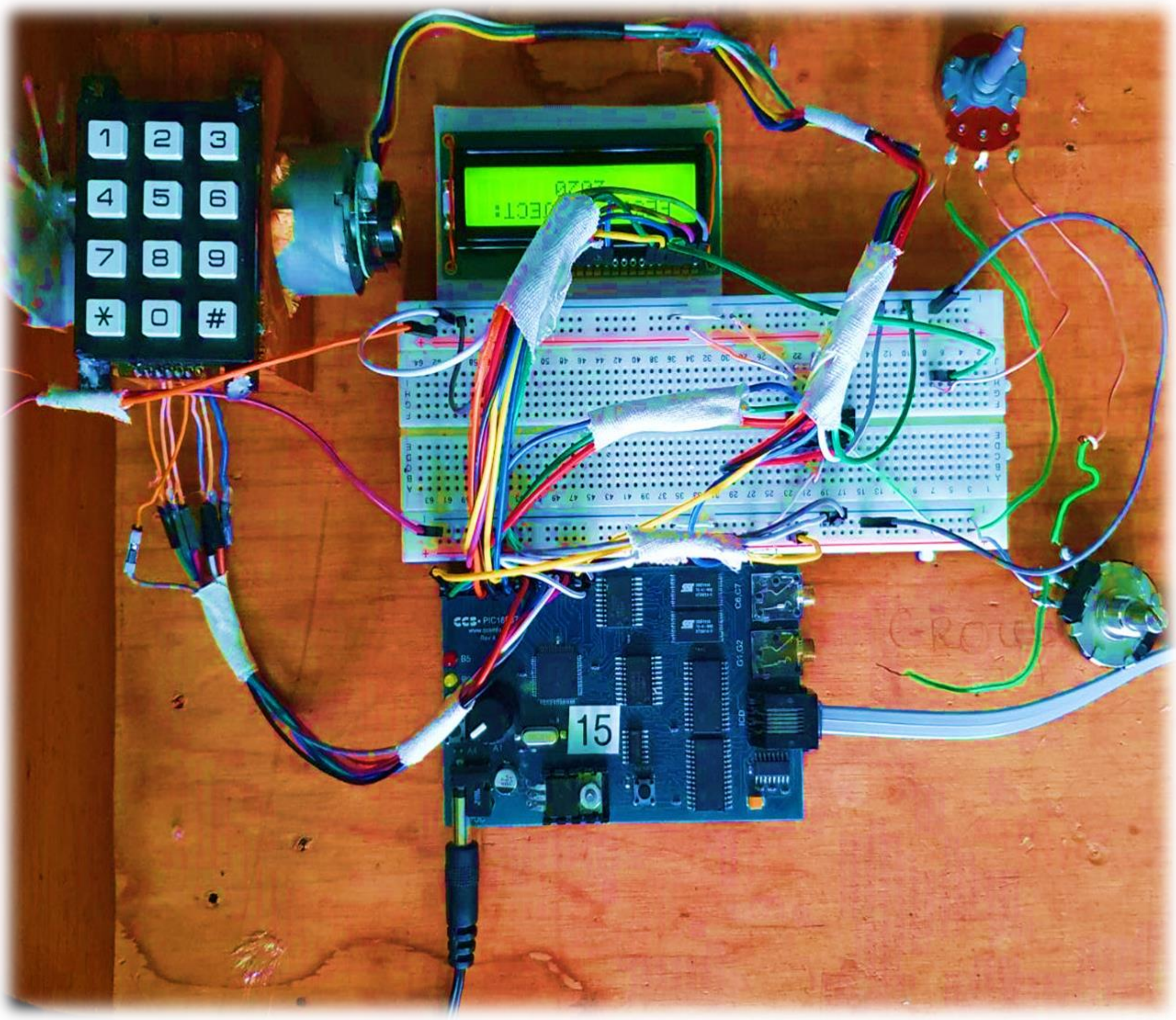
FACULTY OF SCIENCE TECHNOLOGY AND ENVIRONMENT

School of Engineering and Physics

EE313: MICROPROCESSOR APPLICATIONS

PROJECT TOPIC: DC MOTOR SPEED MEASURING AND DISPLAY ON LCD SCREEN



Team Members: **(PID: 9)**

KRISHNEEL RAM SHARMA	S11158558
SUKHDE JOSHI	S11158046
KRISH KUMAR RAJ	S11160036
SHAHEEL KUMAR	S11147454

Table of Contents

ACKNOWLEDGEMENT

With the completion of this project, let’s take a moment to appreciate certain individuals whose assistance and form of directions helped our project to success.

- → Firstly, the God Almighty for his continuous guidance and providing us with the strength to complete our project even in this unfortunate crisis.
- → Secondly, to our Course Coordinator Dr. Utkal Mehta, for guiding the group throughout and for his expertise suggestions and resolutions when the group were facing issues during hardware simulation.
- → Finally, Mr. Rohit - for providing the parts needed to complete the project.

INTRODUCTION

In today's networks of electric systems, an increasingly requirement of higher performance and reliability in the drive systems are being demanded by industrial uses and traction systems. The DC motors fit well with this requirement as the high efficiency and large power density function creates popularity for wide range of applications in the industry.

Moving on, there are 3 types of motors that can be interfaced using microcontrollers. The first one being the DC motor itself. DC motors are readily available as their uses range from household items to industrial applications. Most DC motors operate on 5 – 12V. The second type of motor is known as the stepper motor. This motor has several electromagnetic coils which can be energized and DE energized in a sequence to see the turning effect. The degree to which the motor turns is known as the *step angle*. Stepper motors provide excellent position control and mostly used in printers and scanners. Stepper motors are often compared servo motors, which is the third type of motor. Servo motors are used in devices that require a high precision control of position, velocity and acceleration. They are found in elevators, radio-controlled aircrafts and operating grippers. However, the main interest of this project lies in DC motors. [1]

To understand the working principle of a DC motor, we must first familiarize ourselves with basic electromagnetism laws. In particularly, Lorentz law which states that "*the current carrying conductor placed in a magnetic field will experience a force*". This force induced is proportional to the length (*L*) of the conductor, the amount of current (*I*) passing through, and the magnitude of the magnetic field (*B*) [2]. Additionally, the direction of force can be known by using Fleming's left hand motor rule, which is shown in the figure below.

Though the question which is mainly asked, "Why is there a need for speed control for DC motor". Generally, industries wiling to utilize a DC motor for their applications devise different specifications then others, this may involve shaft course at different speeds and diverse variable power, looking at line following robots where the speed of robot is reduced and increased depending on the complexity of the line. Also observing an electric Drill where the speed control plays a vital role to take precautions for damage to the bit when looking at the surface intended to be drilled. Altering the speed in synchronous and induction motors are simply done by changing the input voltage frequency, this is not the case for DC motor as the examples suggested DC motors utilize speed control mechanism to alter the speed when required. Thus, this project will elaborate what this control mechanism and how is it functioned to set a desired speed.

