below.

3.1 Project Requirements

- The system shall enable wireless control of the auxiliary machine that makes use of either AC or DC motor.
- Wireless motor control implementation shall be compatible with the operation of the system. That is, it should not complicate or introduce more problems to the automobile application i.e. winch application.
- The system should be cost-effective and efficient to build and implement.
- The system shall be responsive (on-time) at a distance within 100 meters of range and work efficiently and effortlessly of the operator; this will help avoid unexpected circumstances/accidents.
- The system shall respond effectively to the speed changes and general commands from the controller unit.
- The system shall be multidirectional in command and receive the command from any distance within a 100 meters limit and any direction of operation
- The auxiliary machine should have a quick starting time.
- Components for the system should be locally available.

The design of the wireless control of motors for industrial or home applications consists of mainly three parts.

- The first part is comprised of the wireless control unit. The wireless control signal is received and decoded by the module at the receiving portion of the system.
- Second is the microcontroller signal outputs that control the output to the motor drive which drives the motor.
- And lastly, the external power supply from the dry cell batteries, a Pulse Width Modulation generated from Keil uVision FRDM board and LCD

Figure 3.1 shown a block diagram of the system parts and operation.

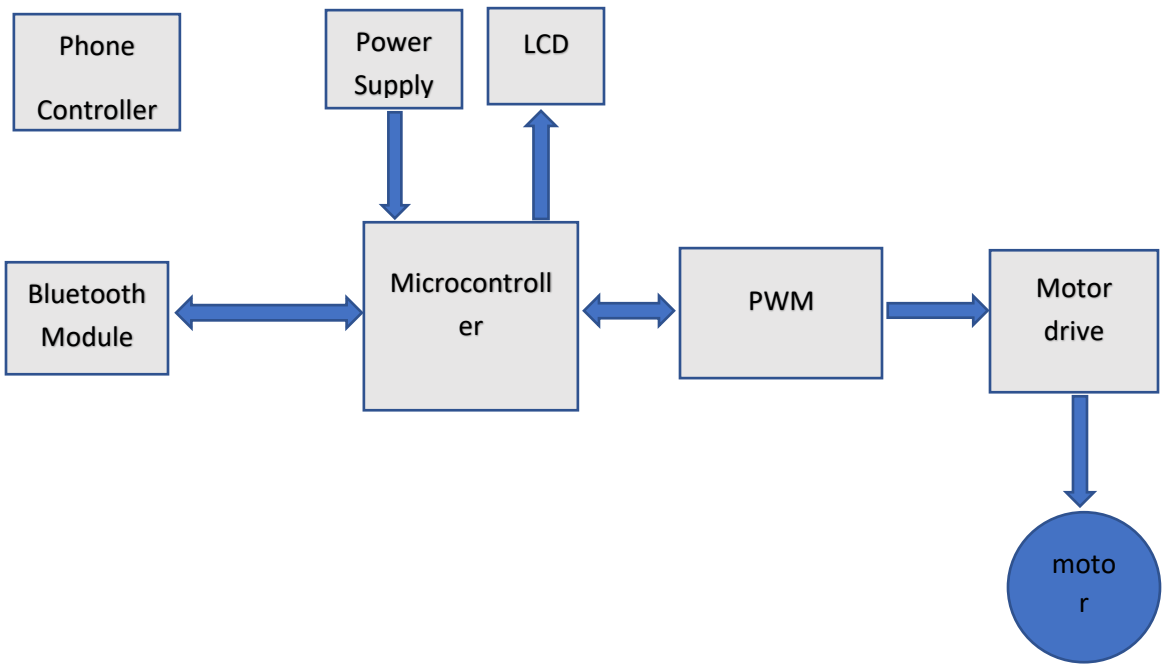

Figure 3.1: Block Diagram of the design

3.1.1 Design Components Selection

3.2 Pugh Chart

3.2.1 Modes of Communication

Table 3.1. Communication module

Selection Criteria	**Baseline**	**Weight**	**Bluetooth**	**WIFI**	**Zigbee**	**Infrared (IR)**
Efficiency/ Responsiveness	0	2	0	+2	-2	-2
Power Consumption	0	1	+1	-1	+1	+1
Cost and availability	0	2	+2	+1	+1	+1
Preferability	0	1	+1	+1	+1	+1
Coverage	0	1	+1	+1	+1	0
Net weight			5	4	2	1

Therefore, the Pugh chart analysis narrowed down the best choice for controlling the system wirelessly to be Bluetooth controller methods according to the project system requirements.

3.2.2 Bluetooth

From the Pugh Chart, Bluetooth falls in as a cheap, available, effective, and preferred means to control AC and DC motors from a distance.

Bluetooth is available on many devices for linking wireless communication within a short-range. As such, more devices use Bluetooth technology and there is a need to make many products compatible [13]. The device is a low cost, easily available, and has low power consumption [14]. Bluetooth can be used to control devices in proximity, and strength of control reduces as the distance increases beyond the recommended distance. Another favorable advantage of Bluetooth is its reduced possibility of interference from other devices in close proximity [15]. This gives security for operation without interruption.

3.2.3 Objectives

In this project, the objectives in relation to the use of Bluetooth is

- use Bluetooth link (Bluetooth device and module) for transmitting and receiving process
- to be able to control the motor from a distance

3.2.4 Bluetooth Module

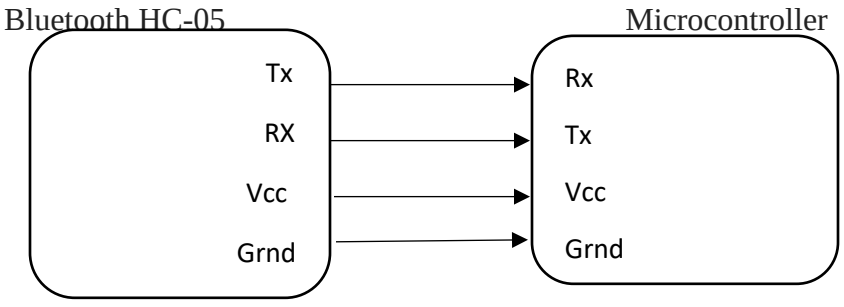

Figure 3.2: Bluetooth interface with MCU

The Bluetooth interface with the KL25Z board is regulated via the UARTs interface on the board. The connection is as shown above in Figure 3.2 with the Bluetooth transmitter to the microcontroller's receiver, and its receiver to the transmitter. The system is also powered via a 5V source from the board or an external dry cell battery.

3.3 Microcontroller Unit

Table 3.2 Subsystem: Microcontrollers/MCU

Selection Criteria	**Basis- ATmega328 (Arduino's)**	**Weight**	**ATmega8L MCU**	**AT89C52 MCU**	**KL25Z MCU**
Cost and availability	0	2	+2	+2	+2
Analog and Digital inputs	0	1	+1	-1	+1
Memory	0	1	+1	+1	+1
Power consumption	0	1	+1	+1	+1
Programming environment	0	2	+2	+2	+2
Speed and operating voltage	0	2	+1	+1	+1
PWM output port	0	1	-1	-1	+1
Net Score			7	5	9

Pugh chart elimination method appropriated KL25Z microcontroller on the FRDM KL25Z board much suited for the project.

3.3.1 KL25Z FRDM Board

KL25Z FRDM MCU has many hardware tools built into it: a CPU to execute software, an efficient interrupt system, etc. Its main advantage to this project is that it has all the components required for PWM and analog outputs, UARTS inputs interface, and many more incorporated in a single printed circuit board (PCB) [6]. The board can be programmed in an embedded C or C++ and compiled to run in the processor's machine language [6] to give the output required i.e., analog waveform or Pulse-Width Modulation.

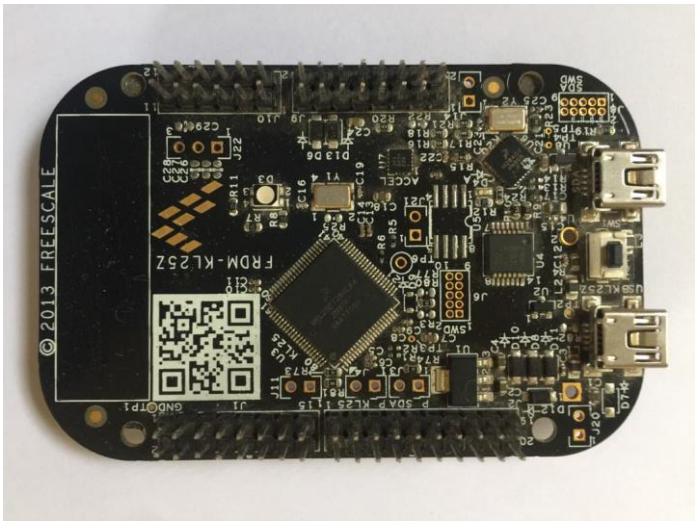

Figure 3.3: KL25Z

3.3.2 Pulse Width Modulation

There are several ways to control the speed of a DC motor. The simplest way is using a variable resistance in series with a DC motor. However, the method is wasteful. As such, the best way to control the speed of DC is to regulate the amount of voltage across its terminal (armature voltage control) using Pulse-Width Modulation (PWM). The method works by driving the motor with a series of ON and OFF pulses and a varying duty cycle (the fraction of time that the output voltage is on) [7] fed into the motor driver. For example, the average DC voltage for 0% duty cycle

is 0 voltage, 25% duty cycle average to 1.25V (25% of 5V), 2.5V for 50% and constant 5 V at 100% duty cycle [16]. A single waveform showing half duty cycle is shown in the figure 3.4.

