**NATIONAL INSTITUTE OF TECHNOLOGY**

** ROURKELA**

**EMBEDDED SYSTEM PROJECT [*EE3401*]**

**Topic:-**

**An automatic attendance system with serial communication using microcontroller 8051**

**Group members:-**

* **Animesh Singh Sikarwar (122EE0843)**
* **Anurag Kumar (122EE0371)**
* **Rohit Kumar Bhunya (122EE0361)**

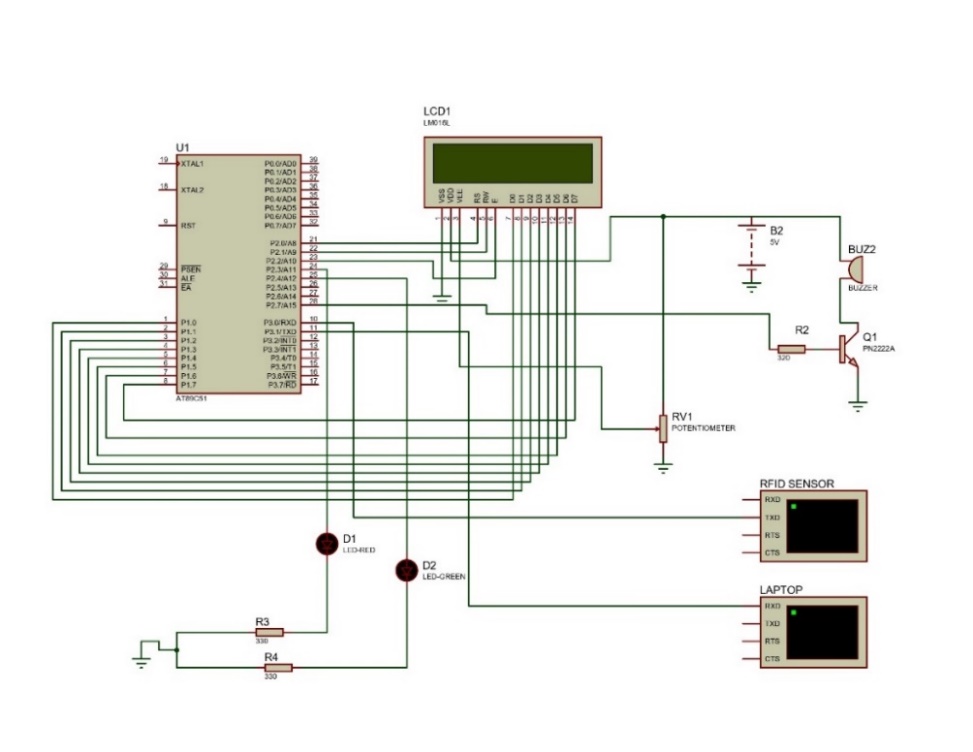
We want to develop an automatic attendance system in class using 8051 microcontroller. Further, the record should be available to computer through serial interface.

