Microcontroller Based System Design

Project Report

Alarm Clock

Submitted by:

Shahriar Tarvir – 180021133

Fardin Sohel – 180021103

Nazmus Sadat – 180021121

Atik Abrar Sourav – 160021049

Abdullahi Hamza Lawal - 180021147

Introduction

Time and tide waits for none. So it is very important for everyone to know what time it is and keep track of time. And to do that clocks are essential.

Clocks are part and parcel of our day to day life. They help us not only to keep track of time but their alarm feature notifies us at the specified times when we are busy doing one thing and something else needs our attention.

Clocks can be both analog and digital. Digital clocks are used almost everywhere. From smartphones, smartwatches to computers everything has a real time clock module installed within. Digital clocks are used as wall clocks and wristwatches as well.

A digital alarm clock has been designed and implemented in this project. All the features of the clock and alarm have been coded using assembly language and the target device for hardware implementation was a 8051 microcontroller.

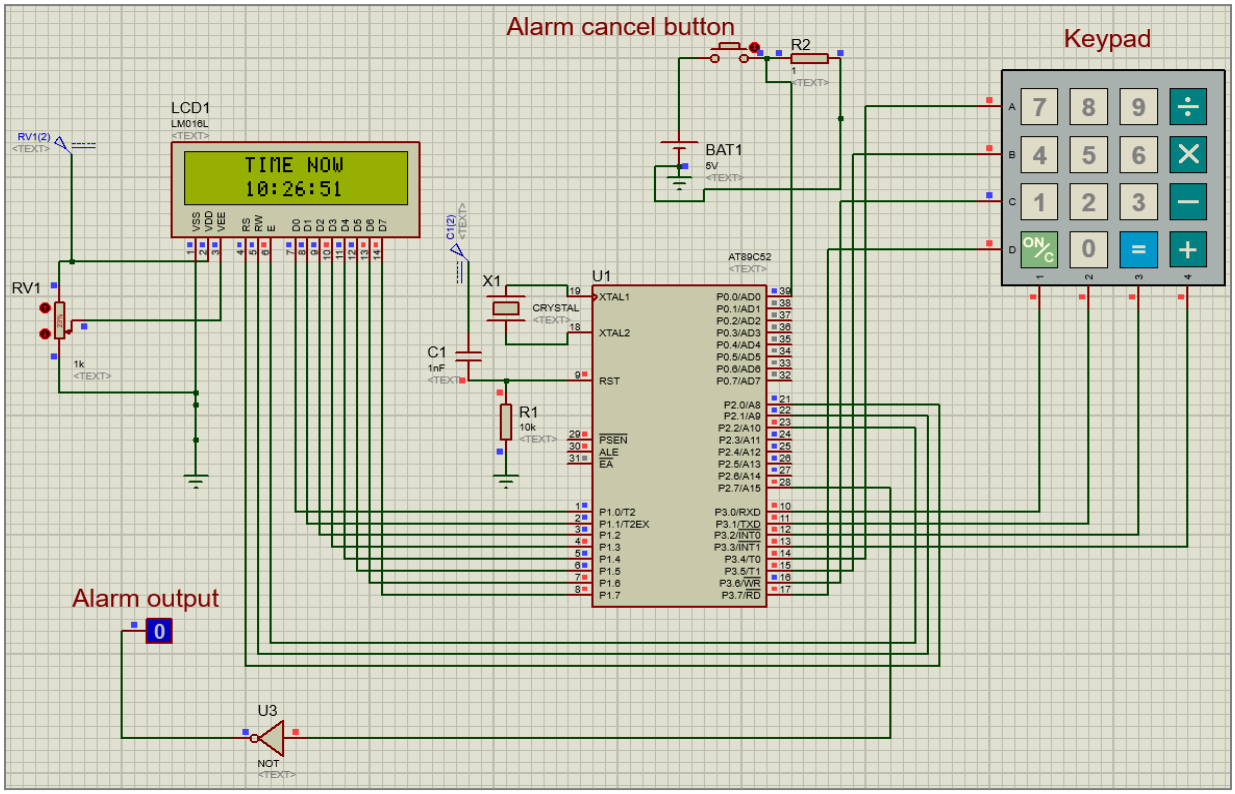
Objective

The objective of this project was to develop a Digital Clock on an 8051 microcontroller using assembly language with the feature of ringing alarm at specified time. Using the keypad to take user input and showing the time on either LCD or 7-segment LED were also requirements of this project.

