Password Protected DC Motor

Course NO. **: EEE 4706**
Course Name **: Microcontroller Based System Design Lab**

Lab Group **: A-2**
Project Group **: G-4**

Group Members:

STUDENT ID	NAME
200021102	Khandaker Adeba Tabassum
200021112	Nahadia Tabassum Oyshi
200021132	MD. Nazmul Aman
200021146	Md. Mehedi Hasan

TABLE OF CONTENTS:

OBJECTIVE:

A DC motor is an electromechanical device that can convert electrical energy into mechanical energy. The main components of a DC motor are stator (provides a constant magnetic field), rotor (carries windings to the shaft), commutator (reverses the direction of current in the armature windings, to ensure smooth rotation), brushes (maintain electrical contact between the stationary and moving parts). DC motors are used in home appliances like fans, washing machines, vacuum cleaners, mixers, in smart devices like electrical doors and windows, scanners, printers, cooling fans, robotics kits, remote-controlled cars, drones etc. DC motors are known for their efficient use of power, especially in battery-powered devices. Adding a password can enhance security, safety, and user control in any system like smart homes or robotics, electric vehicles, e-bikes etc. that uses a DC motor.

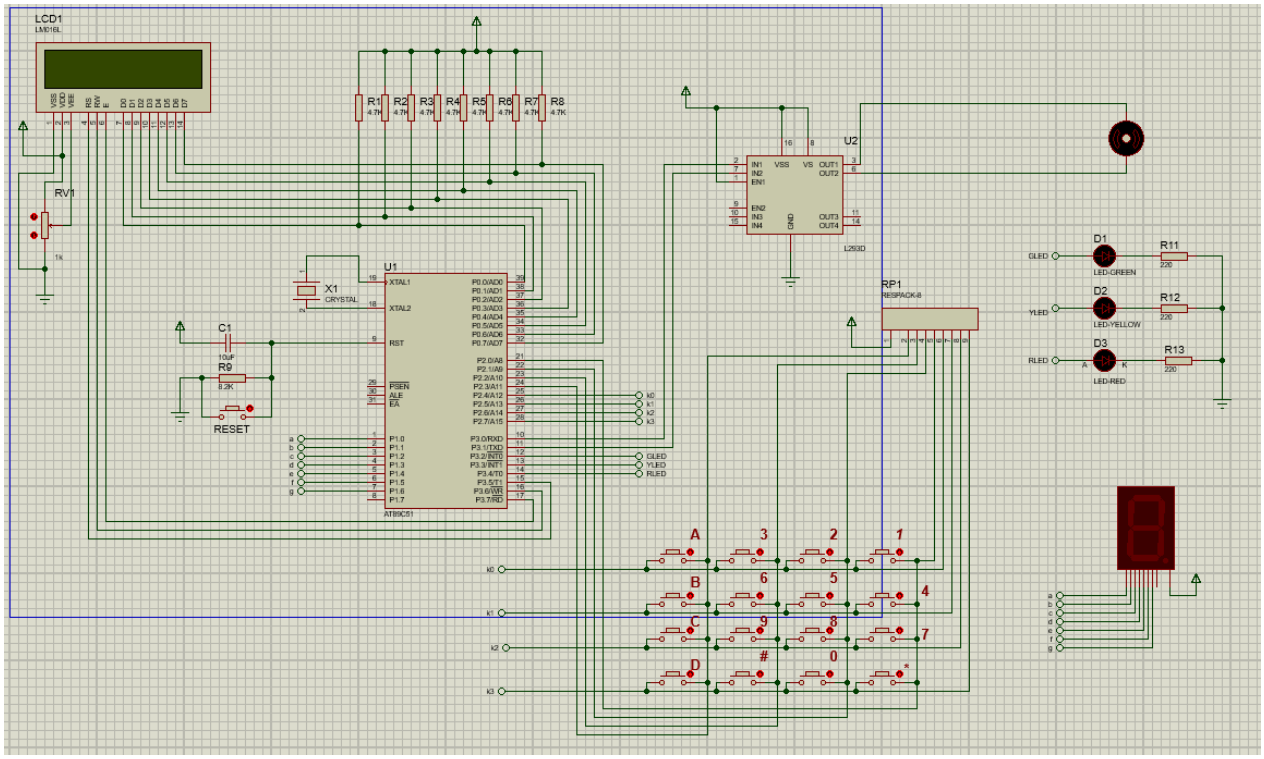
The fundamental aim of the project is to prevent unauthorized access or accidental operation of systems like smart door locks, elevators, or gates that use DC motors. Using a password in places like factories or offices ensures that only trained or authorized personnel can operate machinery or equipment. So our goal is to build a password protected DC motor using AT89S52 Microcontroller. The main objectives are-

- ➢ Performing assembly language programing in MIDE 51
- ➢ To implement a password protection system for a DC motor in software using PROTEUS.
- ➢ Interfacing a keypad and LCD with the microcontroller for entering the password and a motor driver to control the DC motor.

REQUIRED COMPONENTS:

No.	Components	Cost (BDT)
1	AT89S52 Microcontroller board (1 pc) with - 16x2 Serial LCD Module 4x4 Keypad L293D Motor Driver	3500
2	DC Motor (1 pc)	-
3	Jumper wires (Female to Female)	100
Subtotal		3600

CIRCUIT DIAGRAM:



The above circuit consists of an AT89C51(AT89S52 in hardware) microcontroller, a LM016L LCD, a L293D motor driver IC, a keypad to input password, a common anode 7-segment display and a 12V DC motor.

