Islamic University of Technology

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

COURSE No.	: EEE 4706
COURSE TITLE	: Microcontroller Based System Design Lab
NAME OF PROJECT	:DC Motor with speed control
DATE OF SUBMISSION	:28/04/25
Group Information	:Group No. : 2
	Section : A1
Group Members	:

Student ID	Name
200021105	Fabbiha Bushra
200021113	Tasnia Nafs
200021133	Syed Mohammad Ahnaf Faruk
200021141	Mustafid Bin Mostafa
200021153	Sheriffo Badjee

Table of Contents

Introduction and Theory

DC Motor and Motor Driver Concept

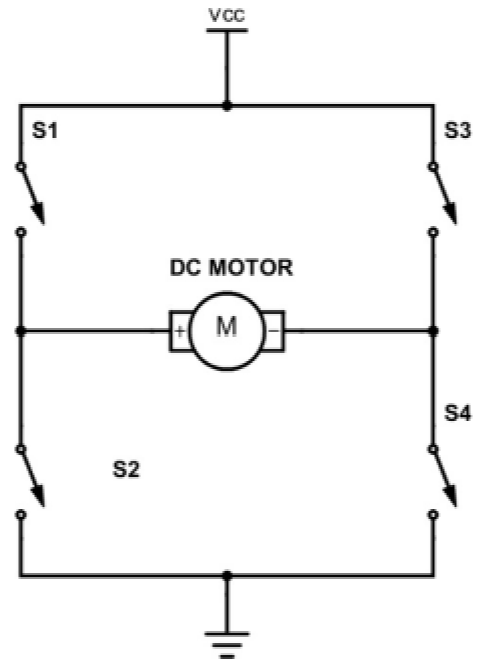
A DC motor functions by converting electrical energy into mechanical movement, utilizing direct current (DC) as its power source. It features two terminals that connect to the control circuit, enabling operation. The direction of rotation ,either clockwise or anticlockwise,is determined by the polarity of the voltage applied to these terminals. By reversing the polarity, the motor's rotational direction can be easily changed.

The motor used in this project operates within a voltage range of 3V to 6V and supports bidirectional rotation by altering the connection of the lead wires.

However, the current and voltage requirements of a DC motor often exceed the output capacity of a microcontroller. Additionally, the reverse electromotive force (back EMF) generated during motor operation poses a risk of damaging the microcontroller. Therefore, an intermediary interface, specifically a motor driver circuit, is required to safely and effectively connect the motor to the microcontroller.

Motor Driver (L293D) and H-Bridge Working Principle

In this system, the motor is interfaced with the microcontroller through the L293D motor driver



IC. The L293D provides an output voltage range of 4.5V to 6V and can deliver a continuous current of up to 600mA per channel. It is a 16-pin IC capable of controlling two DC motors simultaneously. The L293D operates based on the H-bridge principle, a standard technique used for motor direction control.

The H-bridge configuration allows current to flow through the motor in either direction. According to the H-bridge logic:

- When all switches are open, the motor remains off.
- Closing switches S1 and S4 results in clockwise rotation.
- Closing switches S2 and S3 results in anticlockwise rotation.

This flexible switching arrangement enables the motor to start, stop, and change direction efficiently based on microcontroller commands, ensuring reliable motor control.

****** For this entire project code and connections we used chapter 17 of the textbook as our reference.**

Objective

The primary purpose of a DC motor is to convert electrical energy into mechanical rotation. In microcontroller-based embedded systems, controlling such motors efficiently involves not only powering the motor but also managing its speed, direction, and safety. However, microcontrollers such as the AT89S52 are limited in terms of current and voltage output capabilities, making them unsuitable for directly driving motors. Hence,we use a motor driver (specifically the L293D) to bridge the gap, allowing safe and efficient operation.

This project focuses on developing a speed-controlled DC motor system using an 8051-based microcontroller platform. By integrating a 4x4 keypad, an LCD display, and control buttons, users can dynamically select motor direction (clockwise or anticlockwise) and set predefined speed levels (LOW, MEDIUM, HIGH, RATED). The speed is adjusted using PWM (Pulse Width Modulation).

For system protection, both overload and overspeed mechanisms are implemented. Overload protection shuts off the motor at low-speed conditions (interpreted as potential excess load), while a buzzer alert is triggered at maximum speed (100% PWM) to signal an overspeed scenario.