**Components used :-**

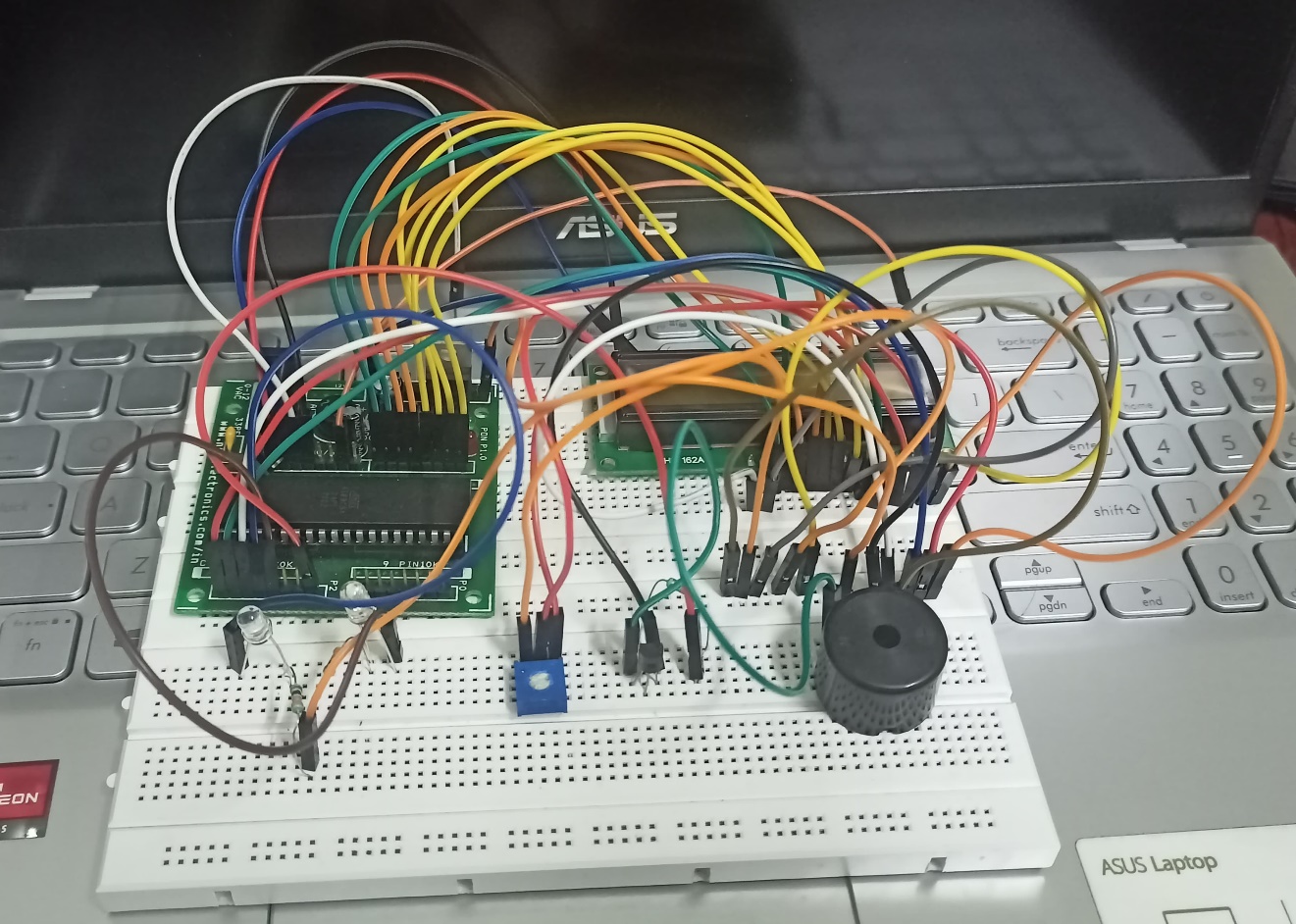
|  |  |  |
| --- | --- | --- |
| **Sl no.** | **Components used** | **Specification** |
| 01 | Microcontroller (AT89C51) | 4V to 5.5V,up to 33MHz |
| 02 | Development Board | 5V, 11.0592MHz |
| 03 | LCD display (JHD 162A) | 16\*2 |
| 04 | Mini USB TO USB Cable | -------------- |
| 05 | Buzzer | -------------- |
| 06 | Potentiometer | 0-10kOhm |
| 07 | LED | Red, Blue |
| 08 | Resistor | 10kOhm, 320Ohm |
| 09 | Transistor (N2222A) | I\_c=800mA,250MHz |
| 10 | UART to USB converter (FT232RL) | -------------- |
| 11 | Push Button | --------------- |
| 12 | Jump wires | --------------- |
| 13 | Bread Board | --------------- |

**Circuit Diagram :-**

**Figure 1:** *Schematic of the proteus design (inactive)*



**Figure 2:** *Hardware design of Schematic(inactive)*



**Code with comments**

RS EQU P2.0 ; RS OF LCD CONNECTED TO P2.0

RW EQU P2.1 ; Read/Write Control

EN EQU P2.2 ; EN OF LCD CONNECTED TO P2.1

DATA\_LINE EQU P1 ; Data Port (DB0-DB7 on Port 1)

RFID\_STORAGE EQU 30H ; TEMPORARY STORAGE FOR RFID DATA

BUZZ EQU P2.7 ; BUZZER CONNECTED TO P2.7

RED EQU P2.3 ; RED LED CONNECTED TO P2.3

GREEN EQU P2.4 ; GREEN LED CONNECTED TO P2.4

BLUE EQU P2.5 ; BLUE LED CONNECTED TO P2.5

COUNT\_ANIMESH EQU R3 ; COUNTER REGISTER FOR NUMBER OF PRESENT FOR ANIMESH

COUNT\_ANURAG EQU R4 ; COUNTER REGISTER FOR NUMBER OF PRESENT FOR ANURAG

COUNT\_ROHIT EQU R5 ; COUNTER REGISTER FOR NUMBER OF PRESENT FOR ROHIT

;====================================================================

; RESET VECTOR

;====================================================================

ORG 0000H

CLR BUZZ ; MAKING THE BUZZER PIN AT LOW ON STARTING

CLR RED ; MAKING THE RED LED AT LOW ON STARTING

CLR GREEN ; MAKING GREEN LED AT LOW ON STARTING

MOV COUNT\_ANIMESH, #0d ; INITIALIZING 'PRESENT' COUNTER OF ANIMESH AS ZERO

MOV COUNT\_ANURAG, #0d ; INITIALIZING 'PRESENT' COUNTER OF ANURAG AS ZERO

MOV COUNT\_ROHIT, #0d ; INITIALIZING 'PRESENT' COUNTER OF ROHIT AS ZERO

ACALL LCD\_INIT ; INITIALIZE LCD FOR THE RESPECTIVE MODES

ACALL UART\_INIT ; INITIALIZE SERIAL COMMUNICATION FOR RFID INPUT AND FOR DATA OUTPUT TO LAPTOP

MOV DPTR, #START\_MSG0 ; MOVING THE 16-BIT ADDRESS OF 'START\_MSG' STRING IN DATA POINTER

SETB BUZZ ; SETTING THE BUZZER

ACALL TRIPLE\_DELAY ; CALLING TRIPPLE DELAY SUBROUTINE

CLR BUZZ ; RESETTING THE BUZZER

ACALL DISPLAY\_STRING ; DISPLAYING THE START MESSAGE AS ' WELCOME !!! '