Required Components

Component	Total Cost
8051 microcontroller	7000 taka
LCD	
Buzzer	
LED	
Jumper wires	
Resistors	

Circuit Diagram



Mandatory Features

Digital clock:

A 24h format digital clock has been developed using timers and delays.

Display:

Current time of the clock is shown on a LCD. Instructions for the user inputs are also displayed there.

Alarm:

Users can set a time as alarm time and when the clock reaches that time the user is notified by a buzzer or a lit up LED.

User input:

Users can provide input of current time, alarm time, alarm snooze count and snooze interval (extra features) using the keypad. The system is designed to take all these inputs before starting the clock.

Extra Features

Snooze Alarm:

Users can choose to repeat the alarm at a specified time interval. The number of times that the alarm will be repeated and the time interval between each of those alarms are taken as input before starting the clock.

If a user sets alarm at 10:30 and chooses a snooze count of 3 and snooze interval of 5 min then the program will ring alarm at 10:30, 10:35, 10:40 and 10:45.

Input verification:

Users cannot input invalid time as current time or alarm time. Each of the digits entered by the users are verified. If an invalid input is given then the program doesn't move forward. It'll keep asking for valid input.

A user will not be able to enter 12:72 as an input for current time or alarm time. If anyone tries to do so after entering 7 the program will be stuck there and wait for the user to provide a valid input between 0 to 5. The it will move on to take the next digit input.

Alarm cancel:

By pressing a push button users can terminate all the alarms specified beforehand. If this button is pressed, no alarm will be rung in the future and if an alarm is currently triggered, it'll stop. But the clock will keep running as it is.

Working Principle

The program can be divided into four components.

1. The clock
2. RInging Alarm
3. Repeating Alarm
4. Stopping Alarm

The working principle of these components are described below.

The clock:

The clock has been implemented using time delays. After every seconds Least significant digit of the seconds counter gets incremented by 1. After it goes to 9 it becomes 0 again in the next second and the Most significant digit of the seconds counter gets incremented by 1. This continues until the two digits

showing the seconds value become 59.

After that the least significant digit of minutes counter gets incremented by 1. Both the digits of minutes follow the same principle as seconds digits. After 59 they become 00.

The increment of hour digits is a bit more complicated. If the most significant digit of hour is 0 or 1 that allows the least significant digit to go up to 9. But if it is 2, then the least significant digit can go only up to 3 since for a 24h clock after 23:59 it's 00:00 again.

Ringing Alarm:

Every second program checks if the current time of the clock and the pre specified alarm time matches or not. If they match then the alarm is triggered and it keeps ringing the alarm for 1 min unless the alarm cancel button is pressed.

The alarm control block first checks if the most significant digit of hour for both current time and alarm time match. If they do then it checks if the least significant digits of hour match. Similarly the minute digits are checked as well. If all four of them match then the alarm is triggered.

Repeating Alarm:

After the alarm has been rung once the alarm time needs to be updated in case of a repeating alarm. The program checks if the alarm has been triggered the number of times specified by the user. If yes then it does nothing, if not then it updates the alarm time according to the snooze interval specified by the user.

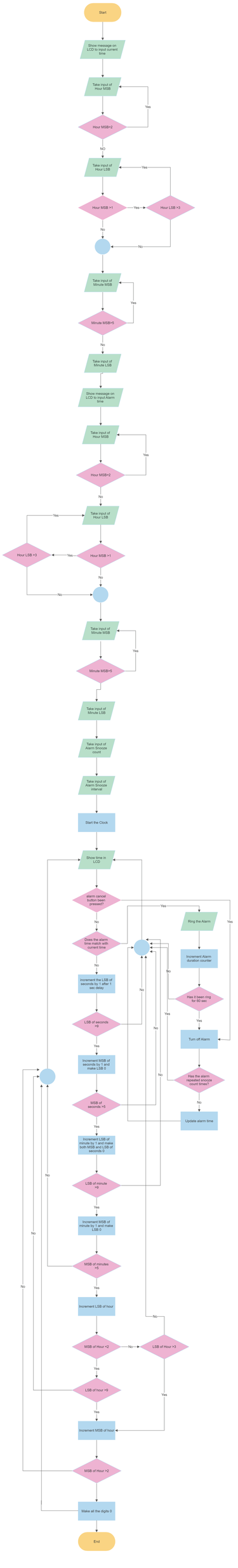
If an alarm has been set at 10:30 and the snooze time interval is 5 min then after ringing the alarm at 10:30, the program will update the alarm time to 10:35 and then ring the alarm at 10:35. This will continue until the alarm snooze count provided by the user has been reached.