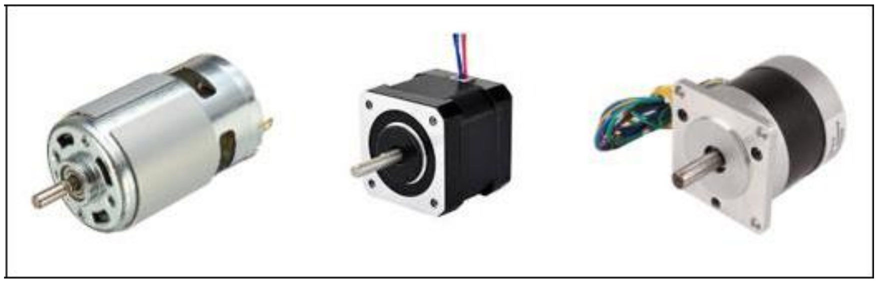


***Figure 1: DC motor, Stepper motor and DC servo motor (from left to right*)*. [3]*

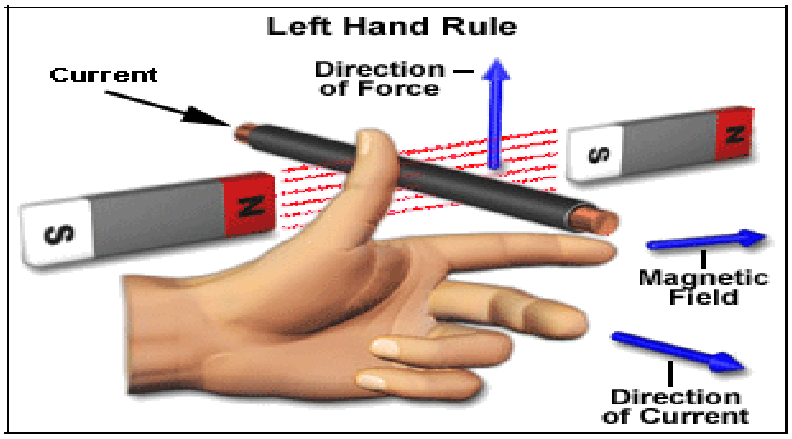


Figure 2: Fleming's left-hand rule for motors. [4]

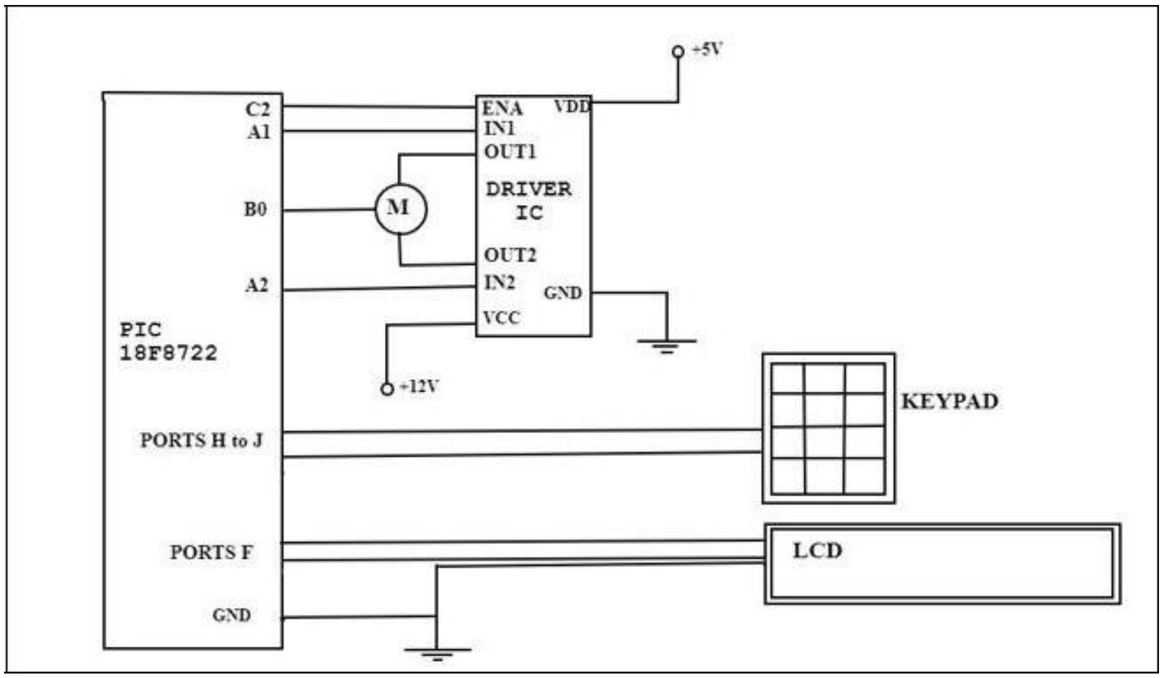


Figure 3: General wiring diagram showing basic connections between circuit elements. [5]

PROBLEM STATEMENT

DC motors commonly used in speed control systems involves heavy power specifications such as rolling mills, double-hulled tankers and automated instruments with high precision. So, controlling the motor speed is crucial to achieving good production. One of the most common ways of running a DC motor is to use PWM signals with respect to the voltage of the motor supply. Also, in the technology era, manual controller is not practical, because it can waste time and cost. Controller operating costs are taken into account from the industrial field. Therefore, it can be suggested that making a computer-based controller which can be also portable, in order to reduce costs and time. At a certain position, the consumer can control their device without having to go to the plant (machine), particularly during industrial implementation. By this the strength of the man can be of and reserved for a more precise and stable machine.

OBJECTIVES

The main objectives of this project were to:

- ➢ Interface a DC motor with PIC via a driver circuit.
- ➢ Calculate current speed of the motor and display on LCD.
- ➢ Set desire speed via user input through the keypad.
- ➢ Encoder, Start and Stop buttons using interrupt features - real time

MATERIALS

Other than basic electronic workbench equipment, the following materials were used to complete this assignment;

- 1 x PIC18F8722 microcontroller;
- PIC-C compiler software;
- 1 x LCD screen;
- 1 x Pololu DC motor;
- 1 x SN754410 driver IC;
- 1 x Breadboard;
- 1 x 218Ω resistor;
- 1 x (4 x 3) Keypad;
- 1 x Potentiometer;
- 12V power supply;
- Connecting wires;

SYSTEM DESCRIPTION

The program begins by displaying the welcome message on the LCD screen. Simultaneously, the motor runs at maximum speed, which is at default. After which the user input is taken. This can be broken down in terms of key presses. Every key press returns an integer which is defined in the *read_keypad()* function.