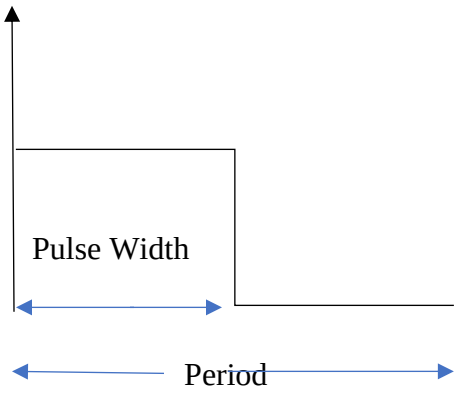

Figure 3.4: 50% duty cycle.

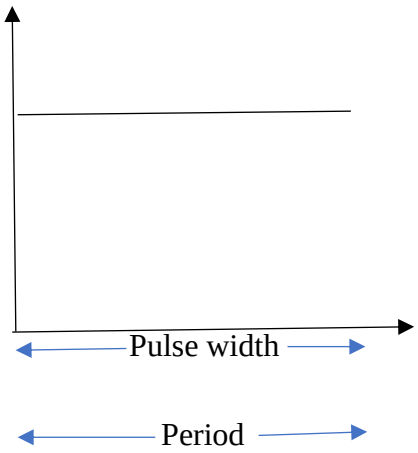

Figure 3.5: 100% duty cycle

As a result of varying the duty cycles, the speed of the motor can easily be varied with a controlled pulse width at a constant frequency, and the motor at full speed when the duty cycle is full or at 100%.

3.3.3 Relays

It is a motor driver which allows the motor to drive in on either direction. It works on the concept of the H-bridge which allows voltage to flow in either direction. The motor receives the voltage supply from the Pulse-Width Modulation output from the board to drive the motor in a direction commanded from the board. The average voltage received determines the speed at which the motor rotates.

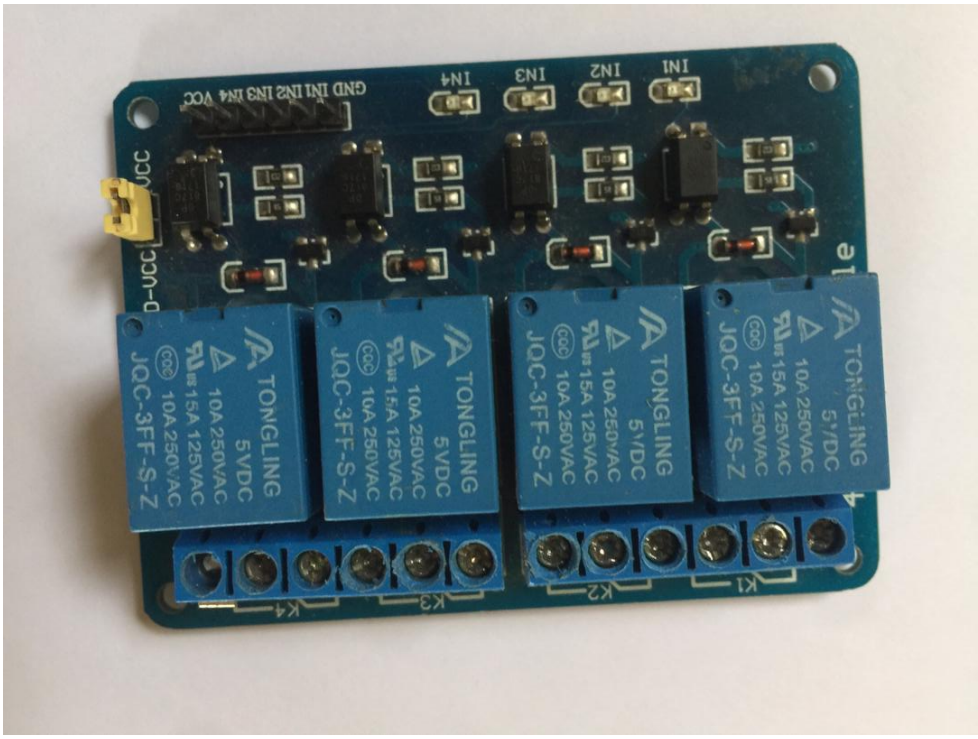

Figure 3.6: 5VDC 10 A 250 VAC Relay

3.4 AC Motor Speed Control

AC motor speed control by altering the frequency of the ac waveform. The higher the frequency, the faster the motor rotates and vice versa. This is achieved by feeding alternating voltage supply into the machine's driver.

Design Components

- Bluetooth module (HC-05)
- KL25Z microcontroller (The FRDM-KL25Z)
- Phone with Bluetooth
- 290816 DC motor
- Jumper wires
- Four 10k Resistors
- Four 10uF capacitors
- 1 Breadboard
- 1 LCD
- 120 VDC 250 VAC relays
- 240 V Split-phase motor
- 120 V DC motor

Chapter 4: Methodology

The project implementation is done through both software and hardware design. First, the software section of the project includes the code (embedded C) – of which the DC motor control code is using Keil uVision. Moreover, the AC signal is programmed through mbed using KL25Z board as well. Through software, the communication between the Bluetooth module and the motor is made possible.

The other design is the hardware design that is directly dependent on the software implementation. First, is the physical connection of the Bluetooth module to the KL25Z board: that enables the user to directly control the motors via a microcontroller. Then, a physical connection between the microcontroller to an external motor driver or a relay that the microcontroller uses to engage the motor. And finally, an external power supply is used to drive the motor.

4.1 Testing procedure I: For DC supply

1. Test if the device can be used to wirelessly control the output commands from the board. Here, the pulse width of the PWM is varied using Bluetooth commands.
2. Test/view the PWM wave generated from the board through Digilent Waveforms.
    a. The dc motor speed and direction of rotation method is implemented with the use of pulse width modulation from the board. The PWM is fed into the relays on the two input terminals which determine the output to the motor. The two output terminals feed the motor as a negative and positive terminal. As such, if one is active and the other is not from the board, the motor rotates in a clockwise direction, and when vice

versa, the motor changes the direction to the anti-clockwise direction. Moreover, the pulse width determines the speed of the motor; the bigger the pulse, the more the speed and vice versa.

b. For DC, check the duty cycles as varied by the Bluetooth controller. The Digilent Waveforms outputs for various pulse width are as follow:

i). A figure showing a 10% duty cycle

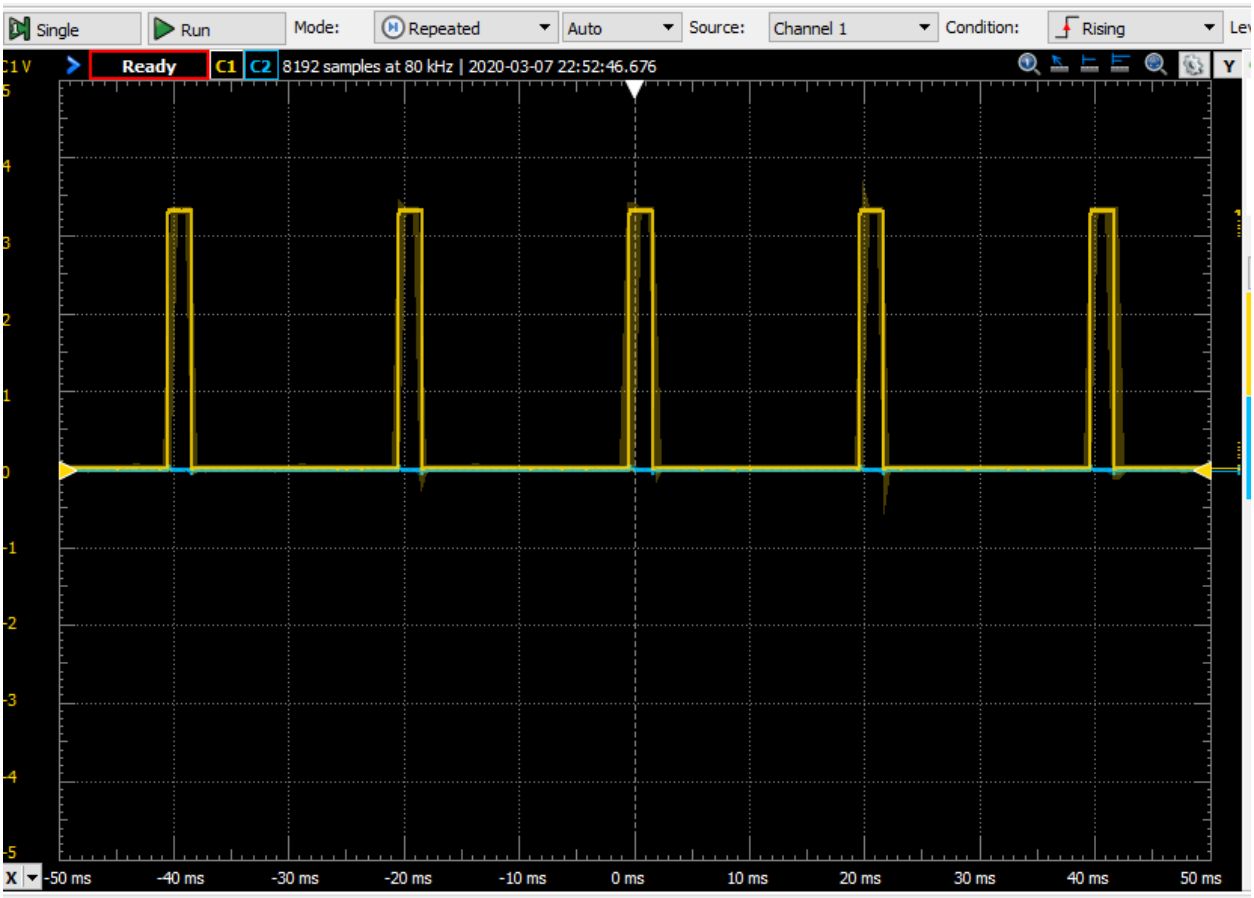

Figure 4.1: Period of 20ms with a pulse width of 2ms

ii). A 50% duty cycle PWM

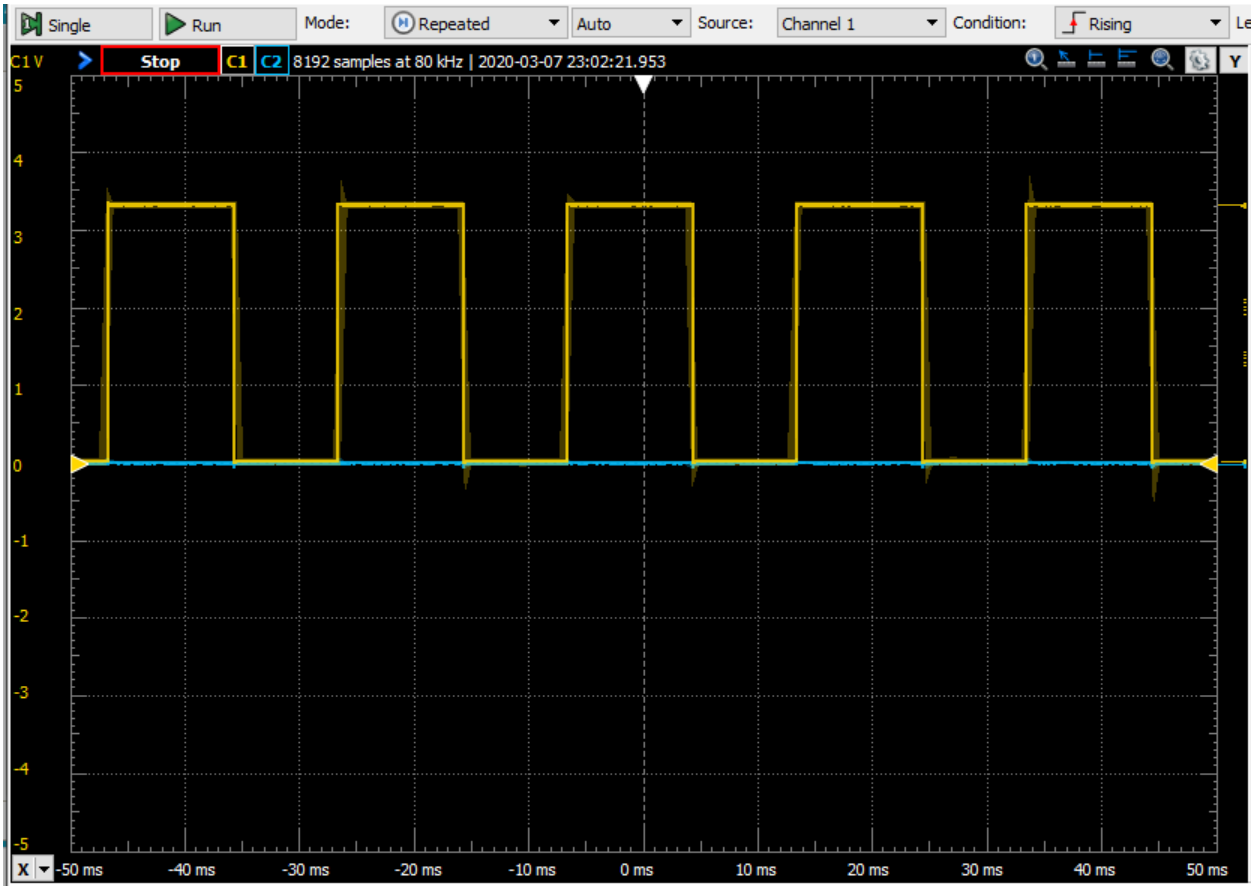

Figure 4.2: Period of 20ms and 10ms pulse width

iii). An 80% duty cycle PWM

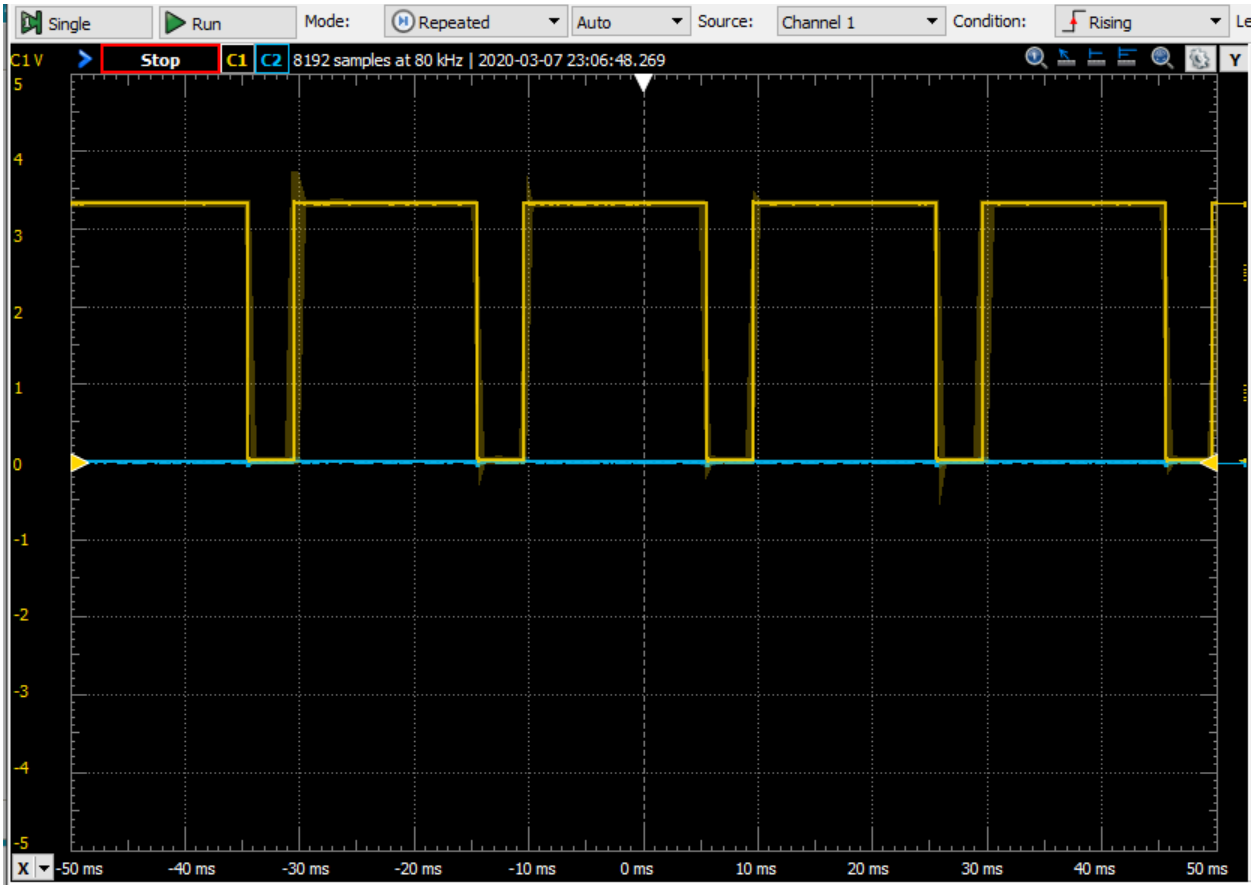

Figure 4.3: 80% duty cycle

iv). 100% duty cycle is a DC power supply.

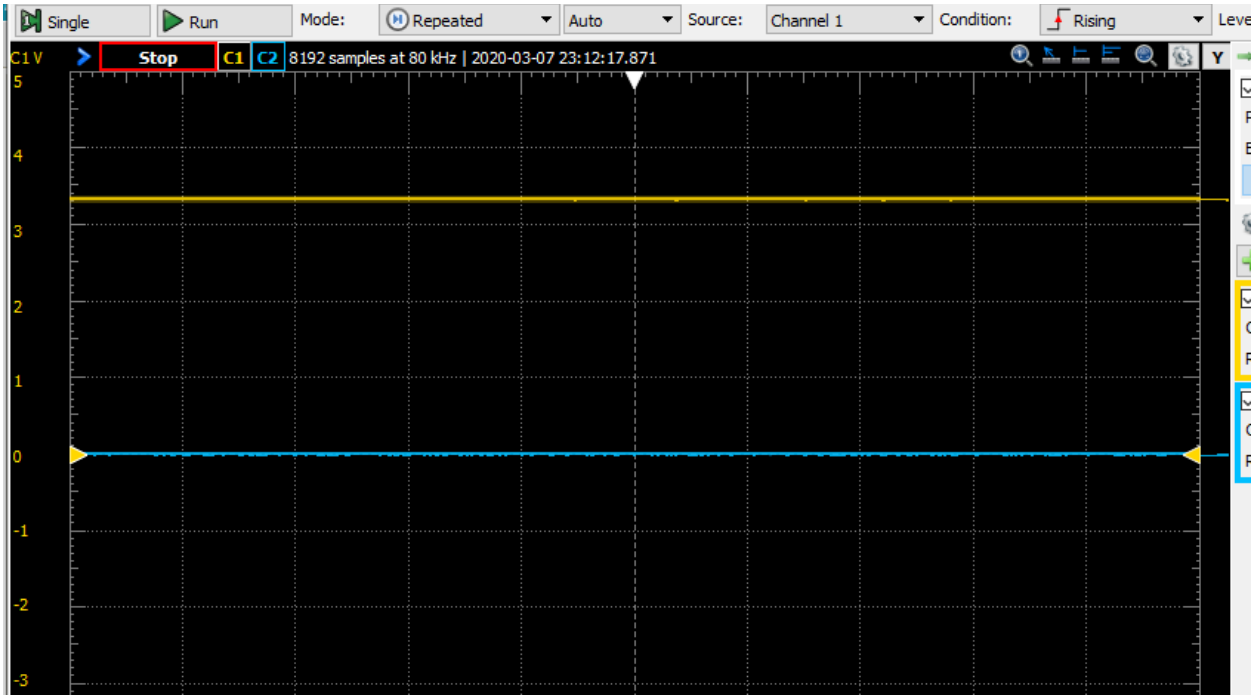

Figure 4.4: 100% duty cycle

4.2 Testing Procedure II: For AC supply

a. The speed control of ac motor in this project is set by varying the frequency of the wave; the higher the frequency, the higher the speed, and vice versa. Therefore, the period of sine wave output from the board is regulated according to the required frequency that defines the speed of the motor.