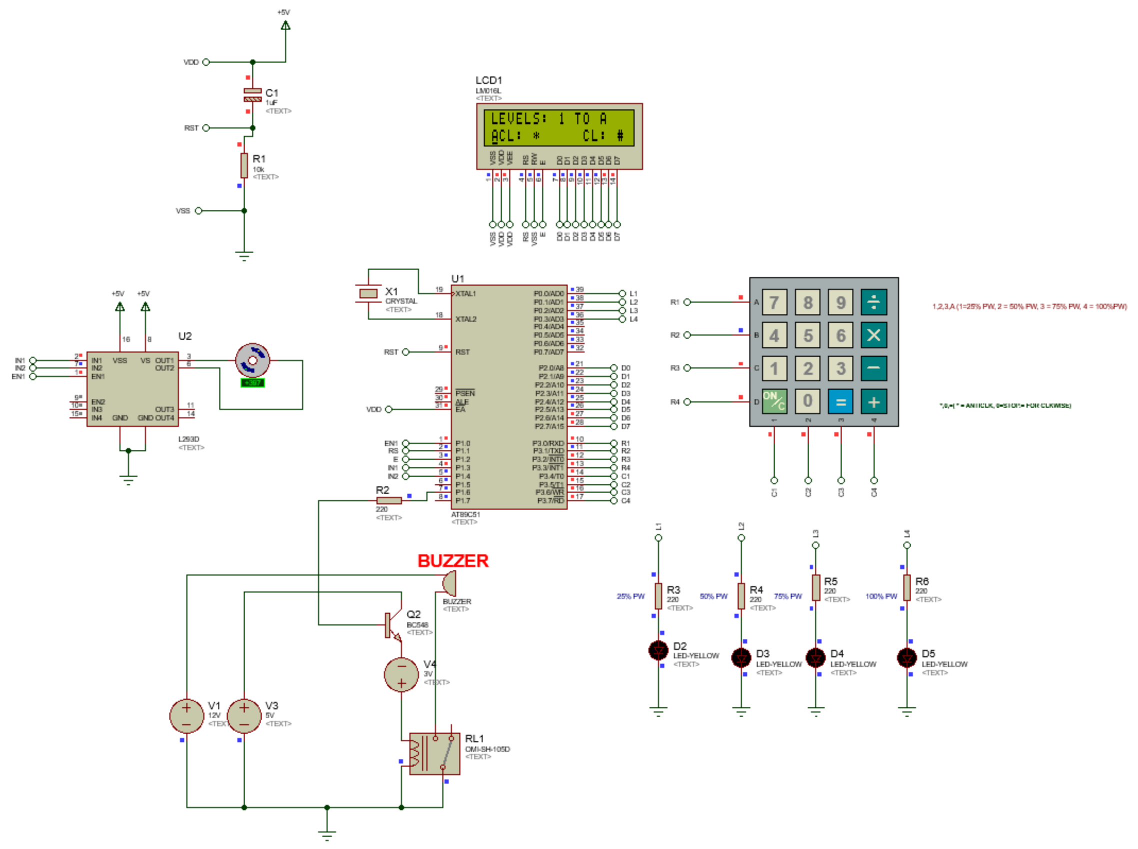
The primary objective of this project is to design and implement a motor control system with the following goals:

- Allow users to select clockwise or anticlockwise motor rotation using a keypad and display the chosen direction on an LCD.
- Provide predefined speed settings (LOW, MED, HIGH, Rated speed) using keypad.
- Simulate a condition where the motor shuts off and triggers a buzzer if a low-speed (25% PWM) state is detected, indicating an overload.
- Indicate the duty cycle (25%, 50%, 75%, 100%) both on the LCD and with four dedicated LEDs.
- Trigger a buzzer when the motor reaches its maximum (100% PWM) speed to signal a potential overspeed condition.

Required Components

	Components	
1.	AT89C52 Microcontroller	Microcontroller Board(includes all the components)
2.	L293D Motor Driver Module	
3.	4x4 Matrix Keypad	
4.	LCD display 16x2	
5.	Jumper Wires (male-male, male-female, female-female)	
6.	12V 2400 RPM DC Motor	
7.	USBasp Programmer	
8.	5V Power Supply Module, LED, Buzzer	
		Cost = TK 7000

Circuit Diagram



The given circuit diagram represents a DC motor speed control system using an AT89C51 (8051) microcontroller. A 4x4 matrix keypad connected to Port 1 allows the user to input commands for selecting different speed levels. The selected level is displayed on a 16x2 LCD connected to Port 2.

Motor speed control is achieved through PWM signals generated by the microcontroller and fed into an L293D motor driver IC, which drives the DC motor. Four LEDs indicate the current PWM duty cycle (25%, 50%, 75%, or 100%) based on the user's input. A buzzer circuit with a PNP transistor provides audio feedback for invalid operations or alerts. A relay mechanism is included for isolating and protecting the motor circuit. The system uses a crystal oscillator for clock generation and a reset circuit for initializing the microcontroller during power-up.

Features

The following is a detailed description of the system's implemented features:

Mandatory Features:

1. Direction Control via Keypad:

The system allows the user to select the direction of motor rotation (clockwise or anticlockwise) using a 4x4 keypad. Pressing the # key sets the motor to clockwise, while pressing * switches it to anticlockwise. The selected direction is immediately displayed on the LCD screen, ensuring real-time feedback.

2. Predefined Speed Levels with Gradual Acceleration:

Three predefined speed levels(LOW, MEDIUM, and HIGH) along with the rated speed level can be selected through dedicated keypad inputs. Upon selecting a level, the motor speed changes gradually.

3. Overload Protection Mechanism:

The system includes an overload protection feature. If the motor speed drops to 25% PWM ,indicating a high load or mechanical resistance ,the motor automatically shuts down, This protects both the motor and control circuitry from potential damage due to excessive current draw. When the motor operates at 100% PWM ,usually under a no-load condition,a buzzer is activated to indicate an overspeed warning.

