ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

NAME	: K M Istiaque	STUDENT ID: 200021220
NAME	: Mayesha Aanan Prapti	STUDENT ID: 200021224
NAME	: Zabin Tazrin Nashita	STUDENT ID: 200021230
NAME	: Md. Muhidur Rahman	STUDENT ID: 200021242
NAME	: Kouyate Mouhamed	STUDENT ID: 200021256
NAME	: Mohamed Saed Jama	STUDENT ID: 200021260

DEPARTMENT : EEE SECTION & GROUP: B2

PROJECT GROUP: 03

DATE OF SUBMISSION : 28.04.25

COURSE NO. : EEE 4706

COURSE TITLE : Microcontroller Based System Design Lab

PROJECT TITLE : Temperature Controlled Motor

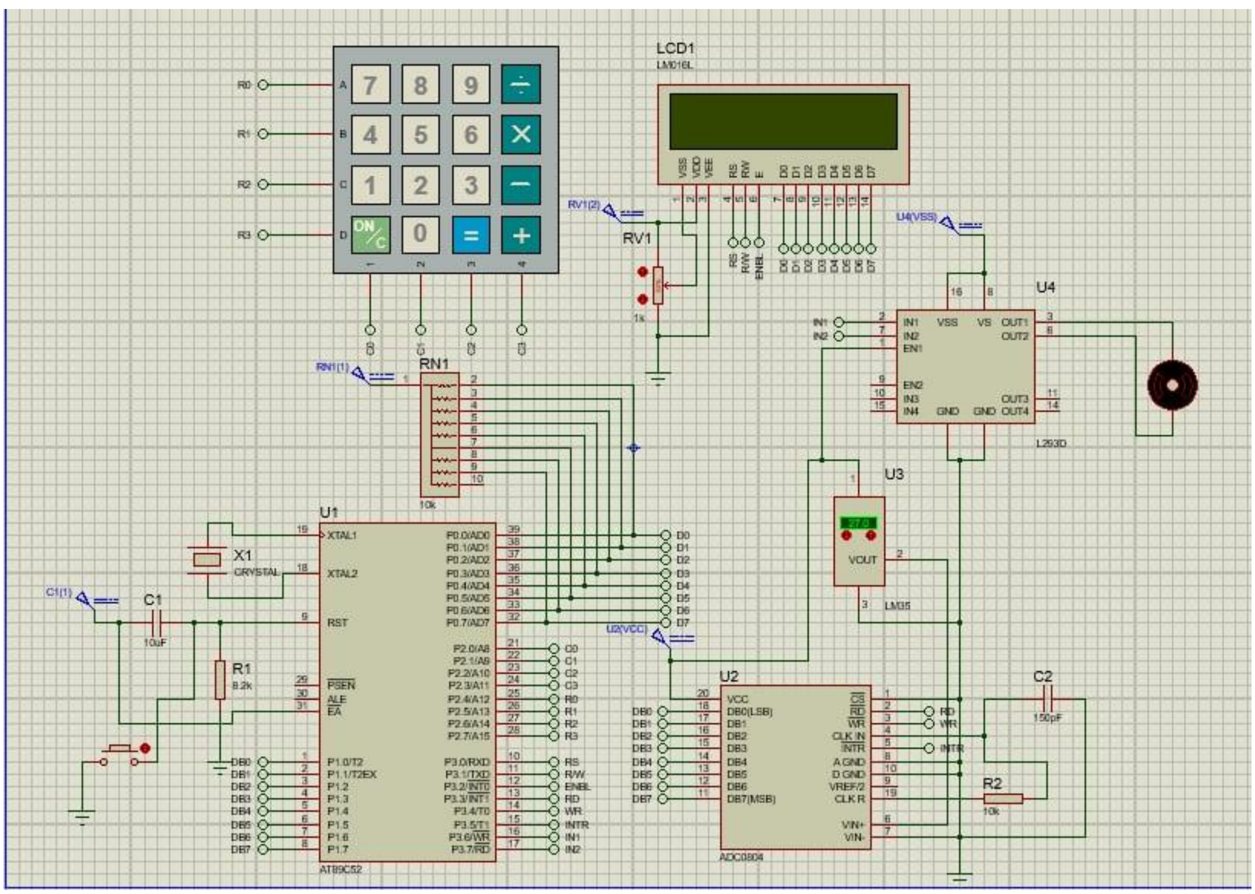
Objective

The objective of this project is to use an 8052 Microcontroller Development to control a DC motor based on temperature. There are three stages of speed: low, medium, and high. The user will define three temperatures: minimum, medium, and maximum. If the temperature found by the sensor is between minimum and medium, the motor runs on low speed. If the temperature is higher than medium but lower than high, then the motor runs on medium speed, and finally, if the temperature is higher than maximum, the motor runs on high speed. There's an additional critical temperature, and if this threshold is crossed, the motor shuts down with an alert message.

