NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

DIVISION OF ELECTRONICS AND COMMUNICATION ENGINEERING

EC316 MICROPROCESSOR LAB PROJECT REPORT

8085 based
Smart Mirror

AN INTEL 8085 MICROPROCESSOR BASED PROJECT

Submitted by
Manoj Sahu (093EC15)
Udit Kumar Agarwal (188EC15)

Submitted to
Prof. Dhananjay V. Gadre

May, 2018

Contents

1 Acknowledgements

Firstly, We would like to express our sincere gratitude to our advisor Prof. Dhananjay V.Gadre for the continuous support on our 8085 project and related research, for his patience, motivation, and immense knowledge.

We thank our fellow labmates in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last few days. In particular, We are grateful to Prof. Ramesh S. Gaonkar for his enlightening and vivid text book named, Microprocessor Architecture, Programming and Applications with the 8085.

Last but not the least, We would like to thank our family: our parents and to our classmates for supporting us spiritually throughout the project preparation.

2 Abstract

Our smart mirror won't just say you're the prettiest of them all. It will also tell you the temperature, daily tasks(to-do list), weather outside, upcoming calendar appointments, and more. Encapsulated within a wooden frame, it will consist of a two-way mirror with an electronic display and wifi module behind the mirror. In form of widgets, it deploys various multi-colour LED's accompanied with stencils to form an iconic indicator for various weather conditions.

An essential part of modern interior designing ideas is the use of mirrors. Long gong are the days, when mirror is just used to see your face. Now, Mirrors being an integral part of every household are now used to provide glamorous user-interface for a variety of smart products. Our project, 8085 based smart mirror is an attempt to intermingle our modern luxury with a so called 'obsolete' Intel's microprocessor 8085.

Main objective of our work is to keep the user updated with his To-Do list, which was once entered by the user, stored securely on a reliable cloud database. With our Smart Mirror android application interface, user can smoothly update the data, with his location (required for weather) along with the SSID and password of the wireless interface, being used for data exchange purpose. Along with this, weather forecast along with temperature is displayed for user's convenience.

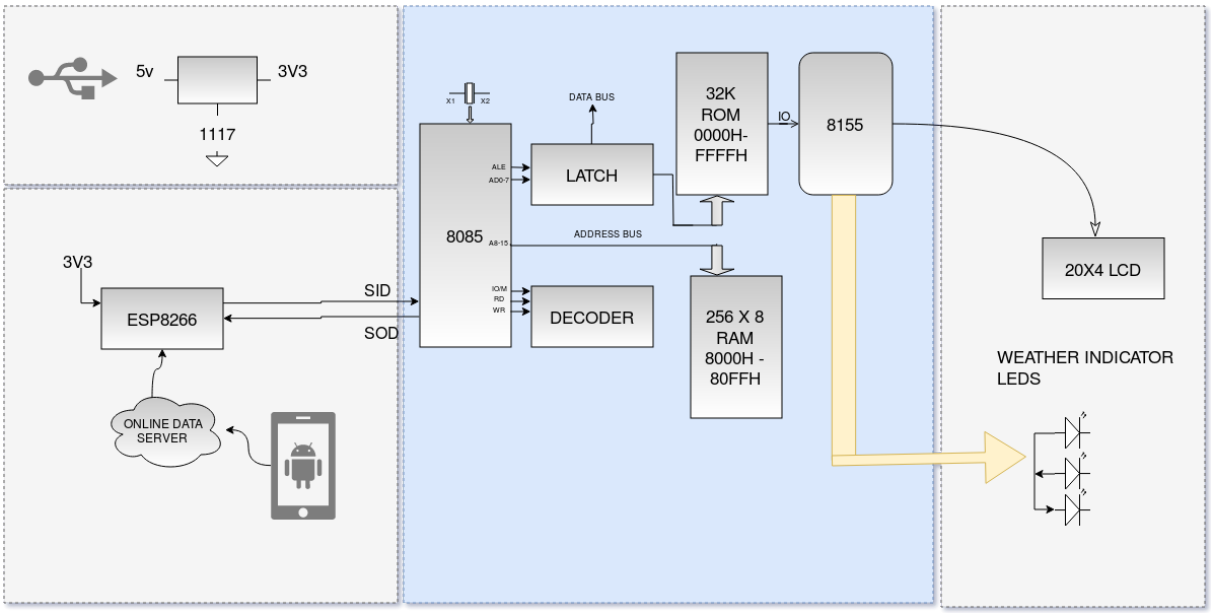
3 Background And Motivation

Utilizing this opportunity of understanding a Microprocessor's architecture and programming and implementing the knowledge gained, by actually making a project, we decided to select a project that would help us learn the essentials of a microprocessor based system design. Our project Smart mirror, presents an enthralling blend of science fiction with reality.