Stopping the alarm:

If the alarm time matches the current time of the clock, the program checks if the alarm cancel button has been pressed or not before ringing the alarm. If it has been pressed then it does not ring the alarm.

Since the alarm functionality checks for the alarm time match every second, if the alarm cancel button is pressed while an alarm is ringing it'll stop immediately. Pressing the cancel button cancels the snooze alarm feature as well but the clock functionality stays uninterrupted . Current time always stays visible on the LCD.

Flowchart


Start
Show message on LCD to input current time
Take input of Hour MSB
Hour MSB>2
Yes
NO
Take input of Hour LSB
Hour MSB >1
Hour LSB >3
Take input of Minute MSB
Minute MSB>5
Take input of Minute LSB
Show message on LCD to input Alarm time
Take input of Hour MSB
Hour MSB>2
Take input of Hour LSB
Hour LSB >3
Hour MSB >1
Take input of Minute MSB
Minute MSB>5
Take input of Minute LSB
Take input of Alarm Snooze count
Take input of Alarm Snooze interval
Start the Clock
Show time in LCD
alarm cancel button been pressed?
Ring the Alarm
Increment Alarm duration counter
Has it been ring for 60 sec
Turn off Alarm
Does the alarm time match with current time
increment the LSB of seconds by 1 after 1 sec delay
LSB of seconds >9
Increment MSB of seconds by 1 and make LSB 0
MSB of seconds >5
Has the alarm repeated snooze count times?
Update alarm time
Increment LSB of minute by 1 and make both MSB and LSB of seconds 0
LSB of minute >9
Increment MSB of minute by 1 and make LSB 0
MSB of minutes >5
Increment LSB of hour
MSB of Hour <2
LSB of Hour >3
LSB of hour >9
Increment MSB of hour
MSB of Hour >2
Make all the digits 0
End