Required Components:

Component Name	Amount	Price (BDT)
Double Shaft DC motor	3	65*3=195
L298N H-bridge Motor Driver	1	183
Jumper Wires	-	-
Battery	1	-
8052 board	1	7000 (apprx)

Circuit Diagram



As we can see, the circuit consists of an AT89C52 microcontroller chip, a keypad, a temperature sensor, a motor driver, a motor, an LCD display, and a 0804 ADC. The keypad is connected to the microcontroller chip using pins 2.0-2.7 or port 2, the LCD is connected to 0.0-0.7 or port 0, and the ADC is connected using port 1. We use port 3 for connections to the motor driver. Pin 3.6 and 3.7 basically carry the output of the motor, and pins 3.0-3.5 are used to enable the LCD display as well as the ADC. There is a reset button that works to shut down the motor incases of emergencies.

Features:

Mandatory Features

1. Temperature Sensing:

- The microcontroller reads temperature data from an ADC connected to a temperature sensor (e.g., LM35).
- The sensor continuously sends analog voltage, which the μC samples and converts into a digital temperature reading.

2. Keypad Input for Thresholds:

- The user inputs three temperature thresholds — TMIN (minimum), TMED (medium), and TMAX (maximum) — using a 4x4 keypad.
- The values are stored in memory and are later used for motor speed control decisions.

3. Motor Speed Control:

- Based on the real-time temperature, the motor runs at three distinct speed stages:
    - If T < TMIN, motor stays OFF.
    - If TMIN < T < TMED, motor runs at LOW speed.
    - If TMED < T < TMAX, motor runs at MEDIUM speed.
    - If T > TMAX, motor runs at HIGH speed.

4. Critical Temperature Handling:

- A critical temperature (TCRIT) is predefined.
- If T > TCRIT, the system immediately stops the motor and displays an ALERT message.
- This is a protective shutdown to prevent overheating damage.

5. LCD Display:

- The system continuously displays:
    - The current temperature.
    - The current motor speed (e.g., “LOW SPEED”, “HIGH SPEED”, etc.)
    - Alert messages if critical conditions are reached.

Extra Features:

1. Manual Reset Button (in Proteus):

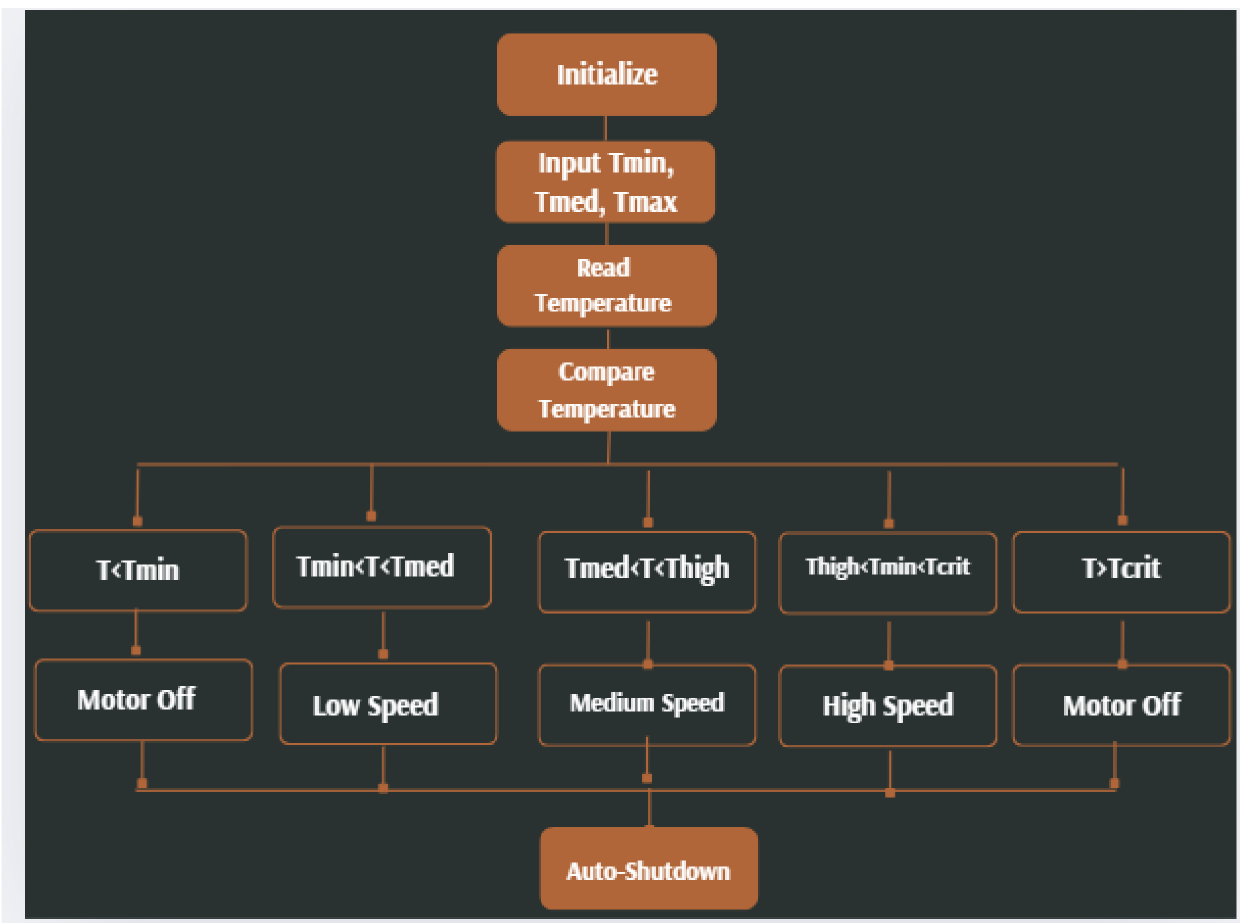
- A RESET button is added in the Proteus simulation.
- It allows the user to restart the entire system manually — reinitializing thresholds and resuming normal operation after a critical shutdown.

2. Auto Shutdown Timer:

- An internal timer continuously runs.
- If the motor runs for too long (e.g., several minutes) without conditions changing significantly, the system automatically shuts down the motor to save power and protect the hardware.

Working Principle:

A flowchart of the algorithm is given below:



1. Initialize System

- LCD is set up to show messages.
- Ports (P0, P1, P2, P3) configured for motor, keypad, ADC.

2. Get Temperature Limits

- The user uses the keypad to enter 3 temperatures: Tmin, Tmed, Tmax.
- These are stored in memory addresses (`40H`, `41H`, `42H`).

3. Start Reading Temperature

- The microcontroller communicates with the ADC (Analog-to-Digital Converter).
- ADC converts analog sensor signal into digital temperature data.

4. Display Temperature

- The temperature is shown on the LCD in two-digit format.

5. Compare and Control Motor

- The measured temperature is compared against Tmin, Tmed, Tmax, and Tcritical.
- Based on range:
  - Below Tmin → Motor OFF
  - Between Tmin and Tmed → Motor LOW speed
  - Between Tmed and Tmax → Motor MEDIUM speed
  - Between Tmax and Tcritical → Motor HIGH speed
  - Above Tcritical → Motor STOP + Alert

6. Auto-Shutdown

- A timer is running (using Timer0).
- After a delay, the motor automatically shuts off to save energy or for safety.

7. Emergency Reset

- A hardware reset button can immediately turn off the motor if pressed.
- Useful in critical situations.

8. Continuous Monitoring

- The system constantly monitors and loops back to read new temperature values and adjust the motor speed accordingly.

