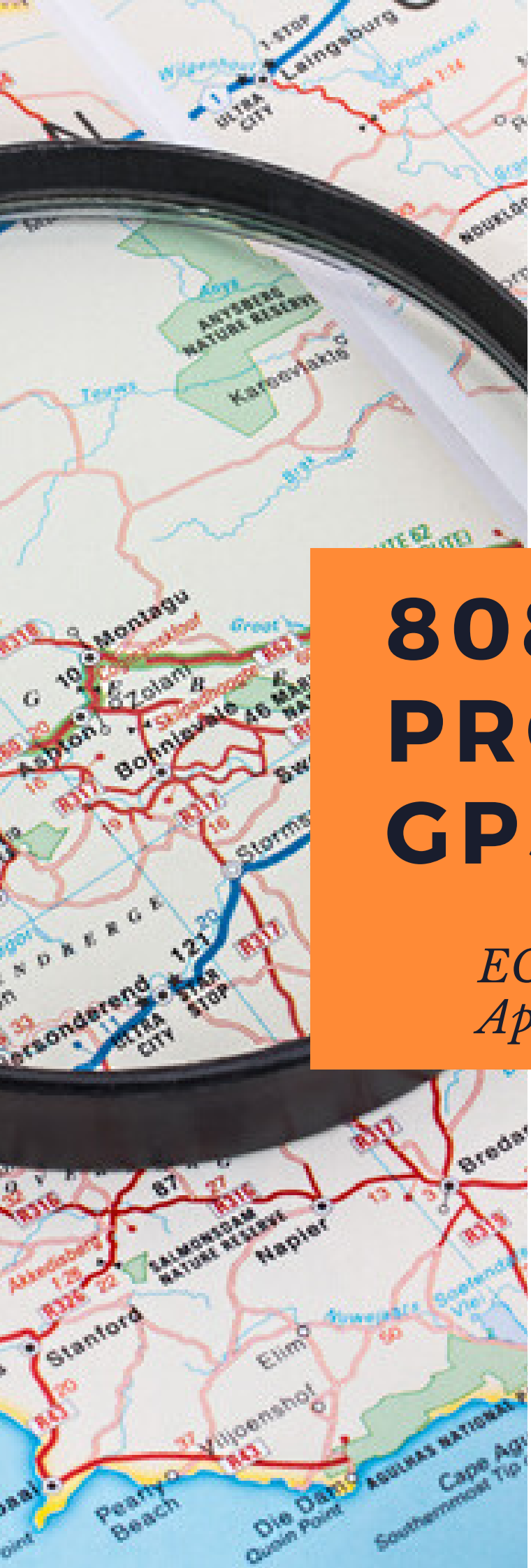


DECEMBER 2019

8085 BASED PROJECT ON GPS GUIDER

ECC17: Microprocessor and its Applications

PREPARED BY:

SANUJ KULSHRESTHA (2017UEC2053)
ANURAG HOODA (2017UEC2019)