4 Justification

Nowadays, implementation of smart mirrors includes using high end processors like in raspberry pi and other single board computers. Using a cheap microprocessor like 8085 for recreating smart mirror, justifies it's processing power.
Moreover, Mirrors are an essential parts of every room, using it to display important information along with To-Do list sounds to be an interesting and a useful idea.

5 Block Diagram



6 Description

The project employs an 8085 microprocessor interfaced with a wifi module named, ESP8266 which fetches the data from the online database, maintained by the user. As far as power supply is concerned, all components except esp8266 works on 5V, while esp needs 3.3V, thus a regulator LM1117 is used to convert 5V to 3V3. Data is displayed through the help of LEDs and LCD.
8155 is being employed as a IO controller with Port A used for Lower Byte data lines of LCD and Port B for higher data Byte. Pins of Port C are being used for LCD control signals and for toggling weather LEDs.
For RAM, we havn't used any external IC, 256 Bytes within 8155 seems sufficient to store immediate information received from esp8266.

7 Technical Details

7.1 Hardware: Board 1

Our 8085 based smart mirror consists of following processing parts:

7.1.1 Core Microprocessor (Intel 80C85)

At the heart of this project is an Intel 80C85 microprocessor(μP) from Toshiba Semiconductors is a CMOS implementation of the original TTL 8085 which has a quiescent current of as low as 10mA. The microprocessor has a 64KB memory address space and 256 bytes of I/O space. In this case the 8085 is running at 3MHz from the clock of a 6MHz crystal oscillator.

7.1.2 Peripherals And Addressing

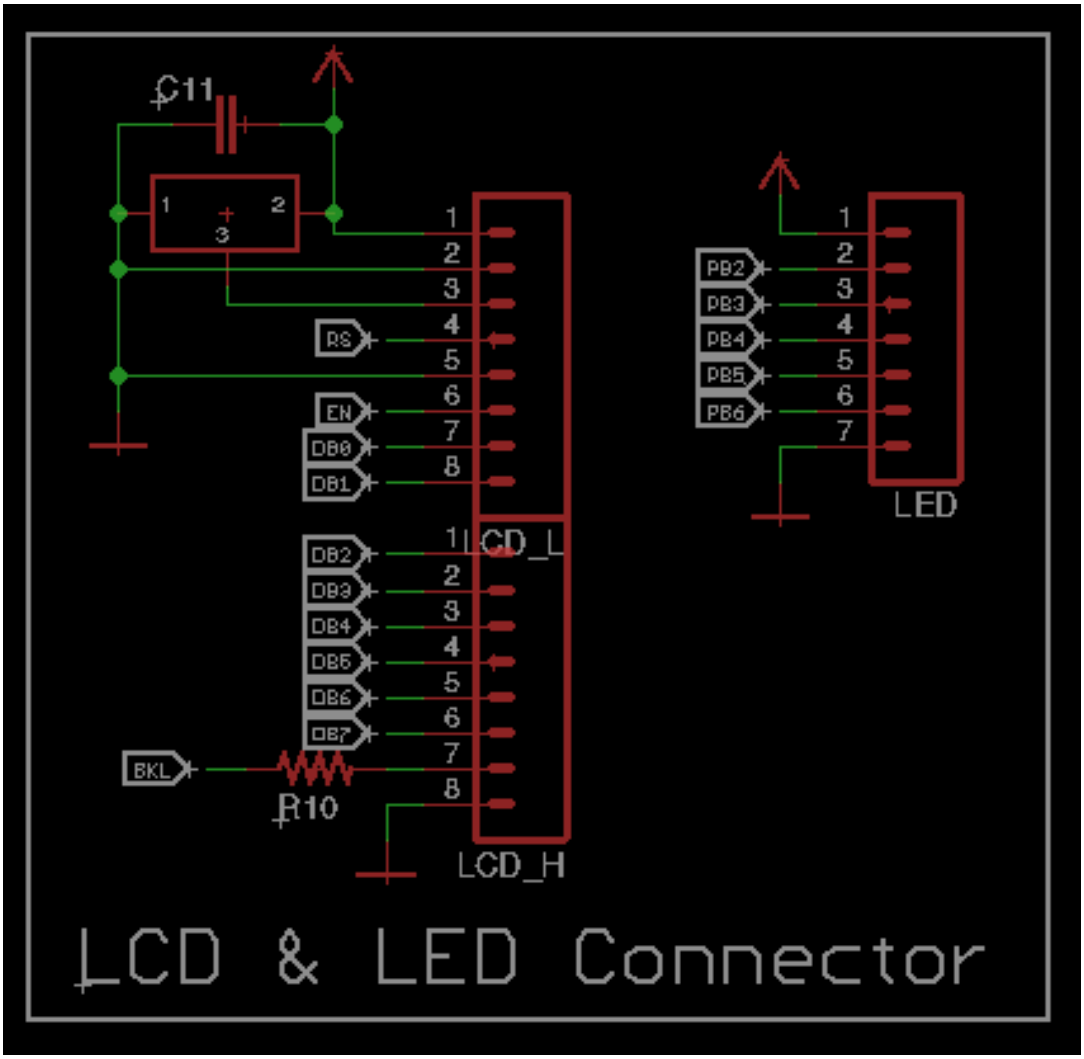
This project employs an Atmel ROM 27C256 and Intel 8155 PPI+RAM+Timer, as peripherals for the microprocessor and hence they occupy some part of the 8085's address space.