Additional Features:

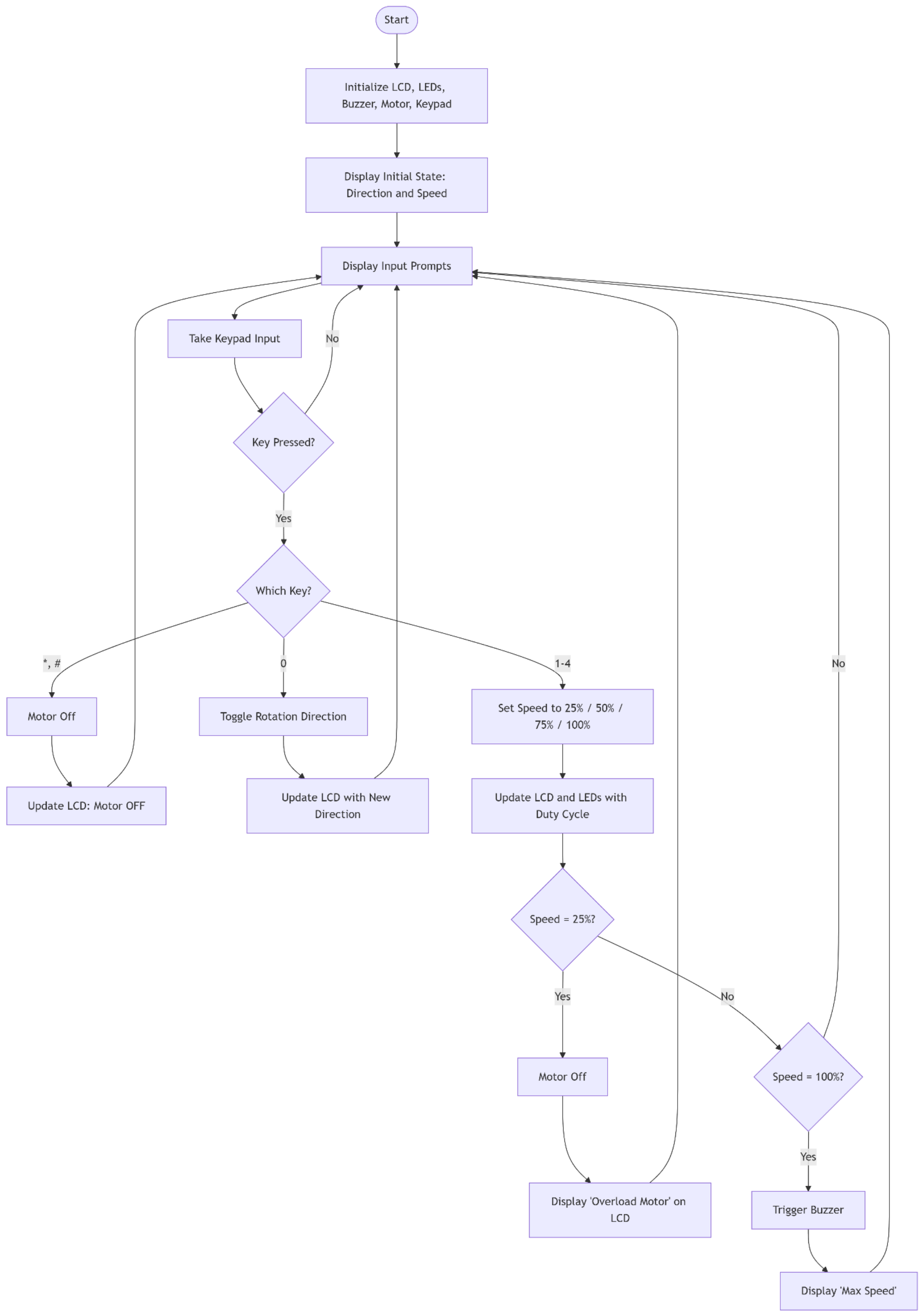
1.Real-Time PWM Display:

The system is designed to continuously monitor and display the motor's operating PWM (Pulse Width Modulation) duty cycle. The PWM levels are divided into four standard percentages: 25%, 50%, 75%, and 100%. Each PWM level corresponds to a specific operating speed of the DC motor.

2.LED Indicators:

To provide immediate visual feedback, four dedicated LEDs are incorporated into the design. Each LED lights up according to the selected duty cycle—allowing users to quickly determine the motor's speed state without relying solely on the LCD.

Working Principle


Start
Initialize LCD, LEDs, Buzzer, Motor, Keypad
Display Initial State: Direction and Speed
Display Input Prompts
Take Keypad Input
Key Pressed?
No
Yes
Which Key?
*, #
0
1-4
Motor Off
Toggle Rotation Direction
Set Speed to 25% / 50% / 75% / 100%
Update LCD: Motor OFF
Update LCD with New Direction
Update LCD and LEDs with Duty Cycle
Speed = 25%?
Yes
No
Motor Off
Speed = 100%?
No
Yes
Display 'Overload Motor' on LCD
Trigger Buzzer
Display 'Max Speed'

1. At first, we are initializing the 8051 microcontroller: setting the stack pointer to 70H to prevent the stack from overwriting variables, clearing the PSW to select register bank 0, and defining names for four LEDs responsible for speed control, one LED for the buzzer, and two pins for LCD control. Then, we clear Ports 0, 1, and 2 to set them as output ports, and configure Port 3 input while the rest remain as outputs.

2. Next, the LCD is initialized by setting it to 2-line mode with a 5x7 dot matrix, turning on the display with the cursor visible, clearing the screen, and adjusted cursor position. Then, a series of predefined messages (MSG to MSG12, START1, START2, ACTIVE) are loaded into the data pointer (DPTR) and printed on the LCD one after another using a subroutine LCD_PRINT, with the screen cleared between different sets of messages.