Code

```
ORG  00H
   RS  EQU P3.0
```

```
   RW  EQU P3.1
   E  EQU P3.2
   RDADC EQU P3.3
   WRADC EQU P3.4
   INTR EQU P3.5
   CLR P3.6
   CLR P3.7
   MOV  R0, #30H  ; Pointer to store first ASCII digit
   MOV  R2, #00H  ; Counter for enter
   MOV R1, #40H ;Pointer to store first temperature
   MOV 43H,#80H; critical temperature at 43H
LCD_IN:MOV  A, #38H   ; init. LCD 2 lines, 5x7 matrix
   ACALL COMNWRT   ; call command subroutine
   ACALL  DELAY   ; give LCD some time
   MOV   A, #0FH   ; display on, cursor on
   ACALL COMNWRT   ; call command subroutine
   ACALL  DELAY   ; give LCD some time
   MOV  A, #01    ; clear LCD
   ACALL COMNWRT   ; call command subroutine
   ACALL  DELAY   ; give LCD some time
   MOV  A, #06H
   ACALL  COMNWRT
   ACALL  DELAY
   MOV A, #80H   ;cursor at line 1 postion 5
LCALL COMNWRT  ;call command subroutine
   LCALL  DELAY
   CLR A
   MOV DPTR,#M1
   ACALL MSG
   MOV A, #0C0H   ;cursor at line 1 postion 5
   LCALL COMNWRT  ;call command subroutine
   LCALL  DELAY

KEYPAD:MOV   A, #0FH     ; make P1.0 - P1.3 input
   MOV  P2, A
K1: 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 if 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:  SETB P2.7
   SETB P2.6
   SETB P2.5
   CLR P2.4 ; Ground row 0 (P1.4 = 0), keep P1.0-P1.3 as inputs
   MOV  A, P2
   ANL  A, #11101111B    ; Mask unused bits
CJNE A, #11101111B  , ROW_0
   SETB P2.7
   SETB P2.6
```

```
    SETB P2.4
    CLR P2.5; Ground row 1 (P1.5 = 0)
    MOV  A, P2
    ANL  A, #11011111B
    CJNE A, #11011111B , ROW_1
    SETB P2.7
    SETB P2.5
    SETB P2.4
    CLR P2.6 ; Ground row 2 (P1.6 = 0)
    MOV  A, P2
    ANL  A, #10111111B
    CJNE A, #10111111B , ROW_2
    SETB P2.6
    SETB P2.5
    SETB P2.4
    CLR P2.7 ; Ground row 3 (P1.7 = 0)
    MOV  A, P2
    ANL  A, #01111111B
    CJNE A, #01111111B, ROW_3
    LJMP  K2
ROW_0:  MOV  DPTR,  #KCODE0  ; set DPTR=start of row 0
    CLR P2.7
    CLR P2.6
    CLR P2.5
    CLR P2.4
    SJMP  FIND    ; find column key belongs to
ROW_1:  MOV  DPTR, #KCODE1  ; set DPTR=start of row 1
    CLR P2.7
    CLR P2.6
    CLR P2.5
    CLR P2.4
    SJMP  FIND    ; find column key belongs to
ROW_2:  MOV  DPTR, #KCODE2  ; set DPTR=start of row 2
    CLR P2.7
    CLR P2.6
    CLR P2.5
    CLR P2.4
    SJMP  FIND    ; find column key belongs to
ROW_3:  MOV  DPTR, #KCODE3  ; set DPTR=start of row 3
    CLR P2.7
    CLR P2.6
    CLR P2.5
    CLR P2.4
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
    MOVC  A, @A+DPTR  ; get ASCII code from table
    ACALL DATAWRT
    ACALL DELAY
    ANL A,#0FH; 33H and 0FH = 03H
    MOV @R0,A
    INC R0
    INC R2
    CJNE R2,#2,FOURDIGIT
    ACALL FIRSTLINE
```

```
 MOV DPTR,#M2
   ACALL MSG
   ACALL SECONDLINE
   SJMP NXT
FOURDIGIT:CJNE R2,#4,SIXDIGIT
   ACALL FIRSTLINE
   MOV DPTR,#M3
   ACALL MSG
   ACALL SECONDLINE
   SJMP NXT
SIXDIGIT:CJNE R2,#6,NXT
   ACALL STORETEMP
   ACALL FIRSTLINE
   MOV DPTR,#M4
   ACALL MSG
   SJMP ADCCODE;R2 is free register now
NXT:LJMP KEYPAD

ADCCODE:MOV P1, #0FFH
    SETB INTR
BACK:CLR WRADC
    SETB WRADC
CHECK:JB INTR,CHECK
    CLR RDADC
    MOV A,P1
    ACALL CONVERSION
    MOV A,#0CH
    ACALL COMNWRT
    ACALL DELAY
    ACALL SECONDLINE
    ACALL DISPLAYDATA
    SJMP COMPARE
    SETB RDADC
    SJMP BACK

COMPARE:CLR C
   MOV A,R7;A=sensor temp
   MOV R1,#40H
   MOV B,@R1;Tmin
   CJNE A,B,BELOWTMIN
BELOWTMIN:JNC RANGE2;A>B=sensor temp>Tmin
   CLR P3.6;sensor temp<Tmin,motor off
   ACALL FIRSTLINE
   MOV DPTR,#M5
   ACALL MSG
   SJMP $
RANGE2:INC R1
   MOV B,@R1;Tmed
   CJNE A,B,TMINTMED
TMINTMED:JNC RANGE3;A>B=sensor temp>Tmed
   ACALL FIRSTLINE
   MOV DPTR,#M6
   ACALL MSG
LOW:ACALL LOW_SPEED;A<B=sensor temp<Tmed;Tmin<sensor temp
   ACALL AUTO_SHUT
   SJMP LOW
RANGE3:INC R1
```