- *Atmel ROM 27C256*: It is a 32KB rom which is over kill for this project but was chosen due to its ready availability and most of the address space of the 8085 microprocessor is not in use.
  -The address space for the ROM ranges from $0000h$ to $7fffh$.
  -Chip Select($\overline{CS}$) for this IC is connected to $A15$ of the μP.

- *Intel 8155*: Is a IC specially developed to be used with the Intel 8085 μP. It has 256bytes of RAM, 3 I/O ports 2 of 8bit and 1 of 6bit, with handshake capability using the former as I/O and the latter for handshaking, and a 8 bit timer with clock out.
  -The address space for the RAM ranges from $8000h$ to $FFFFh$.
  -The IO addresses for 8155CWR-$80h$,PORTA-$81h$,PORTB-$82h$,PORTC-$83h$-Chip Select($\overline{CS}$) for this IC is connected to $A15*$ of the μP.

7.1.3 Display

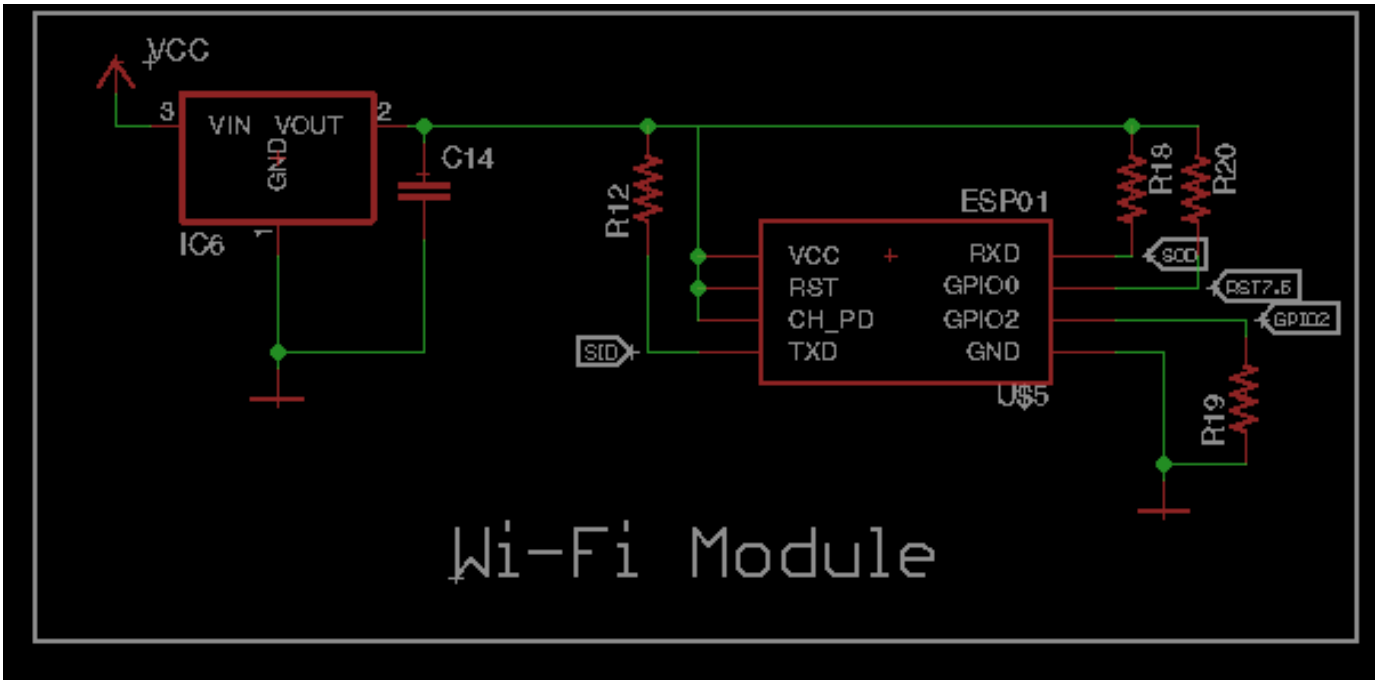
A 20X4 LCD is used for displaying the characters send by the esp to 8085. The appropriate character gets printed based on the values given by the Microprocessor. 8155 is working as a bridge to communicate between 8085 and the LCD.