The first key press value is returned into *digit_1* which is then multiplied by 10 and stored in the *tenth* variable. Moreover, the second key press is stored in *digit_2*. The *user_input* variable then takes the sum of the *tenth* and *digit_2* values and stores this as *cal_speed*. The sum is then converted into duty cycle value by dividing by 80. The obtained quotient is multiplied by 1023. *cal_speed* is updated with the result obtained.

This value needs to be rounded up to the next whole digit. This is taken care by the *CEIL* command. The final duty cycle is stored as *motor_speed*. PWM is configured and the *motor_speed* variable is taken in as the duty cycle. The motor does not yet run at the updated duty cycle value but waits for the third key press. This value is returned to the *digit_3* variable. Generally, any of the 9 keys would be sufficient. Since the task of the microcontroller is to know that the user has entered his duty cycle and wishes to run the motor at that speed. If the entered rpm falls in the range of 40 and 80, the program will accept the user input. Or else, an error message is displayed on the LCD and the user is asked for input again.

If the user input is valid, the motor runs at the specified speed and the speed is displayed. Details of speed reading are as follows:

When the user enters a valid speed, which mean the RPM is in between 40 to 80 the program changes the duty cycle accordingly and changes the motor speed. The formula that was used to calculate the PWM was$\frac{User\ input}{Max\ speed} \times 1023$. PWM is a technique in which we change the pulse width to get the desired duty cycle.

The next step of the program is to check if the motor is actually running on the entered RPM. To check this the sensor pulse is counted for 1 second from one of the encoder output pin and multiplied by 60 to get revolution per minute. One encoder output pin give 16 counts per revolution of the motor shaft. The encoder pin is connected to the external interrupt and timer zero in 8-bit mode was used to create delay for 1 sec. From the calculation, it required 76 overflows to create a 1 sec delay. After counting the pulse from the encoder, the equation $\frac{Sensor\ pulse}{2096} \times 60$ was used to calculate the speed of motor and later this speed was displayed on LCD.

During this motor operation phase, the microcontroller continuously scans the matrix keypad for user input. The value returned from the key press is taken as *digit_4*. If 10 is returned (if user presses * key), the motor goes into stop mode. The instructions executed here are stated in the *stop ()* function. To resume the motor at previous speed user must press the * key again. That is, *digit_5* must return 10.

Note:

- If digit_4 is not 10, the program recognizes this as an attempt to enter a new speed. All the entire process, as stated in paragraph 2 above, is repeated.
- When in stop mode, *digit_5* must strictly return 10. Otherwise, the motor will in stop mode.
- LED indicators are used to indicate different motor states. If the motor is running at user entered duty cycle, the green LED will turn on. When the motor is in stop stage, the red LED will on. Any one LED will be on at a time.

COMPONENTS REQUIRED

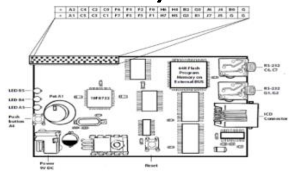
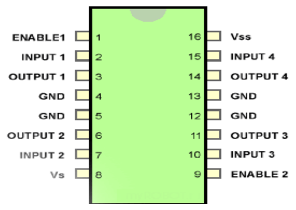
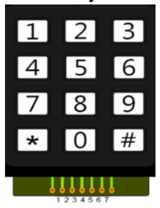
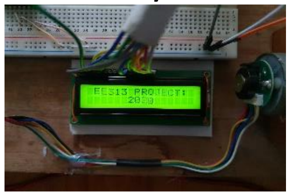
Number	Type
1.) 	➔ **PIC Microcontroller Board (18F8722)**
2.)  ENABLE1 1 · INPUT 1 2 · OUTPUT 1 3 · GND 4 · GND 5 · OUTPUT 2 6 · INPUT 2 7 · Vs 8 · 16 Vss · 15 INPUT 4 · 14 OUTPUT 4 · 13 GND · 12 GND · 11 OUTPUT 3 · 10 INPUT 3 · 9 ENABLE 2	➔ **Driver Circuit (SN75441)**
3.) 	➔ **4x3 Keypad with 7 pin outputs.**
4.) 	➔ **LCD screen**
5.)	➔ **DC Pololu Motor**
6.)	➔ **Potentiometer (26K)**

Table 1: showing components required for the project

METHODOLOGY

The following procedures were followed in order to make this project successful:

- The datasheet of PIC 18F8722 were studied in detail.
- The datasheet for polulu motor, LCD screen, and Driver Circuit (SN75441) were studied in detail.
- A detailed circuit diagram was made with the components mentioned in materials.
- A detailed flowchart of the program was made in order to fully understand its function.
- A program was designed and written for the microcontroller according to the flow chart designed.
- Program the microcontroller and run the system.

CIRCUIT CONNECTION

Interfacing Keypad with Microcontroller

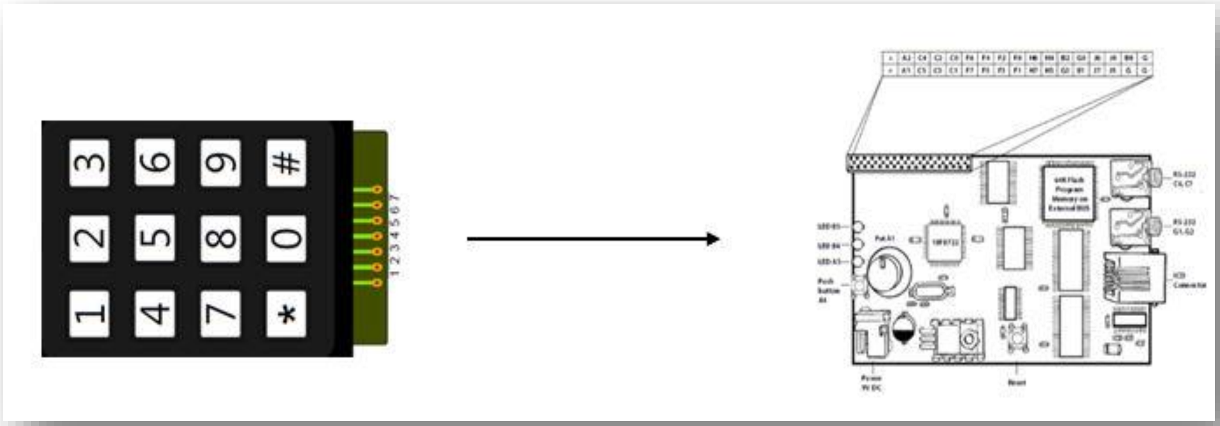


Figure 4: showing the interfacing keypad with microcontroller

KEYPAD	MICROCONTROLLER
PIN 1	**Port J4 (Column 1)**
PIN 2	**Port J5 (Column 2)**
PIN 3	**Port J6 (Column 3)**
PIN 4	**Port H4 (Row 1)**
PIN 5	**Port H5 (Row 2)**
PIN 6	**Port H6 (Row 3)**
PIN 7	**Port H7 (Row 4)**