UNDER THE GUIDANCE OF

PROF. D.V. GADRE

TABLE OF CONTENT

INTRODUCTION

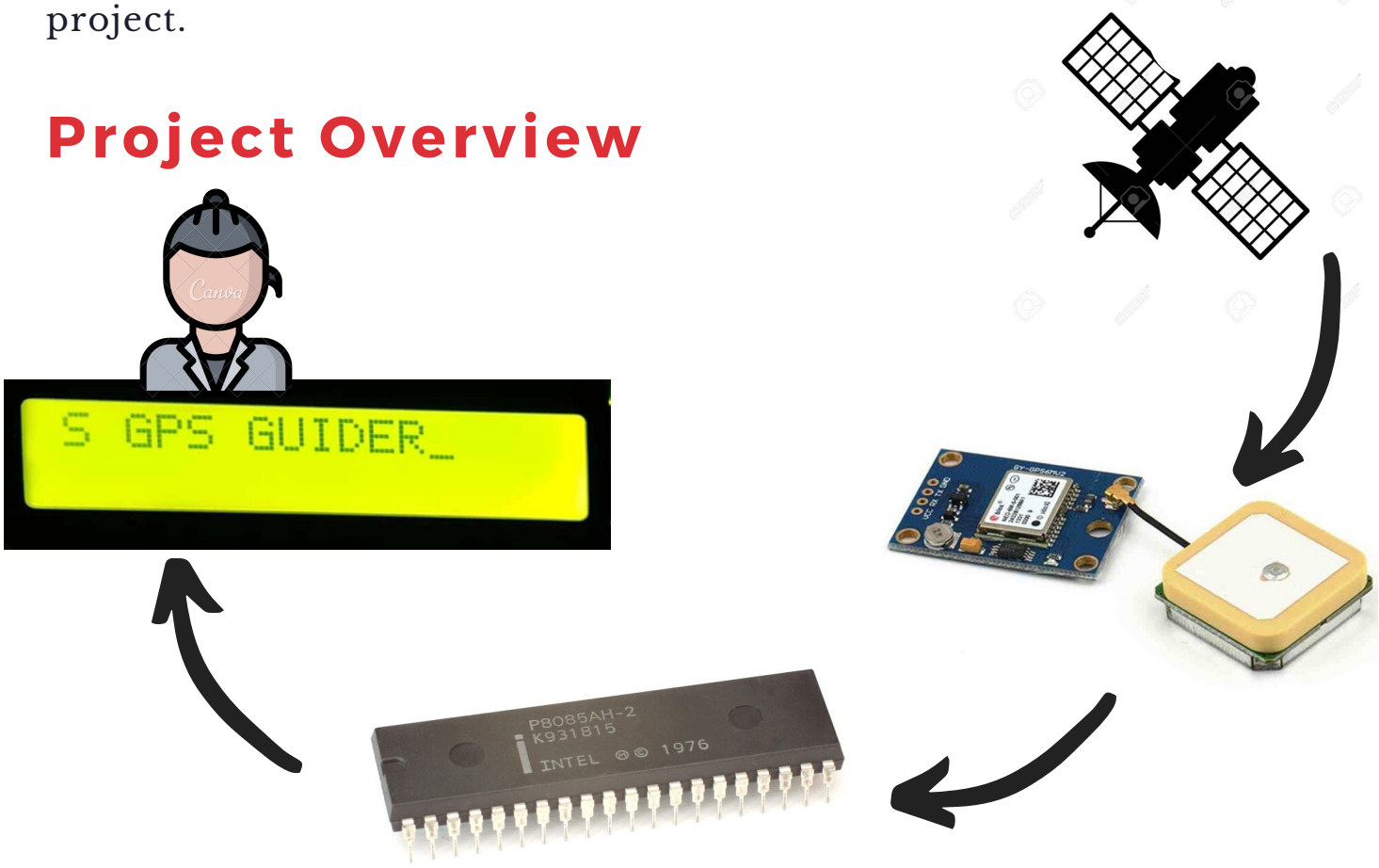
Mission

To design build and implement a GPS Guider system using 8085 Microprocessor

Inspiration

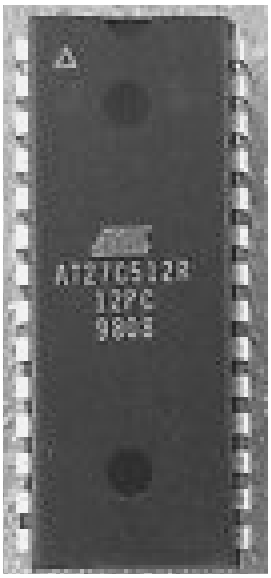
We would like to thank Prof. D.V. Gadre to give us the oppurtunity to work on the project using 8085 Mircroprocessor and to use his teachings in ECC17 course practically in our project.
We would also like to thank him for being a constant motivation and also for the the resources and the facilities he let us to use for our project.

Project Overview



COMPONENTS

S.	Component	Note
1	8085	Microprocessor
2	GPS NEO 6m + Antenna	To get the GPS data from the satellite
3	LCD	16x2 LCD for displaying data
4	ROM	Atmel 28C256 32K volatile ROM
5	RAM	IC 6264 8K RAM
6	PPIO	Intel 8255 Programmable Peripheral Input Output
7	Latch	74373 Address Latch
8	Decoder	74138 Control signals decoder
9	Button switch	For reset, RST 7.5, RST 6.5, SID
10	Crystal	6 MHz crystal that give 3 MHz system frequency
11	N Mosfet	BS 170 is used to generate ~A15 for Memory decoding
12	Resistors	10KOhm resistors for pull up and pull down
13	Capacitors	Ceramic: 22pF with clock and 0.1uF with all ICs and Electrolytic: 10uF smoothening capacitor.



HARDWARE DESIGN AND FABRICATION

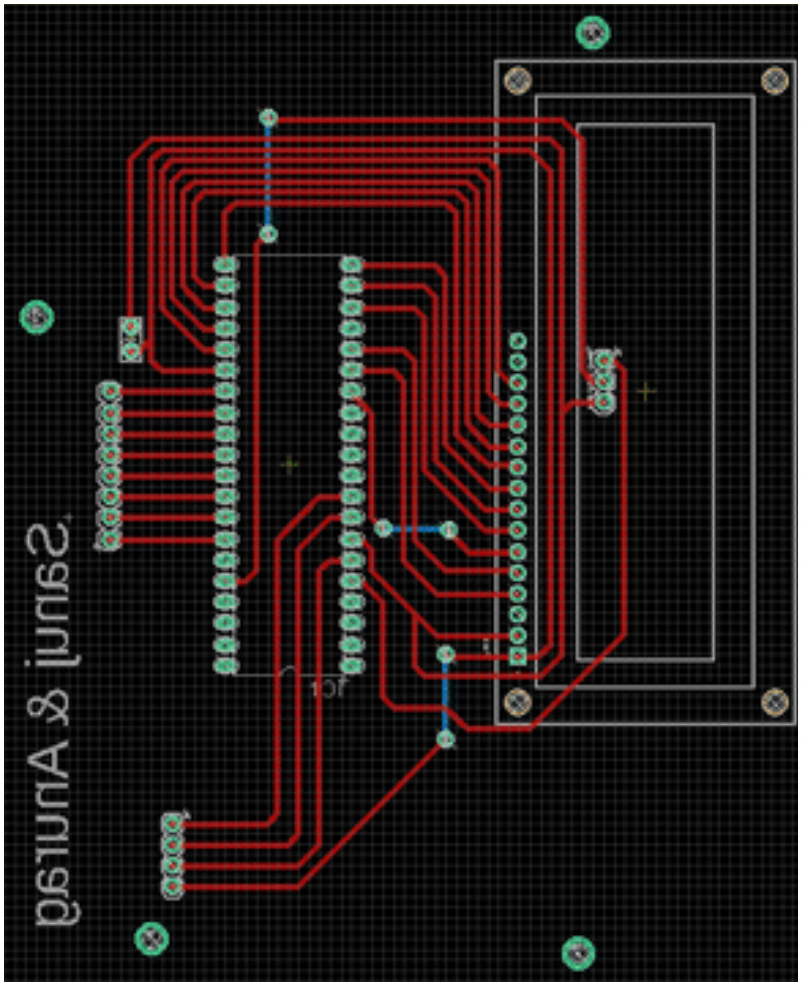
PCB was designed on the Eagle CAD software. We have made 2 PCBs.
First PCB has following subsections:

1. Power subsection
2. Microprocessor Unit
3. Memory subsection
4. GPS input subsection

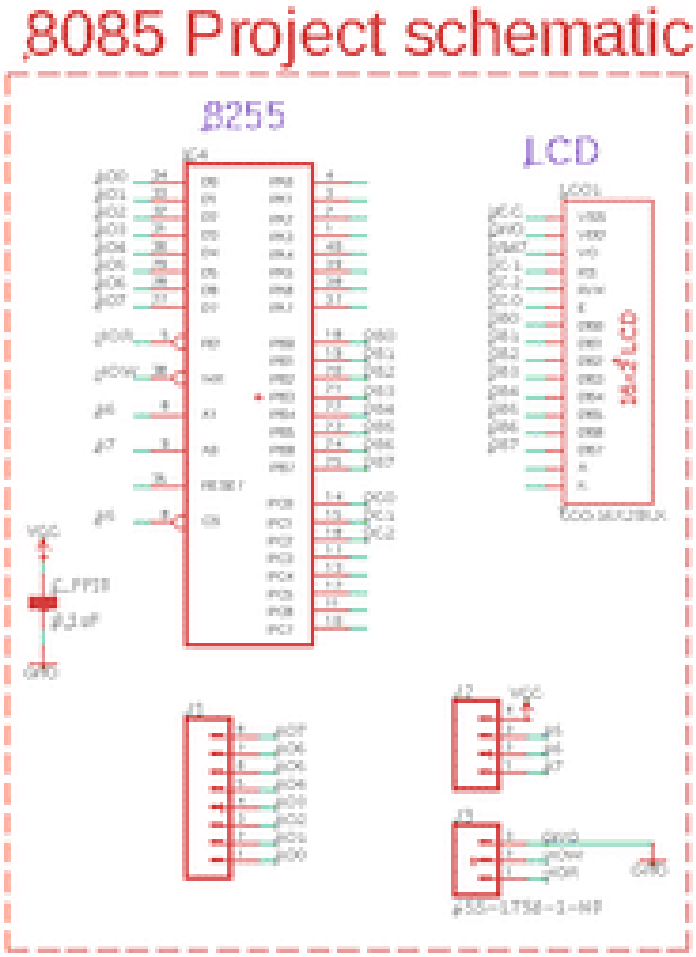
Second PCB has following subsection:

1. Output Display subsetion

OUTPUT PCB

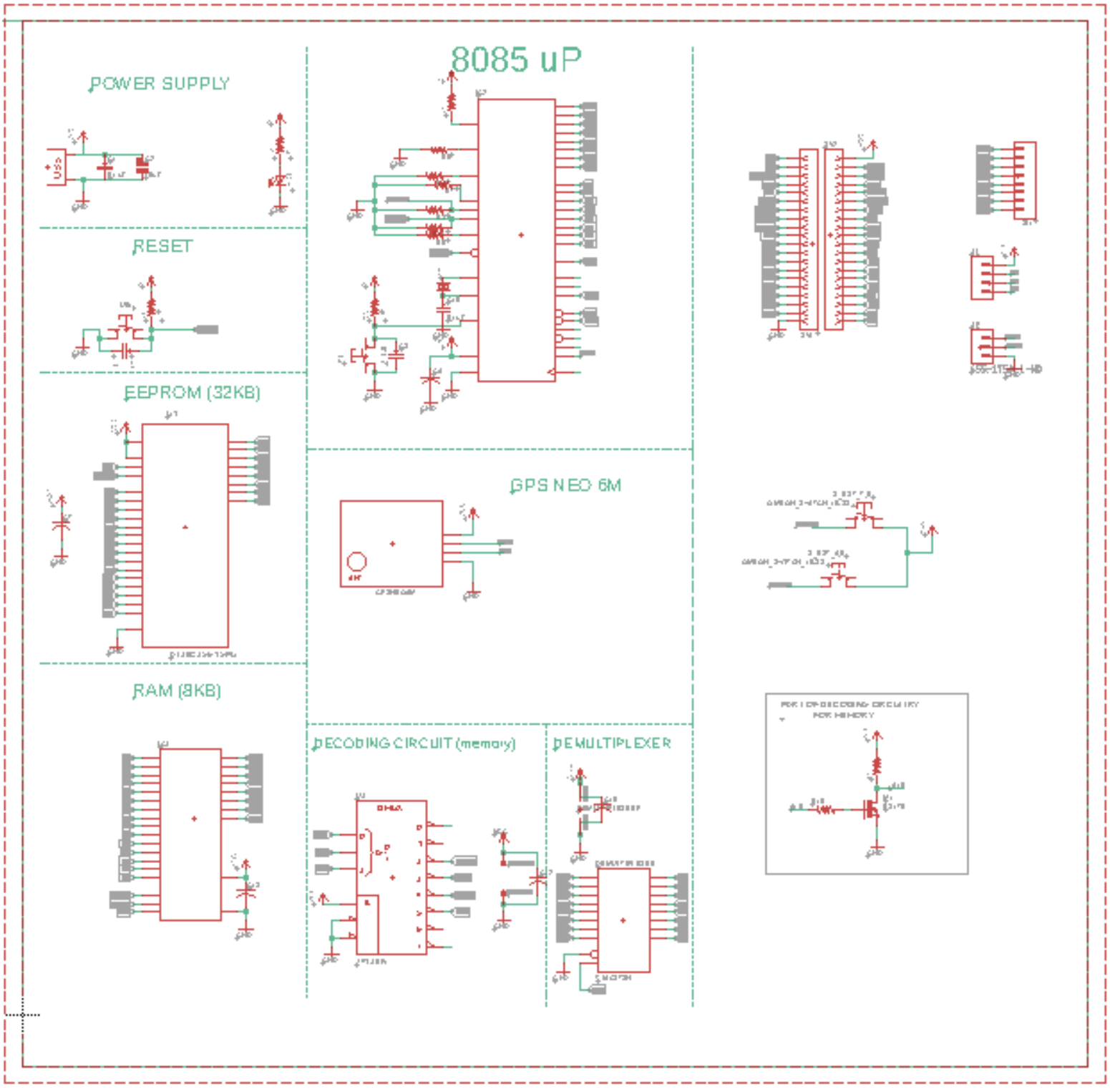


Board File

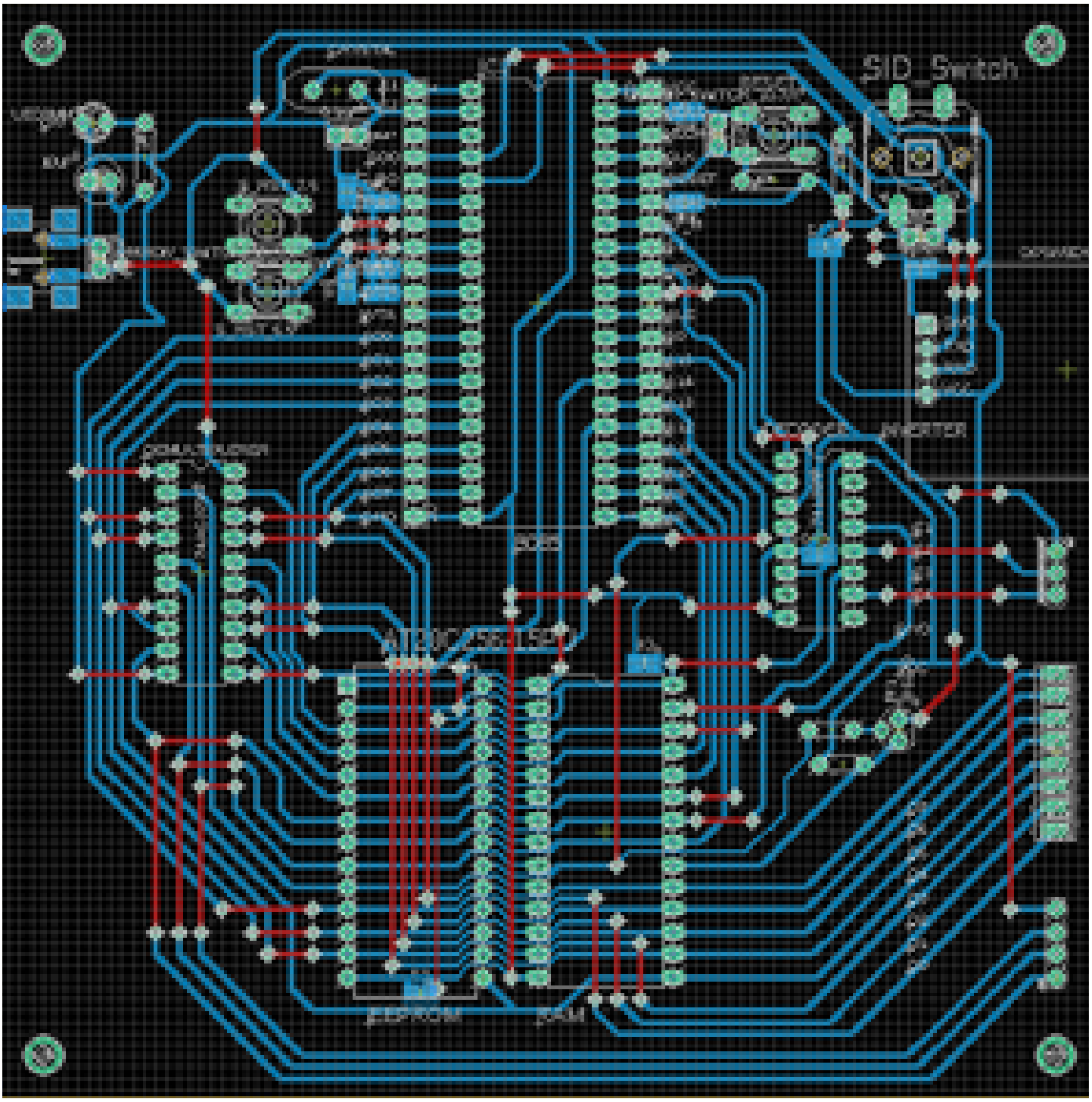


Schematic

PRIMARY PCB



Schematic



Board File

SOFTWARE DESIGN AND IMPLEMENTATION

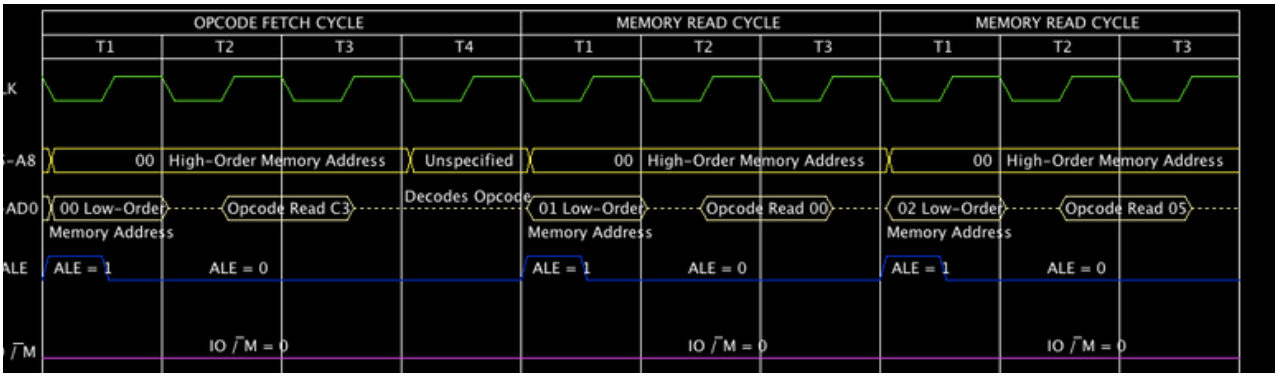
JUBIN 8085 Simulator

Jubin 8085 simulator was used for programming the assembly language code. It helps in get started easily with example codes, and to learn the architecture playfully. It also provides a trainer kit as an appealing functional alternative to real hardware. The users can write assembly code easily and get results quickly without even having the actual hardware.

FEATURES:

- Assembler Editor
- Disassembler Editor
- Assembler Workspace
- Memory Editor
- I/O Editor
- Interrupt Editor
- Debugger
- Simulator
- Helper
- Printing
- Crash Recovery
- 8085 TRAINER KIT
- TOOLS

```
:10000000C3000500000000000000000000000028
:1000300000000000C32E060000000000C3F9050008
:10050000FB3100B03E09303E80D30C3E00D30C3EA0
:100510003047CD9D05CD9905CD99053E38CD99053E
