EEE 4706
Project Report

Real Time Clock

Group Number: G3

Group Members:	Elmul Soad Swopno	200021219
	Mutakabbir Ashfak	200021227
	Mustak Hossain Simanto	200021243
	Sirazul Monir	200021247
	Shouvik Fahim	200021249
	Abubakar Babangida	200021255

Objective

Main Features:

1. Clock Interface and Display:
- Interface a Real-Time Clock (RTC) module to design a functional clock.
- Display time on an LCD.
- Allow users to choose between 12-hour (with AM/PM indicator) and 24-hour time formats.
- Implement a stopwatch feature with start, pause, and reset functionalities.

2. Hourly Chime:
- Use a piezo buzzer to produce a beep or chime sound at the start of every hour.

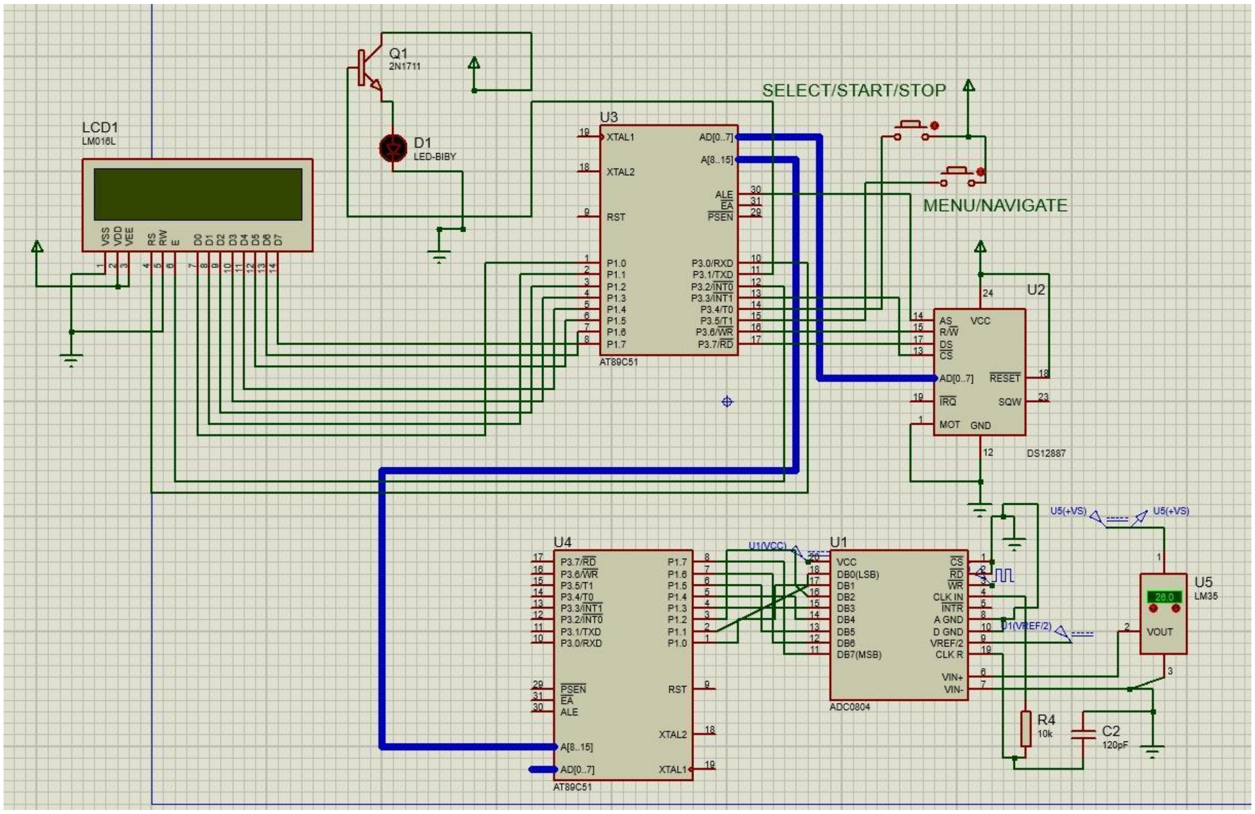
3. Date, Day, and Temperature Display:
- Display the current date and the name of the day on the screen.
- Interface a temperature sensor to measure and display the ambient temperature.
- Provide a basic weather status ("Cold," "Warm," or "Hot") based on the measured temperature using predefined thresholds:

● Temperature < 15°C:	"Cold"
● Temperature between 15°C and 25°C:	"Warm"
● Temperature > 25°C:	"Hot"

Additional Features:

● **Countdown Timer**	→	Shows countdown at 90s, 60s and 30s
● **Tally Counter**	→	Can count from 0 to 99 in a tally

Circuit Diagram



Configuration

1. RTC Initialization and Configuration

- Code starts by initializing the RTC chip (DS12887) with a 200ms delay after power-up.
- Configures the RTC registers:
  - Register A (address 0AH) is set to turn on the oscillator.
  - Register B (address 0BH) controls time format (12/24 hour), BCD mode, etc.

---

2. Time and Date Setting

- Sets initial time to 16:58:55 (4:58:55 PM in 12-hour format).
- Sets initial date to October 19, 2004 (19/10/04).
- Day of the week is also set (though not explicitly shown in the setting section).

---

3. Main Display Loop (OV1)

- Continuously reads time from RTC and displays it on LCD.
- Handles both 12-hour and 24-hour formats based on the FORMAT_SELECT flag.
- Displays date in the format: "Day DD/MM/YY".
- Includes temperature display functionality, reading from Port 2.
- Features an hourly buzzer that activates exactly at hh:00:00.

---

4. Additional Functionalities

a) Menu System

- Accessed via Button_A (P3.5) and Button_B (P3.4).
- Available options:
    - Clock format selection (12/24 hour)
    - Stopwatch
    - Countdown timer (30/60/90 seconds)
    - Tally counter

b) Stopwatch

- Counts up to 99 seconds with 0.1s resolution.
- Controlled by:
    - Button_B: Pause/Resume
    - Button_A: Exit

c) Countdown Timer

- Three preset durations: 30, 60, or 90 seconds.
- Counts down with 1s resolution.