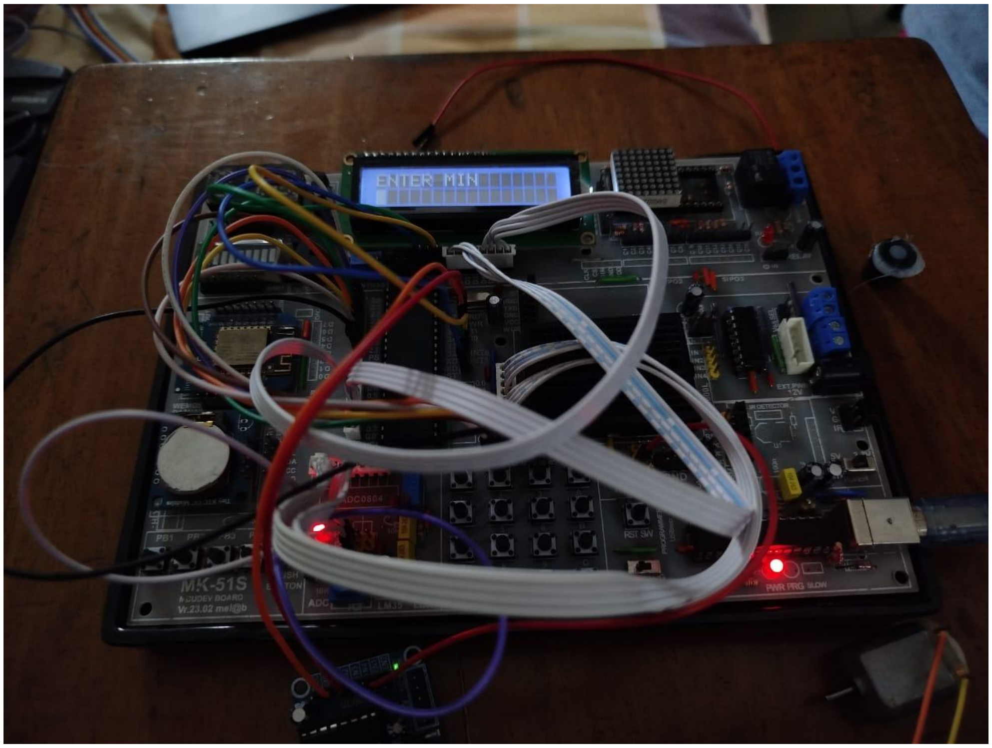
```
   MOV B,@R1;Tmax
   CJNE A,B,TMEDTMAX
TMEDTMAX:JNC RANGE4;A>B=sensor temp>Tmax
   ACALL FIRSTLINE
   MOV DPTR,#M7
   ACALL MSG
MED:ACALL MED_SPEED;A<B= sensor temp<Tmax;Tmed<sensor temp
    ACALL AUTO_SHUT
   SJMP MED
RANGE4:INC R1
   MOV B,@R1;Tcritical
   CJNE A,B,TMAXTCRIT
TMAXTCRIT:JNC STOPALERT;A>B=sensor temp>Tcritical
   ACALL FIRSTLINE
   MOV DPTR,#M8
   ACALL MSG
HIGH:ACALL HIGH_SPEED;A<B= sensor temp<Tcritical;Tmax<sensor temp
    ACALL AUTO_SHUT
   SJMP HIGH
STOPALERT:CLR P3.6
   ACALL FIRSTLINE
   MOV DPTR,#M9
   ACALL MSG
   SJMP $
;AUTO SHUTDOWN ROUTINE-------------------
AUTO_SHUT:
    MOV TMOD, #01H
    MOV R3, #20
    OUTER_LOOP_T: MOV R4, #250
    START_TIMER: MOV TH0, #0B6H
    MOV TL0, #00H
    SETB TR0
    WAIT_TIMER: JNB TF0, WAIT_TIMER
    CLR TR0
   CLR TF0
    DJNZ R4, START_TIMER
    DJNZ R3, OUTER_LOOP_T
    CLR P3.6
    SJMP $
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

MSG: CLR A
   MOVC  A,@A+DPTR
   JZ  DONE
   LCALL  DATAWRT
   LCALL  DELAY
   INC  DPTR
   LJMP  MSG
DONE:RET

FIRSTLINE:MOV  A, #01    ; clear LCD
   ACALL COMNWRT   ; call command subroutine
   ACALL  DELAY   ; give LCD some time
   MOV  A, #06H
   ACALL  COMNWRT
   ACALL  DELAY
   MOV A, #80H   ;cursor at line 1 postion 5
   LCALL COMNWRT  ;call command subroutine
   LCALL  DELAY
   RET

SECONDLINE:MOV  A, #0C0H    ; clear LCD
   ACALL COMNWRT   ; call command subroutine
   ACALL  DELAY
   RET

STORETEMP:MOV R0,#30H
   MOV R5,#3
STORELOOP:MOV A,@R0
    MOV B,#10
    MUL AB
    MOV R6,A;R6 as temporary space
    INC R0
    MOV A,@R0
    ADD A,R6
    MOV @R1,A;40H=Tmin,41H=Tmed,42H=Tmax
    INC R1
    INC R0;R0 is free register
    DJNZ R5,STORELOOP;R5 is free register now
    RET

CONVERSION:MOV B,#2
   MOV A,P1
```

```
   MUL AB
   MOV R7,A;sensor temp in R7
   MOV B,#10
   DIV AB