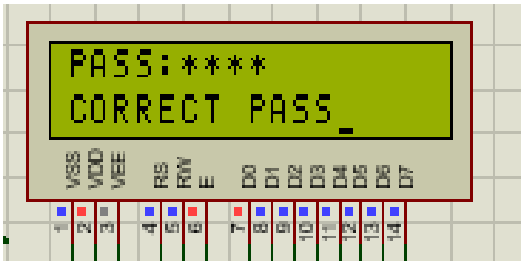
The connection of the ports -

- ➢ Port 0 (pins of P0.0- P0.7) of AT89C51 microcontroller are connected to the LCD pins (D0 - D7). As port 0 doesn't have pull up resistors like other ports so we connected the pins with external 4.7k ohms resistors.
- ➢ Port 1 (pins of P1.0- P1.7) of AT89C51 microcontroller are connected to the 7- SEG display.
- ➢ Pins of P2.0 – p2.3 of port 2 are connected to the columns and pins of P2.4 – p2.7 are connected to the rows of the keypad.
- ➢ Port 3 pins p3.0 and p3.1 are connected to the IN1 and IN2 of the L293D motor driver.
- ➢ Pins P3.2, P3.3 & P3.4 are connected to the Green, yellow and red LEDs respectively.
- ➢ Pins P3.5, P3.6 & P3.7 are connected to the RS, RW and E pins of the LCD respectively.
- ➢ The OUT1 pin of the L293D motor driver is connected to the positive terminal of the DC motor and the OUT2 pin is connected to the other terminal of the motor.

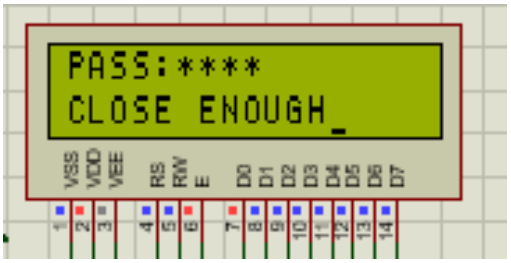
FEATURES:

Mandatory Features:

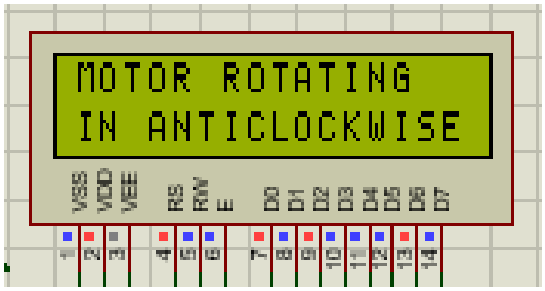
1. **Show * when a key is pressed:**
   The LCD display will show * when we press any character. To maintain privacy, only * symbols will appear on the screen. No actual character will be visible during input.

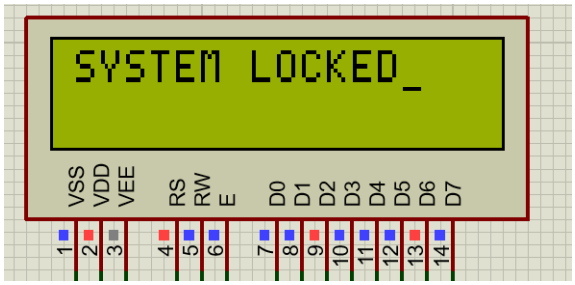


2. **Implement an "Close Enough" message when the input closely matches the password:**
   When the entered password is nearly correct, the LCD will display "Close Enough" message. This message appears if three out of four characters match the correct password. It acts as a hint to show the user they are close to the correct answer. This feature helps to improve user experience by offering guidance.



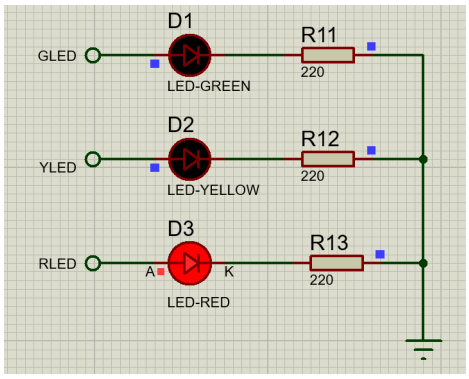
3. **Locking the system for 3 unsuccessful tries and motor will run in the reverse direction:**
   For every wrong attempt the microcontroller will keep the count and after 3 unsuccessful attempts the system will be locked and the motor will start to rotate in the anticlockwise and "MOTOR ROTATING IN ANTICLOCKWISE" and "SYSTEM LOCKED" messages will be displayed in sequence on the LCD display.



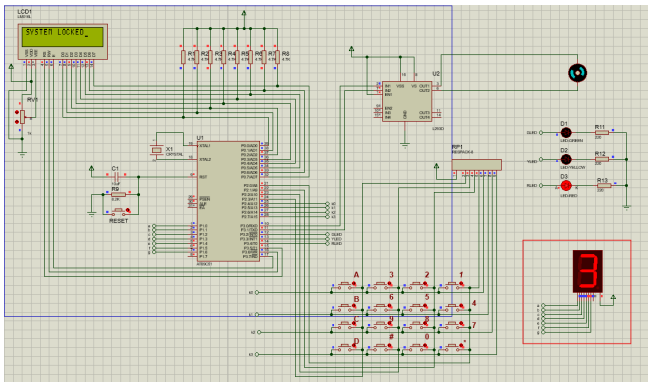


Additional Features:

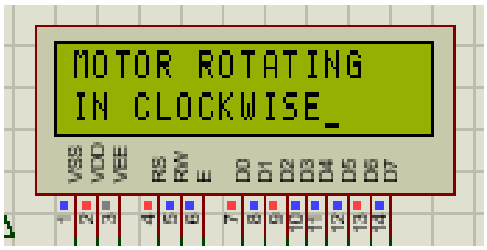
1. **Password based LED indicator:**
   The system will turn on the green LED if the password is correct. If the password is wrong, it will switch on the red LED, and if the password is close enough to the right password, it will turn on the yellow LED. The microcontroller pins P3.2, P3.3, and P3.4 of port 3 are connected to the Green, yellow, and red LEDs, respectively.