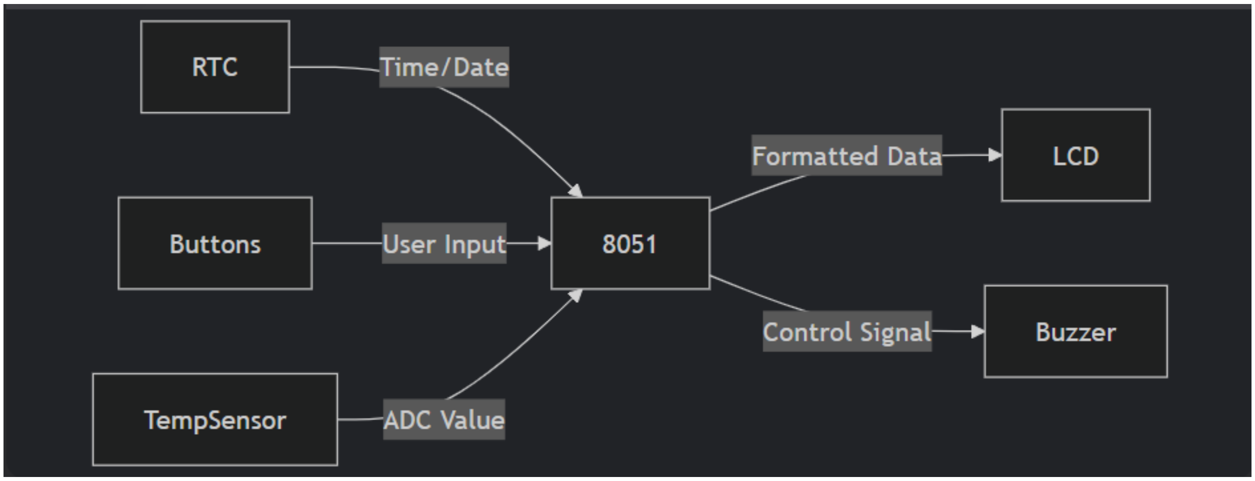
d) Tally Counter

- Simple 2-digit counter (00-99).
- Incremented by Button_B.

e) Temperature Display

- Reads analog value from Port 2.
- Displays raw value and status ("Cold", "Warm", or "Hot").

Working Principle



Core System Architecture

The system is built around three main components:

- **DS12887 Real-Time Clock (RTC)** chip that maintains accurate time/date
- **8051 microcontroller** that manages all operations
- **LCD display** for user output

---

1. System Initialization Process

When power is applied:

- The microcontroller pauses for 200 milliseconds to allow the RTC chip's oscillator to stabilize.
- The RTC is configured through its internal registers:
  - Register A: Starts the oscillator.
  - Register B: Enables BCD (Binary-Coded Decimal) mode and 24-hour time format.

- Various control bits are set for proper operation.

The LCD display is prepared by:

- Setting it to 2-line display mode
- Clearing any existing content
- Configuring cursor behavior

Hardware components are initialized:

- Button inputs set up on pins P3.4 and P3.5
- Buzzer connected to pin P3.1
- Temperature sensor input configured on Port 2

---

2. Time and Date Configuration

The system sets initial values in the RTC's timekeeping registers:

- Time: Set to 16:58:55 (4:58:55 PM in 12-hour mode)
- Date: Programmed as October 19, 2004
- Days of the week stored in memory as text strings ("Sun", "Mon", etc.) for display purposes

---

3. Continuous Operation Loop

The main program loop constantly performs:

A. Time Retrieval

- Communicates with the RTC chip
- Reads hours, minutes, seconds
- Reads day, month, year

B. Time Formatting

- **12-hour mode**:
    - Converts 24-hour time to 12-hour format
    - Determines AM/PM
    - Handles special cases (12 PM, 12 AM)

- **24-hour mode**:
  - Displays hours as 00-23
  - No AM/PM needed

C. Date Display

- Looks up day name from stored table
- Formats date as "Day MM/DD/YY"
- Handles single-digit values properly (e.g., "05" instead of "5")

D. User Input Checking

- Continuously monitors button states
- Implements debouncing to prevent false triggers
- Recognizes button press combinations

E. Alarm Function

- Checks if current time is exactly on the hour (minutes and seconds both zero)
- Activates buzzer for 1 second on the hour change
- Ensures buzzer doesn't sound at other times

---

4. Menu System Operation

When the user presses the menu button:

- Displays available functions:
  - Clock format switching
  - Stopwatch
  - Countdown timer
  - Tally counter

Navigation:

- Button A: Cycles through menu options
- Button B: Selects the highlighted function

Clock Format Adjustment

- Toggles between 12-hour and 24-hour display modes
- Maintains time accuracy during format changes
- Immediately updates display

Stopwatch Function

- Counts upward in 0.1 second increments
- Maximum count: 99.9 seconds
- Button B: Starts/pauses the count
- Button A: Resets and exits

Countdown Timer

- Offers preset durations (30, 60, 90 seconds)
- Counts down to zero
- Sounds buzzer when reaching zero
- Button B: Pauses/resumes
- Button A: Cancels countdown

Tally Counter

- Simple incrementing counter (00-99)
- Button B: Increases count
- Button A: Resets and exits
- Useful for quick counting tasks

5. Additional Features

Temperature Monitoring

- Reads analog voltage from temperature sensor
- Converts reading to approximate temperature
- Displays **"COLD"**, **"WARM"**, or **"HOT"** based on thresholds
- Updates display periodically without affecting timekeeping

---

6. Hardware Interactions

RTC Communication

- Uses dedicated **chip select line (P3.3)**
- Precise timing for register access
- Special sequences for reading/writing time data

LCD Control

- Parallel interface using **Port 1** for data
- Three control lines (**P3.0-P3.2**) for commands
- Optimized routines for fast updates

Button Handling

- **Software debouncing** to prevent false readings
- Clear visual feedback for button presses
- Timeout for menu operations

System Reliability Features

Power Management

- RTC maintains timekeeping during power loss
- Low-power operation when possible
- Clean startup after power restoration

Error Handling

- Validates RTC communications
- Checks for reasonable time/date values
- Recovers gracefully from invalid states