Program

```
ORG 00H
MOV DPTR, #MYCOM

;MOV P3,#0FFH
MOV P2,#00H
MOV P1,#00H
MOV P0,#0FFH

MOV 30H,#00H ;ALARM WILL BE OFF IF THIS IS 1
MOV 60H,#0 ;STORES THE 60S COUNT OF ALARM (SNOOZE PURPOSE)
MOV 61H,#0 ;SNOOZE INTERVAL INPUT STORED HERE
MOV 62H,#0 ;REPEAT COUNT INPUT STORED HERE
MOV 63H,#0 ;REPEAT COUNTER
SETB P2.7


;========TURN ON LCD=========

LCD_ON_COMMAND:
        CLR A
        MOVC A, @A+DPTR
        ACALL COMNWRT
        ACALL DELAY2
        JZ START
        INC DPTR
        SJMP LCD_ON_COMMAND

;========== TAKE INPUTS==========

START:

;======TAKE INPUT OF CURRENT TIME=======

        MOV A,#80H
        ACALL COMNWRT
        ACALL DELAY2

        MOV DPTR, #MESSAGE1
M1:
        CLR A
        MOVC A, @A+DPTR
        ACALL DATAWRT
```

```
        ACALL DELAY2
        INC DPTR
        JNZ M1

        MOV A,#0C8H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#":"
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0EH
        ACALL COMNWRT
        ACALL DELAY2

H1_INP:
        MOV A,#0C6H      ;HOUR1 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 40H, A

        MOV A, 40H          ;INPUT VALIDATION (MSB OF HOUR CANNOT BE >2)
        SUBB A, #3
        JNC H1_INP

H2_INP:
        MOV A,#0C7H      ;HOUR2 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 41H, A

        MOV A,40H ;INPUT VALIDATION (LSB OF HOUR CANNOT BE >3 IF MSB
IS 2)
        CJNE A, #2, M1_INP
```

```
        MOV A, 41H
        SUBB A, #4
        JNC H2_INP

M1_INP:     MOV A,#0C9H     ;MIN1 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 42H, A

        MOV A, 42H;INPUT VALIDATION (MSB OF MIN CANNOT BE >5)
        SUBB A, #6
        JNC M1_INP

        MOV A,#0CAH     ;MIN2 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 43H, A

        ;ACALL PAUSE

;======TAKE INPUT OF ALARM TIME=======

        MOV A,#1H ;clear
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#80H
        ACALL COMNWRT
        ACALL DELAY2

        MOV DPTR, #0
        MOV DPTR, #MESSAGE2
M2:
        CLR A
        MOVC A, @A+DPTR
```

```
        ACALL DATAWRT
        ACALL DELAY2
        INC DPTR
        JNZ M2

        MOV A,#0C8H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#":"
        ACALL DATAWRT
        ACALL DELAY2

H1_A_INP:
        MOV A,#0C6H      ;ALARM_HOUR1 INPUT
        ACALL COMNWRT
        ACALL DELAY2
        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 50H, A
        MOV A, 50H          ;INPUT VALIDATION (MSB OF HOUR CANNOT BE >2)
        SUBB A, #3
        JNC H1_A_INP

H2_A_INP:
        MOV A,#0C7H      ;ALARM HOUR2 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 51H, A

        MOV A,50H ;INPUT VALIDATION (LSB OF HOUR CANNOT BE >3 IF MSB
IS 2)
        CJNE A, #2, M1_A_INP
        MOV A, 51H
        SUBB A, #4
        JNC H2_A_INP

M1_A_INP:
```

```
        MOV A,#0C9H      ;ALARM MIN1 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 52H, A

        MOV A, 52H;INPUT VALIDATION (MSB OF MIN CANNOT BE >5)
        SUBB A, #6
        JNC M1_A_INP

        MOV A,#0CAH      ;ALARM MIN2 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 53H, A

        ;ACALL PAUSE

;TAKE INPUT OF SNOOZE COUNT (HOW MANY TIMES ALARM WILL REPEAT)

        MOV A,#1H ;clear
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#82H
        ACALL COMNWRT
        ACALL DELAY2

        MOV DPTR, #0
        MOV DPTR, #MESSAGE3

M3:
        CLR A
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2
        INC DPTR
```

```
        JNZ M3

        MOV A,#0C8H      ;ALARM_HOUR1 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
        ACALL DELAY2
        ANL A, #0FH
        MOV 62H, A

        MOV A,#0CH ;
        ACALL COMNWRT
        ACALL DELAY2

        ACALL PAUSE


;=TAKE INPUT OF SNOOZE INTERVAL (TIME BETWEEN REPEATING ALARMS)=
        MOV A,#1H ;clear
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#82H
        ACALL COMNWRT
        ACALL DELAY2

        MOV DPTR, #0
        MOV DPTR, #MESSAGE4

M4:
        CLR A
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2
        INC DPTR
        JNZ M4

        MOV A,#0C8H      ;ALARM_HOUR1 INPUT
        ACALL COMNWRT
        ACALL DELAY2