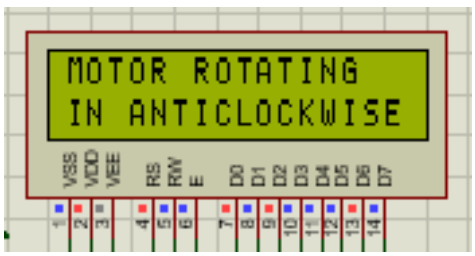


2. **Show the unsuccessful tries:**
   The unsuccessful attempts are counted and displayed on the 7 segment display. It will count up to 3 wrong passwords, after that it will start rotating in the opposite direction.



3. **Display the direction of motor rotation:**
   If we give the right password the motor will start rotating in the clockwise direction and it will display " MOTOR ROTATING IN CLOCKWISE" on the LCD display. If we attempt 3 unsuccessful passwords, the motor will start rotating in the opposite direction and it will display " MOTOR ROTATING IN ANTICLOCKWISE".





WORKING PRINCIPLE:

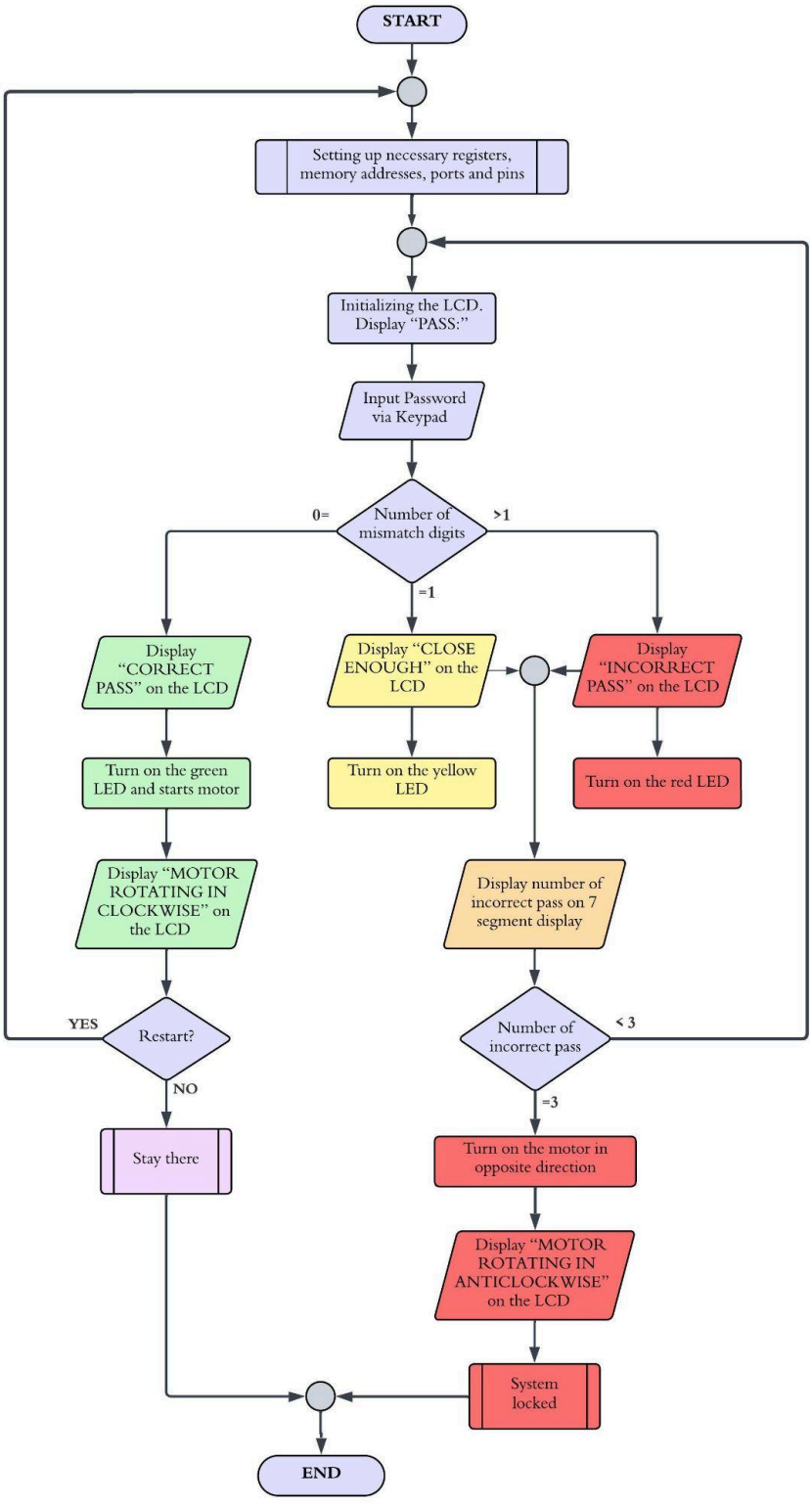


Fig: Flowchart of the whole system

Explanation of the Process:

- ➢ Program will start from 00H
- ➢ At first we initialize the pins where pin P3.5 is for RS pin, P3.6 is for the RW pin, P3.7 is for E pin, green yellow red leds are connected to the P2.3, P3.3, P3.4.
- ➢ Then we initialized R0 at 60H to store password and R1 at 50H for user-entered password.
- ➢ We initialized R5 for digit counting and R6 to check the number of digits.
- ➢ Then it stores a predefined password ('1', '3', '5', '7') in memory locations 60H to 63H.
- ➢ For keypad handling, we set columns of the keypad as inputs and rows as outputs to check if any key is pressed.
- ➢ After identifying the row from where the key was pressed, we got the ASCII value from the KCODE lookup table.
- ➢ If the user inputs any digit it will show * on the LCD display.
- ➢ Then it compares each digit to the stored password.
- ➢ If the password is correct, it shows on the LED displays “CORRECT PASS” and the motor starts to rotate clockwise and the green LED will be turned on and displays “MOTOR ROTATING CLOCKWISE” on the LCD display.
- ➢ If the password is incorrect, the LED displays “INCORRECT PASS” and the red light will be turned on.
- ➢ If the password is almost correct, the “ CLOSE ENOUGH” message will be shown on the LED display and the yellow LED will be turned on.
- ➢ The counter updates the 7-segment display to show the number of wrong attempts (1, 2, or 3). After 3 wrong attempts it will start rotating anticlockwise the LCD displays “MOTOR ROTATING ANTICLOCKWISE” and “SYSTEM LOCKED” .
- ➢ The subroutine COMNWRT is used for command to the LCD, DATAWRT used for writing the data, DELAY is used to create a small delay for debouncing.