Table 2: showing the connection between interfacing keypad with microcontroller

The keypad pin 1 is connected to the microcontrollers port J4 (column 1). Keypad pin 2 is connected to the microcontrollers port J5 (column 2). Keypad pin 3 is connected to the microcontrollers port J6 (column 3). Keypad pin 4 is connected to the microcontrollers port H4 (row 1). Keypad pin 5 is connected to the microcontrollers port H5 (row 2). Keypad pin 6 is connected to the microcontrollers port H6 (row 3). Keypad pin 7 is connected to the microcontrollers port H7 (row 4).

Interfacing DC motor with Driver circuit and PIC 18F8722

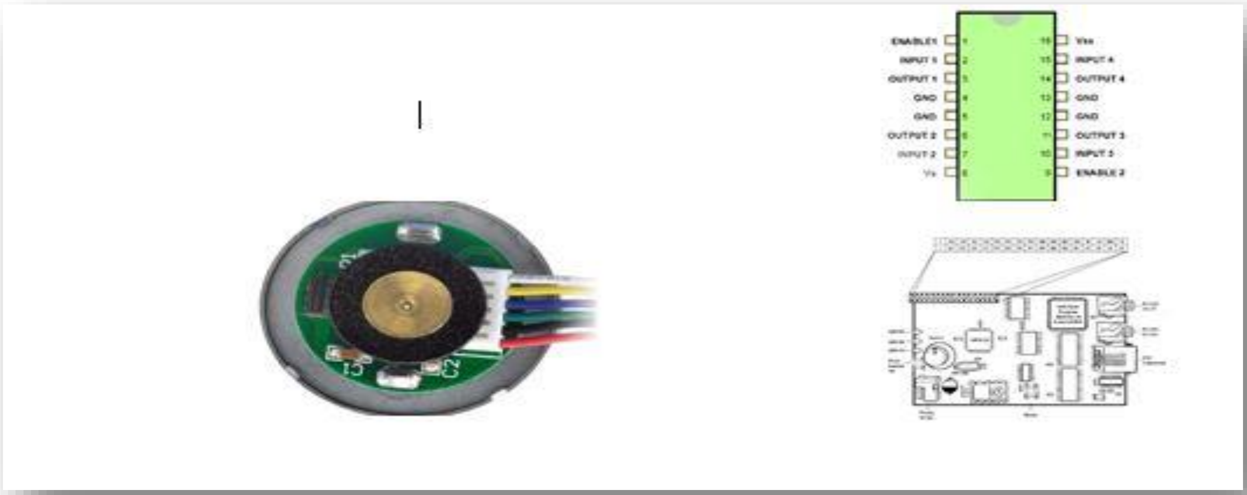


Figure 5: shows the interfacing DC motor with Driver circuit and PIC18F8722.

DC MOTOR	DRIVER CIRCUIT	PIC MICROCONTROLLER 18F8722
RED WIRE	**Input Port 1**	
BLACK WIRE	**Input Port 2**	
GREEN WIRE → CONNECTED TO COMMON GROUND		
BLUE WIRE		**+ 5V supply**
YELLOW WIRE		**To port PIN B0**
WHITE WIRE → UNUSED		
	Output Port 1	**Pin A2**
	Output Port 2	**Pin C4**
	Enable Pin	**Pin C2**
	Vs → Connected to +12V external supply	
	Vss	**+5V supply**
	GND→ Connected to common ground	
	GND→ Connected to common ground	

Table 3: shows the connections between DC motor, Driver circuit and PIC18F8722.

The red wire of the DC motor goes to the Driver circuits input port 1. The black wire of the DC motor goes to the Driver circuits input port 2. The green wire of the DC motor is connected to the common ground. The blue wire of the DC motor goes to the +5 V supply of the PIC18F8722. The white wire of the DC motor is unused. The output port 1 of the Driver circuit goes to Pin A2 of PIC18F8722. The output port 2 of the Driver circuit goes to the pin C4 of the PIC18F8722. The enable pin of the Driver circuit goes to pin C2 of PIC18F8722. The Vs of the Driver circuit is connected to +12 V external supply. The Vss of the Driver circuit is connected to the +5 V of the PIC18F8722. Both the grounds of the Driver circuit are connected to the common ground.

Interfacing LCD screen with PIC 18F8722 and potentiometer

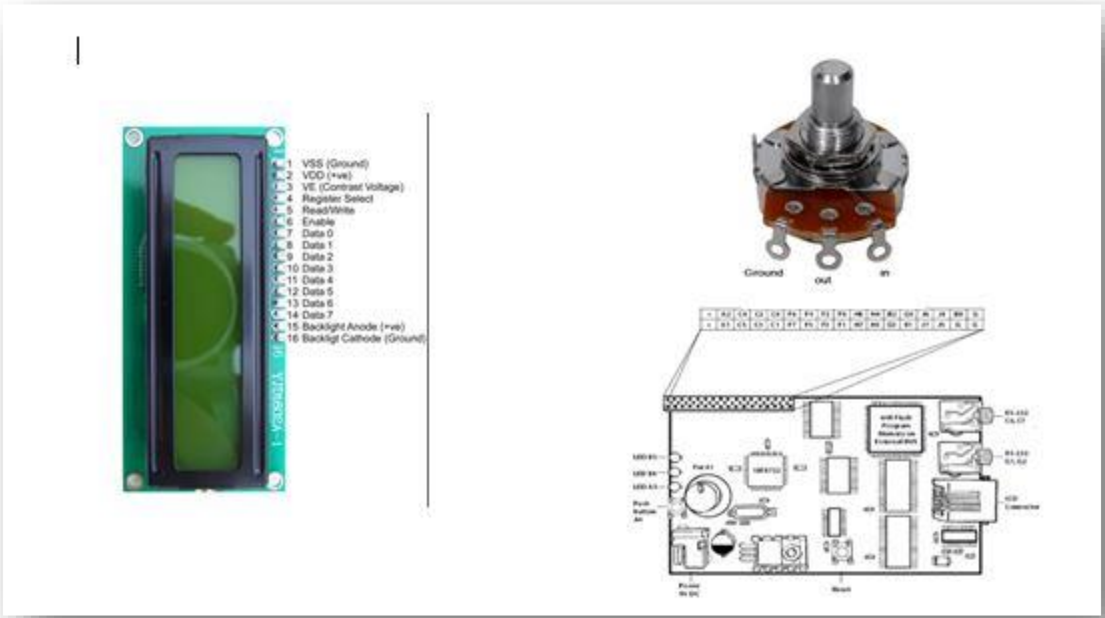


Figure 6: shows the interfacing LCD screen with PIC 18F8722 and potentiometer.