ACALL TRIPLE\_DELAY

;====================================================================

; MESSAGE SUBROUTINE

;====================================================================

REBOOT\_MSG:

MOV A, #01H ; HEX CODE FOR CLEARING LCD

ACALL LCD\_CMD ; CALLING LCD\_CMD SUBROUTINE FOR GIVING COMMAND TO LCD

MOV DPTR, #START\_MSG1 ; MOVING THE 16-BIT ADDRESS OF STRING IN DATA POINTER

ACALL DISPLAY\_STRING ; CALLING SUBROUTINE TO DISPLAY STRING ' PLEASE SCAN '

ACALL DELAY ; CALLING DELAY SUBROUTINE

MOV A, #0C0H ; HEX CODE FOR MOVING THE CURSOR OF LCD TO SECOND ROW

ACALL LCD\_CMD ; CALLING LCD\_CMD SUBROUTINE FOR GIVING COMMAND TO LCD

MOV DPTR, #START\_MSG2 ; MOVING THE 16-BIT ADDRESS OF STRING IN DATA POINTER

ACALL DISPLAY\_STRING ; CALLING SUBROUTINE TO DISPLAY STRING ' YOUR ID CARD '

SJMP MAIN\_LOOP ; JUMPING TO MAIN LOOP

MAIN\_LOOP:

ACALL READ\_RFID ; CALLING READ\_RFID SUBROUTINE FOR READING RFID

MOV A, #01H ; HEX CODE FOR CLEARING LCD DISPLAY

ACALL LCD\_CMD ; THIS WILL GIVE COMMAND TO CLEAR LCD DISPLAY

ACALL COMPARE\_RFID ; CALLING COMPARE\_RFID SUBROUTINE TO COMPARE THE UID RECEIVED FROM RFID WITH THE REGISTERED UID

ACALL REBOOT\_MSG ; CALLING REBOOT\_MSG

SJMP MAIN\_LOOP ; INFINITE LOOP

;====================================================================

; LONG DELAY SUBROUTINE

DELAY:

MOV R0, #255D

OUTER\_LOOP:

MOV R1, #150D

INNER\_LOOP:

DJNZ R1, INNER\_LOOP

DJNZ R0, OUTER\_LOOP

RET

;====================================================================

; SHORT DELAY SUBROUTINE

SHORT\_DELAY:

MOV R0, #0C8H

SHORT\_LOOP:

DJNZ R0, SHORT\_LOOP

RET

DELAY\_100MS:

MOV R0, #200D

HERE1:

MOV R1, #250D

HERE2:

DJNZ R1, INNER\_LOOP

DJNZ R0, OUTER\_LOOP

RET

;====================================================================

; LCD COMMAND SUBROUTINE

LCD\_CMD:

ACALL LCD\_busy ; CALLING LCD\_BUSY SUBROUTINE TO CHECK WHETHER THE LCD IS READING FOR TAKING NEW COMMAND

MOV DATA\_LINE, A ; MOVE COMMAND TO DATA BUS

CLR RS ; CLEARING THE REGISTER SELECT PIN OF LCD TO SELECT COMMAND MODE

CLR RW ; CLEARING THE READ/WRTIE PIN TO WRITE TO THE LCD

SETB EN ; ENABLE HIGH (START COMMAND)

CLR EN ; ENABLE LOW (END COMMAND)

RET

LCD\_busy:

setb P1.7 ;Make D7th bit of LCD data port as i/p to use it as buffer flag of lcd

setb EN ;setting enable pin for reading the buffer flag

clr RS ;Select command register

setb RW ;read/write pin is used as reading purpose

check:

clr EN ;clearing enable

setb EN ; setting enable to make active high edge triggered

jb P1.7,check ;read busy flag again and again till it becomes 0

ret ;Return from busy routine

;====================================================================

; LCD DATA SUBROUTINE

LCD\_DATA:

ACALL LCD\_busy ; WAIT FOR DATA TO COMPLETE

MOV DATA\_LINE, A ; MOVE DATA TO DATA BUS

SETB RS ; SETTING RS MODE FOR DATA MODE ON LCD

CLR RW ; CLEARING R/W FOR WRITE

SETB EN ; ENABLE HIGH (START DATA WRITE)

CLR EN ; ENABLE LOW (END DATA WRITE)

RET

;====================================================================

; LCD INITIALIZATION SUBROUTINE WITH LONGER DELAYS

LCD\_INIT:

MOV A, #38H ; FUNCTION SET: 8-BIT MODE, 2-LINE DISPLAY

ACALL LCD\_CMD

MOV A, #0EH ; DISPLAY ON, CURSOR ON MODE

ACALL LCD\_CMD

MOV A, #01H ; CLEAR DISPLAY

ACALL LCD\_CMD

MOV A, #06H ; SHIFT CURSOR EACH TIME DATA IS WRITTEN TO LCD

ACALL LCD\_CMD

MOV A, #80H ; SET CURSOR TO BEGINNING OF FIRST LINE OF FIRST ROW

ACALL LCD\_CMD

RET

;====================================================================

; UART INITIALIZATION SUBROUTINE FOR SERIAL COMMUNICATION WITH RFID AND COMPUTER

UART\_INIT:

MOV TMOD, #20H ; TIMER 1, MODE 2

MOV TH1, #0FDH ; 9600 BAUD RATE WITH 11.0592 MHZ CRYSTAL

SETB TR1 ; START TIMER 1

MOV SCON, #50H ; SERIAL MODE 1, REN ENABLED

RET

;====================================================================

; SUBROUTINE TO DISPLAY STRING

DISPLAY\_STRING:

NEXT\_CHAR:

CLR A

MOVC A, @A+DPTR ; LOAD CHARACTER FROM CODE MEMORY WHERE DATA POINTER POINTS TO

JZ END\_STRING ; IF CHARACTER IS NULL (0), END OF STRING

ACALL LCD\_DATA ; DISPLAY CHARACTER ON LCD

ACALL DELAY

INC DPTR ; POINT TO NEXT CHARACTER STORED IN CODE MEMORY

SJMP NEXT\_CHAR ; JUMP TO NEXT\_CHAR TILL THE WHOLE STRING IS DISPLAYED ON LCD

END\_STRING:

RET

;====================================================================

; RFID READING SUBROUTINE

READ\_RFID:

MOV R0, #RFID\_STORAGE ; MOVING THE ADDRESS OF RAM MEMORY TO R0 REGISTER BANK WHERE 12-BYTE DATA RECEIVED FROM RFID IS STORED

MOV R1, #12D ; MOVING 12 IN R1 REGISTER BANK TO USE IT AS COUNTING PURPOSE

READ\_CHAR:

JNB RI, READ\_CHAR ; WAIT UNTIL A CHARACTER IS RECEIVED

MOV A, SBUF ; MOVE RECEIVED DATA FROM SERIAL BUFFER INTO ACCUMULATOR

CLR RI ; CLEAR THE RECEIVE INTERRUPT FLAG

MOV @R0, A ; STORE CHARACTER IN THE RAM LOCATION DISCRIBED BY THE ADDRESS IN R0 REGISTER

INC R0 ; INCREMENT ADDRESS POINTER

DJNZ R1, READ\_CHAR ; REPEAT UNTIL 12 CHARACTERS ARE RECEIVED

RET

TRIPLE\_DELAY:

ACALL DELAY

ACALL DELAY

ACALL DELAY

RET

; Subroutine to compare RFID data with stored IDs

COMPARE\_RFID:

; Compare with ID\_1

ID\_1:

MOV R0, #RFID\_STORAGE ; Load the temporary storage address for UID into R0

CJNE @R0, #"1", ID\_2 ; Compare first byte, jump to ID\_2 if not equal

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare second byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare third byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare fourth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare fifth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare sixth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare seventh byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare eighth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare ninth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"1", ID\_2 ; Compare tenth byte

; If matched, call match\_found routine and display matched information

ACALL MATCH\_FOUND ; calling match\_found subroutine to display ' MATCHED ' on lcd

MOV DPTR, #ID1\_MATCHED ; Load address of ID1\_MATCHED for display

ACALL DISPLAY\_STRING ; Display "ANIMESH" on LCD

MOV A, #0C0H ; Set LCD cursor to second row

ACALL LCD\_CMD ; calling command subroutine to to set cursor to second row

ACALL DELAY

MOV DPTR, #ID1\_ROLLNO ; Load address of ID1\_ROLLNO for display

ACALL DISPLAY\_STRING ; Display roll number "122EE0843"

ACALL TRIPLE\_DELAY

ACALL INFO\_STRING ; calling info\_string to display total attendance of the student

; Increment attendance count for "ANIMESH"

INC COUNT\_ANIMESH

MOV A, COUNT\_ANIMESH ; Load attendance count into accumulator

ADD A, #30H ; Convert count to ASCII to display on lcd and on laptop

ACALL LCD\_DATA

ACALL TRIPLE\_DELAY

CLR GREEN

; SENDING THE ATTENDANCE DATA OF 'ANIMESH' TO PC

MOV DPTR, #DATA\_ONPC1 ; move the address of first character byte stored in code memory to data pointer

ACALL DISPLAY\_ONPC ; display the string as ' ANIMESH - 122EE0843/ '

MOV A, COUNT\_ANIMESH ; Load the count of presents

ADD A, #30H ; Convert count to ASCII (adding 30H)

MOV SBUF,A ; move the count to laptop using serial communication

LOOP:JNB TI ,LOOP ; wait till transmit interupt is set indicating the transfer complete

CLR TI ; clear transmit interupt

RET ; Return from subroutine

; Compare with ID\_2

ID\_2:

MOV R0, #RFID\_STORAGE ; Load the temporary storage address for UID into R0

CJNE @R0, #"2", ID\_3 ; Compare first byte, jump to ID\_3 if not equal

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare second byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare third byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare fourth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare fifth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare sixth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare seventh byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare eighth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare ninth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"2", ID\_3 ; Compare tenth byte

; If matched, call match\_found routine and display matched information

ACALL MATCH\_FOUND ;calling match\_found subroutine to display ' MATCHED ' on lcd