        ACALL KEYPAD
        ACALL DATAWRT
```

```
        ACALL DELAY2
        ANL A, #0FH
        MOV 61H, A

        MOV A,#0CH ;
        ACALL COMNWRT
        ACALL DELAY2

        ACALL PAUSE

;=========DISPLAY MESSAGE 5 AND CURRENT TIME=============

        MOV A,#1H ;clear
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#82H
        ACALL COMNWRT
        ACALL DELAY2
        MOV DPTR, #0
        MOV DPTR, #MESSAGE5

M5:
        CLR A
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2
        INC DPTR
        JNZ M5

        MOV A,#0C6H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#":"
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0C9H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,#":"
        ACALL DATAWRT
        ACALL DELAY2
```

```
        MOV A,#0CH
        ACALL COMNWRT
        ACALL DELAY2

;=========TRANSFER THE INPUT VALUES OF CURRENT TIME=============

SET_TIME:
        MOV R0,#0
        MOV R1,#0
        MOV R2,43H
        MOV R3,42H
        MOV R4,41H
        MOV R5,40H

;=========CHECK FOR ALARM=============

ALARM_CHECK:
        SETB P2.7

; HAVING 1 IN PORT0 AKA LOCATION 30H WILL TURN OFF THE ALARM
        MOV A, 30H
        CJNE A, #00H, NO_ALARM
        MOV A, P0
        MOV 30H, A

        MOV A, R5
        CJNE A,50H, NO_ALARM
        MOV A, R4
        CJNE A,51H, NO_ALARM
        MOV A, R3
        CJNE A,52H, NO_ALARM
        MOV A, R2
        CJNE A,53H, NO_ALARM

        CLR P2.7

;NEXT BLOCK CONTROLS THE UPDATE OF ALARM TIME IN CASE OF SNOOZE

        INC 60H             ;WHEN ALARM IS ON INCREMENTS FOR EVERY SEC
        MOV A, 60H
        MOV B, #60D
        DIV AB
        MOV A, B
        JNZ NO_ALARM    ; REMAINDER ZERO FOR COUNT/60
```

```
        MOV A, 63H
        CJNE A, 62H, CALL_UPDATE_ALARM
        SJMP NO_ALARM

;UPDATE ALARM TIME
CALL_UPDATE_ALARM:
        ACALL UPDATE_ALARM
        INC 63H

NO_ALARM:

;===========DISPLAY TIME ON LCD=======

DISPLAY:
        MOV DPTR, #MYDATA
        MOV A,#0C4H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,R5
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0C5H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,R4
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0C7H
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,R3
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0C8H
        ACALL COMNWRT
        ACALL DELAY2
```

```
        MOV A,R2
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0CAH
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,R1
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY2

        MOV A,#0CBH
        ACALL COMNWRT
        ACALL DELAY2

        MOV A,R0
        MOVC A, @A+DPTR
        ACALL DATAWRT
        ACALL DELAY

;===========KEEP UPDATING THE CURRENT TIME=======

                INC R0
                CJNE R0, #10D, JUNC14
                SJMP JUNC15
JUNC14:     LJMP ALARM_CHECK
JUNC15:     MOV R0, #0

                INC R1
                CJNE R1, #6D, JUNC12
                SJMP JUNC13
JUNC12:     LJMP ALARM_CHECK
JUNC13: MOV R1, #0

                INC R2
                CJNE R2, #10D, JUNC10
                SJMP JUNC11
JUNC10:     LJMP ALARM_CHECK
JUNC11:     MOV R2, #0

                INC R3
```

```
                CJNE R3, #6D, JUNC6
                SJMP JUNC7
JUNC6:          LJMP ALARM_CHECK
JUNC7:          MOV R3, #0

                INC R4
                CJNE R5, #2D, LOGIC_1
                SJMP LOGIC_2

LOGIC_1:
                CJNE R4, #10D,JUNC8
                SJMP JUNC9
JUNC8:          LJMP ALARM_CHECK
JUNC9:          MOV R4, #00H
                SJMP JUNC5

LOGIC_2:
                CJNE R4, #4D, JUNC0
                SJMP JUNC1
JUNC0:          LJMP ALARM_CHECK
JUNC1:          MOV R4, #00H

JUNC5:          INC R5
                CJNE R5, #3D, JUNC2
                SJMP JUNC3
JUNC2:          LJMP ALARM_CHECK
JUNC3:          MOV R5, #00H

                LJMP ALARM_CHECK

;===========SUBROUTINE FOR LCD=======

COMNWRT:
        MOV P1,A
        CLR P2.0
        CLR P2.1
        SETB P2.2
        ACALL DELAY2
        CLR P2.2
        RET
DATAWRT:
        MOV P1,A
        SETB P2.0
        CLR P2.1
        SETB P2.2
```