LCD SCREEN	POTENTIOMETER	PC 18F8722
VSS		**Ground**
V_{DD}	**Out Pin**	
V_E		**Ground**
REGISTER SELECT (RS)		**Port F1**
READ/WRITE (RW)		**Port F2**
ENABLE (E)		**Port F0**
DATA PIN 4 - DATA PIN 7 (D4-D7)		**Port F4 – Port F7**
BACKLIGHT ANODE		**+5V supply**
BACKLIGHT CATHODE		**Ground**
	Ground Pin → Grounded	
	In Pin	**+5V supply**

Table 4: shows the connections between LCD screen, Potentiometer, and PIC18F8722.

The LCD screens Vss is connected to the ground to the PIC18F8722. The Vdd of the LCD screen is connected to the out pin of the Potentiometer. The Ve of the LCD screen is connected to the ground of the PIC18F8722. The register select (RS) of the LCD screen goes to the port F1 of the PIC18F8722. The read/write (RW) of the LCD screen goes to port F2 of the PIC18F8722. The enable (E) of the LDC screen goes to port F0 of the PIC18F8722. The data pin D4 and data pin D7 of the LCD screen goes to the port F4 and port F7 of the PIC18F8722. The backlight anode of the LCD screen goes to the +5 V supply of the PIC18F8722. The backlight cathode of the LCD screen goes to the ground of the PIC18F8722. The ground pin of the potentiometer goes to the common ground. The in pin of the potentiometer goes to the +5 V supply of the PIC18F8722.

SOFTWARE FLOW CHART

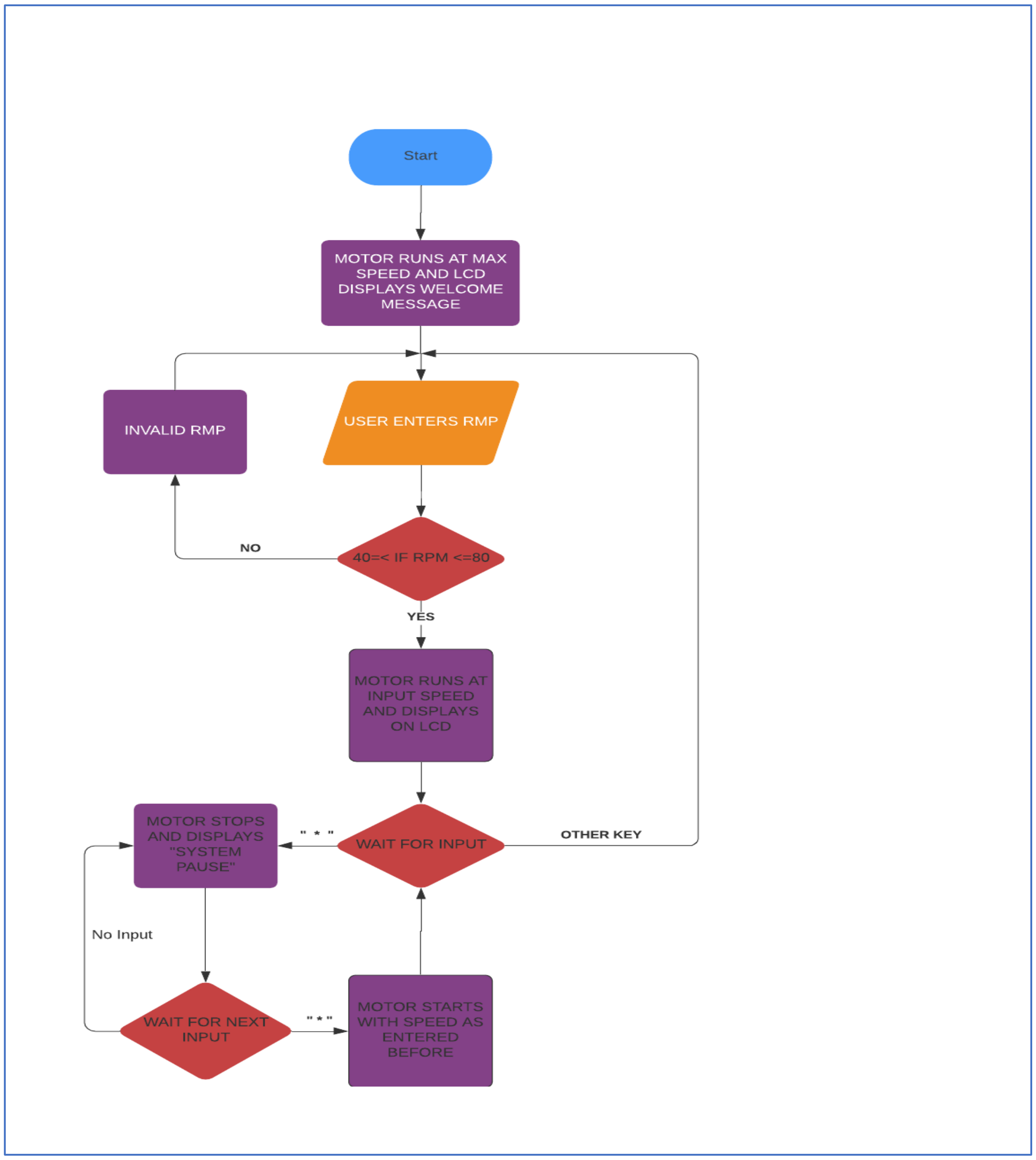

Start
MOTOR RUNS AT MAX SPEED AND LCD DISPLAYS WELCOME MESSAGE
INVALID RMP
USER ENTERS RMP
NO
40=< IF RPM <=80
YES
MOTOR RUNS AT INPUT SPEED AND DISPLAYS ON LCD
MOTOR STOPS AND DISPLAYS "SYSTEM PAUSE"
" * "
WAIT FOR INPUT
OTHER KEY
No Input
WAIT FOR NEXT INPUT
" * "
MOTOR STARTS WITH SPEED AS ENTERED BEFORE