:100520003E08CD99053E01CD99053E07CD99053E82
:100530000ECD99053E8FCD99053E47CDB1053E5074
:10054000CDB1053E53CDB1053E20CDB1053E47CDE1
:10055000B1053E55CDB1053E49CDB1053E44CDB1C5
:10056000053E45CDB1053E52CDB1052100A1365322
:1005700023365421 00A10E0279C630CDB105CD46F7
:10058000670230DC278052100A10E027ECDB105B3
:1005900023 0DC28C05C373057647CDCC053E02D32F
:1005A0000C3E04D30C3E01D30C78D3083E00D30C90
:1005B000C9CDEB0547CDCC053E03D30C3E04D30C8F
:1005C0003E01D30C78D3083E00D30CC93E92D30C25
:1005D0003E02D30C3E05D30C3E01D30CDB08073E94
:1005E00000D30CDAD8053E80D30CC9F5D511307A8A
:1005F0001B7AB3C2F005D1F1C9F5D5C53E20CDB106
:10060000053E20CDB1053E20CDB1053E20CDB10542
:100610003E53CDB1053E41CDB1053E4ECDB1053E77
:1006200055CDB1053E4ACDB105C1D1F1FBC9F5D5D6
:10063000C5E52100A10E047ECDB105230DC237060C
:10064000E1C1D1F1FBC9F5D52017DA4606CD6F0619
:100650001609CD6406201715CA6106781F47C352D4
:1006600006D1F1C9E5F5261125C26806F1E1C9E513
:10067000F5260625C26806F1E1C901FFFF0B78B136
:10068000C27D06C900000000000000000000000005C
:00000001FF
```