   For AC, check for the frequency of the wave varied by changing the wave period of the signal, and the outputs viewed in Waveforms.

   Figure 4.5 and Figure 4.6 are the sine wave signals viewed from the Scope screen.

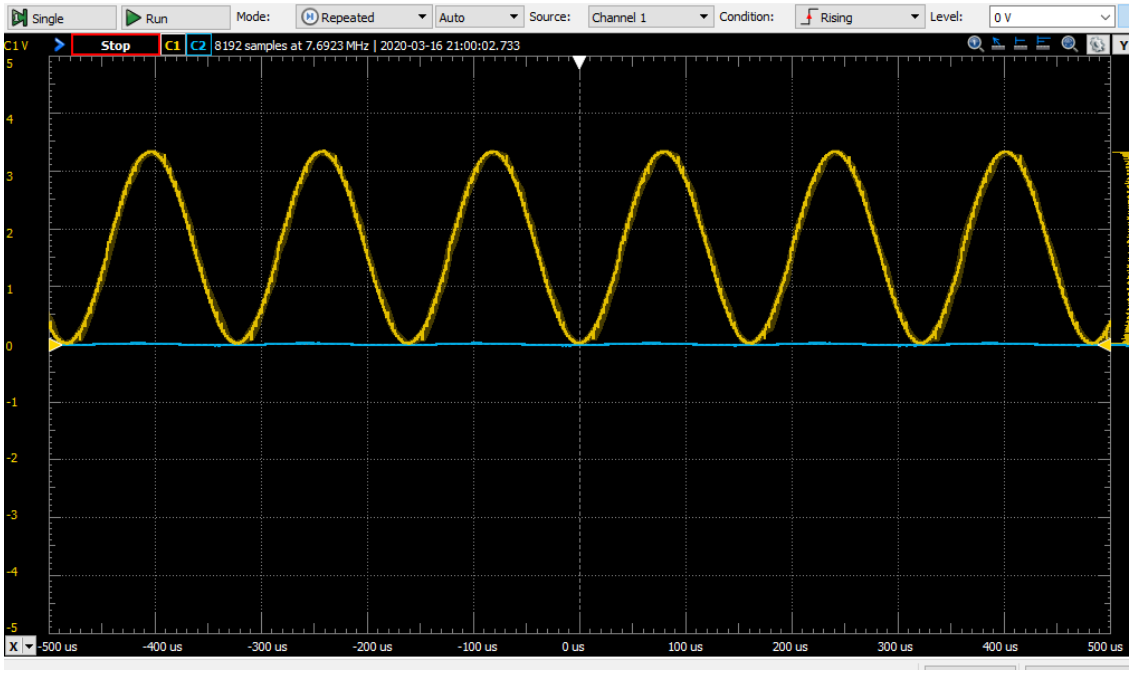

Figure 4.5: Frequency =1/150 microseconds

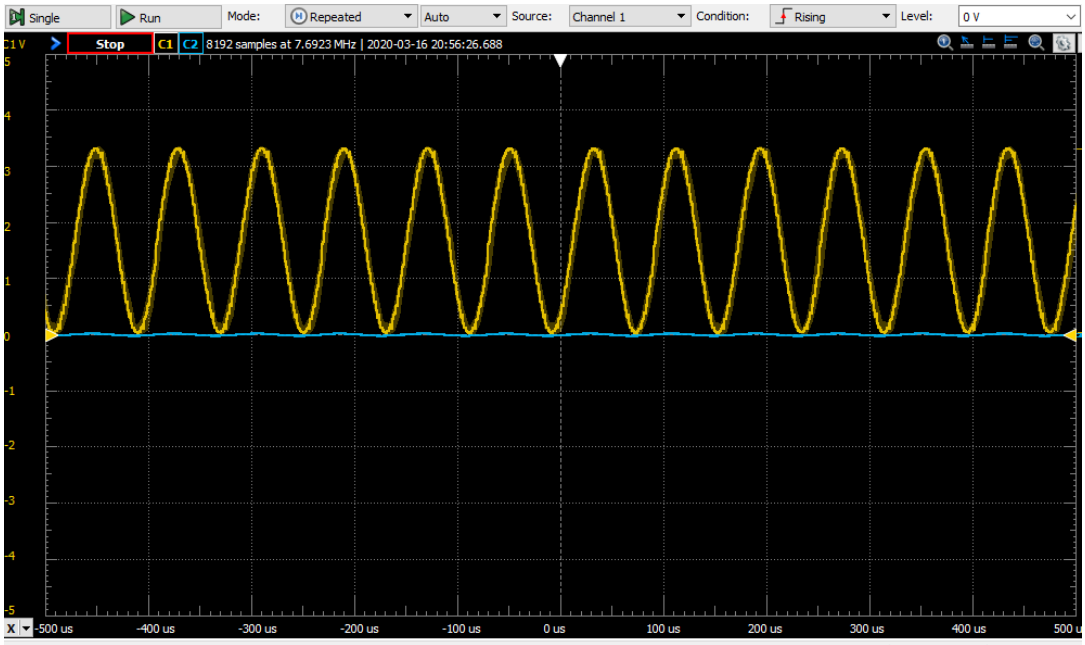

Figure 4.6: Frequency=1/100uSeconds

b. The AC signals in Figure 4.5 and Figure 4.6 are non-zero crossing signals. As such, a suitable zero-crossing circuit is required to alternate the waves across the zero lines.
c. A simple alternative to derive a zero-crossing signal is passing the signals through a 1:1 transformer. In a transformer, an ac signal provides an alternating magnetic field (either from a zero-crossing or non-zero-crossing alternating currents) [17]. The changing magnetic fields also induce a changing voltage in the secondary coil thus inducing an alternating current in the secondary circuit; a zero-crossing signal is achieved since the transformer recognizes the pulses and gives out the pure sine waves on the secondary side [18] [19].
d. The output from the transformer (the secondary side alternating current), is used as the input to the relay coil that engages the relay.

Software section

The microcontroller (KL25Z) is programmed in embedded C to function remotely in the system. The signal of the Bluetooth android application is sent and received by the Bluetooth module interfaced with the microcontroller. The corresponding signals (commands) is sent to the motor and the motor output is given as required. Figure 4.7 shows the flow chart for the embedded code in the microcontroller unit. The C code is attached in the appendix.

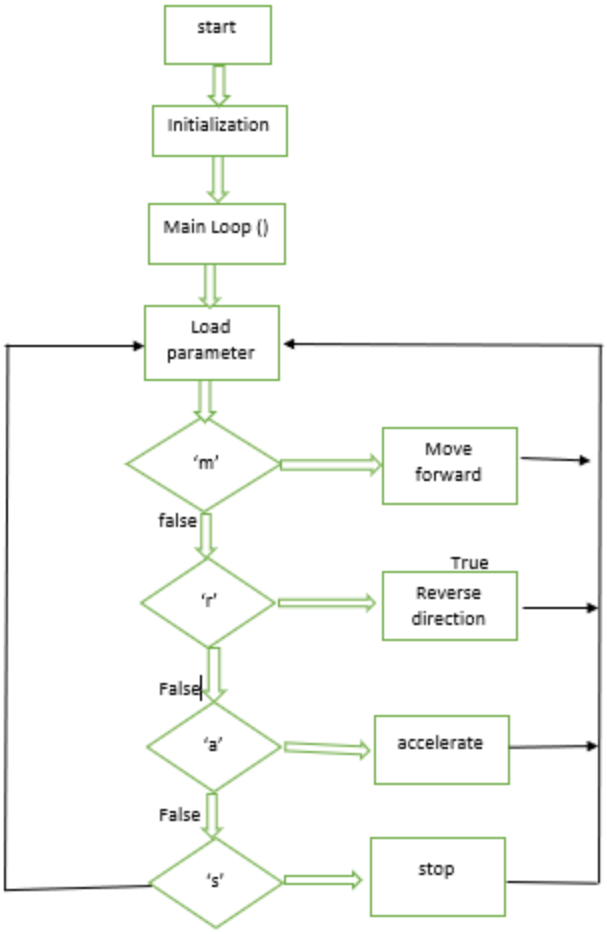

Figure 4.7: Flow Chart

4.3 Relays

Electric relays in Electrical Engineering mainly act as switches between the board and the high voltage motor. The board is unable to provide enough power that is required to run the motors. As such, an external power supply is used to drive the motor and provide enough power required for driving electric motors via motor relays. The relays act as an electric switch that the board uses to engage [20]. The interface between the relays and the motors is the mechanism of electromagnetism.