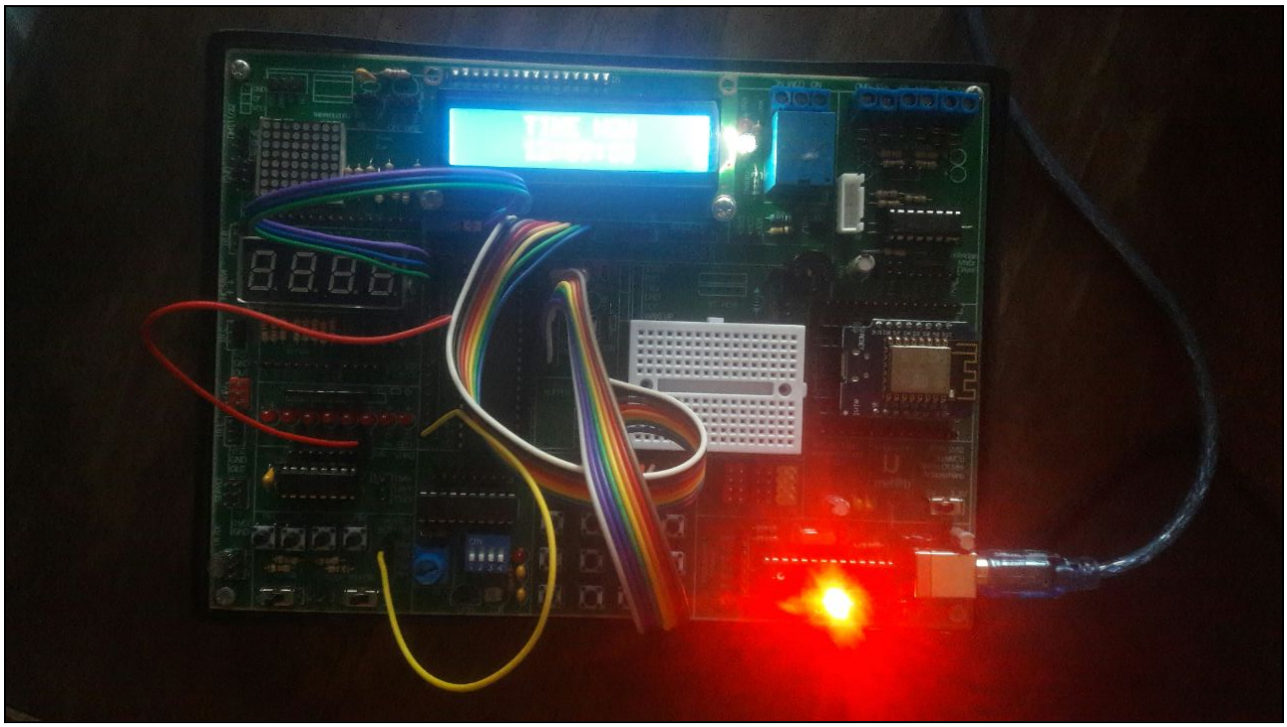
```
        ACALL DELAY2
        CLR P2.2
        RET

;=======SUBROUTINE FOR TUNING OFF ALARM======
ALARM_OFF:
        CLR P2.7
        RET


;=========SUBROUTINE FOR UPDATING ALARM TIME FOR SNOOZE=======

UPDATE_ALARM:

        MOV A, 53H
        ADD A, 61H
        MOV 70H,A
        CLR CY
        SUBB A, #10D
        ; CHEKCING  IF ADDS UP TO MORE THAN 10(9:28+5=9:33, 8+5=13)
        JC UPDATE_MIN2
        ; IF <10 THEN WE CAN STRAIGHT UP PUT THE VALUE (9:12+5=9:17, 2+5=7)

        MOV A,70H           ; IF >10 THEN UPADTE LSB OF MINS THE HARD WAY
        SUBB A, #10D
        MOV 53H, A ; UPDATE THE LSB OF MINS

        MOV A,52H ; WE ALSO NEED THE INC THE MSB OF MINS
        INC A
        CJNE A, #6, UPDATE_MIN1
        MOV 52H, #0         ; IF ADDTION GOES BEYOND 59 THEN MSB IS 0

        ;NOW WE NEED TO UPDATE THE HOUR DIGITS
        MOV A, 50H
        CJNE A, #2, HOUR_UPDATE_LOGIC1 ; LOGIC FOR 20:00 ANE BEYOND

        MOV A,51H
        INC A
        CJNE A, #4, UPDATE_HOUR2
        MOV 51H, #0
        MOV 50H, #0
        SJMP END_UPDATE_ALARM

HOUR_UPDATE_LOGIC1:
        MOV A,51H
```

```
        INC A
        CJNE A, #10, UPDATE_HOUR2
        MOV 51H, #0

        INC 50H
        SJMP END_UPDATE_ALARM

UPDATE_HOUR2:
        INC 51H
        SJMP END_UPDATE_ALARM

UPDATE_MIN1:
        INC 52H
        SJMP END_UPDATE_ALARM

UPDATE_MIN2:
        MOV 53H, 70H
END_UPDATE_ALARM:
        RET

;===========DELAY SUBROUTINES=======

; THIS ONE FOR UPDATING SECONDS OF CLOCK (36,100,100)
DELAY:      MOV B, #36D
LOOP_3:     MOV R7, #100D
LOOP_2:     MOV R6, #100D
LOOP_1:     DJNZ R6, LOOP_1
            DJNZ R7, LOOP_2
            DJNZ B, LOOP_3
            RET

;SMALL DELAY FOR LCD BUSY STATUS

DELAY2:     MOV R6,#25
HERE2:      MOV R7,#10
HERE:       DJNZ R7, HERE
            DJNZ R6, HERE2
            RET
;DELAY AFTER SHOWING TEXT IN LCD

PAUSE:      MOV B, #100D
LOOP_4:     MOV R7, #20D
LOOP_5:     MOV R6, #100D
LOOP_6:     DJNZ R6, LOOP_6
            DJNZ R7, LOOP_5
```

```
        DJNZ B, LOOP_4
        RET


;===========KEYPAD SUBROUTINES=======

KEYPAD:
        MOV A,#0FH
        MOV P3,A
K1:     MOV P3,#00001111B
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,K1

K2:     ACALL DELAY_KP
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,OVER
        SJMP K2

OVER: ACALL DELAY_KP
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,OVER1
        SJMP K2

OVER1:      MOV P3,#11101111B
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,ROW_0
        MOV P3,#11011111B
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,ROW_1
        MOV P3,#10111111B
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,ROW_2
        MOV P3,#01111111B
        MOV A,P3
        ANL A,#00001111B
        CJNE A,#00001111B,ROW_3
        LJMP K2
```

```
ROW_0:      MOV DPTR,#KCODE0
            SJMP FIND
ROW_1:      MOV DPTR,#KCODE1
            SJMP FIND
ROW_2:      MOV DPTR,#KCODE2
            SJMP FIND
ROW_3:      MOV DPTR,#KCODE3

FIND:       RRC A
            JNC MATCH
            INC DPTR
            SJMP FIND

MATCH:      CLR A
            MOVC A,@A+DPTR
            RET

DELAY_KP:
            MOV R3,#50
            HEREZ2: MOV R4,#255
            HEREZ: DJNZ R4,HEREZ
            DJNZ R3,HEREZ2
            RET

;ASCII LOOK-UP TABLE FOR EACH ROW
KCODE0: DB '7','8','9','/' ;ROW 0
KCODE1: DB '4','5','6','*' ;ROW 1
KCODE2: DB '1','2','3','-' ;ROW 2
KCODE3: DB 'A','0','C','+' ;ROW 3


;===========ON SCREEN TEXTS=========

ORG 450H

        MYCOM:    DB 38H, 0EH, 0CH, 1H, 6H,82H, 0
        MYDATA:   DB "0","1","2","3","4","5","6","7","8","9"
        MESSAGE1: DB " SET CLOCK TIME: ",0
        MESSAGE2: DB "   SET ALARM:",0
        MESSAGE3: DB "  REPEAT COUNT:",0
        MESSAGE4: DB "SNOOZE INTERVAL:",0
        MESSAGE5: DB "    TIME NOW",0

END
```