MOV DPTR, #ID2\_MATCHED ; Load address of ID2\_MATCHED for display

ACALL DISPLAY\_STRING ; Display "ANURAG" on LCD

MOV A, #0C0H ; Set LCD cursor to second row

ACALL LCD\_CMD

ACALL DELAY

MOV DPTR, #ID2\_ROLLNO ; Load address of ID2\_ROLLNO for display

ACALL DISPLAY\_STRING ; Display roll number "122EE0843"

ACALL TRIPLE\_DELAY

ACALL INFO\_STRING ; calling info\_string to display total attendance of the student

; Increment attendance count for "ANURAG"

INC COUNT\_ANURAG

MOV A, COUNT\_ANURAG ; Load attendance count into accumulator

ADD A, #30H ; Convert count to ASCII

ACALL LCD\_DATA

ACALL TRIPLE\_DELAY

ACALL TRIPLE\_DELAY

CLR GREEN

; SENDING THE ATTENDANCE DATA OF 'ANURAG' TO PC

MOV DPTR, #DATA\_ONPC2 ; move the address of first character byte stored in code memory to data pointer

ACALL DISPLAY\_ONPC ; display the string as ' ANURAG - 122EE0371/ '

MOV A, COUNT\_ANURAG ; Load the count of presents

ADD A, #30H ; Convert count to ASCII (adding 30H)

MOV SBUF,A

LOOP1:JNB TI ,LOOP1

CLR TI

RET ; Return from subroutine

; Compare with ID\_1

ID\_3:

MOV R0, #RFID\_STORAGE ; Load the temporary storage address for UID into R0

CJNE @R0, #"3", NO\_MATCH ; Compare first byte, jump to NO\_MATCH if not equal

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare second byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare third byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare fourth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare fifth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare sixth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare seventh byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare eighth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare ninth byte

INC R0 ; Increment R0 to next byte

CJNE @R0, #"3", NO\_MATCH ; Compare tenth byte

; If matched, call match\_found routine and display matched information

ACALL MATCH\_FOUND ; calling match\_found subroutine to display ' MATCHED ' on lcd

MOV DPTR, #ID3\_MATCHED ; Load address of ID3\_MATCHED for display

ACALL DISPLAY\_STRING ; Display "ROHIT" on LCD

MOV A, #0C0H ; Set LCD cursor to second row

ACALL LCD\_CMD ; calling command subroutine to to set cursor to second row

ACALL DELAY

MOV DPTR, #ID3\_ROLLNO ; Load address of ID3\_ROLLNO for display

ACALL DISPLAY\_STRING ; Display roll number "122EE0361"

ACALL TRIPLE\_DELAY

ACALL INFO\_STRING ; calling info\_string to display total attendance of the student

; Increment attendance count for "ROHIT"

INC COUNT\_ROHIT

MOV A, COUNT\_ROHIT ; Load attendance count into accumulator

ADD A, #30H ; Convert count to ASCII to display on lcd and on laptop

ACALL LCD\_DATA

ACALL TRIPLE\_DELAY

CLR GREEN

; SENDING THE ATTENDANCE DATA OF 'ROHIT' TO PC

MOV DPTR, #DATA\_ONPC3 ; move the address of first character byte stored in code memory to data pointer

ACALL DISPLAY\_ONPC ; display the string as ' ROHIT - 122EE0361/ '

MOV A, COUNT\_ROHIT ; Load the count of presents

ADD A, #30H ; Convert count to ASCII (adding 30H)

MOV SBUF,A ; move the count to laptop using serial communication

LOOP3:JNB TI ,LOOP3 ; wait till transmit interupt is set indicating the transfer complete

CLR TI ; clear transmit interupt

RET ; Return from subroutine

; No match found routine

NO\_MATCH:

MOV DPTR, #NOT\_MATCHED ; Load address for "NOT MATCHED" message

SETB BUZZ ; set buzzer to indicate wrong ID

SETB RED ; display red led to indicate wrong ID

ACALL DISPLAY\_STRING ; Display "NOT MATCHED"

ACALL TRIPLE\_DELAY

ACALL TRIPLE\_DELAY

ACALL DELAY

CLR BUZZ

CLR RED

RET ; Return from subroutine

; Match found routine

MATCH\_FOUND:

SETB BUZZ ; setting buzzer for short duration to indicate successful detection of ID

SETB GREEN ; setting green LED for indicating successful detection

ACALL TRIPLE\_DELAY

CLR BUZZ

MOV DPTR, #MATCHED ; Load address for "MATCHED" message

ACALL DISPLAY\_STRING ; Display "MATCHED"

ACALL TRIPLE\_DELAY

MOV A, #01H ; Clear display command

ACALL LCD\_CMD

ACALL DELAY

RET ; Return from subroutine

; INFO STRING TO DISPLAY ' TOTAL ATTENDANCE'

INFO\_STRING:

; Clear display and show additional information

MOV A, #01H ; Clear display command

ACALL LCD\_CMD

ACALL DELAY

MOV DPTR, #NUMBER ; Display attendance number

ACALL DISPLAY\_STRING

ACALL TRIPLE\_DELAY

MOV A, #0C0H ; Set cursor to second row

ACALL LCD\_CMD

ACALL DELAY

MOV DPTR, #EQUAL ; Display equal sign

ACALL DISPLAY\_STRING

ACALL DELAY

RET ;====================================================================

;SUBROUTINE TO DISPLAY CHARACTER ON THE COMPUTER USING SERIAL COMMUNICATION

DISPLAY\_ONPC :