   ORL A,#30H    ; Convert to ASCII for display
   MOV 70H,A
   MOV A,B
   ORL A,#30H
   MOV 71H,A
   RET

DISPLAYDATA:MOV A ,70H
   ACALL DATAWRT
   ACALL DELAY
   MOV A,71H
   ACALL DATAWRT
   ACALL DELAY
   RET

LOW_SPEED:SETB P3.6
    MOV R6, #30
    ACALL DELAY2
    CLR P3.6
    MOV R6, #70
    ACALL DELAY2
    RET
MED_SPEED:SETB P3.6
    MOV R6, #60
    ACALL DELAY2
    CLR P3.6
    MOV R6, #40
    ACALL DELAY2
    RET
HIGH_SPEED: SETB P3.6
    MOV R6, #90
    ACALL DELAY2
    CLR P3.6
    MOV R6, #10
    ACALL DELAY2
    RET
DELAY2:
H11: MOV R2, #100
H12: MOV R5, #255
H13: DJNZ R5, H13
    DJNZ R2, H12
    DJNZ R6, H11
    RET

KCODE0: DB '7','8','9','*'    ; ROW 0
KCODE1: DB '4','5','6','*'   ; ROW 1
KCODE2: DB '1','2','3','*'   ; ROW 2
KCODE3: DB 'S','0','T','*'    ; ROW 3
M1: DB 'ENTER MIN',0
M2: DB 'ENTER MED',0
M3: DB 'ENTER MAX',0
M4: DB 'ROOM TEMPERATURE',0
M5: DB 'MOTOR OFF',0
```

```
M6: DB 'LOW SPEED',0
M7: DB 'MEDIUM SPEED',0
M8: DB 'HIGH SPEED',0
M9: DB 'CRITICAL TEMP',0
END
```

Hardware Implementation



Problems Faced

Software: While coding, our initial approach was as such: after the simulation has been run, we would need to press the enter button to start the keypad and then equal button after each temperature to store the values. However, because of this, a blank value was always being stored in our target registers, leading to high speed operation of motor regardless of input. This was solved by getting rid of the buttons, and configuring so that the program starts taking values off the get go.

Hardware: The ADC of our microcontroller board was not working properly. The LM35 temperature sensor cannot seem to detect the temperature, hence we couldn't get the room temperature reading.

Conclusion:

On the basis of our simulation results and hardware progress, we can conclude that we have built a system of DC motor that can be controlled by the temperature.