Timing Accuracy

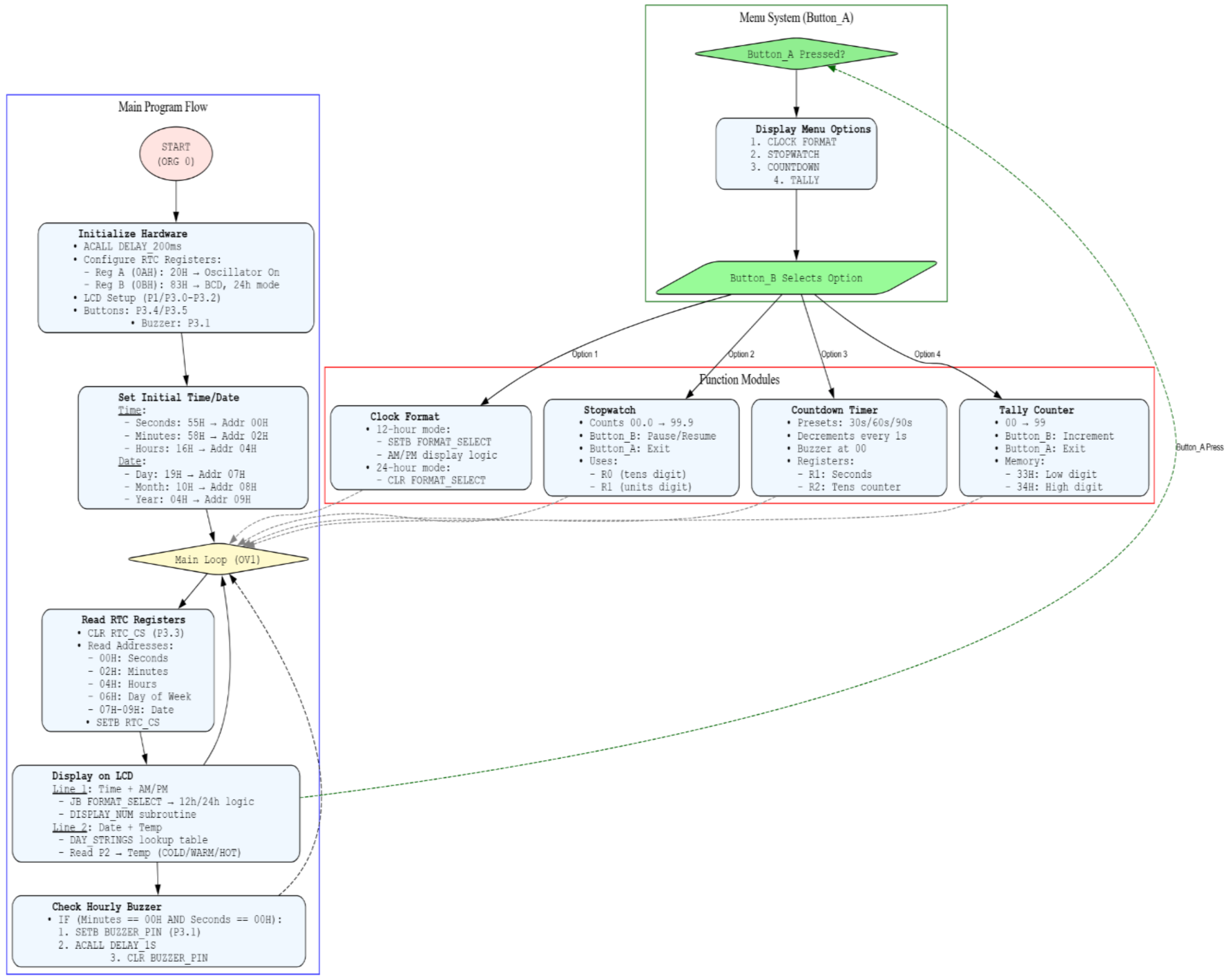
- Careful delay calibration
- Compensation for instruction cycles
- Minimal interrupt disruption

User Experience Considerations

- Immediate feedback for all button presses
- Clear visual distinction between modes
- Intuitive navigation between functions
- Consistent display formatting
- Audible confirmation for important actions

Performance Optimization

- Efficient register usage
- Minimized delay periods
- Compact code structure
- Balanced polling frequencies
- Prioritized operations