The set-up diagram shown in Figure 4.7 shows the connection of motors (ac or dc) with relays. The relay-based motor controller works with an H-bridge arrangement [21] [22]. The two common poles from the two relays go to the motor terminals. The normally closed terminals connected to the ground or negative terminal. The relay coils terminals are connected to PWM outputs from the board. The PWM input to the relay coils engages the Normally Open terminal of the relay and completes the H-bridge circuit whenever each terminal is activated [22]. The relay acts as a separating module between the board and the external power supply to the motor.

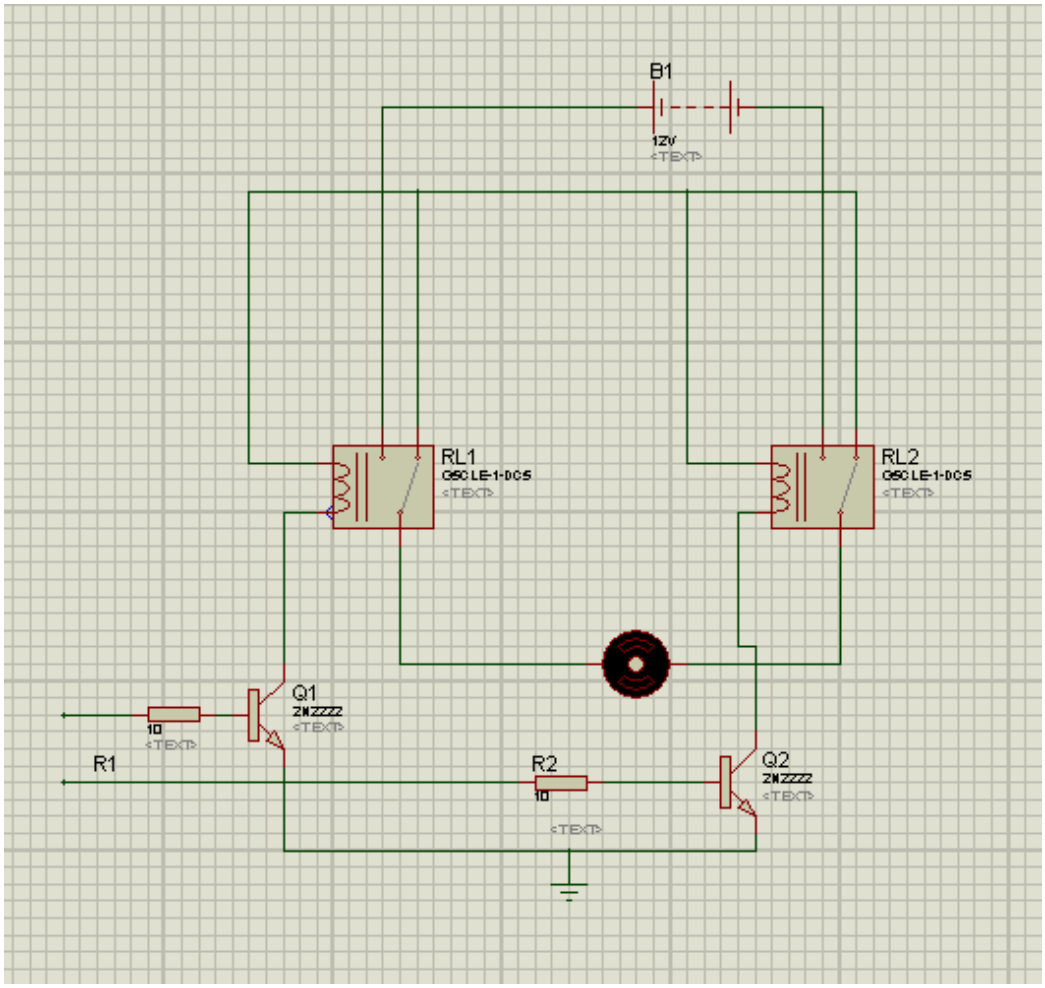

Figure 4.7: Interface connection between relays and motors

Relays can handle voltages of up to 240 VDC and 250VAC and as high as 30 A current depending on the type. Therefore, they are quite efficient and effective in controlling high voltages from a low voltage supply such as the one from the board.

The figure below shows the hardware set-up for dc motor control using 5VDC 250VAC relays, KL25Z board, power supply (batteries), Bluetooth module, and a Bluetooth device.

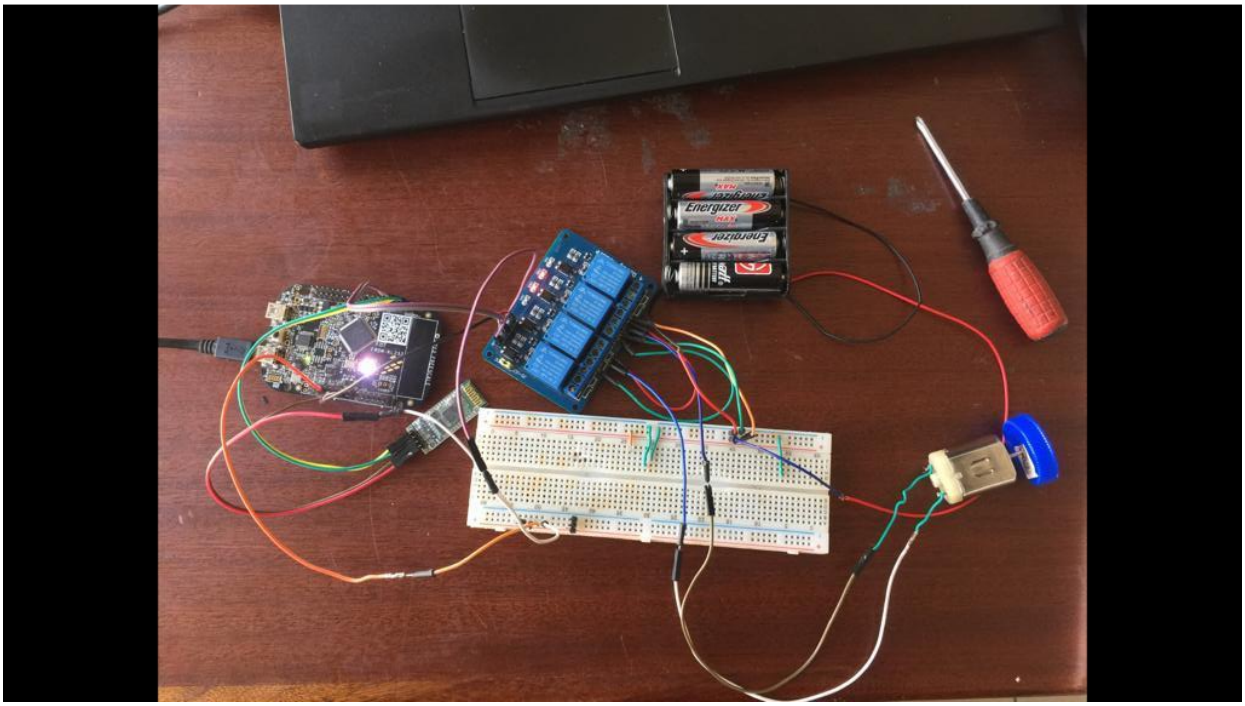

Figure 4.8: Hardware Connection

Chapter 5: Results and Discussions

5.1 For AC motor

Single-phase induction motors have challenges in reversing the direction since the magnetic field does not rotate, rather pulses and remains in one direction. However, the implementation of special single-phase motors such as split-phase motor and capacitor-start motors make it possible to reverse the directions of rotation as well as control the speed of the motor. In the split-phase motor, there are two windings: the main stator winding and an auxiliary starting winding, both set at 90 electrical degrees from each other [1]. The direction of rotation is determined if the auxiliary winding is either 90 electrical degrees ahead of or behind the main stator windings. Supplying power to the terminals sets the main stator winding 90 degrees ahead of the auxiliary winding, therefore, the motor rotates in the clockwise direction. Switching the terminals, the terminals of auxiliary winding set the auxiliary winding ahead of the main winding, and the direction of rotation is reversed [1].

From this concept, the board supplies sinewaves from two outputs to the two relays set in an ‘H-bridge like’ manner. In the mechanism for the forward rotation, using the split-phase motor, the board supply sinewaves from one output while the other is set to the ground and vice versa for the reverse direction rotation. Moreover, for the speed control of the synchronous motor, the frequency of the sine wave input to the relay is varied. The higher the frequency, the higher the speed of rotation and vice versa.

5.2 For DC motor

Reversing the direction of a DC motor can simply be by reversing the motor terminal connection. As such, relays are set up in an ‘H-bridge like’ mechanism. Making one terminal active rotates the motor in a clockwise direction and closing it and making the other active reverses the direction of the motor. Moreover, the speed of the motor in either direction can be achieved by controlling the amount of power to the motor, achieved through the pulse width modulation.

Chapter 6: Conclusion, Limitations, and Future Work

6.1 Conclusion

The wireless control of electric motors (ac and dc) is implemented to improve the intractability of the machine at a distance, and significantly enhance effective and efficient control. The system helps the user control multiple machines more easily and not get exposed to the dangers that are associated with the motors at times of faults. Bluetooth is one method of wireless control that cannot be easily interrupted once the connection is made. As such, there is an assurance there is no breach or interruption in the system once a connection is successful. Therefore, there is total control of the machines with no unexpected circumstances in operation.