CHAR:

CLR A

MOVC A, @A+DPTR ; LOAD CHARACTER FROM CODE MEMORY

JZ END\_CHAR ; IF CHARACTER IS NULL (0), END OF STRING

MOV SBUF,A ; MOVING ASCII VALUE OF CHARACTER TO SBUF

PC\_LOOP:JNB TI , PC\_LOOP ; WAITING TILL TRASNMIT INTERUPT IS SET

CLR TI ; CLEARING TRANSMIT INTERUPT

INC DPTR ; POINT TO NEXT CHARACTER

SJMP CHAR

END\_CHAR:

RET

;====================================================================

; String Data Block

START\_MSG0: DB ' WELCOME !!! ',0 ; Initial greeting message

START\_MSG1: DB ' PLEASE SCAN ',0 ; Prompt to scan the ID card

START\_MSG2: DB ' YOUR ID CARD ',0 ; Additional instruction for scanning

MATCHED: DB ' MATCHED ',0 ; Displayed when ID matches

NOT\_MATCHED: DB ' NOT MATCHED ',0 ; Displayed when ID does not match

ID1\_MATCHED: DB ' ANIMESH ',0 ; Name displayed for ID1 match

ID1\_ROLLNO: DB ' 122EE0843 ',0 ; Roll number for ID1

ID2\_MATCHED: DB ' ANURAG ',0 ; Name displayed for ID2 match

ID2\_ROLLNO: DB ' 122EE0371 ',0 ; Roll number for ID2

ID3\_MATCHED: DB ' ROHIT ',0 ; Name displayed for ID3 match

ID3\_ROLLNO: DB ' 122EE0361 ',0 ; Roll number for ID3

DATA\_ONPC1: DB ' ANIMESH - 122EE0843/ ',0

DATA\_ONPC2: DB ' ANURAG - 122EE0371/ ',0

DATA\_ONPC3: DB ' ROHIT - 122EE0361/ ',0

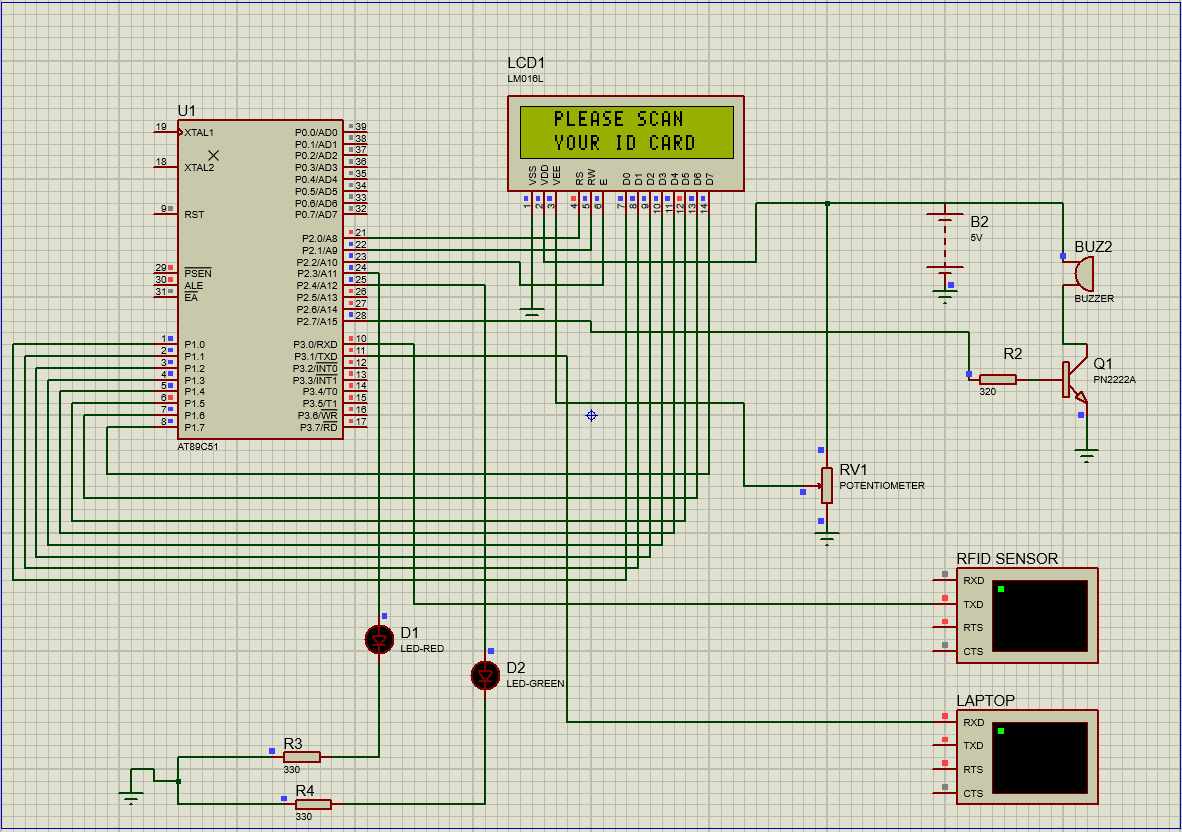
NUMBER: DB 'TOTAL ATTENDANCE',0 ; Display for attendance count