CODE:

```
   ORG  00H
   RS  EQU P3.5
   RW  EQU P3.6
   E  EQU P3.7
   MPIN1 EQU P3.0
   MPIN2 EQU P3.1
   GLED EQU P3.2
   YLED EQU P3.3
   RLED EQU P3.4

INITIALIZE0:
   MOV P1, #0C0H  ;DISPLAY 0 ON 7 SEGMENT DISPLAY
   MOV  PSW, #00H
   MOV R2, #0   ;COUNTS INCORRECT PASSWORD

INITIALIZE:
   MOV R0, #60H ;PASSWORD LOCATION
   MOV R1, #50H ;GIVEN PASSWORD

   MOV R5, #0   ;COUNTS DIGIT MISMATCH
   MOV R6, #4   ;COUNTS 4 DIGIT PASSWORD

;STORED PASSWORD
   MOV 60H, #'1'
   MOV 61H, #'3'
   MOV 62H, #'5'
   MOV 63H, #'7'

;CLEARING OUTPUT OF LED AND MOTOR DRIVER AT THE BEGINNING
   CLR GLED
   CLR YLED
   CLR RLED
   CLR MPIN1
   CLR MPIN2

;INITIALIZING THE LCD WITH NECESSARY COMMANDS
LCD_IN:
   MOV DPTR,#LCD
DISPLAY_ON:
   CLR A
   MOVC A,@A+DPTR
   LCALL COMNWRT
   LCALL DELAY
   JZ S1
   INC DPTR
   SJMP DISPLAY_ON
```

```
;MSG FOR DIRECTION OF ROTATION OF MOTOR AFTER A
;SUCCESSFUL ATTEMPT
S1:
   JNB GLED, S2
   MOV DPTR, #MSG5
M5:
   CLR A
   MOVC A, @A+DPTR
   JZ M51
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M5

M51:
   MOV A, #0C0H
   ACALL COMNWRT
   ACALL DELAY
   MOV DPTR, #MSG6
M6:
   CLR A
   MOVC A, @A+DPTR
   CJNE A, #0, M61
   ACALL DELAY
   ACALL DELAY
   ACALL DELAY
   ACALL DELAY
   ACALL DELAY
   SJMP M8
M61:
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M6

;MSG FOR DIRECTION OF ROTATION OF MOTOR AFTER
;3 UNSUCCESSFUL ATTEMPTS
S2:
   JNB YLED, S3
S21:
   MOV DPTR, #MSG5
M52:
   CLR A
   MOVC A, @A+DPTR
   JZ M53
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M52
```

```
M53:
   MOV A, #0C0H
   ACALL COMNWRT
   ACALL DELAY
   MOV DPTR, #MSG7
M7:
   CLR A
   MOVC A, @A+DPTR
   CJNE A, #0, M71
   ACALL DELAY
   ACALL DELAY
   ACALL DELAY
   ACALL DELAY
   ACALL DELAY
   SJMP M9
M71:
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M7

S3:
   JB RLED, S21

;PRINTING 1ST MSG AFTER STARTING THE SYSTEM
   MOV DPTR, #MSG1
M1:
   CLR A
   MOVC A, @A+DPTR
   JZ K0
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M1

;MSG FOR RUNNING THE SYSTEM AGAIN AFTER ANY CORRECT PASS
M8:
   MOV A, #01H
   ACALL COMNWRT
   ACALL DELAY
   MOV A, #80H
   ACALL COMNWRT
   ACALL DELAY
   MOV DPTR, #MSG8

M81:
   CLR A
   MOVC A, @A+DPTR
   JZ K0
   ACALL DATAWRT
```

```
   ACALL DELAY
   INC DPTR
   SJMP M81

;MSG FOR SYSTEM LOCKED
M9:
   MOV A, #01H
   ACALL COMNWRT
   ACALL DELAY
   MOV A, #80H
   ACALL COMNWRT
   ACALL DELAY
   MOV DPTR, #MSG9

M91:
   CLR A
   MOVC A, @A+DPTR
   JZ LOCK
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M81

LOCK: SJMP LOCK

;KEYPAD
K0:
;SETTING THE COLUMNS AS INPUT
   SETB P2.0
   SETB P2.1
   SETB P2.2
   SETB P2.3
K1:
;SETTING THE ROWS AS OUTPUT
   CLR P2.4
   CLR P2.5
   CLR P2.6
   CLR P2.7
   MOV  A, P2      ;read all columns.ensure all keys open
   ANL  A, #00001111B  ;mask unused bits
   CJNE  A, #00001111B,K1  ;check till all keys released
K2:
   ACALL  DELAY   ;call 20ms delay
   MOV  A, P2      ;see if any key is pressed
   ANL  A, #00001111B  ;mask unused bits
   CJNE   A, #00001111B, OVER ;key pressed, await closure
   SJMP  K2 ;check is key pressed
OVER:
   ACALL  DELAY   ;wait 20ms debounce time
   MOV  A, P2      ;check key closure
```

```
   ANL  A, #00001111B  ;mask unused bits
   CJNE  A, #00001111B, OVER1 ;key pressed, find row
   SJMP  K2 ;if none, keep polling
OVER1:
   CLR P2.4
   SETB P2.5
   SETB P2.6
   SETB P2.7
   MOV  A, P2     ;read all columns
   ANL  A, #00001111B  ;mask unused bits
   CJNE  A, #00001111B, ROW_0 ;key row 0, find the column
   SETB P2.4
   CLR P2.5
   SETB P2.6
   SETB P2.7
   MOV  A, P2     ;reall all columns
   ANL  A, #00001111B  ;mask unused bits
   CJNE  A, #00001111B, ROW_1 ;key row 1, find the column
   SETB P2.4
   SETB P2.5
   CLR P2.6
   SETB P2.7
   MOV  A, P2     ;read all columns
   ANL  A, #00001111B  ;mask unused bits
   CJNE  A, #00001111B, ROW_2 ;key row 2, find column
   SETB P2.4
   SETB P2.5
   SETB P2.6
   CLR P2.7
   MOV  A, P2     ;read all columns
   ANL  A, #00001111B  ;mask unused bits
   CJNE  A, #00001111B, ROW_3 ;key row 3, find column
   LJMP  K2 ;if none, false input, repeat
ROW_0:
   MOV  DPTR,  #KCODE0  ;set DPTR=start of row 0
   SJMP  FIND    ;find column.key belongs to
ROW_1:
   MOV  DPTR, #KCODE1  ;set DPTR=start of row 1
   SJMP  FIND    ;find column.key belongs to
ROW_2:
   MOV  DPTR, #KCODE2  ;set DPTR=start of row 2
   SJMP  FIND    ;find column.key belongs to
ROW_3:
   MOV  DPTR, #KCODE3  ;set DPTR=start of row 3
FIND:
   RRC  A   ;see if any CY bit is low
   JNC  MATCH   ;if zero, get the ASCII code
   INC  DPTR;point to the next column address
   SJMP  FIND    ;keep searching
MATCH:
```

```
   CLR  A    ;set A=0 (match found)
   MOVC  A, @A+DPTR  ;get ASCII code from table

   JNB GLED, MATCH1
   CJNE A, #'0', RES
   ACALL DATAWRT
   ACALL DELAY
   SJMP $
   ;IF USER SELECTS 0 STAY HERE
RES:
   ACALL DATAWRT
   ACALL DELAY
   ACALL DELAY
   LJMP INITIALIZE0 ;IF USER SELECTS 1 RESTART THE SYSTEM


MATCH1:
   MOV @R1, A  ;STORING THE GIVEN PASS FROM 50H LOCATION

   MOV A, #'*' ;DISPLAYING * WHILE GIVING PASSWORD
   ACALL DATAWRT   ;call display subroutine
   ACALL  DELAY   ;give LCD some time

   MOV A, @R1
   ;ACALL DATAWRT  ;FOR DISPLAYING THE PASS GIVEN BY USER
   ;ACALL  DELAY

   MOV B, @R0  ;PLACING THE PASS IN B REGISTER
   CJNE A, B, L  ;COMPARING THE GIVEN PASS WITH THE ORIGINAL PASS
   INC R5 ;STORING THE NUMBER OF CORRECT DIGITS
L:
   INC R1 ;GO TO NEXT DIGIT
   INC R0 ;GO TO NEXT DIGIT
   DEC R6 ;DECREMENTING THE COUNTER FOR NUMBER OF DIGITS IN PASS
   MOV A, R6
   CJNE A, #0, NEX ;CHECKS UPTO 4 DIGITS OF PASS
   SJMP NEXTT

NEX:
   LJMP K0 ;INPUT NEXT DIGIT

NEXTT:
   MOV A, #0C0H
   ACALL COMNWRT
   ACALL DELAY
   MOV A, R5
   CJNE A, #4, CE ;CHECKING IF ALL 4 DIGITS ARE CORRECT OR NOT
   MOV DPTR, #MSG2
   SETB GLED ;GREEN LED ON
   SETB MPIN1 ;SETTING MOTOR DRIVER FOR
```

```
   CLR MPIN2  ;CLOCKWISE ROTATION

;DISPLAYING MSG FOR CORRECT PASS
M2:
   CLR A
   MOVC A, @A+DPTR
   CJNE A, #0, M21
   LJMP LCD_IN ;INITIALIZE THE LCD AGAIN FOR FURTHER MSG
M21:
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M2

CE:
   MOV A, R5
   CJNE A, #3, IC ;CHECKING IF 3 DIGITS ARE CORRECT OR NOT
   MOV DPTR, #MSG3
   SETB YLED ;YELLOW LED ON

;DISPLAYING MSG FOR CLOSE ENOUGH PASS
M3:
   CLR A
   MOVC A, @A+DPTR
   JZ FINIC
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M3

;DISPLAYING MSG FOR INCORRECT PASS
IC:
   MOV DPTR, #MSG4
   SETB RLED
M4:
   CLR A
   MOVC A, @A+DPTR
   JZ FINIC
   ACALL DATAWRT
   ACALL DELAY
   INC DPTR
   SJMP M4

FINIC:
   INC R2 ;INCREMENTING NUMBER OF INCORRECT PASS
   MOV A, R2
   CJNE A, #1, FIN1
   MOV P1, #0F9H ;DISPLAY NUMBER(1) OF INCORRECT PASS
   ;IN 7 SEGMENT DISPLAY
   LJMP INITIALIZE ;NEXT ATTEMPT AFTER AN INCORRECT PASS
```