Up until this point in the project, a major part of the project has certainly been completed. The wireless control of the DC motor (the demo) and the concept of the process of an AC motor control via Bluetooth and the design and research details about AC motor control via Bluetooth are completed. However, what is yet to be to be done is the implementation of the system with a high voltage DC motor control, and the entire practical implementation of AC motor control via Bluetooth.

6.2 Limitations

First, the entire practical aspect of the ac motor control was interrupted due to COVID-19 pandemic, and limited resources available at disposal at the time of completion. As such, the process of the ac motor control practical section was not realized.

Second, the embedded C program that enables the microcontroller to control the motor via relays is implemented using polling serial communication. As such, there's a slight delay in the system's response to commands since there is a 'function calling from above' whenever there

is a command to the module. Moreover, when a motor is running at a speed, say a 20% duty cycle, and the user wants to increase the speed; say 80% duty cycle, the motor start over. That is, it stops and accelerates from zero value to that speed value commanded. This is one disadvantage realized with the use of a polling method of serial communication.

6.3 Future Works

The system explains the implementation of electric motor control using Bluetooth. Bluetooth, however, is limited in the range of control; that is, up to about 100 meters of control. As such, to improve effective control in long ranges, a better method that uses Wi-Fi control can be used which is of greater proximity than Bluetooth can be used.

Moreover, to realize a better and efficient response to the commands, it is advisable to use interrupts as the choice of the serial communications over polling to avoid the problems associated with the polling method mentioned in the limitations above in section 6.2.

Finally, a hardware approach of controlling an ac motor is highly recommended to realize a real-life practical implementation of ac motor control via Bluetooth.

References

[1] S. J. Chapman, "Electric Machinery Fundamentals," New York, McGill-Hill Companies Inc., 2012, pp. 152-187.

[2] A. R. Daniels, Introduction to Electric Machines, New York: Macmillan, 1976.

[3] X. F. P. J. A. A. Rediana Koc¸i, "Classification of Changes in API Evolution," *IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC),* vol. 23rd, pp. 243-249, 2019.

[4] A. SILL, "When to Use Standards Based APIs (part1 & 2)," *STANDARDS NOW,* vol. I & II, pp. 76-80, 2015.

[5] S. L. P. a. S. S. Paul Raj Devadoss, "Managing Knowledge Integration in a National Health-Care Crisis: Lessons Learned From Combating SARS in Singapore," *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE,* vol. 9, no. 2, pp. 266-275, 2005.

[6] A. G. Dean, Embedded Systems Fundamentals with ARM Cortex-M based Microcontrollers: A Practical Approach, ARM Education Media, 2017, 2017.

[7] E. T. Team, "ElectronicsTutorials," April 2019. [Online]. Available: https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html. [Accessed 19 December 2019].

[8] R. B. Daniel Sarb, "Wireless Motor Control in Automotive Industry," *IEEE Explore,* 2016.

[9] A. B. T. V. F. B. R. Kouki, "IoT Predictive Application for DC Motor Control using Radio Frequency links," *IEEE explore,* 2017.

[10] J. R. V. K. S. Ravi Kumar, "Microprocessor Based Closed Loop Speed Control of DC Motor using PWM," *IEEE Xplore,* 2015.

[11] L. J. J. F. Jia Yunfei, "Design of DC Motor Wireless Controller Based on MCU MSP430," *IEEE Xplore,* 2013.

[12] M. J. D. G. Mrs. A. D. Kadage, "Wireless Control System for Agricultural Motor," *Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09,* 2009.

[13] R. M. P. P. G. W. R. C. Mayuresh Shimpi, "Speed Control of DC Motor Using Bluetooth Devices," *International Journal for Research in Applied Science & Engineering Technology (IJRASET),* March 2018.

[14] G. G. V. K. T. P. C. Arindam Bhattacharjee, "Speed Control of BLDC Motor through mobile Application via Secured Bluetooth," *IEEE xplore,* 2017.

[15] J.-I.-P. Ulugbek Umirov, "Efficient Use of Bluetooth in Networked Control Systems," *12th International Conference on Control, Automation and Systems ,* 17-21 October 2012.

[16] "Electronicsforu.com," ElectronicsForu.com, 23 July 2019. [Online]. Available: https://www.electronicsforu.com/electronics-projects/hardware-diy/speed-control-dc-motor-using-pwm. [Accessed 11 February 2020].

[17] T. Agarwal, "Edgefxkits," Edgefx Technologies Pvt Ltd., 17 August 2017. [Online]. Available: www.edgefxkits.com. [Accessed 09 May 2020].

[18] I. P. &. E. Society, "IEEE Guide for the Application,," *IEEE STANDARDS ASSOCIATIONS,* pp. 1-71, 19 August 2011.

[19] K. L. G. L. B. X. Y. K. Jiefu Zhang, "Research on the Influence of Primary Load," *POWERCON,* no. Paper NO. 201804270000511, pp. 1504-1511, 2018.

[20] K. G. S. M. Nidhi Verma, "Implementation Of Solid State Relays For Power," *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH ,* p. 6, 2015.

[21] "DC motor direction control using relay circuit," Mechatrofice, 24 March 2017. [Online]. Available: www.mechatrofice.com. [Accessed 07 May 2020].

[22] P. B. Ishtiaque Amin, "Relay Replacement for Brushed DC Motor Drive in," *Texas Instruments,* pp. 1-13, 2016.

Appendix

// mbed DC motor control

```
#include "mbed.h"

Serial pc(USBTX, USBRX); //tx, rx
Serial device(PTA2,PTA1);

PwmOut DC_motor1(PTD4);  // other pwm ports= PTA12, PTA4, PTA5
PwmOut DC_motor2(PTA12);

DigitalOut led1(LED_RED);
DigitalOut led2(LED_GREEN);
DigitalOut led3(LED_BLUE);


//DigitalOut pc_activity(LED3);
unsigned char receivedchar;
void send_bytes(uint8_t len, uint8_t data){

   device.putc(len);
```

```
    while(len>0){
        device.putc(data);
        len--;
        }
    }

int main()
{

    unsigned char rx;
    device.baud(9600);
    while(1){
        if(device.readable()){
            rx=device.getc();
           // pc.printf("%c\n, %c", rx, rx);
            switch(rx){
                case 'r': // 2/5 speed forward
                    led1=0;
                    led2=1;
                    led3=1;
                    DC_motor1.period_ms(20);
                    DC_motor1.pulsewidth_ms(8);
                    DC_motor2.period_ms(0);
                    DC_motor2.pulsewidth_ms(0);
                    break;

                case 'm': // 2/5 speed reverse
                    led1=0;
                    led2=1;
                    led3=0;
```

```
    DC_motor1.period_ms(0);
    DC_motor1.pulsewidth_ms(0);

    DC_motor2.period_ms(20);
    DC_motor2.pulsewidth_ms(8);
    break;
case 'h': //3/4 speed for forward

    led1=0;
    led2=1;
    led3=1;
    DC_motor1.period_ms(20);
    DC_motor1.pulsewidth_ms(16);
    DC_motor2.period_ms(0);
    DC_motor2.pulsewidth_ms(0);
    break;

case 'a': // 3/4 speed reverse

    led1=0;
    led2=1;
    led3=0;
    DC_motor1.period_ms(0);
    DC_motor1.pulsewidth_ms(0);
    DC_motor2.period_ms(20);
    DC_motor2.pulsewidth_ms(16);
    break;

case 'b': // accelerate to full speed for forward
    led1=1;
```

```
    led2=1;
    led3=0;

    for(int i=0; i<=20; i++){
       DC_motor1.period_ms(20);
       DC_motor2.period_ms(0);
       DC_motor2.pulsewidth_ms(0);
       DC_motor1.pulsewidth_ms(i);
       wait(0.3);
       }
    break;
case 'y': // accelerate to full speed for reverse
    led1=0;
    led2=0;
    led3=1;

    for(int i=0; i<=20; i++){
       DC_motor1.period_ms(0);
       DC_motor1.pulsewidth_ms(0);
       DC_motor2.period_ms(20);
       DC_motor2.pulsewidth_ms(i);
       wait(0.3);
          }
       break;
case 's':  //stop

    led1=1;
    led2=0;
    led3=0;
     DC_motor1.period_ms(0);
```

```
                DC_motor1.pulsewidth_ms(0);
                DC_motor2.period_ms(0);
                DC_motor2.pulsewidth_ms(0);
                break;

            default:
               break;

            }
         }
      }
}
```