Code

```
;----RTCTIME.ASM: SETTING TIME, READING AND DISPLAYING IT
ORG 0
ACALL DELAY_200ms ;RTC needs 200ms upon power-up

; Define LCD commands
LCD_CMD_CLR      EQU 01H    ; Clear display
LCD_CMD_HOME     EQU 02H    ; Return home
LCD_CMD_ENTRY    EQU 06H    ; Entry mode, increment, no shift
LCD_CMD_ON       EQU 0CH    ; Display on, cursor off
LCD_CMD_LINE1    EQU 80H    ; Address of the first line
LCD_CMD_LINE2    EQU 0C0H   ; Address of the second line
RTC_CS           EQU P3.3
FORMAT_SELECT    EQU 27H.0   ; Pin to select 12-hour (1) or 24-hour (0) format
Button_A EQU P3.5
Button_B EQU P3.4
BUZZER_PIN       EQU P3.1   ; Pin connected to the active buzzer

; Day of the Week Strings
DAY_STRINGS:
    DB "Sat", 0
    DB "Sun", 0
    DB "Mon", 0
    DB "Tue", 0
    DB "Wed", 0
    DB "Thu", 0
    DB "Fri", 0
    DB "Sat", 0

;-------------TURNING ON THE RTC

CLR RTC_CS  ; Enable RTC (Active Low)

MOV R0,#10 ;R0=0AH, Reg A address
MOV A,#20H ;010 in D6-D4 to turn on osc.
MOVX @R0,A ;send it to Reg A of DS12887
SETB RTC_CS ; Disable RTC

SETB FORMAT_SELECT ;control bit for 12h/24h format
CLR BUTTON_A
CLR BUTTON_B
CLR Buzzer_PIN

MOV P2,#0FFH
;------------------
CLR 24H.1 ;for the tally counter reset control bit

;-------------Turning on RTC---------------------
CLR RTC_CS
MOV   R0, #10        ; R0 = 0AH, Reg A address
MOV   A, #2EH        ; Turn on osc., 1110=RS4-RS0 4Hz SQW
MOVX  @R0, A         ; Send it to Reg A of DS12887
SETB RTC_CS
ACALL DELAY
MOV   R0, #11        ; R0 = 0BH, Reg B address
CLR RTC_CS
ACALL DELAY
MOVX  A, @R0         ; Get Reg B of DS12887 to ACC
SETB RTC_CS
ACALL DELAY          ; Need delay for fast 8051
SETB  ACC.3         ; Let 4Hz come out
CLR RTC_CS
MOVX  @R0, A         ; Send it back to Reg B
ACALL DELAY
SETB RTC_CS
```

```
;--------------Setting the Time mode
CLR RTC_CS
MOV R0,#11 ;Reg B address
MOV A,#83H ;BCD, 24 hrs, daylight saving
MOVX @R0,A ;send it to Reg B
SETB RTC_CS
NOP
NOP
NOP
;---------Setting the Time---------------------
CLR RTC_CS
MOV   R0, #0       ; point to seconds address
MOV   A, #55H      ; seconds = 55H (BCD numbers need H)
MOVX  @R0, A       ; set seconds

MOV   R0, #02      ; point to minutes address
MOV   A, #58H      ; minutes = 58
MOVX  @R0, A       ; set minutes

MOV   R0, #04      ; point to hours address
MOV   A, #16H      ; hours = 16
MOVX  @R0, A       ; set hours

MOV   R0, #11      ; Reg B address
MOV   A, #03      ; D7=0 of reg B to allow update
MOVX  @R0, A       ; send it to reg B

SETB RTC_CS

NOP
NOP
;---------Setting the DATE
CLR RTC_CS

MOV   R0, #07      ; load pointer for DAY OF MONTH
MOV   A, #19H      ; DAY = 19H (BCD numbers need H)
MOVX  @R0, A       ; set DAY OF MONTH
ACALL DELAY        ;

MOV   R0, #08      ; point to MONTH
MOV   A, #10H      ; 10 = OCTOBER
MOVX  @R0, A       ; set MONTH
ACALL DELAY        ;

MOV   R0, #09      ; point to YEAR address
MOV   A, #04      ; YEAR = 04 for 2004
MOVX  @R0, A       ; set YEAR to 2004
ACALL DELAY        ;

MOV   R0, #11      ; Reg B address
MOV   A, #03      ; D7=0 of reg B to allow update
MOVX  @R0, A       ; send it to reg B
SETB RTC_CS

NOP
NOP

; Main Routine to Display Time and Date
OV1:

CLR RTC_CS

MOV   R0, #10       ; R0 = 0AH, Reg A address
MOV   A, #2EH       ; Turn on osc., 1110=RS4-RS0 4Hz
SQW
MOVX  @R0, A        ; Send it to Reg A of DS12887
SETB RTC_CS
MOV   R0, #11       ; R0 = 0BH, Reg B address
CLR RTC_CS
MOVX  A, @R0        ; Get Reg B of DS12887 to ACC
```

```
ACALL DELAY
SETB RTC_CS        ; Need delay for fast 8051
SETB  ACC.3        ; Let 4Hz come out
CLR RTC_CS
MOVX  @R0, A       ; Send it back to Reg B
SETB RTC_CS


  ACALL LCD_INIT

  ; Display "HH:MM:SS" format on the first line
  CLR RTC_CS
  CLR A
  MOV R0,#4     ; Point to HR location
  MOVX A,@R0    ; Read hours

  JB FORMAT_SELECT, DISPLAY_12_HOUR ; If
FORMAT_SELECT is high, use 12-hour format
  SJMP DISPLAY_24_HOUR ; Otherwise, use 24-hour
format

DISPLAY_12_HOUR:
  ANL A,#7FH    ; Mask the MSB (AM/PM bit)
  CJNE A,#12, NOT_NOON ; If hour is 12, it's noon
  MOV A,#12     ; Convert 12 to 12 for noon
  SJMP DISPLAY_HOURS

NOT_NOON:
  CJNE A,#00, DISPLAY_HOURS ; If hour is 0, it's midnight
  MOV A,#12H     ; Convert 0 to 12 for midnight

DISPLAY_HOURS:
  ACALL DISPLAY_NUM
  MOV A,#':'    ; Display ':'
  ACALL LCD_WRITE
  SETB RTC_CS

  CLR RTC_CS
  CLR A
  MOV R0,#02     ; Point to minute location
  MOVX A,@R0    ; Read minutes
  ACALL DISPLAY_NUM
  MOV A,#':'    ; Display ':'
  ACALL LCD_WRITE
  SETB RTC_CS

  CLR RTC_CS
  CLR A
  MOV R0,#00     ; Point to seconds location
  MOVX A,@R0    ; Read seconds
  ACALL DISPLAY_NUM
  SETB RTC_CS

  ; Display AM/PM
  CLR RTC_CS
  MOV R0,#4      ; Point to HR location
  MOVX A,@R0    ; Read hours
  JB ACC.7, DISPLAY_PM ; If MSB is set, it's PM
  MOV A,#'A'    ; Display 'A' for AM
  ACALL LCD_WRITE
  MOV A,#'M'
  ACALL LCD_WRITE
  SJMP DISPLAY_DONE
DISPLAY_PM:
  MOV A,#'P'    ; Display 'P' for PM
  ACALL LCD_WRITE
  MOV A,#'M'
  ACALL LCD_WRITE
DISPLAY_DONE:
    SETB RTC_CS
    SJMP DISPLAY_DATE

DISPLAY_24_HOUR:
    MOV B,A        ; Save original hour value
    ANL A,#7FH     ; Mask the MSB (AM/PM bit)