3. Following this, we have defined subroutines to control the LCD display. The LCD_PRINT subroutine reads characters from memory pointed by DPTR, sends them one by one to the LCD using the DISPLAY subroutine, and stops when it detects a null character. After printing, it moves the cursor to the second line. The DISPLAY subroutine sends actual data (characters) to the LCD, setting the RS pin high to indicate data write. The COMMAND subroutine sends instructions to the LCD by setting RS low. Both DISPLAY and COMMAND use a small DELAY subroutine to ensure proper timing for the LCD to process the data or command. The DELAY routine creates an approximate 50 ms delay using nested loops.

4. After LCD initialization, the motor is started by setting P1.0 (the enable pin) high. The default direction is clockwise, achieved by setting IN1 high and IN2 low. In the H-bridge, IN1 controls switches S1 and S4, while IN2 controls S2 and S3. Activating IN1 turns on S1 and S4, causing the motor to rotate clockwise.

5. The STARTUP_MOTOR routine clears the LCD and sequentially prints startup instruction messages, including keys to define the RPM level, select clockwise or anticlockwise rotation, and stop the motor.

6. Afterwards, we started scanning the keypad. In our project we are not following the conventional method described in the lab sheet. Instead of using one port for input and another for output, we use Port 3 for both the rows and columns of the keypad (P3.0–P3.3 for rows, P3.4–P3.7 for columns). Initially, MOV P3, #0FEH sets the upper four bits (P3.4–P3.7) to 1111 (columns idle) and the lower bits (P3.0–P3.3) to 1110 to begin checking the first row. We first check if there is a '0' detected in the first row; if so, we proceed to scan the corresponding columns (C1–C4). Based on where the '0' appears, we jump to different labels corresponding to the keypad symbols. If no key is detected in the first row, we set the second row low by

sending 1101 (setting second row to '0', first to '1'), and continue scanning, repeating the same logic for the remaining rows.
7. To control motor operation, inputs are taken only from the 1st (1,2,3,A) and last (*,0,=,D) rows of the keypad, while inputs from the 2nd and 3rd rows are ignored by jumping back to scanning (LJMP ML).
8. Motor control is based on keypad inputs: pressing '0' clears the LCD and jumps to execute motor stop, keys '1' to 'A' select different RPM levels, update the display, and jump to subroutines to run the motor at different speeds, while 'E' and 'F' change the motor's rotation direction (clockwise or anticlockwise). After each operation, the system returns to scanning the keypad. A small delay subroutine (DELAYM) is included to support PWM timing.
9. To halt the motor, both IN1 and IN2 (P1.3 and P1.4) are cleared. After stopping, the program scans the keypad: if a key from the first (1,2,3,A) or last row (,#) is pressed, the motor restarts. The motor operates at four RPM levels based on duty cycle: 25%, 50%, 75%, and 100%, with corresponding LEDs (LED25, LED50, LED75, LED100) indicating the active level. At 25% duty cycle (MOTOR_1), default rotation is clockwise, achieved by setting IN1 high and IN2 low with controlled delay timings. If an anticlockwise input (") is detected, it switches to anticlockwise rotation (MOTOR_1_ACL) by setting IN2 high and IN1 low. Direction change commands are managed through ACL_MAIN (anticlockwise) and CL_MAIN (clockwise) labels, which update the LCD display accordingly. Similar logic applies for higher duty cycles (50%, 75%, and 100%) by adjusting the number of delay calls.
10.Additionally, intermediate states are invoked to avoid address out-of-range problems during program execution.
11. Finally for overload protection, when the motor is operating at 25% PWM at the MOTOR_1_CLK or MOTOR_1_ACLK level, indicating that the load is high and the speed is very low, the system automatically disconnects the motor. After a few seconds of operation at 25% PWM, a warning message is displayed on the LCD indicating that the motor is stopping due to overload, and then the program jumps to the MOTOR_0 level to halt the motor & preventing potential damage.

Program Code

```
ORG 0000H
MOV SP, #70H
MOV PSW, #00H
LED EQU P0.4 ; Define BUZZER pin as P0.4
LED25 EQU P0.0
LED50 EQU P0.1
LED75 EQU P0.2
LED100 EQU P0.3
;------------------------------------------------------------
RS EQU P1.1
EN EQU P1.2
;------------------------------------------------------------
INITIALIZATION:
MOV P0, #00H
MOV P1, #00H
MOV P2, #00H
MOV P3, #0FEH ;11111110
;------------------------------------------------------------

;-------------------------------------------------------------
LCD_INIT:
MOV R2, #38H ;INITIALIZING LCD, 2 LINES, 5X7 MATRIX
ACALL COMMAND ;CALLING COMMAND SUBROUTINE

MOV R2, #0EH ;DISPLAY ON, CURSOR ON
ACALL COMMAND

MOV R2, #01H ;CLEARING SCREEN
ACALL COMMAND
MOV A,#06H ;INCREMENT CURSOR POSITION
ACALL COMMAND
MOV R2, #80H ;CURSOR AT START OF 1ST LINE
ACALL COMMAND
MOV A,#3CH
ACALL COMMAND
MOV DPTR,#MSG ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG2
ACALL LCD_PRINT
MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG3 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG4
ACALL LCD_PRINT