// Keil uVision DC motor control

```
//unfxied as yet

#include <MKL25Z4.H>
#include <stdio.h>

#define PWM_PERIOD (48000)
#define FULL_ON (PWM_PERIOD-1)
#define FULL_OFF (0)

#define RED_LED_POS (18)      // on port B
#define GREEN_LED_POS (19)    // on port B
#define BLUE_LED_POS (1)      // on port D

// connected to motor enabler on the driver
#define DC_Motor_Enable (1)    //on port D1
#define AC_Motor_Enable (2)    //on port D2
```

```
// determine the direction of the motor
// DC Motor PWMs
#define DC_motor1 (11)   //on port B11
#define DC_motor2 (10)   //on port B10



#define MASK(x) (1UL << (x))

#define USE_UART_INTERRUPTS (0)     // 0 for polled UART communications, 1 for interrupt-driven
#define UART_OVERSAMPLE_RATE  (16)  //a divisor
#define SYS_CLOCK   (48e6)
volatile char rxChar;

// clockwise rotation
void DC_Forward(){
        PTB->PCOR=MASK(DC_motor1);
        PTB->PSOR=MASK(DC_motor2);



}

//Function for backwards
void DC_Reverse(){
        PTB->PCOR=MASK(DC_motor2);
        PTB->PSOR=MASK(DC_motor1);



}
```

```
//Function for stop
void Stop(){
	PTB->PCOR=MASK(DC_motor1);
	PTB->PCOR=MASK(DC_motor2);



}


void Init_RGB_LEDs(void) {
	// Enable clock to ports B and D
	SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTD_MASK;;

	// Make 3 pins GPIO
	PORTB->PCR[RED_LED_POS] &= ~PORT_PCR_MUX_MASK;
	PORTB->PCR[RED_LED_POS] |= PORT_PCR_MUX(1);
	PORTB->PCR[GREEN_LED_POS] &= ~PORT_PCR_MUX_MASK;
	PORTB->PCR[GREEN_LED_POS] |= PORT_PCR_MUX(1);
	PORTD->PCR[BLUE_LED_POS] &= ~PORT_PCR_MUX_MASK;
	PORTD->PCR[BLUE_LED_POS] |= PORT_PCR_MUX(1);

	// Set ports to outputs
	PTB->PDDR |= MASK(RED_LED_POS) | MASK(GREEN_LED_POS);
	PTD->PDDR |= MASK(BLUE_LED_POS);

	PTB->PSOR |= MASK(RED_LED_POS) | MASK(GREEN_LED_POS);
	PTD->PSOR |= MASK(BLUE_LED_POS);
}

//for blinking the lights RGB
```

```
void Control_RGB_LEDs(unsigned int red_on, unsigned int green_on, unsigned int blue_on) {
        if (red_on) {
                                                        PTB->PCOR =
MASK(RED_LED_POS);
        } else {
          PTB->PSOR = MASK(RED_LED_POS);
        }
        if (green_on) {
          PTB->PCOR = MASK(GREEN_LED_POS);
        }else {
          PTB->PSOR = MASK(GREEN_LED_POS);
        }
        if (blue_on) {
          PTD->PCOR = MASK(BLUE_LED_POS);
        }else {
          PTD->PSOR = MASK(BLUE_LED_POS);
        }
}

// Code listing 8.8, p. 231
void Init_UART0(uint32_t baud_rate) {
        uint16_t sbr;       //structs
        uint8_t temp;

        // Enable clock gating for UART0 and Port A
        SIM->SCGC4 |= SIM_SCGC4_UART0_MASK;
        SIM->SCGC5 |= SIM_SCGC5_PORTA_MASK;

        // Make sure transmitter and receiver are disabled before init
        UART0->C2 &= ~UART0_C2_TE_MASK & ~UART0_C2_RE_MASK;
```

```
        // Set UART clock to 48 MHz clock
        SIM->SOPT2 |= SIM_SOPT2_UART0SRC(1);
        SIM->SOPT2 |= SIM_SOPT2_PLLFLLSEL_MASK;

        // Set pins to UART0 Rx and Tx
        PORTA->PCR[1] = PORT_PCR_ISF_MASK | PORT_PCR_MUX(2); // Rx
        PORTA->PCR[2] = PORT_PCR_ISF_MASK | PORT_PCR_MUX(2); // Tx

        // Set baud rate and oversampling ratio ( a divisor)
        sbr = (uint16_t)((SYS_CLOCK)/(baud_rate *
UART_OVERSAMPLE_RATE));//48M/(baudRate *oversample rate of 16) //i think samples 16
bit ati
        UART0->BDH &= ~UART0_BDH_SBR_MASK;
        UART0->BDH |= UART0_BDH_SBR(sbr>>8);//higher 4 bits of 12 bit no
        UART0->BDL = UART0_BDL_SBR(sbr);  //lower 8bits of 12 bit no
        UART0->C4 |= UART0_C4_OSR(UART_OVERSAMPLE_RATE-1); //specify as N-
1

// Diasble interrupts for RX active edge and LIN break detect, select one stop bit //anza hapa
//UART0->BDH |=UART0_BDH_RXEDGIE(0) | UART0_BDH_SBNS(0) |
UART0_BDH_LBKDIE(0);
// Don't enable loopback mode, use 8 data bit mode, don't use parity
//UART0->C1 = UART0_C1_LOOPS(0) | UART0_C1_M(0) | UART0_C1_PE(0);   // humu pia
// Dont invert transmit data, do enable interrupts for errors
//UART0->C3 = UART0_C3_TXINV(0) | UART0_C3_ORIE(1) | UART0_C3_NEIE(1) |
UART0_C3_FEIE(1) | UART0_C3_PEIE(1); //humu pia

 // clear error flags // use this or one below
 //UART0->S1 = UART0_S1_OR(1) | UART0_S1_NF(1) | UART0_S1_FE(1) |
UART0_S1_PF(1);  //haikuwa kwa chaBluetooth //humu pia
```

```
        // Clear error flags
                                                    //OR- rx buff not read bf new data was received
                                                    //NF - noise in receiver, FE - framing error, PF - Parity error
 UART0->S1 |= UART0_S1_OR_MASK | UART0_S1_NF_MASK | UART0_S1_FE_MASK | UART0_S1_PF_MASK;


 // send LSB first, d not invert received data // humu tuaona
 //UART0->S2 = UART0_S2_MSBF(0) | UART0_S2_RXINV(0);   // haikuwa kwa chaBluetooth


                                    // Enable UART receiver and transmitter
                                                    UART0->C2 |= UART0_C2_RE(1) | UART0_C2_TE(1);
                                                    // Clear the UART RDRF flag
                                                    temp = UART0->D;
                                                    UART0->S1 &= ~UART0_S1_RDRF_MASK;
}



// Code listing 8.9, p. 233
void UART0_Transmit_Poll(uint8_t data) {
        while (!(UART0->S1 & UART0_S1_TDRE_MASK));
        UART0->D = data;
}

uint8_t UART0_Receive_Poll(void) {
        while (!(UART0->S1 & UART0_S1_RDRF_MASK));
        return UART0->D;
}
```

```
//void TPM0_IRQHandler();

void PWM_Gen_TPM(uint16_t period){
        // Enable clock to port D
        SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK;
        // Enable clock to port B
        SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK;

        // Blue FTM0_CH1, Mux Alt 4
        // Set pin to FTMs
        PORTD->PCR[DC_Motor_Enable] &= ~PORT_PCR_MUX_MASK;
        PORTD->PCR[DC_Motor_Enable] |= PORT_PCR_MUX(4);

        // Blue FTM0_CH2, Mux Alt 4
        // Set pin to FTMs
        PORTD->PCR[AC_Motor_Enable] &= ~PORT_PCR_MUX_MASK;
        PORTD->PCR[AC_Motor_Enable] |= PORT_PCR_MUX(4);

        // For first input of DC_Motor1
        PORTB->PCR[DC_motor1] &= ~PORT_PCR_MUX_MASK;
        PORTB->PCR[DC_motor1] |= (1UL<<8);  //setup to be GPIO

        //for the second input of DC_motor2
        PORTB->PCR[DC_motor2] &= ~PORT_PCR_MUX_MASK;
        PORTB->PCR[DC_motor2] |= (1UL<<8);  //setup to be GPIO
```

```
//*****************************************************************************
*******
          //Red LED
           PORTB->PCR[RED_LED_POS ] &= ~PORT_PCR_MUX_MASK; //Clear mux
           PORTB->PCR[RED_LED_POS ] |= (1UL<<8);  //setup to be GPIO

           //Green LED
           PORTB->PCR[GREEN_LED_POS ] &= ~PORT_PCR_MUX_MASK; //Clear mux
           PORTB->PCR[GREEN_LED_POS ] |= (1UL<<8);  //setup to be GPIO

           //Blue LED
           PORTD->PCR[BLUE_LED_POS ] &= ~PORT_PCR_MUX_MASK; //Clear mux
           PORTD->PCR[BLUE_LED_POS ] |= (1UL<<8);  //setup to be GPIO


//*****************************************************************************
********

           // Set ports to output
           //PTD->PDDR|= MASK(Motor_Enable);
           PTB->PDDR |= MASK(DC_motor1 );
           PTB->PDDR |= MASK(DC_motor2 );\
           PTB->PDDR |= MASK(RED_LED_POS);
           PTB->PDDR |= MASK(GREEN_LED_POS);
           PTD->PDDR |= MASK(BLUE_LED_POS);

           //Following is unneeded, but puts the leds in a known state
           PTC->PSOR =MASK(DC_motor1);
           PTB->PSOR =MASK(DC_motor2); //BETTER to use an OR operation,
```

```
	PTB->PSOR =MASK(RED_LED_POS);
	PTB->PSOR =MASK(GREEN_LED_POS);
	PTD->PSOR =MASK(BLUE_LED_POS);

	// Configure TPM
	SIM->SCGC6 |= SIM_SCGC6_TPM0_MASK;
	//set clock source for tpm: 48 MHz
	SIM->SOPT2 |= (SIM_SOPT2_TPMSRC(1) | SIM_SOPT2_PLLFLLSEL_MASK);
	//load the counter and mod
	TPM0->MOD = period-1;   //set the period
	//set TPM count direction to up with a divide by 2 prescaler
	TPM0->SC =  TPM_SC_PS(1); // prescale to 2
	// Continue operation in debug mode
	TPM0->CONF |= TPM_CONF_DBGMODE(3);
	// Set channel 1 to edge-aligned low-true PWM
	TPM0->CONTROLS[1].CnSC = TPM_CnSC_MSB_MASK |
TPM_CnSC_ELSA_MASK| TPM_CnSC_ELSB_MASK; //both rising and falling edges
	// Set channel 2 to edge-aligned low-true PWM
	TPM0->CONTROLS[2].CnSC = TPM_CnSC_MSB_MASK |
TPM_CnSC_ELSA_MASK| TPM_CnSC_ELSB_MASK;
	// Set initial duty cycle
	TPM0->CONTROLS[1].CnV = 0;
	// Set initial duty cycle
	TPM0->CONTROLS[2].CnV = 0;
	// Start TPM
	TPM0->SC |= TPM_SC_CMOD(1)|TPM_SC_TOIE_MASK;  //Timer overflow
interrupt enable

	//Enable Interrupts in NVIC  // set up interrupt to NVIC
	NVIC_SetPriority(TPM0_IRQn, 3);
	NVIC_ClearPendingIRQ(TPM0_IRQn);
```

```
            NVIC_EnableIRQ(TPM0_IRQn);
}



void full_speed(){  //Speed_UpG()

            __enable_irq();
           // uint16_t i=0;
            volatile int32_t delay;

            if ((TPM0->SC & TPM_CnSC_ELSB_MASK)){
                        //Control_RGB_LEDs(1,0,0);
                        }
                                                TPM0->CONTROLS[1].CnV =
PWM_PERIOD-1;
                                                TPM0->CONTROLS[2].CnV =
PWM_PERIOD-1;
            //Control_RGB_LEDs(1,0,0);
            for (delay=0; delay<1000; delay++);
}


void half_speed(){ //Speed_DownG()

            __enable_irq();
           // uint16_t i=0;
            volatile int32_t delay;
```

```
        if ((TPM0->SC & TPM_CnSC_ELSB_MASK)){

                                                        }
                                                        TPM0->CONTROLS[1].CnV =
PWM_PERIOD/2;
                                                        TPM0->CONTROLS[2].CnV =
PWM_PERIOD/2;



}
 void one_fourth_speed(){

         __enable_irq();
        // uint16_t i=0;
         volatile int32_t delay;

         if ((TPM0->SC & TPM_CnSC_ELSB_MASK)){

                  }
                                                        TPM0->CONTROLS[1].CnV =
PWM_PERIOD/4;
                                                        TPM0->CONTROLS[2].CnV =
PWM_PERIOD/4;


              }
void Speed_Up(){
         __enable_irq();
         uint16_t i=0;
         volatile int32_t delay;
```

```
        if ((TPM0->SC & TPM_CnSC_ELSB_MASK)){

                }
                                                        for (i=(PWM_PERIOD)/8;
i>PWM_PERIOD-1; i++) {
                                                            TPM0-
>CONTROLS[1].CnV = i;
                                                            TPM0-
>CONTROLS[2].CnV = i;

  }
}

void Speed_Down(){

        __enable_irq();
        uint16_t i=0;
        volatile int32_t delay;

        if ((TPM0->SC & TPM_CnSC_ELSB_MASK)){

                                                        }
                                                        for (i=PWM_PERIOD-1;
i>(PWM_PERIOD)/8; i--) {
                                                            TPM0-
>CONTROLS[1].CnV = i;
                                                            TPM0-
>CONTROLS[2].CnV = i;

                                                        }
}
```

```
void lightLED(void){      //motor_control() its.'inline void lightLED(void){
            switch (rxChar){
                                                                    case 'y': full_speed();
                                                                    break;

              case 'h': half_speed();
              break;
              case 'q': one_fourth_speed();
              break;
                                                                    case 'm': DC_Reverse();
                                                                    break;
                                                                    case 'r': DC_Forward();
                                                                    break;

                  case 'b': Speed_Up();
                                                                    break;
                                                                    case 'd': Speed_Down();
                                                                    break;
                                                                    case 's': Stop();
                                                                    break;
              default:
                                                                          Stop();
            }
}


// ---------------------------------------------------------------------
//        MAIN
//---------------------------------------------------------------------
int main (void) {
```

```
        __enable_irq();

        PWM_Gen_TPM(PWM_PERIOD);
                                                uint8_t c;

        Init_UART0(9600);//115200, the bluetooth HC-06 pin is 1234, buad is 9600
        lightLED();

// Polling version of code
// Send_String_Poll("Ashesi Embedded!");
Control_RGB_LEDs(0, 0, 1);

    rxChar='_';//initially,

// Code listing 8.9, p. 233
          while (1) {
       PWM_Gen_TPM(PWM_PERIOD);
                    c = UART0_Receive_Poll();

                    rxChar =c;
                    UART0_Transmit_Poll(c+1);
                    //Init_UART0(9600);

                     //lightLED();
     //DC_Forward();
                }
}
```