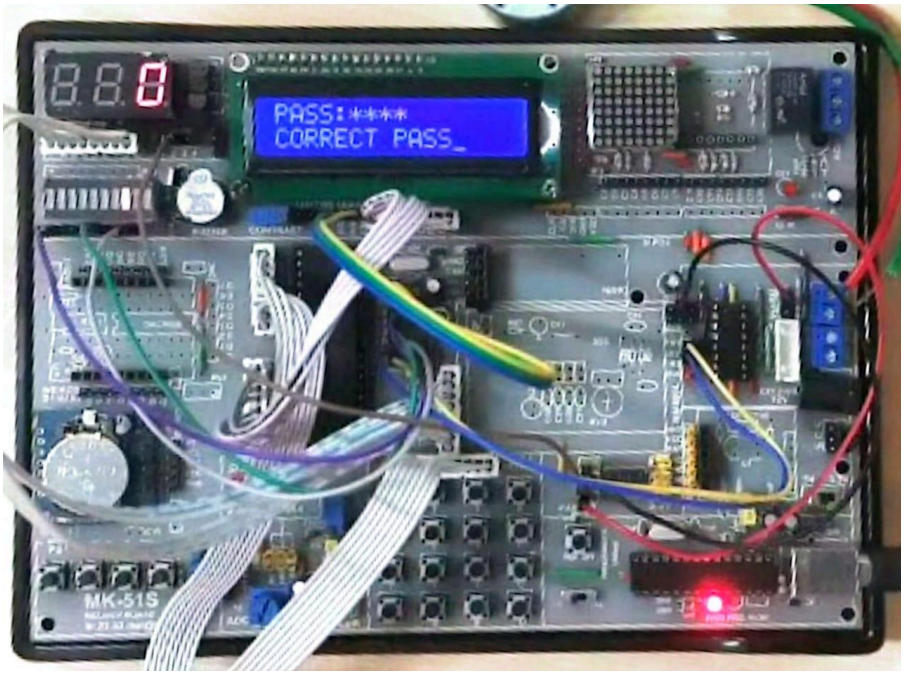
```
FIN1:
   CJNE A, #2, FIN2
   MOV P1, #0A4H ;DISPLAY NUMBER(2) OF INCORRECT PASS
   ;IN 7 SEGMENT DISPLAY
   LJMP INITIALIZE ;NEXT ATTEMPT AFTER AN INCORRECT PASS
FIN2:
   MOV P1, #0B0H ;DISPLAY NUMBER(3) OF INCORRECT PASS
   ;IN 7 SEGMENT DISPLAY
   SETB MPIN2   ;SETTING MOTOR DRIVER FOR
   CLR MPIN1 ;ANTICW ROTATION
   LJMP LCD_IN  ;INITIALIZE LCD AGAIN FOR FURTHER MSG

FIN3:
   LJMP INITIALIZE


COMNWRT:
   LCALL READY   ;send command to LCD
   MOV  P0, A     ;copy reg A to port 1
   CLR  RS  ;RS=0 for command
   CLR  RW  ;R/W=0 for write
   SETB  E  ;E-1 for high pulse
   ACALL DELAY   ;give LCD some time
   CLR  E   ;E=0 for H-to-L pulse
   RET


DATAWRT:
   LCALL READY   ;write data to LCD
   MOV  P0, A     ;copy reg A to port1
   SETB  RS ;RS=1 for data
   CLR  RW  ;R/W=0 for write
   SETB  E  ;E=1 for high pulse
   ACALL DELAY   ;give LCD some time
   CLR  E   ;E=0 for H-to-L pulse
   RET

READY:
   SETB  P0.7
   CLR  RS
   SETB  RW
WAIT:
   CLR  E
   LCALL DELAY
   SETB  E
   JB  P0.7, WAIT
   RET

DELAY:  MOV  R3, #50   ;50 or higher for fast CPUs
HERE2:  MOV  R4, #255   ;R4=255
```

```
HERE:   DJNZ  R4, HERE   ;stay untill R4 becomes 0
        DJNZ   R3, HERE2
        RET

ORG 300H
;ASCII LOOK-UP TABLE FOR EACH ROW
KCODE0: DB '1','2','3','A'     ;ROW 0
KCODE1: DB '4','5','6','B'     ;ROW 1
KCODE2: DB '7','8','9','C'     ;ROW 2
KCODE3: DB '*','0','#','D'     ;ROW 3
ORG 400H
MSG1: DB 'PASS:', 0
MSG2: DB 'CORRECT PASS', 0
MSG3: DB 'CLOSE ENOUGH', 0
MSG4: DB 'INCORRECT PASS', 0
MSG5: DB 'MOTOR ROTATING', 0
MSG6: DB 'IN CLOCKWISE', 0
MSG7: DB 'IN ANTICLOCKWISE', 0
MSG8: DB 'RESTART? 0/1___', 0
MSG9: DB 'SYSTEM LOCKED', 0
LCD : DB 38H,0EH,01,06,80H,0

   END
```