    JB B.7, IS_12PM ; If PM bit is set, add 12 to the hour
    SJMP DISPLAY_HOURS_24

IS_12PM:
    CJNE A,#12H, IS_8PM
    SJMP DISPLAY_HOURS_24

IS_8PM:
    CJNE A,#08H, IS_9PM
    SJMP ADD_18

IS_9PM:
    CJNE A,#09H, ADD_12
    SJMP ADD_18

ADD_12:
    ADD A,#12H      ; Add 12 to convert to 24-hour format
    SJMP DISPLAY_HOURS_24

ADD_18:
    ADD A,#18H      ; Add 18 to convert to 24-hour format

DISPLAY_HOURS_24:
    ACALL DISPLAY_NUM
    MOV A,#':'     ; Display ':'
    ACALL LCD_WRITE
    SETB RTC_CS

    CLR RTC_CS
    CLR A
    MOV R0,#02     ; Point to minute location
    MOVX A,@R0     ; Read minutes
    ACALL DISPLAY_NUM
    MOV A,#':'     ; Display ':'
    ACALL LCD_WRITE
    SETB RTC_CS

    CLR RTC_CS
    CLR A
    MOV R0,#00     ; Point to seconds location
    MOVX A,@R0     ; Read seconds
    ACALL DISPLAY_NUM
    SETB RTC_CS


DISPLAY_DATE:

    Display_Temperature:
    MOV A, P2          ; Read ADC value from Port 2
    ACALL DISPLAY_TEMP   ; Update LCD with ADC value and status

    ; Display Date on the second line
    MOV A,#LCD_CMD_LINE2 ; Move cursor to the second line
    ACALL LCD_CMD

    ; Display Day of the Week
    CLR RTC_CS
    MOV R0,#06     ; Point to DAY OF WEEK location (Register 06H)
    MOVX A,@R0     ; Read day of the week
    ANL A,#07H     ; Mask to get only the lower 3 bits (0-6)
    MOV DPTR,#DAY_STRINGS ; Point to day strings
    MOV B,#04      ; Each day string is 4 bytes long (3 chars + null terminator)
    MUL AB         ; Calculate offset (A * 4)
    ADD A,DPL      ; Add offset to DPTR
    MOV DPL,A
    MOV A,DPH
    ADDC A,#0
    MOV DPH,A
    ACALL LCD_STRING ; Display day string
    MOV A,#' '     ; Display space
    ACALL LCD_WRITE
    SETB RTC_CS

    ; Display Day
    CLR RTC_CS
    MOV R0,#07     ; Point to DAY location
    MOVX A,@R0     ; Read day
    ACALL DISPLAY_NUM
    MOV A,#'/'     ; Display '/'
    ACALL LCD_WRITE
    SETB RTC_CS

    ; Display Month
    CLR RTC_CS
    MOV R0,#08     ; Point to MONTH location
    MOVX A,@R0     ; Read month
    ACALL DISPLAY_NUM
    MOV A,#'/'     ; Display '/'
    ACALL LCD_WRITE
    SETB RTC_CS

    ; Display Year
    CLR RTC_CS
    MOV R0,#09     ; Point to YEAR location
    MOVX A,@R0     ; Read year
    ACALL DISPLAY_NUM
    MOV A,#' '
    SETB RTC_CS

    Display_Weath:
    MOV A, P2          ; Read ADC value from Port 2
    ACALL DISPLAY_WEATHER   ; Update LCD value to weather

    ; Check if it's hh:00:00 to activate the buzzer
    CLR RTC_CS
    MOV R0,#02     ; Point to minute location
    MOVX A,@R0     ; Read minutes
    JNZ BUZZER_OFF ; If minutes != 0, skip buzzer
    MOV R0,#00     ; Point to seconds location
    MOVX A,@R0     ; Read seconds
    JNZ BUZZER_OFF ; If seconds != 0, skip buzzer
    SETB BUZZER_PIN ; Activate the buzzer
    ACALL DELAY_1S ; Beep duration

    CLR BUZZER_PIN   ; Deactivate the buzzer
BUZZER_OFF:
    SETB RTC_CS

    JB P3.5,MODES

    ACALL DELAY_50MS

    LJMP OV1       ; Read and display forever

;-------------------setting up the different functionalities

MODES:

DEBOUNCE4: JB Button_A,DEBOUNCE4
```

```
ACALL LCD_INIT
MOV DPTR,#Option1
ACALL LCD_STRING
HOLD1:
JB Button_B,Time_Mode_Stoppage
JNB Button_A,HOLD1

DEBOUNCE5: JB Button_A,DEBOUNCE5

MODE2:
ACALL LCD_INIT
MOV DPTR,#Option2
ACALL LCD_STRING
HOLD2:
JB Button_B,STOP_STATION
JNB Button_A,HOLD2
DEBOUNCE6: JB Button_A,DEBOUNCE6

MODE3:
ACALL LCD_INIT
MOV DPTR,#Option3
ACALL LCD_STRING
HOLD3:
JB Button_B,Countdowner_Stoppage
JNB Button_A,HOLD3
DEBOUNCE7: JB Button_A,DEBOUNCE7

MODE4:
ACALL LCD_INIT
MOV DPTR,#Option4
ACALL LCD_STRING
HOLD24:
JB Button_B,TALLY
JNB Button_A,HOLD24
DEBOUNCE72: JB Button_A,DEBOUNCE72

LJMP OV1

LJMP OV1

STOP_STATION:
LJMP STOPWATCH_LEV

Countdowner_Stoppage:
LJMP Countdowner

Time_Mode_Stoppage:
LJMP Time_Mode

TALLY:
DEBOUNCE78: JB Button_B,DEBOUNCE78
MOV 33,#0  ;Low digit
MOV 34,#0  ;High digit

TALLY_CON:
ACALL LCD_INIT
MOV A,34
ACALL DISPLAY_NUM_DEC
MOV A,33
ACALL DISPLAY_NUM_DEC
TAP:
JB Button_A,EXET
JNB Button_B,TAP
DEBOUNCE200: JB Button_B,DEBOUNCE200

INC 33
MOV A,33
CJNE A,#10,Dig2
MOV 33,#0
INC 34
```

```
MOV A,34
CJNE A,#10,Dig2
SJMP TALLY
Dig2:

SJMP TALLY_CON

EXET:
DEBOUNCE220: JB Button_A,DEBOUNCE220
JB 24H.1,OK_DONE
SETB 24H.1
SJMP TALLY
OK_DONE:
CLR 24H.1
LJMP OV1

Time_Mode:
DEBOUNCE8: JB Button_B,DEBOUNCE8

H12:
ACALL LCD_INIT
MOV DPTR,#TWELVE
ACALL LCD_STRING
HOLD9:
JB Button_B,H12_LEV
JNB Button_A,HOLD9
DEBOUNCE14: JB Button_A,DEBOUNCE14

H24:
ACALL LCD_INIT
MOV DPTR,#TWENTYFOUR
ACALL LCD_STRING
HOLD10:
JB Button_B,H24_LEV
JNB Button_A,HOLD10
DEBOUNCE23: JB Button_A,DEBOUNCE23

LJMP OV1

H12_LEV:
DEBOUNCE21: JB Button_B,DEBOUNCE21
SETB FORMAT_SELECT
LJMP OV1

H24_LEV:
DEBOUNCE22: JB Button_B,DEBOUNCE22
CLR FORMAT_SELECT
LJMP OV1

Countdowner:
DEBOUNCE9: JB Button_B,DEBOUNCE9

CN30:
ACALL LCD_INIT
MOV DPTR,#Thirty
ACALL LCD_STRING
HOLD4:
JB Button_B,CN30_LEV
JNB Button_A,HOLD4
DEBOUNCE10: JB Button_A,DEBOUNCE10

CN60:
ACALL LCD_INIT
MOV DPTR,#Sixty
ACALL LCD_STRING
HOLD5:
JB Button_B,CN60_LEV
JNB Button_A,HOLD5
DEBOUNCE11: JB Button_A,DEBOUNCE11
```