// mbed AC motor control

```
#include "mbed.h"

Serial pc(USBTX, USBRX); //tx, rx
Serial device(PTA2,PTA1);

AnalogOut Aout(PTE30);
DigitalOut Output(PTD6);

float x;
unsigned int i,j,out;
int array[100];

DigitalOut led1(LED_RED);
DigitalOut led2(LED_GREEN);
DigitalOut led3(LED_BLUE);


//DigitalOut pc_activity(LED3);
unsigned char receivedchar;
void send_bytes(uint8_t len, uint8_t data){

   device.putc(len);
   while(len>0){
      device.putc(data);
      len--;
      }
   }

int main()
{
```

```
        x=0;
i=0;
out=0;
unsigned char rx;
device.baud(9600);
while(1){
   if(device.readable()){
       rx=device.getc();
      // pc.printf("%c\n, %c", rx, rx);
       switch(rx){
           case 'r':
               led1=0;
               led2=1;
               led3=1;
               for(i=0;i<99;i++) {
   Output=out;
   x=16*2047.0*(1+sin(2*3.142*i/50));// regulate the divider to change the frequency
   array[i]=x;
}

while(1){
   i=0;
   for(i=0;i<99;i++){
   j=array[i];
   Aout.write_u16(j);
   }
   }
     break;

           case 'm':
```

```
            led1=0;
            led2=1;
            led3=0;
            for(i=0;i<99;i++) {
   Output=out;
   x=16*2047.0*(1+sin(2*3.142*i/100);//regulate the divider to change the frequency
   array[i]=x;
}

while(1){
   i=0;
   for(i=0;i<99;i++){
   j=array[i];
   Aout.write_u16(j);
   }
   }
            break;
        case 'h': //reverse

            led1=0;
            led2=1;
            led3=1;
            for(i=0;i<99;i++) {
   Output=out;
   x=16*2047.0*(1+sin(2*3.142*i/50));// regulate the divider to change the frequency
   array[i]=x;
}

while(1){
   i=0;
```

```
   for(i=0;i<99;i++){
   j=array[i];
   output.write_16(j);
   }
   }
            break;

        case 'a': //  speed reverse

            led1=0;
            led2=1;
            led3=0;
            for(i=0;i<99;i++) {
   Output=out;
   x=16*2047.0*(1+sin(2*3.142*i/50));// regulate the divider to change the frequency
   array[i]=x;
}

while(1){
   i=0;
   for(i=0;i<99;i++){
   j=array[i];
   output.write_16(j);
   }
   }
            break;
         }
      }
   }
```

```
}


// ac motor control
#include "mbed.h"

Serial pc(USBTX, USBRX); //tx, rx
Serial device(PTA2,PTA1);

AnalogOut Aout(PTE30);
DigitalOut Output(PTD6);

float x;
unsigned int i,j,out;
int array[100];

DigitalOut led1(LED_RED);
DigitalOut led2(LED_GREEN);
DigitalOut led3(LED_BLUE);


//DigitalOut pc_activity(LED3);
unsigned char receivedchar;
void send_bytes(uint8_t len, uint8_t data){

   device.putc(len);
   while(len>0){
      device.putc(data);
      len--;
      }
```

```
    }

int main()
{
    x=0;
    i=0;
    out=0;
    unsigned char rx;
    device.baud(9600);
    while(1){
        if(device.readable()){
            rx=device.getc();
           // pc.printf("%c\n, %c", rx, rx);
            switch(rx){
                case 'r':
                    led1=0;
                    led2=1;
                    led3=1;
                    for(i=0;i<99;i++) {
        Output=out;
        x=16*2047.0*(1+sin(2*3.142*i/50));// regulate the divider to change the frequency
        array[i]=x;
    }

    while(1){
        i=0;
        for(i=0;i<99;i++){
        j=array[i];
        Aout.write_u16(j);
        }
```

```
    }
      //break;

          case 'm':
             led1=0;
             led2=1;
             led3=0;
             for(i=0;i<99;i++) {
                Output=out;
               x=16*2047.0*(1+sin(2*3.142*i/100));//regulate the divider to change the
frequency
               array[i]=x;
  }

  while(1){
    i=0;
    for(i=0;i<99;i++){
    j=array[i];
    Aout.write_u16(j);
    }
    }
             break;
          case 'h': //reverse

             led1=0;
             led2=1;
             led3=1;
             for(i=0;i<99;i++) {
    Output=out;
    x=16*2047.0*(1+sin(2*3.142*i/50));// regulate the divider to change the frequency
```

```
    array[i]=x;
}

while(1){
    i=0;
    for(i=0;i<99;i++){
    j=array[i];
    Output.write(j);
    }
    }
                break;

            case 'a': //  speed reverse

                led1=0;
                led2=1;
                led3=0;
                for(i=0;i<99;i++) {
    Output=out;
    x=16*2047.0*(1+sin(2*3.142*i/50));// regulate the divider to change the frequency
    array[i]=x;
}

while(1){
    i=0;
    for(i=0;i<99;i++){
    j=array[i];
    Output.write(j);
    }
    }
```

```
                break;
            }
        }
    }

}
```