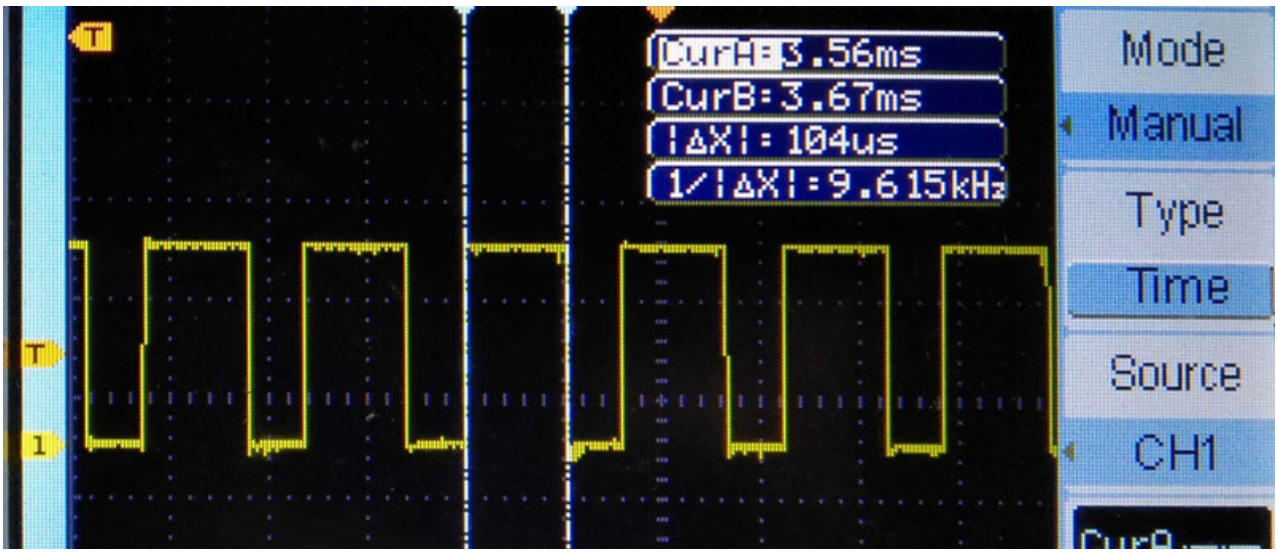


SOFTWARE

```
# PORTA EQU 00H
# PORTB EQU 08H
# PORTC EQU 04H
# CNTRL EQU 0CH

# ORG 0000H
JMP MAIN

# ORG 0034H
JMP ISR6_5

# ORG 003CH
JMP ISR7_5

# ORG 0500H
MAIN:       ;THIS CODE INITIALIZES LCD AND 8255
            EI
            LXI SP, B000
            MVI A, 09H;
            SIM;

            MVI A, 80H; ALL PORTS IN OUTPUT MODE
            OUT CNTRL;
            MVI A, 00H; EN = LOW (PC0 = EN = 0)
            OUT CNTRL;

INIT:       MVI A, 30H; PRECAUTIONARY CODE
            MOV B,A; SAVE THE CODE
            CALL OUTPUT; DB7 CANT BE CHECKED WRITE NOW
            CALL CMDOUT;
            CALL CMDOUT;
            ;(Actual initialisation) SETTING CONTROL REG
            MVI A, 38H;
            CALL CMDOUT;
            MVI A, 08H; Display off
            CALL CMDOUT;
            MVI A, 01H; CODE FOR CLEAR DISPLAY
            CALL CMDOUT;
            MVI A, 07H; ENTRY MODE SET; SHIFTING ON
            CALL CMDOUT;
            MVI A, 0EH; TURN ON CURSOR, DISPLAY, BLINK
            CALL CMDOUT;
```

```
DISPLAY:    MVI A, 8FH; WRITE TO 7TH LOCATION
            CALL CMDOUT;
            MVI A, 47H;                 Print G
            CALL DTAOUT;
            MVI A, 50H;  Print P
            CALL DTAOUT;
            MVI A, 53H;  Print S
            CALL DTAOUT;
            MVI A, 20H;  Print " "
            CALL DTAOUT;
            MVI A, 47H; `               Print G
            CALL DTAOUT;
            MVI A, 55H;  Print U
            CALL DTAOUT;
            MVI A, 49H;  Print I
            CALL DTAOUT;
            MVI A, 44H;  Print D
            CALL DTAOUT;
            MVI A, 45H;  Print E
            CALL DTAOUT;
            MVI A, 52H;  Print R
            CALL DTAOUT

            LXI H, A100H;
            MVI M, 53H;   S
            INX H;
            MVI M, 54H;   T

MAIN_LOOP: ; Doing some work like reading GPS
            LXI H, A100H;
            MVI C, 02H; COUNTER TO READ 7 BYTES
GPSREAD:    MOV A,C;                    PRINTING THE COUNTER
            ADI 30H;      CONVERTING TO ASCII
            CALL DTAOUT;
            CALL SIDATA;
            MOV M,B;
            INX H;
            DCR C;
            JNZ GPSREAD:
```

```
                LXI H, A100H;
                MVI C, 02H; COUNTER TO READ 7 BYTES
PRINTGPS:       MOV A,M;
                CALL DTAOUT;
                INX H;
                DCR C;
                JNZ PRINTGPS;

                JMP MAIN_LOOP;

                HLT;
;=============================================
;==================LCD  FUNCTIONS=============
;=============================================
CMDOUT:         MOV B,A; SAVE DATA BYTE
                CALL CHKDB7; CALLING CHKDB7
OUTPUT:         MVI A, 02H; Select data Reg, SETTING PC1 = RS = 0
                OUT CNTRL;
                MVI A, 04h; Enable Write, PC2 = R/W = 0;
                OUT CNTRL;
                MVI A, 01H; Set Enable HIGH
                OUT CNTRL;
                MOV A,B;    Get the data
                OUT PORTB;
                MVI A, 00H;  Set EN LOW
                OUT CNTRL;
                RET;
```