EQUAL: DB ' = ',0 ; Equal sign for display formatting

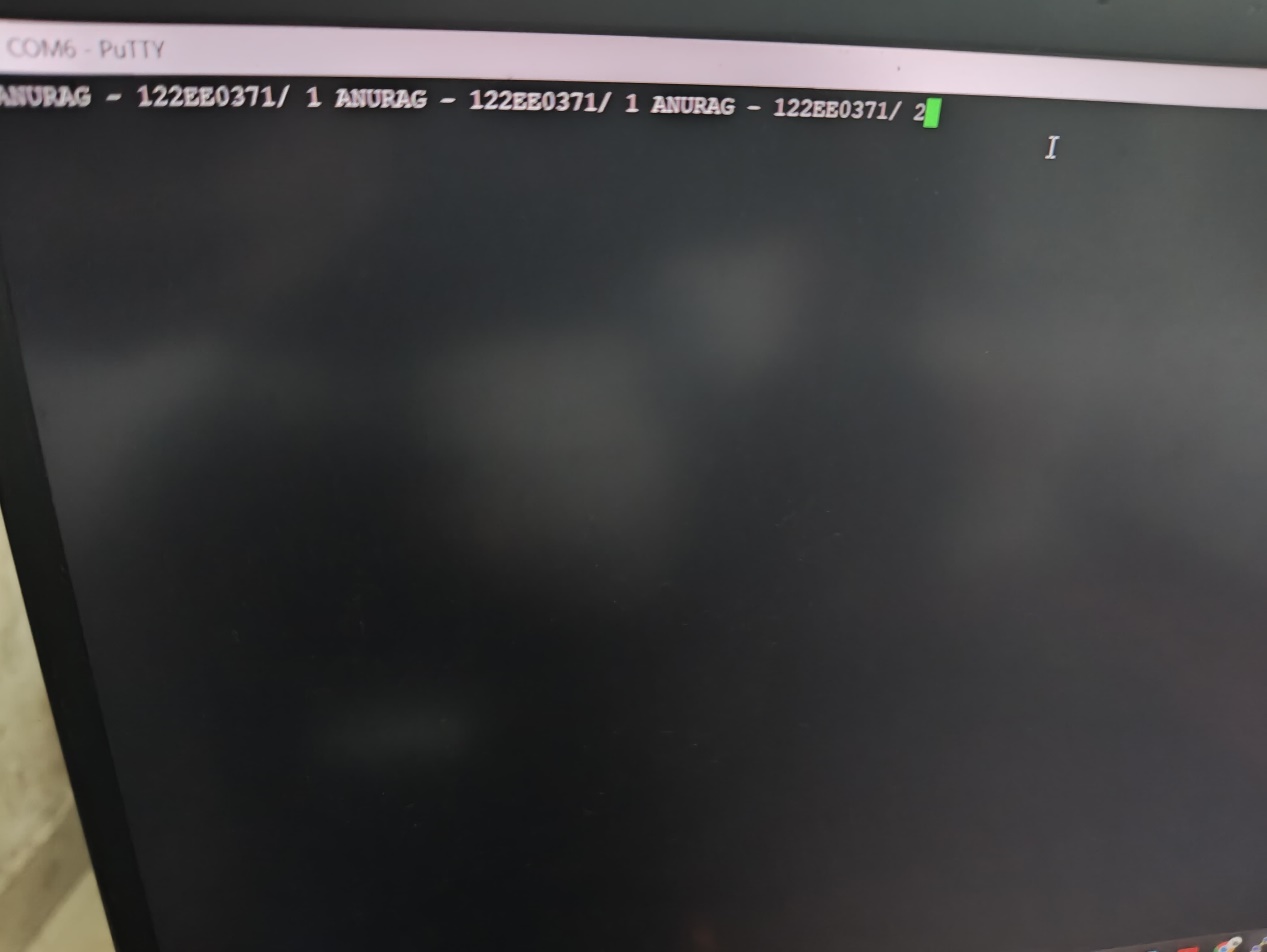
END

**Test Results :-**

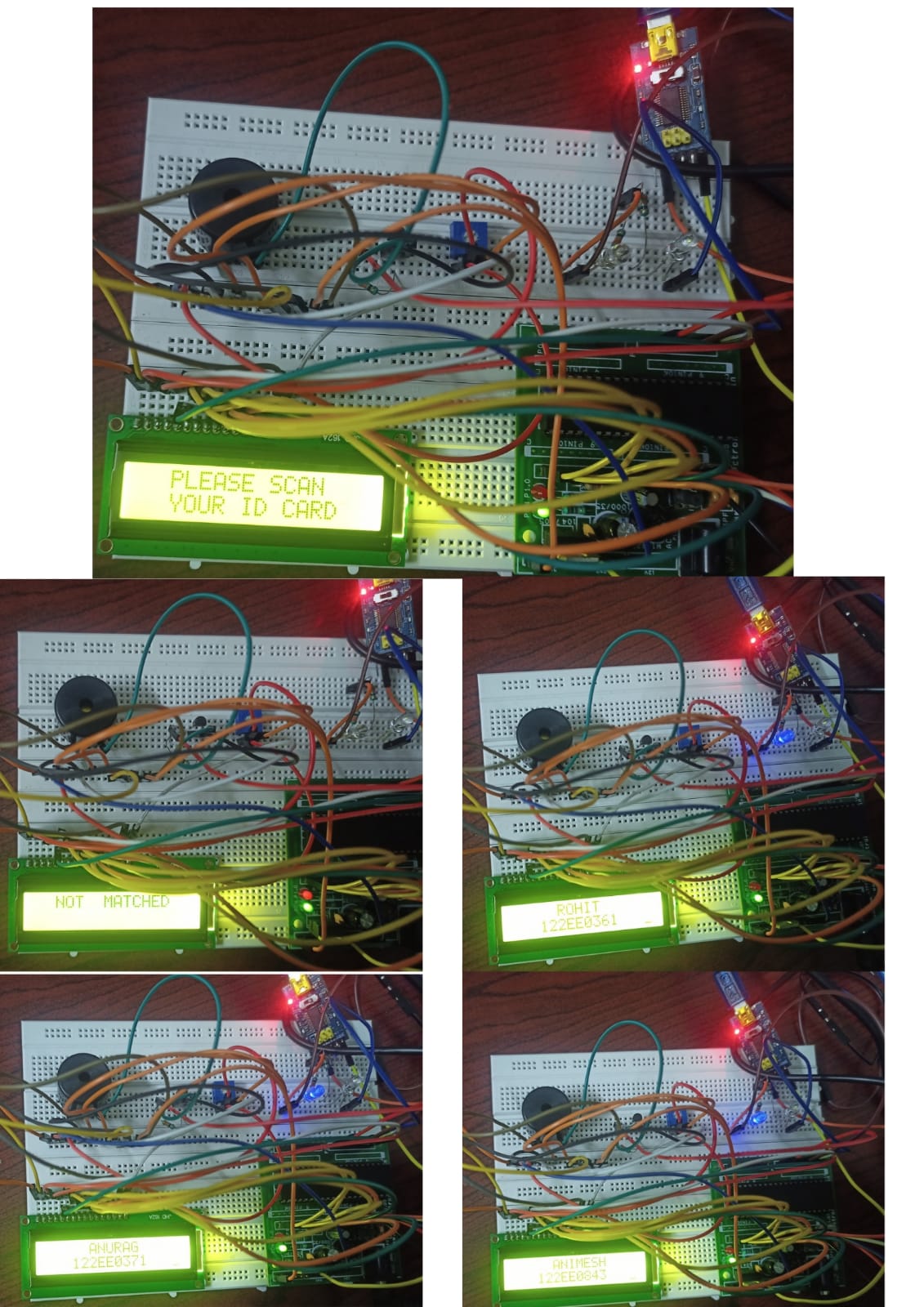
1. **Figure 3:** *Schematic of designed Proteus Circuitry (Active stage)*



1. **Figure 4:** Attendance data is being transmitted to computer through serial communication

**

1. **Figure 5:** *Hardware design of Schematic (Active stage)*

**

1. **Output table**

|  |  |
| --- | --- |
| **Input from RFID sensor** | **Output** |
| 1111111111111 | Attendance for Animesh |
| 2222222222222 | Attendance for Anurag |
| 3333333333333 | Attendance for Rohit |
| ELSE | NOT MATCHED |

RFID sensor sends 12 byte UID data corresponding to individual student to the 8051. In our case 12 byte UID corresponding to Animesh is “111111111111” for Anurag is “222222222222” for Rohit is “333333333333”

So whenever student scans their I’D card then 8051 compare the UID with the stored UID’s data if the ID is found then attendance for that particular student gets marked, if scan not matched then buzzer will play along with red LED for some time and LCD displays “NOT MATCHED”.

**Conclusion :-**

This project focused on developing an Automatic Attendance System using the AT89C51 microcontroller, integrating key components such as an RFID module, LCD display, push button, and serial communication with a PC. The system successfully reads RFID tags using UART communication, matches them with stored IDs, and displays attendance details on the LCD while sending data to a PC for record-keeping. The use of a push button allows manual confirmation of attendance, adding flexibility to the system.

Several challenges arose during development, particularly with software and hardware timing issues and communication protocols. The LCD initially failed to display data correctly on proteus which was resolved by using specific delay compatible with the speed of LCD given in datasheet and then when implemented on real hardware it failed to display character, which was resolved by implementing a busy flag check to ensure data was sent only when the LCD was ready. For data transfer to the PC, UART communication was established with the FT232RL USB-to-Serial adapter, and baud rate (9600) constraints were addressed to ensure reliable operation. Debugging techniques, such as visual feedback using the LCD, proved invaluable in identifying and resolving issues during system integration.

This system highlights the potential of embedded microcontroller solutions for real-world applications, offering a practical, efficient method for attendance management. Future enhancements could include integrating Internet of Things (IoT) capabilities for remote monitoring, adding a Real-Time Clock (RTC) for timestamping, or expanding functionality with more advanced communication protocols. Overall, this project serves as a robust foundation for building more sophisticated and scalable embedded systems.

**Thank you**