Figure 7: Program flow diagram

CODE

```
//Header File is included in program
#include <18F8722.h>       // Library for PIC microcontroller 18F8722
#device ICD=TRUE
#fuses HS,NOLVP,NOWDT
#use delay(clock=20000000)
#include "lcd1.c"          // inbuilt LCD library
#include <math.h>      // header file for basic mathematics operation

//Definition of global varables
float cal_speed;          //for the calculation of speed
int16 motor_speed; //for PWM duty cycle
int digit_1;               // For user input digit 1
int digit_2;             // For user input digit 2
int digit_3;           // For user input to exit key input program
int digit_4;         // For user input to enter new RPM
int digit_5;       // For stop function
int user_input;     // For saving the entered value by user
int tenth;             // For having the first user value multiplied by 10 and saving it to a variable.
int final;             // Variable used in calculations

//Declarations of Sensor Calculations
unsigned int  loops=0, meas_done=0;

void stop1();

float sensor_pulses;
float dex;

// 3x4 Keypad Function
int read_keypad(void)
{   output_J(0x00);    // clear the ports
    output_H(0x00); //clear the ports

    while(1)  // Loop to infinity
    {
       output_high(PIN_j4);  // first column high for scanning purposes
       if(input(PIN_h4))       // if row 1 goes high then
       {while (input(pin_h4));
          return(1);           // key being pressed is one
       }
       else if(input(PIN_h6))  // if row 2 goes high then
       {while (input(pin_h6));
          return(4);       // key being pressed is four
       }
       else if(input(PIN_h5))  // if row 3 goes high then
       {while (input(pin_h5));
          return(7);                    // key being pressed is seven
       }
       else if(input(PIN_h7))  // if row 4 goes high then
       {while (input(pin_h7));
          return(10);            // key being pressed is ten
       }
       output_j(0x00);     // Clear Port
       output_H(0x00);
   output_high(PIN_j5);  // second column high for scanning purposes
     if(input(PIN_h4))    // if row 1 goes high then
       {while (input(pin_h4));
          return(2);       // key being pressed is two
       }
       else if(input(PIN_h6))   // if row 2 goes high then
            {while (input(pin_h6));
             return(5);           // key being pressed is five
            }
       else if(input(PIN_h5))      // if row 3 goes high then
             {while (input(pin_h5));
               return(8);           // key being pressed is eight
             }
       else if(input(PIN_h7))     // if row 4 goes high then
            {while (input(pin_h7));
              return(0);           // key being pressed is eleven
             }
       output_j(0x00);       // Clear Port
       output_H(0x00);
   output_high(PIN_j6);       // third column high for scanning purposes
       if(input(PIN_h4))      // if row 1 goes high then
       {while (input(pin_h4));
          return(3);     // key being pressed is three
       }
       else if(input(PIN_h6))  // if row 2 goes high then
       {while (input(pin_H6));
          return(6);     // key being pressed is six
       }
       else if(input(PIN_h5))  // if row 3 goes high then
       {while (input(pin_h5));
          return(9);     // key being pressed is nine
       }
       else if(input(PIN_h7))  // if row 4 goes high then
       {while(input(pin_h7));
          return(11);    // key being pressed is twelve
       }
       output_J(0x00);    // Clear Port
       output_H(0x00);
}}

//rotation of motor (clockwise)
void anticlockwise(){
 output_low(PIN_A1);
 output_high(PIN_A2);
}

//Interrrupt function - count pulses
#int_EXT
void EXT_isr()
{  if(!meas_done)
    {sensor_pulses=sensor_pulses+1.08; }         // Sensor pulses (TTL level) must arrive at PIN_B0
} //Due to sensor pulses having a float variable, it is possible to have it increment by decimal
number

//Initializing TIMER 0
#int_TIMER0
void TIMER0_isr()
{ if(loops>0)
    {loops--;}
  else
    {meas_done=true; }
}

//Meauring funtion
void start_measurement()
{
   meas_done=false;
   sensor_pulses=0;
   loops = 76;   // 76 overflows using timer 0 8 bit mode ,required number of overflows to have
an interrupt generated

   enable_interrupts(INT_TIMER0);          // To create a measurement window for the sensor
   enable_interrupts(INT_EXT);               // Generate an interrupt every pulse from the Sensor
   enable_interrupts(GLOBAL);

}

void welcome_message() { // this function displays welcome message
   lcd_init();                         // Initialise the LCD
   lcd_putc('\f');                // clear the screen
   lcd_gotoxy(4,1);         // moves insertion pointer to column 4, row 1 of lcd
   lcd_putc("GREETINGS!"); // displays "GREETINGS!"
   delay_ms(2000);              // waits for 2 seconds
   lcd_init();          // Initialize the LCD
   lcd_putc('\f');        // clears the screen
   lcd_gotoxy(4,1);    // moves insertion pointer to column 4, row 1
   lcd_putc("EE313-2020"); // displays "EE313 - 2020" on the centre of the first row LCD
   delay_ms(2000);           // waits for 2 seconds
   lcd_putc('\f');              // clears the screen
   lcd_putc("GROUP MEMBERS:");
   lcd_gotoxy(1,2);    // moves insertion pointer to column 1, row 2 of lcd
   lcd_putc("KRISH");
   delay_ms(2000);
   lcd_putc('\f');    // clears the screen
   lcd_putc("GROUP MEMBERS:");
   lcd_gotoxy(1,2); // moves insertion pointer to column 1, row 2 of lcd
   lcd_putc("KRISHNEEL");
   delay_ms(2000);
   lcd_putc('\f');    // clears the screen
   lcd_putc("GROUP MEMBERS:");
   lcd_gotoxy(1,2); // moves insertion pointer to column 1, row 2 of lcd
   lcd_putc("SHAHEEL");
   delay_ms(2000);
   lcd_putc('\f');    // clears the screen
   lcd_putc("GROUP MEMBERS:");
   lcd_gotoxy(1,2); // moves insertion pointer to column 1, row 2 of lcd