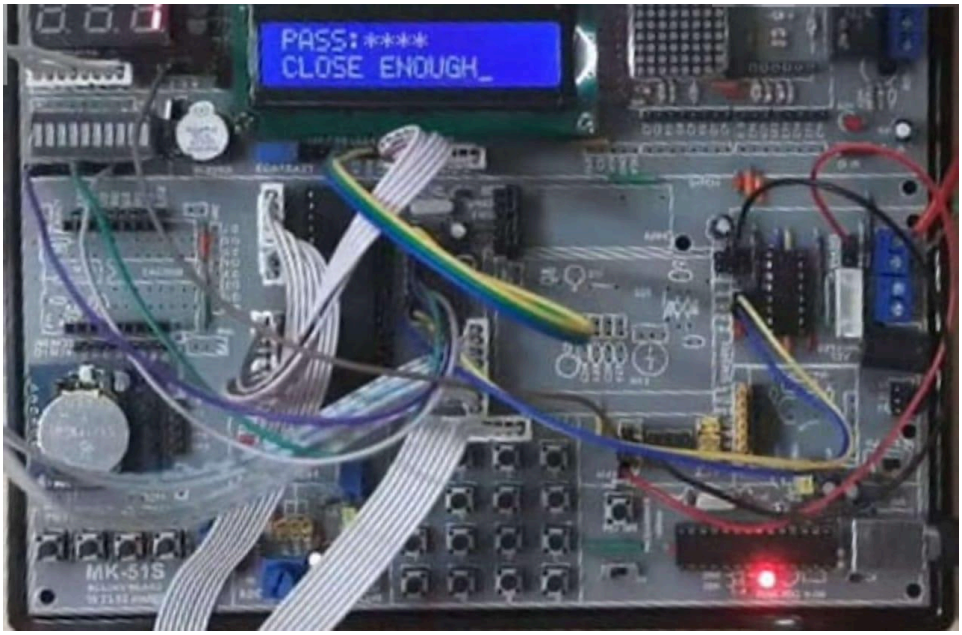
HARDWARE IMPLEMENTATION:

Mandatory Features:

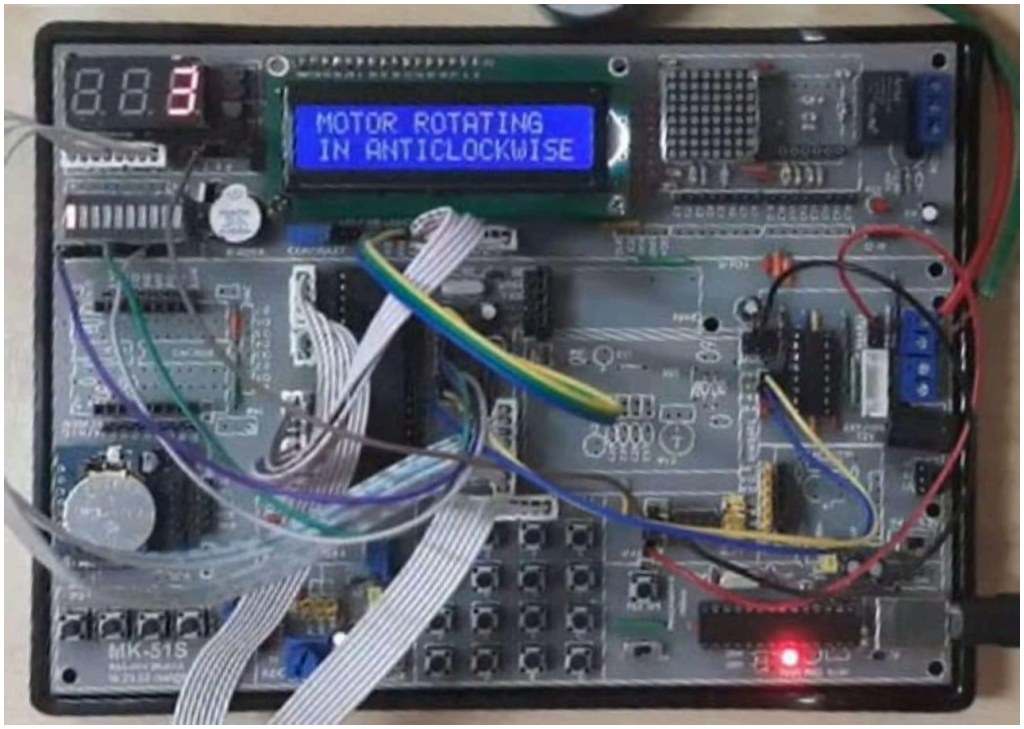
1. **Show * when a key is pressed:**

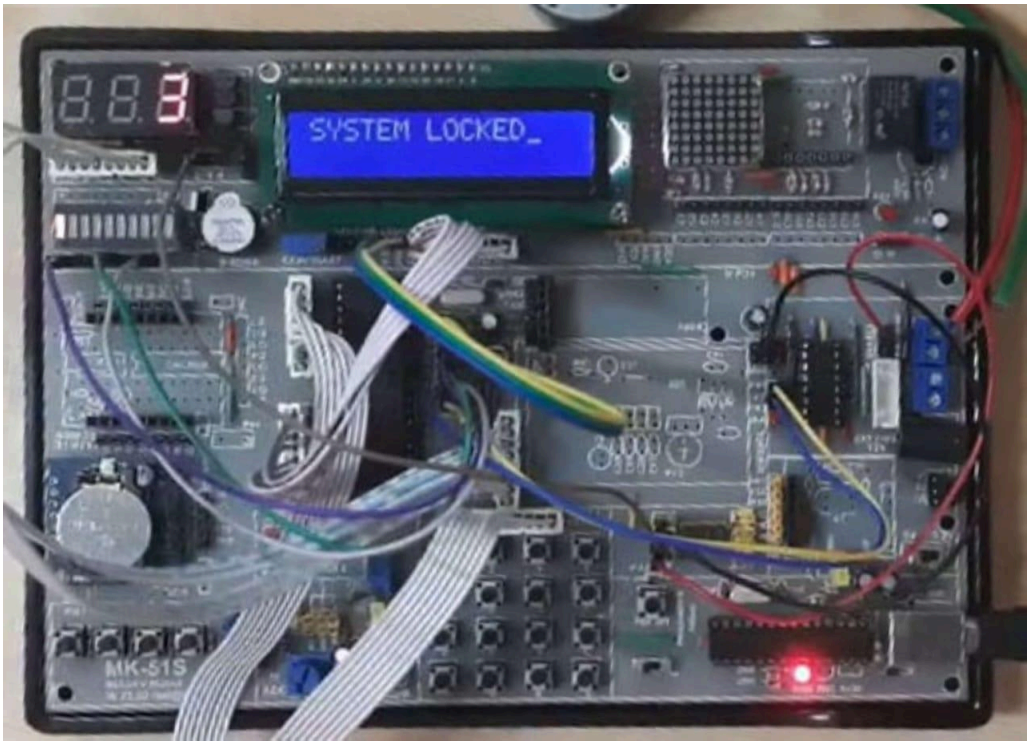


2. **Implement an “Close Enough” message when the input closely matches the password:**