Checking 1 BIT delay at 9600 bps

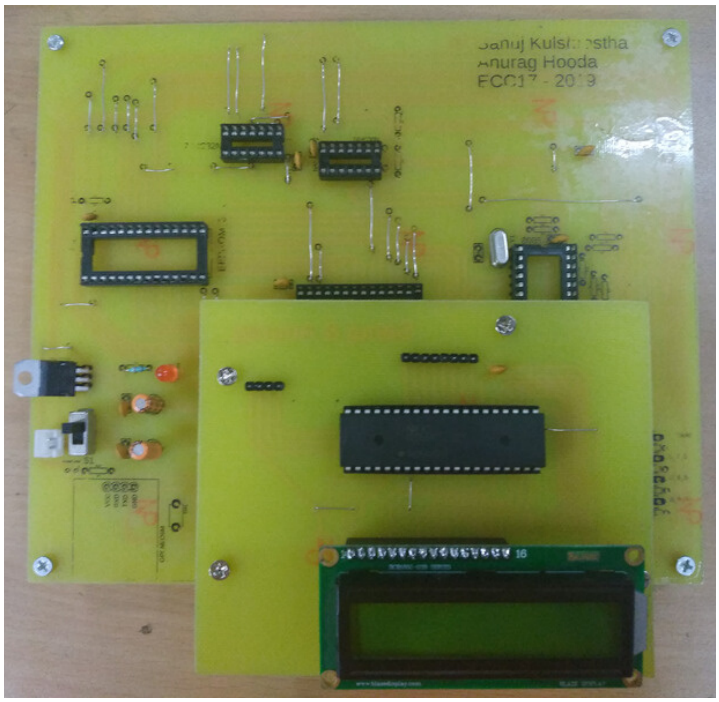
```
DTAOUT:     CALL SCROLLDELAY
            MOV B,A; SAVE DATA BYTE
            CALL CHKDB7; CALLING CHKDB7
            MVI A, 03H; Select data Reg, SETTING PC1 = RS = 1
            OUT CNTRL;
            MVI A, 04h; Enable Write, PC2 = R/W = 0;
            OUT CNTRL;
            MVI A, 01H; Set Enable HIGH
            OUT CNTRL;
            MOV A,B;    Get the data
            OUT PORTB;
            MVI A, 00H; Set EN LOW
            OUT CNTRL;
            RET;

CHKDB7:     MVI A, 92H; Set up portB as inputPort
            OUT CNTRL
            MVI A, 02H; Seting PC1 = RS = 0;
            OUT CNTRL;
            MVI A, 05H; PC2 = R/w = 1
            OUT CNTRL;
READ:       MVI A, 01H; SET ENABLE HIGH
            OUT CNTRL;
            IN PORTB;     READ PORTB AND CHECK DB7
            RLC;
            MVI A, 00H;  SETTING ENABLE LOW
            OUT CNTRL;
            JC READ;      IF DB7=1, GO BACK AND CHECK AGAIN
            MVI A, 80H; SET UP PORTB AS OUTPUT AGAIN
            OUT CNTRL
            RET;

SCROLLDELAY: PUSH PSW;
            PUSH D;
            LXI D,7A30; FOR 250MS delay
SCROLLDELAYLoop:
              DCX D
              MOV A,D
              ORA E
              JNZ SCROLLDELAYLoop
            POP D
            POP PSW
              RET
```

```
;=========================================
;============== ISR ======================
;=========================================
ISR7_5:     PUSH PSW;
            PUSH D;
            PUSH B;

            MVI A, 20H;
            CALL DTAOUT;
            MVI A, 20H;
            CALL DTAOUT;
            MVI A, 20H;
            CALL DTAOUT;
            MVI A, 20H;
            CALL DTAOUT;

            MVI A, 53H;
            CALL DTAOUT;
            MVI A, 41H;
            CALL DTAOUT;
            MVI A, 4EH;
            CALL DTAOUT;
            MVI A, 55H;
            CALL DTAOUT;
            MVI A, 4AH;
            CALL DTAOUT;

            POP B;
            POP D;
            POP PSW;
            EI;
            RET;

ISR6_5:     PUSH PSW;
            PUSH D;
            PUSH B;
            PUSH H;

            LXI H, A100H;
            MVI C, 04H; COUNTER TO READ 4 BYTES
PRINTMEM:MOV A,M;
            CALL DTAOUT;
            INX H;
```