   lcd_putc("SUKHDE");
   delay_ms(2000);
   lcd_putc('\f'); // clears the screen
   lcd_gotoxy(2,1);
   lcd_putc("PROJECT TOPIC:"); // displays "DC SPEED CONTROL"
   lcd_gotoxy(1,2);
   lcd_putc("DC SPEED CONTROL");
   delay_ms(2000);        // waits for 2 seconds
   lcd_putc('\f');          // clears the screen
}

void stop(){

jump1:
    output_low(PIN_A1);                // setting both pins A1 and A2 to low
    output_low(PIN_A2);               // so that motor stops rotating
    output_high(PIN_A5);             // turning off green LED
    output_low(PIN_B5);             // turning on red LED to indicate halt
    lcd_putc('\f');                    // clear the screen
    lcd_gotoxy(3,1);                // Determines the position of the words on the screen
    lcd_putc("SYSTEM PAUSE"); // displays “SYSTEM PAUSE” on LCD

 digit_5 = read_keypad();        // reading user input
     {if(digit_5 == 10){              // if user presses “*” key again
     output_low(PIN_A5);          // green LED turns on
```

```
      output_high(PIN_B5);      // red LED turns on
      anticlockwise(); // calling the anticlockwise function which turns on the motor at previously
      }                     // entered duty cycle
    else {                // if any other key is pressed, motor goes to jump1
      goto jump1;
    }}}

//Main Function

      void main()
      {
        anticlockwise();
        welcome_message();
jump:                                    // Jump used to re-enter user rpm
        lcd_putc('\f');                  // clear the screen
        lcd_gotoxy(1,1);            // Determines the position of the words on the screen
        lcd_putc("ENTER RPM:");         // Print Enter RMP message on lcd
        delay_ms(1000);
        digit_3=0;

        {digit_1 = read_keypad();      // calls function for scanning for the first key press
        lcd_gotoxy(1,2);                  // input from user occupies the space at row 2, column 1 and
prints the value onto LCD
        printf(lcd_putc,"%d",digit_1);

        digit_2 = read_keypad();       // calls function for scanning for the second key press
        lcd_gotoxy(2,2);                  // input from user occupies the space at row 2, column 2 and
prints the value onto LCD
        printf(lcd_putc,"%d",digit_2);

        digit_3 = read_keypad();  // calls function for scanning for the Enter key press
        output_low(PIN_A5);
        }

     tenth = digit_1 * 10 ;                          // if someone enters 2 as second digit then 2 * 10 = 20
     user_input = (digit_2+tenth);             // Then if someone enters 3 as third digit then 20+3 = 23

     setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256|RTCC_8_BIT);   // Timer 0 setup 8 bit mode
     enable_interrupts(INT_TIMER0);

     //Validation RPM limitations
    if (user_input>80)                     // Limits the user input to be not more than 80rpm
        {
       lcd_putc('\f');                             // clear the screen
       lcd_putc("INVALID RPM");           // To notify user if out of bounds input
       lcd_gotoxy(2,2);                  // Position on LCD screen
       lcd_putc("MAX RPM is 80");
       delay_ms(4000);
       goto jump;  // Ensures that if user enters an out of bound input, the program jumps to
where the user will re-enter desired rpm
      }
    else if(user_input<40){
       lcd_putc('\f');                  // clear the screen
       lcd_putc("INVALID RPM");
       lcd_gotoxy(2,2);

       lcd_putc("MIN RPM is 40");
       delay_ms(4000);
       goto jump;
      }

//PWM calculation
   cal_speed=(user_input);
   cal_speed=((cal_speed/80)*1023);                          // Equation for finding the required PWM
   motor_speed=CEIL(cal_speed);                            // Using the CEIL function to round off any
value to an integer type

// PWM Configurations
   setup_timer_2(T2_DIV_BY_1,255,1);                 // Set PWM frequency
   setup_ccp1(CCP_PWM);                                 // Configure CCP1 to PWM mode
   set_pwm1_duty(motor_speed);                       // Duty cycle using user input

//enable external interrupt
   enable_interrupts(INT_EXT);                          //enable external interrupt
   ext_int_edge(H_TO_L);                                //detect high to low edge
   enable_interrupts(GLOBAL);                        //enable global interrupt

 while(1)
    {
      start_measurement();
      do {
          delay_ms(1000);
         }

      while(!meas_done);

    disable_interrupts(INT_TIMER0);         // Disable Timer0 Interrupts
    disable_interrupts(INT_EXT);               // Disable External Interrupts
    dex = ((sensor_pulses/2096)* 60);     // Total Pulses divided by Revolution Count and gear
ratio (16*131 = 2096) and multiplied by 60 to get the minute intervals of revolution
    final = CEIL(dex);                   // CEIL function is used to round up the value of the rpm
```

```
  //LCD display output
    lcd_putc('\f');                              // clear the screen
    lcd_putc("ENTERED RPM: ");                   // Print Entered RMP message on lcd
    printf(lcd_putc,"%d",user_input);    // Prints the rpm desired by the user
    lcd_gotoxy(1,2);
    lcd_putc("ACTUAL RPM: ");
    printf(lcd_putc,"%d",final);            // To display the actual rpm of the motor onto the LCD
screen
    delay_us(6000);

      digit_4 = read_keypad();
    {if (digit_4 == 10){                // when the user enters the '*' button
       stop();                 //  jump to the previous function asking the user to re-enter the rpm values
       delay_ms(100);}

       else {                              // if any other key is pressed:
         goto jump;                    // takes the user to "jump"
         delay_ms(100);
       }
    }}}                                // End of program
```