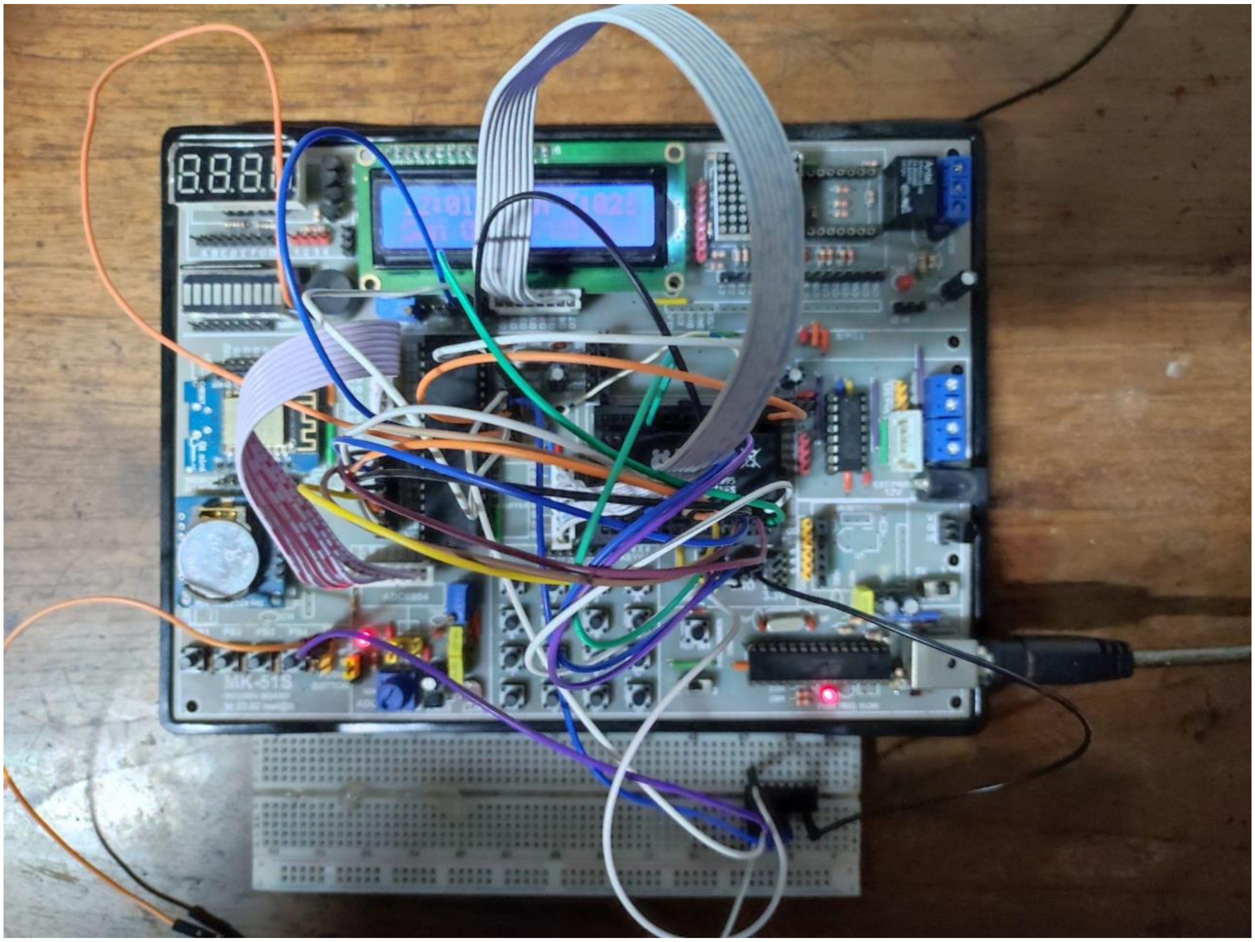
```
CN90:
ACALL LCD_INIT
MOV DPTR,#Ninety
ACALL LCD_STRING
HOLD6:
JB Button_B,CN90_LEV
JNB Button_A,HOLD6
DEBOUNCE12: JB Button_A,DEBOUNCE12

LJMP OV1

CN30_LEV:
DEBOUNCE19: JB Button_B,DEBOUNCE19
MOV R0,#0
MOV R1,#3
ACALL Counter
LJMP OV1

CN60_LEV:
DEBOUNCE18: JB Button_B,DEBOUNCE18
MOV R0,#0
MOV R1,#6
ACALL Counter
LJMP OV1

CN90_LEV:
DEBOUNCE17: JB Button_B,DEBOUNCE17
MOV R0,#0
MOV R1,#9
ACALL Counter
LJMP oV1

STOPWATCH_LEV:
DEBOUNCE: JB P3.4,DEBOUNCE
ACALL STOPWATCH
DEBOUNCE3: JB P3.4,DEBOUNCE3
LJMP OV1


;-----------------------------SUBROUTINES------------------------------
DISPLAY_WEATHER:
  ; --- Display Temperature Status (Line 1) ---
  ;MOV A, #80H          ; Move cursor to Line 1
  ;ACALL LCD_CMD
  ACALL GET_TEMP_MSG    ; Get temperature message (COLD, WARM, HOT)
  ACALL LCD_STRING    ; Print the message

  RET

DISPLAY_TEMP:
  ; --- Display ADC Value (Line 2) ---
  ;MOV A, #0C0H         ; Move cursor to Line 2
  ;ACALL LCD_CMD
  MOV DPTR, #MSG_TEMP  ; Point to "temperature: "
  ACALL LCD_STRING   ; Print "temperature: "
  MOV A, P2            ; Read ADC value again
  ACALL BIN_TO_ASCII   ; Convert to 3-digit ASCII
  ACALL PRINT_ADC      ; Display ADC value

  RET

BIN_TO_ASCII:
  MOV B, #100         ; Extract hundreds digit
  DIV AB
  ADD A, #30H
  MOV 30H, A
  MOV A, B
  MOV B, #10          ; Extract tens digit
  DIV AB
  ADD A, #30H
  MOV 31H, A
  MOV A, B            ; Extract units digit
  ADD A, #30H
  MOV 32H, A
  RET

; Print ADC value (from 30H-32H)
PRINT_ADC:
  MOV A, 30H          ; Hundreds digit
  ACALL LCD_WRITE
  MOV A, 31H          ; Tens digit
  ACALL LCD_WRITE
  MOV A, 32H          ; Units digit
  ACALL LCD_WRITE
  RET

; Print string from ROM (string pointer in DPTR)
PRINT_STRING:
  CLR A
  MOVC A, @A+DPTR     ; Read character from ROM
  JZ STRING_END       ; Exit if null terminator
  ACALL LCD_WRITE
  INC DPTR
  SJMP LCD_STRING
STRING_END:
  RET

; Determine temperature message (COLD, WARM, HOT)
GET_TEMP_MSG:
  MOV A, P2           ; Read ADC value
  CJNE A, #15, CHECK_LOW
CHECK_LOW:
  JC COLD             ; If A < 15, jump to COLD
  CJNE A, #25, CHECK_HIGH
CHECK_HIGH:
  JNC HOT             ; If A >= 25, jump to HOT
  MOV DPTR, #MSG_WARM  ; Else, WARM
  RET
COLD:
  MOV DPTR, #MSG_COLD
  RET
HOT:
  MOV DPTR, #MSG_HOT
  RET

;--------------------------------------------------------------------------

; Display Two-Digit Number on LCD
DISPLAY_NUM:
  MOV B,A
  SWAP A        ; A = quotient (tens), B = remainder (units)
  ANL A,#0FH     ; Convert to ASCII
  ORL A,#30H
  ACALL LCD_WRITE
  ACALL DELAY
  MOV A,B
  ANL A,#0FH
  ORL A,#30H     ; Convert to ASCII
  ACALL LCD_WRITE
  ACALL DELAY
  RET

; Display String on LCD
LCD_STRING:
  CLR A
  MOVC A,@A+DPTR ; Load character from string
  JZ LCD_STRING_END ; If null terminator, end
  ACALL LCD_WRITE ; Display character
```

```
   INC DPTR       ; Move to next character
   SJMP LCD_STRING
LCD_STRING_END:
   RET

;-----------------DECIMAL NUMBER
DISPLAY_NUM_DEC:
   ;MOV B,A
   ;SWAP A         ; A = quotient (tens), B = remainder (units)
   ANL A,#0FH     ; Convert to ASCII