Hardware Implementation



Problems Faced

A good number of challenges were faced during the implementation of the project and most of them were related to hardware.

For the coding part, as more and more features were added and the code started to grow bigger conditional jumps began to break. As they are short jumps, they were unable to cover the range of the jump. And Long Jump doesn't check for any condition. So those jumps had to be broken into parts so that the conditions were checked and the jump was made as well.

The microcontroller kit has only four 7-segment LEDs which we didn't get to know before getting the kit in hand. We completed the coding and software simulation in PROTEUS once using LED to display current time. But due to the shortage of 7-seg LED in the kit we had to redo a huge portion of the code and configure it for LCD.

The PCB footprint of port 1 was wrong in the microcontroller kit which we didn't get to know earlier. This gave us a lot of trouble since the port was neither taking input or giving any output. Even now this port can only work as an output port. The Keypad works only if connected to port 3. Also port0 of the microcontroller in PROTEUS software works only as an input port. It took us a lot of time to figure out these limitations and find a workaround for them.

Conclusion

All the objectives and mandatory requirements of the project were fulfilled. Some extra features were added as well. But this project does have some limitations. Alarm can be set only once, which is before starting the clock. Users cannot set an alarm while the clock is running. This would be a very convenient feature to have. Being able to switch between 12h and 24h format would be another interesting addition to the project.