```

```
MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG5 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG6
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG7 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG8
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG9 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG10
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG11 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG12
ACALL LCD_PRINT

MOV R2, #01H
ACALL COMMAND
MOV DPTR,#START1 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#START2
ACALL LCD_PRINT
MOV R2, #01H
ACALL COMMAND
MOV DPTR,#ACTIVE
ACALL LCD_PRINT
LJMP DECISION

;LCD display subroutine-------------------------------------------
LCD_PRINT:
LOOP:
CLR A
MOVC A, @A+DPTR
MOV R2, A
JZ EXIT ;stop writing when null char is detected
ACALL DISPLAY ;main display part
ACALL DELAY ;giving a short delay for LCD
INC DPTR
LJMP LOOP
EXIT:
MOV R2, #0C0H ;go to the 2nd line of LCD
ACALL COMMAND
RET

DISPLAY:
MOV P2, R2 ;subroutine for displaying in LCD
SETB RS ;RW is grounded in Hardware
SETB EN
ACALL DELAY
CLR EN
RET
;----------------------------------------------------------------
COMMAND:
MOV P2, R2 ;giving instructions to LCD
CLR RS
SETB EN
ACALL DELAY
CLR EN
ACALL DELAY
RET
;----------------------------------------------------------------
DELAY:
MOV 52H, #100 ;50 ms delay (approx)
D1: MOV 51H, #255
DJNZ 51H, $
DJNZ 52H, D1
RET
;----------------------------------------------------------------

;----------------------------------------------------------------
DECISION:
LJMP MOTOR_ROTATION ;Start Motor part

;MOTOR PART------------------------------------------------------
MOTOR_ROTATION:
SETB P1.0
SETB P1.3 ;Input to motor driver
CLR P1.4 ;Clockwise direction
;----------------------------------------------------------------
STARTUP_MOTOR:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #INT_MSG ;print 'ENTER RPM LEVEL'
LCALL LCD_PRINT
```

```
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LEVEL ;print 'LEVEL: 1 TO A'
LCALL LCD_PRINT
LCALL DELAY
MOV DPTR, #STOP ;print 'PRESS 0 TO STOP'
LCALL LCD_PRINT
MOV DPTR, #ACLCL
LCALL LCD_PRINT

;keyboard RPM Input Scanning---------------------------------------
;------------------------------------------------------------------
;SCANNING ALL COLUMN ---------------------------------------------
ML:
JNB P3.0, CC1
JNB P3.1, CC2
JNB P3.2, CC3
JNB P3.3, CC4
SJMP ML
;------------------------------------------------------------------
;SCANNING COLUMN1 ------------------------------------------------
CC1:
JNB P3.4, JMP_TO_1
JNB P3.5, JMP_TO_2
JNB P3.6, JMP_TO_3
JNB P3.7, JMP_TO_A

SETB P3.0
CLR P3.1
SJMP ML
;SCANNING COLUMN2 -----------------------------------------------

CC2:
JNB P3.4, JMP_TO_4
JNB P3.5, JMP_TO_5
JNB P3.6, JMP_TO_6
JNB P3.7, JMP_TO_B

SETB P3.1
CLR P3.2
SJMP ML
;SCANNING COLUMN3 -----------------------------------------------

CC3:
JNB P3.4, JMP_TO_7
JNB P3.5, JMP_TO_8
JNB P3.6, JMP_TO_9
JNB P3.7, JMP_TO_C
```

```

SETB P3.2
CLR P3.3
SJMP ML
;SCANNING COLUMN4 --------------------------------------------

CC4:
JNB P3.4, JMP_TO_F
JNB P3.5, JMP_TO_0
JNB P3.6, JMP_TO_E
JNB P3.7, JMP_TO_D

SETB P3.3
CLR P3.0
LJMP ML
;---------------------------------------------------------------
JMP_TO_0: LJMP M_0 ;MOTOR OFF
JMP_TO_1: LJMP M_1 ;LEVEL 1
JMP_TO_2: LJMP M_2 ;LEVEL 2
JMP_TO_3: LJMP M_3 ;LEVEL 3
JMP_TO_4: LJMP ML ;NOT NEEDED
JMP_TO_5: LJMP ML ;continue scanning
JMP_TO_6: LJMP ML
JMP_TO_7: LJMP ML
JMP_TO_8: LJMP ML
JMP_TO_9: LJMP ML
JMP_TO_A: LJMP M_A ;LEVEL 4
JMP_TO_B: LJMP ML
JMP_TO_C: LJMP ML
JMP_TO_D: LJMP ML
JMP_TO_E: LJMP M_E ;CLOCKWISE
JMP_TO_F: LJMP M_F ;ANTI-CLOCKWISE
;---------------------------------------------------------------

M_0:
JNB P3.5,M_0 ;wait for key release
MOV R2, #01H
ACALL COMMAND ;clear LCD screen
MOV DPTR, #OFF_MSG ;display 'Motor off'
ACALL LCD_PRINT
LJMP MOTOR_0 ;MOTOR OFF STATE
CLR LED ; Turn off LED when '0' is pressed
M_1:
JNB P3.4,M_1
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL1
ACALL LCD_PRINT
```

```
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_1 ;RPM LEVEL 1

M_2:
JNB P3.5,M_2
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL2
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_2 ;RPM LEVEL 2

M_3:
JNB P3.6,M_3
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL3
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_3 ;RPM LEVEL 3

M_A:
JNB P3.7,M_A
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL4
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_4 ;RPM LEVEL 4
;------------------------------------------------------------------
M_F:
JNB P3.4,M_F
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #ANTICLK ;Display 'Anticlockwise'
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
SETB P1.0
SETB P1.4 ;anticlockwise rotation
CLR P1.3
LJMP ML ;scan for new input

M_E:
```

```
JNB P3.6,M_E
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #CLKWISE
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
SETB P1.0
SETB P1.3 ;clockwise rotation (In1=1,In2=0)
CLR P1.4
LJMP ML
;----------------------------------------------------------------
;Delay subroutine for PWM
;----------------------------------------------------------------
DELAYM:
MOV R5, #1
Hl: MOV R4, #100
H2: MOV R3, #255
H3: DJNZ R3, H3
DJNZ R4, H2
DJNZ R5, Hl
RET
;----------------------------------------------------------------
LC: MOV P3, #0FEH ;INITIAL CONDITION
LJMP ML ;SCAN KEYPAD AGAIN
;----------------------------------------------------------------
MOTOR_0:
CLR P1.0 ;MOTOR OFF
CLR P1.3
CLR P1.4
JNB P3.4,LL ;JNB IS SHORT JUMP,
JNB P3.5,LL ;SO LL IS USED AS AN INTERMEDIATE STAGE
JNB P3.6,LL
JNB P3.7,LL
SETB P3.0
CLR P3.3 ;CHECK ROW 4 (* 0 # D)
JNB P3.4,LC ;'*' PRESSED
JNB P3.5,LC ;'0' PRESSED
JNB P3.6,LC ;'#' PRESSED
SETB P3.3
CLR LED ; Turn off LED when '0' is pressed
CLR LED25
CLR LED50
CLR LED75
CLR LED100
CLR P3.0 ;AGAIN CHECK ROW 1(1 2 3 A)
SJMP MOTOR_0
;----------------------------------------------------------------
```

```
LL: LJMP ML ;JUMP TO L1
;--------------------------------------------------------------
MOTOR_1: ;25% DUTY CYCLE
SETB LED25
CLR LED50
CLR LED75
CLR LED100
CLR LED
SETB P1.0
SETB P1.3
CLR P1.4
ACALL DELAYM
CLR P1.3
CLR P1.4
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL ;CHECK FOR NEW INPUT
JNB P3.5,LL ;KEY 1,2,3,A FOR LEVEL 1-4
JNB P3.6,LL
JNB P3.7,LL
SETB P3.0
CLR P3.3
JNB P3.4,ACL1_1 ;'*' FOR ANTI-CLOCKWISE
JNB P3.5,LC ;GO OUT WHEN '0' IS PRESSED
JNB P3.6,CL1_1 ;# FOR CLOCKWISE
SETB P3.3
CLR P3.0

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG13 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG13
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG14 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG15
ACALL LCD_PRINT
;JNB P1.0, LIM1
LJMP M_0

LC1_2: LJMP LC
ACL1_1:LJMP ACL1
```

```
CL1_1: LJMP CL1
LL_2:LJMP LL

MOTOR_1_ACL: ;25% DUTY CYCLE
SETB LED25
CLR LED50
CLR LED75
CLR LED100
CLR LED
SETB P1.0
SETB P1.4
CLR P1.3
ACALL DELAYM
CLR P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL_2 ;CHECK FOR NEW INPUT
JNB P3.5,LL_2 ;KEY 1,2,3,A FOR LEVEL 1-4
JNB P3.6,LL_2
JNB P3.7,LL_2
SETB P3.0
CLR P3.3
JNB P3.4,ACL1 ;'*' FOR ANTI-CLOCKWISE
JNB P3.5,LC1_2 ;GO OUT WHEN '0' IS PRESSED
JNB P3.6,CL1 ;# FOR CLOCKWISE
SETB P3.3
CLR P3.0

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG13 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG13
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG14 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG15
ACALL LCD_PRINT

;JNB P1.0, LIM1A
LJMP M_0