CIRCUIT DIAGRAM

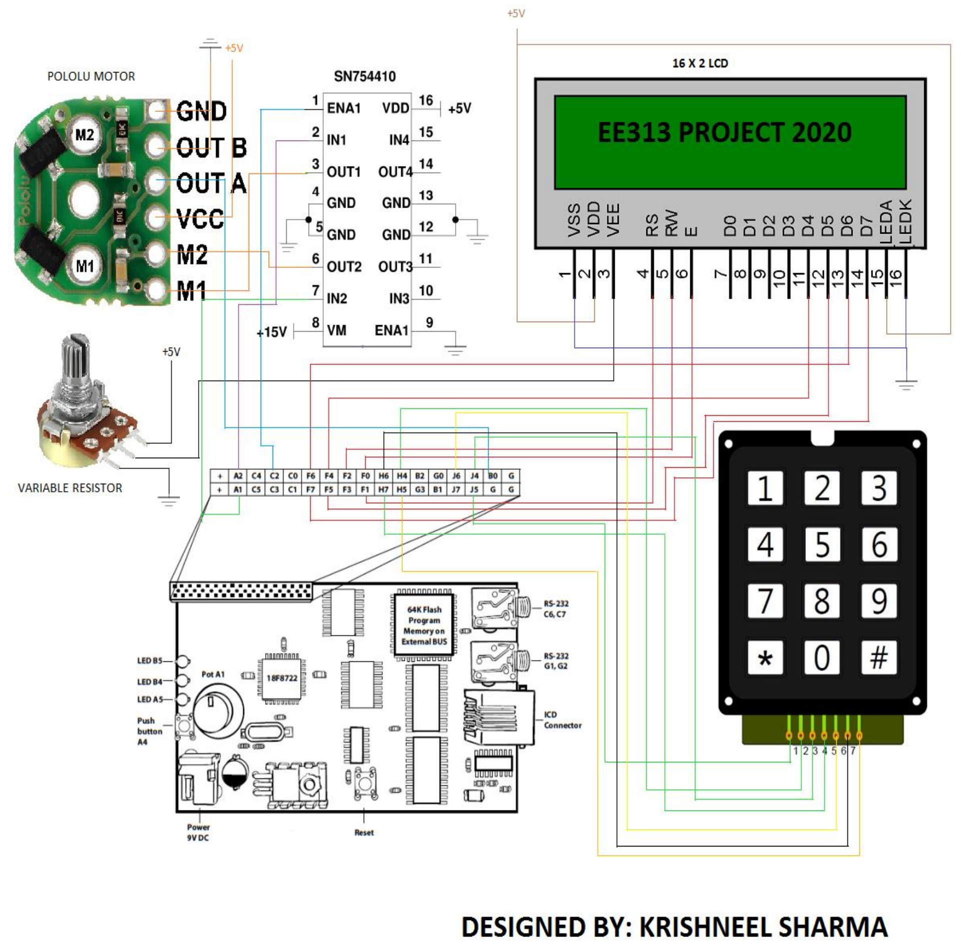


Figure 8: Final circuit wiring diagram

DISSCUSSION

The 21st century is known for the rapid increase in the number of innovations taking place. One of which includes the increasing number of automated systems we see today from basic devices that we use every day to advance level such as spacecraft. This project was the step into the world of automated system.

To begin with, the project required us to control the speed of the DC motor using peripheral interface controller (PIC). Direct Current (DC) motors has many applications ranging from toys to robots. The advantage of the DC motor is that the speed and direction can be controlled easily. The motor that was used in this project was Pololu motor with encoder. The gear ratio of this motor was 1:131, which informs that for every one turn of the input shaft the output shaft makes 131 turns approximately. The speed of DC motors can be controlled by varying the voltage or by duty cycle of the power supply. The speed controlling method that was used in the project was pulse width modulation (PWM). This is a technique used to get the desired duty cycle.

To interface the motor with microcontroller we needed driver IC this is because the motor requires high voltage and high current which the microcontroller will not be able to provide. The microcontroller is only capable to providing 5V and approximately 25mA of currents. However, the motor requires 12V and 300mA of current to run at maximum speed of 80rpm. To overcome this problem we use SN754410 to run our motor. The driver IC can be used to run two motors, since we had to control the speed of one motor the connection of the other motor was grounded. The connection of the pololu motor with the driver IC is shown in the circuit diagram in figure 5.

Moreover, to control the speed through the microcontroller the Capture, Compare and PWM module was used since our project required us to take the desired speed from the user and make changes to the speed respectively. Therefore, this particular module changes the duty cycle.

In addition, one of the objective of our project was to measure the speed of the motor and display on LCD so that the user can see if the speed that entered is same or approximately, to the speed the motor is running. To implement this it required the encoder. The motor we used in the project had the encoder with it. Required connections were made according to the datasheet of the motor. All connection have been shown in the circuit diagram in figure 8. One of the encoder pin was used to calculate the speed of motor. Using the external interrupt and timer 0, 8 bit mode the high to low edge of the pulse was counted for 1 second and then using the equation (sensor pulses/2096)* 60 the speed was calculated for 1 minute. This speed was later displayed on LCD.

Furthermore, an additional feature was implemented in this particular project. This additional feature was to stop the motor at any time while it is running and then resume the motor with the same speed the user entered before. To take the user input a 4x3 keypad was used and using the same keypad the additional feature of start and stop was implemented.

Finally, after making all proper connection of motor with driver IC, Keypad with the microcontroller and LCD with the microcontroller, a C language program was written in PIC C Compiler software, which followed all the objectives of our project.

There were also a few challenges faced in this project. Firstly, the software that was used previously had to go through the process of testing the ICD, checking the target and checking the ports. However, due to some software glitches the process of ICD checking use to stuck and took time to find the errors. This problem was solved by upgrading to a new version of the software, which did not required us to perform the testing phase. Secondly, there was an issue of LCD not displaying the characters even though our program and connection was correct. This issue was solved by disconnecting the connection of LCD with microcontroller and checking the continuity of the wires and then connecting it again with a higher value of variable resistor.

CONCLUSION

To conclude, all our major project objectives were met, we were effectively able to control the speed of the DC motor by varying the duty cycle which was taken as user-input via the 4 x 3 matrix keypad. Also, displaying the input speed on the 16 x 2 LCD screen and as well as implementing a interrupt of an emergency start/stop button which would stop the motor if pressed once and then resume at the previous input speed if this key is pressed again.

Overall, this project provided a great hands-on learning experience of the PIC 18F8722 Board and helped broaden our troubleshooting skills. Furthermore, we also learnt several coding practices since the programming language implemented was of C language.

Additionally, from observations we were able to pin a limitation of this project, which was the inability of the PIC to read speed in real time. This can be a scope of improvement for future projects.

REFERENCES

[1 U. Mehta, "Lecture notes - Motor Interface with PIC Controller," [Online]. Available:
] https://elearn.usp.ac.fj/pluginfile.php/589569/course/section/70551/EE313%20Note%202018.pdf?time=1581380855778. [Accessed 11 May 2020].

[2 "Working principle of a DC motor," Circuit Globe, [Online]. Available:
] https://circuitglobe.com/working-principle-of-a-dc-motor.html. [Accessed 2020 May 2020].

[3 [Online]. Available: https://ae01.alicdn.com/kf/H43d1b1c019cf425bb4361150959717e32.jpg,
] https://www.ato.com/content/images/thumbs/0002446_cnc-bipolar-nema-17-stepper-motor-21v-21a-2-phase-4-wires_550.jpeg, https://sc02.alicdn.com/kf/HTB1IGpAXzzuK1RjSspp760z0XXag.png. [Accessed 11 May 2020].

[4 [Online]. Available: https://www.electrical4u.com/wp-content/uploads/fleming-left-hand-rule.png.
] [Accessed 11 May 2020].

[5 [Online]. Available:
] https://elearn.usp.ac.fj/pluginfile.php/589569/course/section/70551/EE313%20Note%202018.pdf?time=1581380855778. [Accessed 11 May 2020].

[6 "MicroChip," [Online]. Available: https://www.microchip.com/wwwproducts/en/PIC18F8722.
] [Accessed 01 March 2020].

[7 "Electronics Tutorials," 2020. [Online]. Available: https://www.electronics-tutorials.ws/blog/7-
] segment-display-tutorial.html. [Accessed 01 March 2020]

APPENDIX

Click the link below to view the project demonstration video on YouTube:
https://www.youtube.com/watch?v=meyTp2DMRVA