   ORL A,#30H
   ACALL LCD_WRITE
   ACALL DELAY

   RET

;----------SMALL DELAY
DELAY:
   SETB PSW.4
   MOV R6,#50
D2: MOV R7,#10
D1: DJNZ R7,D1
   DJNZ R6,D2
   CLR PSW.4
   RET

; LCD Initialization Routine
LCD_INIT:
   MOV A,#38H    ; Function set: 8-bit mode, 2 lines, 5x7 dots
   ACALL LCD_CMD
   ACALL DELAY
   MOV A,#LCD_CMD_ON
   ACALL LCD_CMD
   ACALL DELAY
   MOV A,#LCD_CMD_CLR
   ACALL LCD_CMD
   ACALL DELAY
   MOV A,#LCD_CMD_ENTRY
   ACALL LCD_CMD
   RET

; Write Command to LCD
LCD_CMD:
   MOV P1,A      ; Send command to LCD (P1 connected to data lines)
   CLR P3.0      ; RS=0 for command
   CLR P3.1      ; RW=0 for write
   SETB P3.2     ; Enable pulse
   ACALL DELAY   ; Small delay
   NOP
   NOP
   CLR P3.2
   RET

; Write Data to LCD
LCD_WRITE:
   MOV P1,A      ; Send data to LCD (P1 connected to data lines)
   SETB P3.0     ; RS=1 for data
   CLR P3.1      ; RW=0 for write
   SETB P3.2     ; Enable pulse
   ACALL DELAY   ; Small delay
   NOP
   NOP
   CLR P3.2
   RET

;----------------------------STOPWATCH------------------------
STOPWATCH:
MOV R0, #0
MOV R1, #0

COUNT_LOOP2:

ACALL LCD_INIT

MOV A,R0
ACALL DISPLAY_NUM_DEC

MOV A, R1            ; Load current count
ACALL DISPLAY_NUM_DEC    ; Display count on LCD

JB P3.4,HALT
JB BUTTON_A,GET_OUT
ACALL DELAY_200MS
JB P3.4,HALT
JB BUTTON_A,GET_OUT
ACALL DELAY_200MS
JB P3.4,HALT
JB BUTTON_A,GET_OUT
ACALL DELAY_200MS
JB P3.4,HALT
JB BUTTON_A,GET_OUT
ACALL DELAY_200MS
JB P3.4,HALT
JB BUTTON_A,GET_OUT
ACALL DELAY_200MS

INC R1               ; Increment seconds
;ACALL DELAY_1S        ; Delay for 1 second
CJNE R1,#10, COUNT_LOOP2 ; Repeat until 100 seconds
INC R0
MOV R1,#0
CJNE R0,#10,COUNT_LOOP2

HALT:
DEBOUNCE2: JB P3.4,DEBOUNCE2
HALTING:
JB BUTTON_A,LEAVE
JNB P3.4,HALTING
DEBOUNCE238: JB Button_B,DEBOUNCE238
SJMP COUNT_LOOP2
LEAVE:
;ACALL DELAY_200MS
DEBOU: JB BUTTON_A,DEBOU
SJMP STOPWATCH
GET_OUT:
DEBOU2: JB BUTTON_A,DEBOU2
RET                   ; Return after 100 seconds

;-----------------------------------------------------------------------------

COUNTER:

ACALL LCD_INIT
MOV A,R1
ACALL DISPLAY_NUM_DEC
MOV A,R0
ACALL DISPLAY_NUM_DEC

MOV A,R0
JZ ADJ

SJMP BACK

NOW:
DEC R2
MOV R1,#10
```

```
COME:
DEC R1
MOV R0,#9

BACK:
JB Button_B,HOLD_ON
ACALL LCD_INIT
MOV A,R1
ACALL DISPLAY_NUM_DEC
MOV A,R0
ACALL DISPLAY_NUM_DEC
ACALL DELAY_1S
DJNZ R0,BACK
JB Button_B,HOLD_ON

ADJ:
JB Button_B,HOLD_ON
ACALL LCD_INIT
MOV A,R1
ACALL DISPLAY_NUM_DEC
MOV A,R0
ACALL DISPLAY_NUM_DEC
ACALL DELAY_1S
CJNE R1,#0,COME
CJNE R2,#0,NOW
JB Button_B,HOLD_ON
ACALL DELAY_1S
JB Button_B,HOLD_ON
ACALL LCD_INIT
MOV A,R1
ACALL DISPLAY_NUM_DEC
MOV A,R0
ACALL DISPLAY_NUM_DEC
JB Button_B,HOLD_ON

HOLD_ON:
DEBOUNCE16: JB Button_B,DEBOUNCE16
RELAX: JNB Button_B,RELAX
   RET

;-------------------Delay Subroutine for 1 second--------------------
-----------
DELAY_1S:
   MOV TMOD, #20H       ; Set Timer 1 in Mode 2 (8-bit
auto-reload)
   MOV TH1, #0          ; Reload value for Timer 1 (0 for full
256 counts)

   SETB TR1             ; Start Timer 1

   MOV R7, #10          ; Upper byte of overflow counter (4232
= 16 x 256 + 152)