```

```

;-------------------------------------------------------------------
LL_1: LJMP LL
LC1_1: LJMP LC1_2
;......................................
CL1:ACALL CL_MAIN ;FOR CLOCKWISE ROTATION
LJMP MOTOR_1 ;CONTINUE SAME RPM LEVEL
ACL1:ACALL ACL_MAIN
LJMP MOTOR_1_ACL
;-------------------------------------------------------------------
MOTOR_2: ;50% DUTY CYCLE
CLR LED25
CLR LED75
CLR LED100
CLR LED
SETB LED50
SETB P1.0
SETB P1.3
CLR P1.4
ACALL DELAYM
ACALL DELAYM
CLR P1.3
CLR P1.4
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL_1
JNB P3.5,LL_1
JNB P3.6,LL_1
JNB P3.7,LL_1
SETB P3.0
CLR P3.3
JNB P3.4,ACL2_1 ;'*' pressed
JNB P3.5,LC1_1
JNB P3.6,CL2_1
SETB P3.3
CLR P3.0
SJMP MOTOR_2

LL_12:LJMP LL_1

ACL2_1: LJMP ACL2
CL2_1: LJMP CL2

MOTOR_2_ACL: ;50% DUTY CYCLE
CLR LED25
CLR LED75
CLR LED100
CLR LED
```