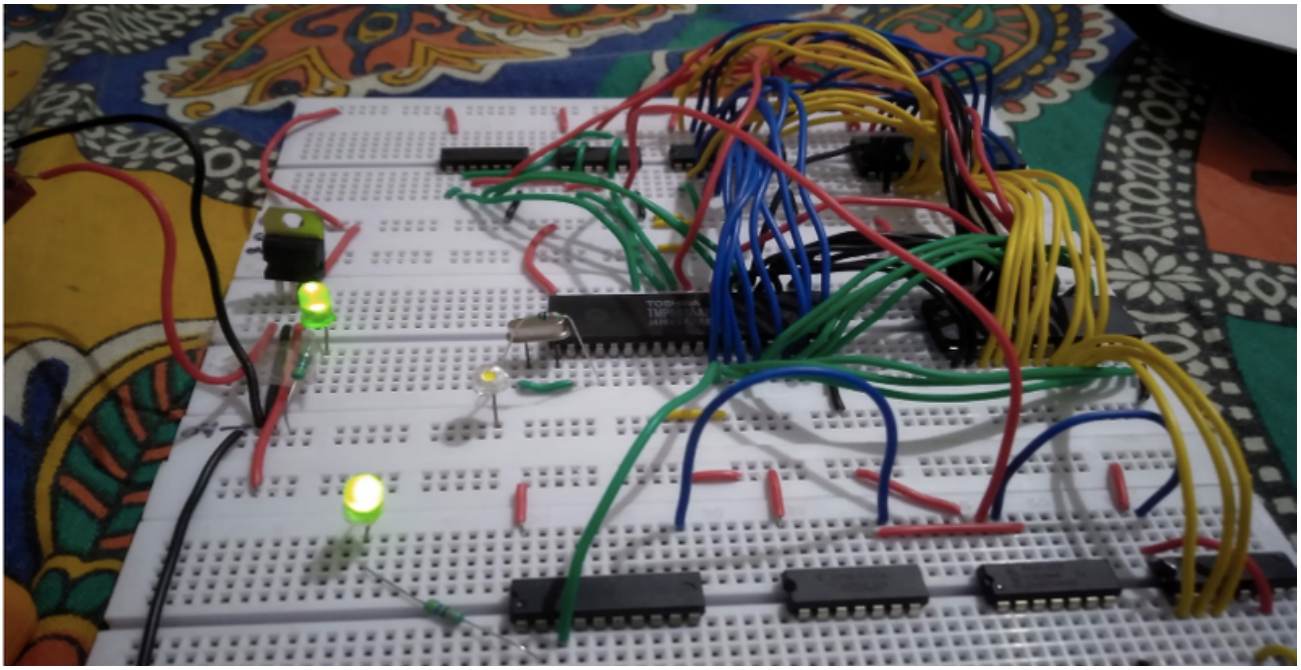
```
                DCR C;
                JNZ PRINTMEM

                POP H;
                POP B;
                POP D;
                POP PSW;
                EI;
                RET;

;=============================================
;============== GPS SID READ ================
;=============================================
;FUNCTION- SID DATA
;This function returns a byte in B register, Then main function need
store it in memory
SIDATA:         PUSH PSW;
                PUSH D;
                RIM;   Read SID Line
                RAL;   Put this bit in CY flag
                JC SIDATA; if bit = 1, this means not started
                           ;if bit = 0; this means start bit
                CALL HAFBIT; We got start bit; wait for half bit time

                MVI D, 09H; Counter for 9 bits

NXTBIT:         CALL BITTIME; Wait for 1 bit period
                RIM;
                RAL;
                DCR D;
                JZ OVER;

                MOV A,B;
                RAR;
                MOV B,A;
                JMP NXTBIT;

OVER:           POP D;
                POP PSW;
                RET;
```

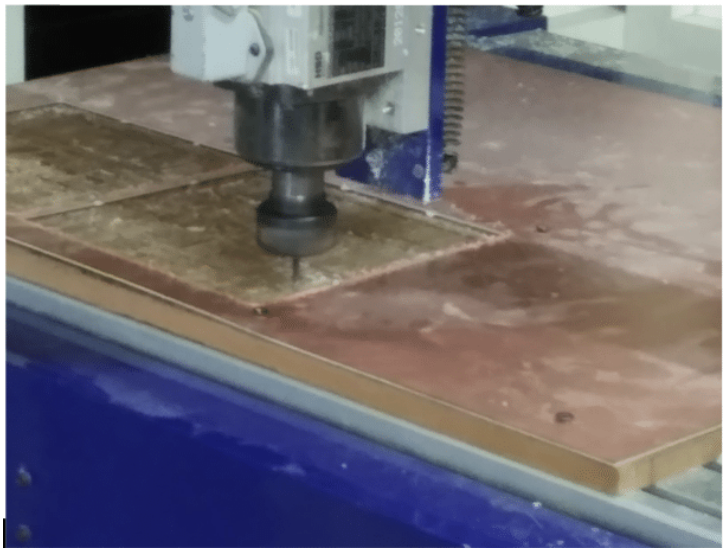
```
;FUNCTION- BIT TIME
;Calling this subroutine will cause delay of 103us (incluing call)
BITTIME:    PUSH H;
            PUSH PSW;
            MVI H, 11H;
bitTimeloop: DCR H;
             JNZ bitTimeloop
             POP PSW;
             POP H;
             RET;

;FUNCTION- HAFBIT (Half bit delay)
;Calling this subroutine will cause delay of 103us (incluing call)
HAFBIT:    PUSH H;
           PUSH PSW;
           MVI H, 06H;
hafTimeloop: DCR H;
             JNZ bitTimeloop;

           POP PSW;
           POP H;
           RET;

;=============================================
;============== DELAY ========================
;=============================================
DELAY500:
           LXI B,FFFF
DELAY500Loop:
           DCX B
           MOV A,B
           ORA C
           JNZ DELAY500Loop
           RET
```

PROJECT MANAGEMENT



We started earlier than anyone else and were the first to fabricate the PCB. Though our first board (Top) didn't worked as per expectation, we had enough time to go back to drawing board and fabricate a new one (Right).
After finishing the hardware we move on to the software. We started with testing waveform on SOD, Reading a byte from the SID, Copying SID waveform on SOD, Printing on LCD.
After these small code testing, we merged them for the complete software development.





REFERENCES AND BIBLIOGRAPHY

Following resources helped us a lot in our project:

1. http://8085projects.in/
2. Ramesh S Gaonkar, Microprocessor Architecture, Programming, and Applications with the 8085, Penram International Publishing (India) Pvt. Ltd.
3. http://mrityunjai.in/tag/8085-project/