   MOV R6, #0           ; Lower byte of overflow counter

DELAY_LOOP:
   JNB TF1, $           ; Wait for Timer 1 overflow
   CLR TF1              ; Clear Timer 1 overflow flag

   DJNZ R6, DELAY_LOOP  ; Decrement R6; if not zero,
loop again
   DJNZ R7, DELAY_LOOP  ; Decrement R7; if not zero,
loop again
```

```
   CLR TR1              ; Stop Timer 1
   RET                  ; Return from subroutine
;--------------------------------------------------------------------------------
-----


;----------------- Delay Subroutine for 200ms-------------------------
----------------
DELAY_200MS:
   MOV 16, #175      ; Outer loop counter (100 iterations)
OUTER_LOOP:
   MOV 17, #250      ; Inner loop counter (200 iterations)
INNER_LOOP:
   NOP               ; No operation (1 µs delay)
   NOP               ; No operation (1 µs delay)
   DJNZ 17, INNER_LOOP ; Decrement R1, jump if not zero
(2 cycles = 2 µs)
   DJNZ 16, OUTER_LOOP ; Decrement R2, jump if not
zero (2 cycles = 2 µs)
   RET               ; Return from subroutine

; Optimized Delay Subroutine for 50ms
DELAY_50MS:
   MOV R2, #50       ; Outer loop counter (50 iterations)
OUTER_LOOP_50:
   MOV R1, #200      ; Inner loop counter (200 iterations)
INNER_LOOP_50:
   NOP               ; No operation (1 µs delay)
   NOP               ; No operation (1 µs delay)
   DJNZ R1, INNER_LOOP_50 ; Decrement R1, jump if not
zero (2 cycles = 2 µs)
   DJNZ R2, OUTER_LOOP_50 ; Decrement R2, jump if not
zero (2 cycles = 2 µs)
   RET               ; Return from subroutine


;--------------------------------TEXTS------------------------------

; Messages (null-terminated)
MSG_COLD: DB " Cld ", 0
MSG_WARM: DB " Wrm ", 0
MSG_HOT:  DB " Hot ", 0
MSG_TEMP: DB " T:", 0


Option1: DB '  CLOCK FORMAT',0
Option2: DB '   Stopwatch',0
Option3: DB '   Countdown',0
Option4: DB '     TALLY',0

Thirty: DB '  30s Countdown',0
Sixty:  DB '  60s Countdown',0
Ninety: DB '  90s Countdown',0

TWELVE: DB ' 12H FORMAT',0
TWENTYFOUR: DB ' 24H FORMAT',0
;-------------------------------------
END
```

Hardware Implementation



Problems Faced

1. Button bouncing
2. Noise in ADC
3. Issues during 12/24 hour formatting
4. Incorrect Pullups
5. Inconsistent temperature reading
6. Jumper Wire issues

Conclusion

Overall, the project was a success. The hardware portion proved to be a lot more difficult than the software implementation. Whether it be errors in the code, the connections or even the components themselves; we pulled through and were able to comprehend how intricate and tedious the most basic functionalities are on low level languages. Although there could have been better workflows to reduce the usage of registers, the code works on a rudimentary level, which is more than what we could have hoped for when we set out in the beginning.

Contributions

Elmul Soad Swopno	200021219	RTC Clock, Date and Day, Clock Format
Mutakabbir Ashfak	200021227	Stopwatch, Counter, Documentation
Mustak Hossain Simanto	200021243	Proteus, Hardware Setup
Sirazul Monir	200021247	Hardware Setup and Troubleshooting
Shouvik Fahim	200021249	Temperature, Weather, ADC, Documentation
Abubakar Babangida	200021255	Tally, Code Debugging