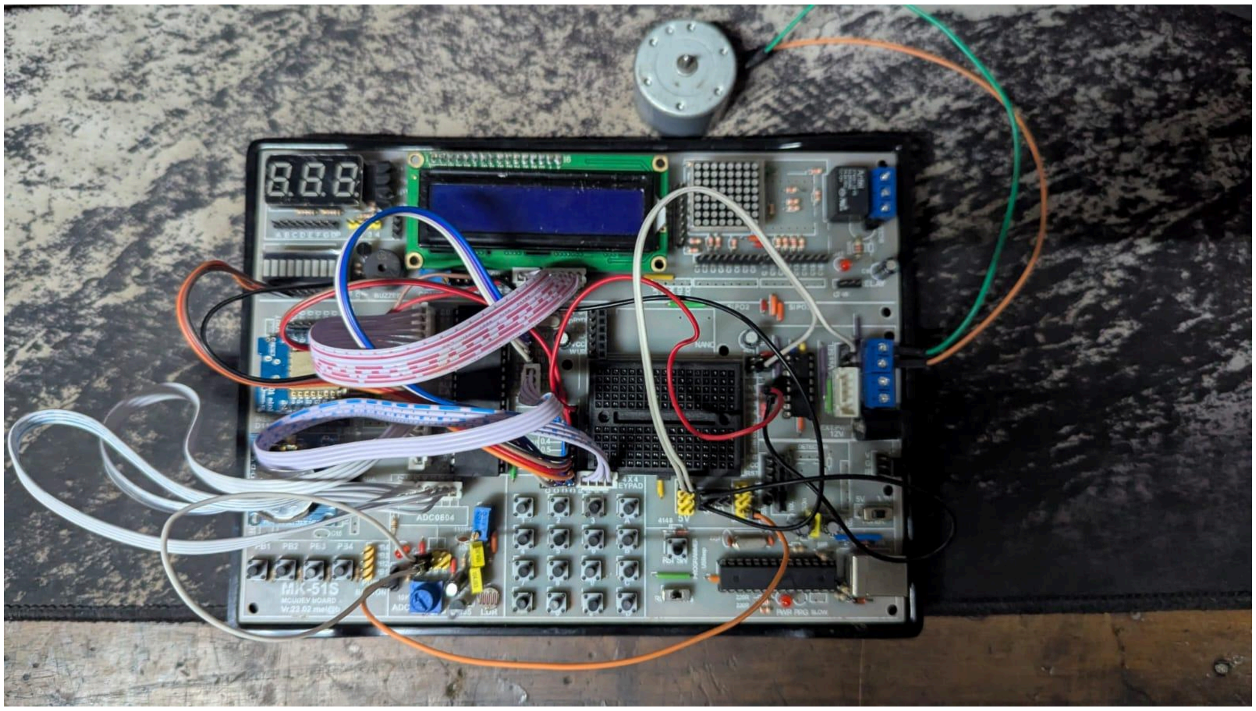
```
SETB LED50
SETB P1.0
SETB P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
CLR P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL_12
JNB P3.5,LL_12
JNB P3.6,LL_12
JNB P3.7,LL_12
SETB P3.0
CLR P3.3
JNB P3.4,ACL2 ;'*' pressed
JNB P3.5,LC1
JNB P3.6,CL2
SETB P3.3
CLR P3.0
SJMP MOTOR_2_ACL

;------------------------------------------------------------------
CL2: ACALL CL_MAIN
LJMP MOTOR_2
ACL2: ACALL ACL_MAIN ;rotation anticlockwise
LJMP MOTOR_2_ACL ;continue with the same RPM
;------------------------------------------------------------------
LC1: LJMP ML ;used as a redirect
;------------------------------------------------------------------
;CLOCKWISE ROTATION-----------------------------------------------
CL_MAIN:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #CLKWISE
ACALL LCD_PRINT
MOV DPTR, #ACLM
ACALL LCD_PRINT
RET
;ANTI-CLOCKWISE ROTATION------------------------------------------
ACL_MAIN:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #ANTICLK
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
```

```
RET
;----------------------------------------------------------------

MOTOR_3: ;75% DUTY CYCLE
CLR LED
CLR LED25
CLR LED50
CLR  LED100
SETB LED75
SETB P1.0
SETB P1.3
CLR P1.4
;SETB LED75
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
CLR P1.3
CLR P1.4
ACALL DELAYM
JNB P3.4,LL1_12
JNB P3.5,LL1_12
JNB P3.6,LL1_12
JNB P3.7,LL1_12
SETB P3.0
CLR P3.3
JNB P3.4,ACL3_1
JNB P3.5,LC1
JNB P3.6,CL3_1
SETB P3.3
CLR P3.0
SJMP MOTOR_3


LC11_1: LJMP LC1
LL1_12:LJMP LL1_1
ACL3_1:LJMP ACL3
CL3_1:LJMP CL3

MOTOR_3_ACL: ;75% DUTY CYCLE
CLR LED
CLR LED25
CLR LED50
CLR  LED100
SETB LED75
SETB P1.0
SETB P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
CLR P1.4
CLR P1.3
ACALL DELAYM
JNB P3.4,LL1_1
JNB P3.5,LL1_1
JNB P3.6,LL1_1
JNB P3.7,LL1_1
SETB P3.0
CLR P3.3
JNB P3.4,ACL3
JNB P3.5,LC11_1
JNB P3.6,CL3
SETB P3.3
CLR P3.0
;JNB P1.0, LIM3A
LJMP MOTOR_3_ACL

;-----------------------------------------------------------------
CL3:ACALL CL_MAIN
LJMP MOTOR_3
ACL3:ACALL ACL_MAIN
LJMP MOTOR_3_ACL

LL1_1: LJMP LL1_2
LC111_1: LJMP LC11_1
;-----------------------------------------------------------------
MOTOR_4: ;KEY 'A'
CLR LED25
CLR LED50
CLR LED75
SETB LED100
SETB P1.0 ;100% DUTY CYCLE
SETB P1.3
CLR P1.4
SETB LED ; Turn on buzzer when motor reaches 3000 RPM

ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL1_2
JNB P3.5,LL1_2
JNB P3.6,LL1_2
JNB P3.7,LL1_2
SETB P3.0
CLR P3.3
```

```
JNB P3.4,ACL4_1
JNB P3.5,LC111_1
JNB P3.6,CL4_1
SETB P3.3
CLR P3.0
;JNB P1.0, LIM4
CLR P1.0
SJMP MOTOR_4

LL1_2: LJMP LL1
ACL4_1:LJMP ACL4
CL4_1: LJMP CL4
LC111_2: LJMP LC111_1

MOTOR_4_ACL: ;KEY 'A'
CLR LED25
CLR LED50
CLR LED75
SETB LED100
SETB P1.0 ;100% DUTY CYCLE
SETB P1.4
CLR P1.3
SETB LED ; Turn on buzzer when motor reaches 3000 RPM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL1
JNB P3.5,LL1
JNB P3.6,LL1
JNB P3.7,LL1
SETB P3.0
CLR P3.3
JNB P3.4,ACL4
JNB P3.5,LC111_2
JNB P3.6,CL4
SETB P3.3
CLR P3.0
;JNB P1.0, LIM4A
CLR P1.0
LJMP MOTOR_4_ACL

;-------------------------------------------------------------
CL4:ACALL CL_MAIN
LJMP MOTOR_4
ACL4:ACALL ACL_MAIN
SJMP MOTOR_4_ACL
;-------------------------------------------------------------
```

```
LL1:LJMP ML
;-----------------------------------------------------------------

;Display messages-----------------------------------------------------
INT_MSG: DB 'ENTER RPM LEVEL',0 ;INITIAL MESSAGE
OFF_MSG: DB 'MOTOR OFF',0
LVL1: DB 'L1:75 RPM,25%PW',0
LVL2: DB 'L2:150 RPM,50%PW ',0
LVL3: DB 'L3:225 RPM,75%PW ',0
LVL4: DB 'L4:300 RPM,100PW',0
CLKWISE: DB 'CLOCKWISE',0
ANTICLK: DB 'ANTI-CLOCKWISE',0
;-----------------------------------------------------------------
MSG:    DB 'EEE-4706',0
MSG2:   DB 'A1-GROUP-02',0
MSG3:   DB 'AFIF',0
MSG4:   DB 'ID:141',0
MSG5:   DB 'TASNIA',0
MSG6:   DB 'ID:113',0
MSG7:   DB 'FABBIHA',0
MSG8:   DB 'ID:105',0
MSG9:   DB 'FARUK',0
MSG10:   DB 'ID:133',0
MSG11:   DB 'SHERIFFO',0
MSG12:   DB 'ID:153',0
MSG13:   DB 'OVERLOAD!!',0
MSG14:   DB 'GOING TO STOP',0
MSG15:   DB 'THE MOTOR',0
START1: DB 'DC MOTOR WITH',0
START2: DB 'SPEED CONTROL',0
MSG16:   DB 'MOTOR IS OFF',0
ACTIVE: DB 'MOTOR IS ON',0
;DENY_MSG: DB 'WRONG PASSWORD',0
;TRY_AGN: DB 'TRY AGAIN',0
LEVEL: DB 'LEVELS: 1 TO A',0
STOP: DB 'PRESS 0 TO STOP',0
ACLM: DB 'ANTICLK: PRESS *',0
CLM: DB 'CLK : PRESS #',0
ACLCL: DB 'ACL: *      CL: #',0
END
```