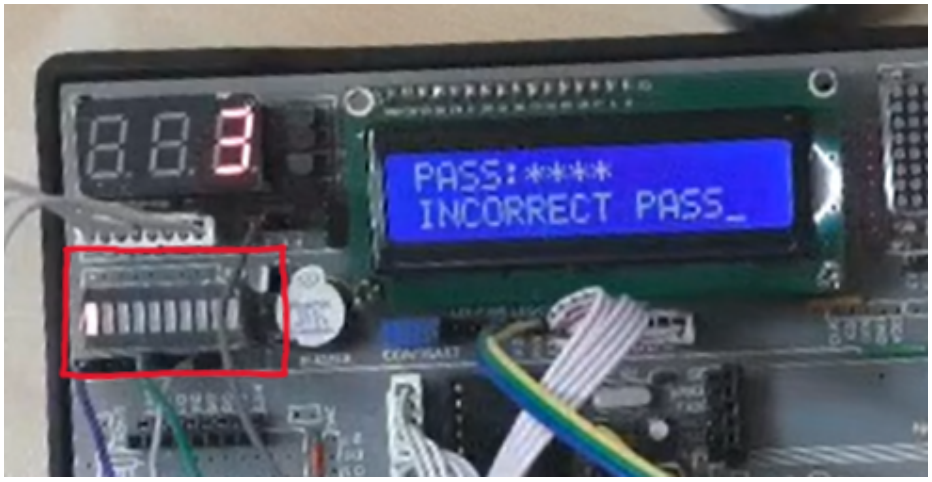
3. **Locking the system for 3 unsuccessful tries and motor will run in the reverse direction:**



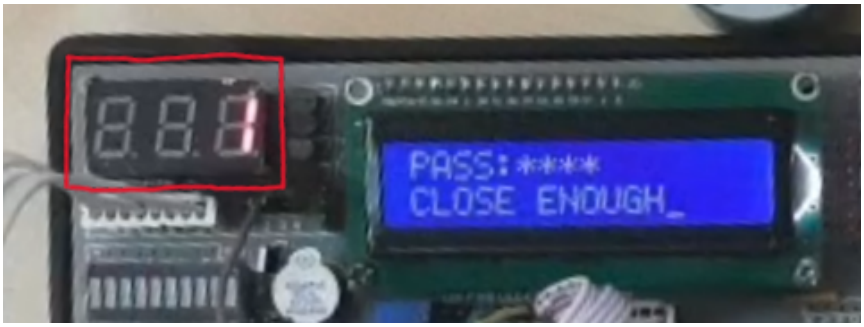


Additional Features:

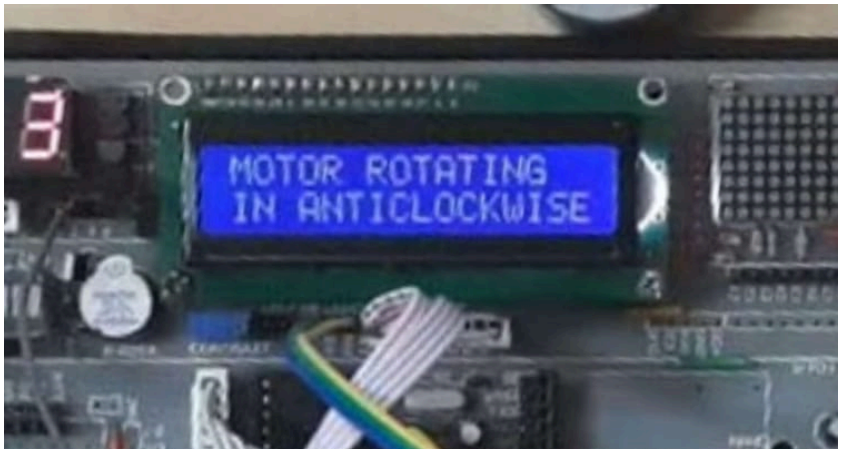
1. **Password based LED indicator:**
   The system will turn on the LED connected with P3.2 if the password is correct. If the password is wrong, it will switch on the LED, connected with P3.4 and if the password is close enough to the right password, it will turn on the LED connected with P3.3.



2. **Show the unsuccessful tries:**



3. **Display the direction of motor rotation:**



PROBLEMS FACED:

- Initially, both Port 0 and Port 1 were found to be faulty. So we sought assistance from our lab instructor for troubleshooting. Despite repeated efforts, Port 1 remained non-functional; neither the LCD nor the keypad responded when connected to it. Even after debugging and connecting a 74HC244D buffer to Port 1 to ensure unidirectional current flow, the port failed to function correctly as an input and could only operate as an output.
- The motor driver's enable pin was not labeled correctly as the ground pin was marked as enable pin. So when we connected it to the 5V, the whole board got short circuited and stopped working. After debugging we came to know that the enable labeled pin was actually the ground pin. After that we found the enable pin and connected it properly to the circuit.
- There were some labeling errors on the hardware microcontroller board. It was really a challenge for us to debug the errors and then construct the circuit. For example, both port 0 and port 1 were labelled as port 0 which left us confused. So it greatly complicated our development process.
- It's really frustrating to see that this single board cost us a huge amount of 7,000 BDT—an amount that could've bought 7 Arduino Uno R3, which are not only packed with more advanced features but also far more reliable with minimal bugs. Yet here we are, stuck with this problematic board that's been causing the same recurring errors year after year. For such a hefty price, we expected at least a smooth, hassle-free experience, but instead, we're left debugging persistent issues that should've been resolved long ago. It just doesn't feel like we're getting our money's worth, especially when this board barely gets used beyond a single project. What a waste of resources for something so unreliable!
- During the coding phase, we encountered an unexpected issue where the R3 and R4 registers failed to increment their values properly. These registers were originally intended to track the password length and count mismatched digits, but their malfunction prevented the system from correctly verifying passwords. To troubleshoot this, we

displayed the contents of R3 and R4 on the LCD, which helped us identify the problem. After some trial and error, we found that R5 and R6 could reliably serve as alternative counters, resolving the issue and restoring the system's functionality.

CONCLUSION:

In conclusion, this project successfully achieved the goal of creating a password-protected motor using the 8051 microcontroller. By combining secure access control with LCD feedback, motor operation and LED indicators, we created a system that is both efficient and user-friendly. The main features like password generation, checking, and security evaluation were handled smoothly through our code. Throughout the process, debugging and hardware troubleshooting helped us gain a much deeper understanding of assembly language and system design. Although we faced several technical challenges, working through them made the experience even more valuable. Our final system, tested with a 12V DC motor, can easily be adapted for larger industrial machines which will ensure the safety of the motor. Overall, this project helped us connect practical skills with theoretical knowledge and to realize how important careful design and testing are in building secure systems.