Hardware Implementation



Problems Faced

(1) One of the fundamental problems that we faced was to properly interface the hardware part in the board. Such as-

(i) Port 1 of the board cannot be connected to the LCD display.

(ii) Improper labelling of different segments of the board. For example both port 0 and port 1 were labeled as port 0. Then enable pin of the L293D driver was labeled in the same port which was allocated for ground,

(iii) Between the 2 Vcc pins of the driver (pin 8, 16) only pin 8 was connected but pin 16 was left open as a port to be connected externally which was unnecessary.

(iv) Buzzer on the board was active low. So we had to change the code a little bit for running the hardware part .

(2) Regarding the software part there were no visible difficulties. Only one problem that we faced was to operate the buzzer at 12V cause in 8.17 initially the 5V package was not included. Later on we added that in the library. However the 12V buzzer circuit was kept as it is. In hardware there is no such issue.

(3) In the code to avoid memory out of range issues we used small intermediate stages/labels for jumping.

Conclusion

In this project, we successfully designed and implemented a microcontroller-based DC motor speed control system using the AT89C52, covering both hardware and software aspects. By integrating a keypad, LCD display, motor driver (L293D), LEDs, buzzer, and protective features like overload protection, the system enables real-time direction control and speed selection at four different levels. Despite facing some hardware interfacing challenges, all functionalities were achieved, and proper safeguards were implemented to ensure reliable and efficient motor operation.