LCD Display

7.1.4 ESP interfacing Circuit

This is the circuit which is actually used for communication with ESP8266 using UART. ESP8266 supplies a rectangular pulse to 8085 RST7.5 interrupt pin, then 8085 enters the data receiving mode and waits in a loop for a start bit on SID. Data, then transferred by the ESP is taken by 8085 and stored at a memory location starting from $8001H$ until it receives a \t character which marks the end of transmission. ESP8266 on a single go, will send the user data along with weather conditions to 8085. Both of these are separated by \a character, which is used to mark the starting of weather information.



ESP interfacing Circuit

7.2 Software Implementation and Interesting Concepts

7.2.1 UART implementation

Communication between 8085 and ESP is done via UART at a baud rate of 9600, no parity and 1 stop bit. UART is implemented on 8085 by, first waiting in an infinite loop for start bit, marked by falling edge on SID pin. Once the start bit is found, registers B,E,and D are initialized. Register B is used as an argument for delay control. Register E will store the incoming byte and register D is used to maintain a count of next 7 bits. After start bit next bit is read after 150us delay. Bits are then read sequentially each after 104us delay and are stored in E register with shifting towards right after each bit. Since, LSB is transferred first so after receiving 7 bits, we have the exact byte in register E, which is then compared with \t character (which marks the end of transmission) it false, content of E is stored on memory location and then loop continues again by detecting the start bit. Delay is calculated from 9600 baud rate, which comes out to be 104us.

```
START_BIT:        RIM  // CHECK FOR START BIT
          ANI 80
          JNZ START_BIT
          MVI E,00
          MVI D,07
          MVI B,1C
          CALL DELAY   // 150uSEC DELAY

BIT_READ:         RIM
          ANI 80
          JNZ HIGH
          XRA A        // LOW BIT

HIGH:     ORA E
          RRC
          MOV E,A
          XRA A        // JUST TO CONSUME EXTRA TIME
          MVI B,10
          CALL DELAY
          DCR D        // BIT COUNTER
          JNZ BIT_READ
          RIM  // READS LAST BIT
          ANI 80
          ORA E
// ALL BITS RECIEVED
          MOV E,A
          CPI 09
          JZ LOOPEXIT  // STOP RECIEVING IF WE GET '/t'
          MOV M,E
          INX H
          INR C
```

UART implementation

7.2.2 Interfacing a 20X4 LCD

In our project 20X4 LCD is being used in 8 bit mode. LCD initialization begins with first setting up the font size. Also, it's worthy to specify here that LCD latches data on it's data bus only during a falling edge on EN pin. Then the LCD is cleared followed by an appropriate delay. LCD needs around 2ms to clear out completely before taking any new data. Then Display is switched ON, along with appropriate cursor settings. Then as a welcome text, "8085 based smart mirror" is displayed on it. Before calling printloop we need to supply the number of characters it's needed to be print. Cursor is incremented after initializing. In order to switch lines we need to send the corresponding cursor location to LCD like, C4 for second row fourth column etc.

```
LCD_INIT:          MVI D,38     // DL - 2,FONT 5X7
            CALL SEND_CMND
            MVI D,01
            CALL SEND_CMND
            MVI B,FF
            CALL DELAY
            MVI B,2F
            CALL DELAY
            MVI D,0C     // DISPLAY ON,CURSOR OFF,BLINK OFF
            CALL SEND_CMND
            MVI D,06     // CURSOR INCREMENT
            CALL SEND_CMND
            LXI H,0C00
            MVI C,04     // NO OF CHARACTERS '4'
            CALL PRINTLOOP
            MVI D,C4     // SECOND ROW 4 COLM
            CALL SEND_CMND
            MVI C,05
            CALL PRINTLOOP
            MVI D,9D     // THIRD ROW
            CALL SEND_CMND
            MVI C,05
            CALL PRINTLOOP
            MVI D,E2     // FOURTH ROW
            CALL SEND_CMND
            MVI C,06
            CALL PRINTLOOP
            RET
```

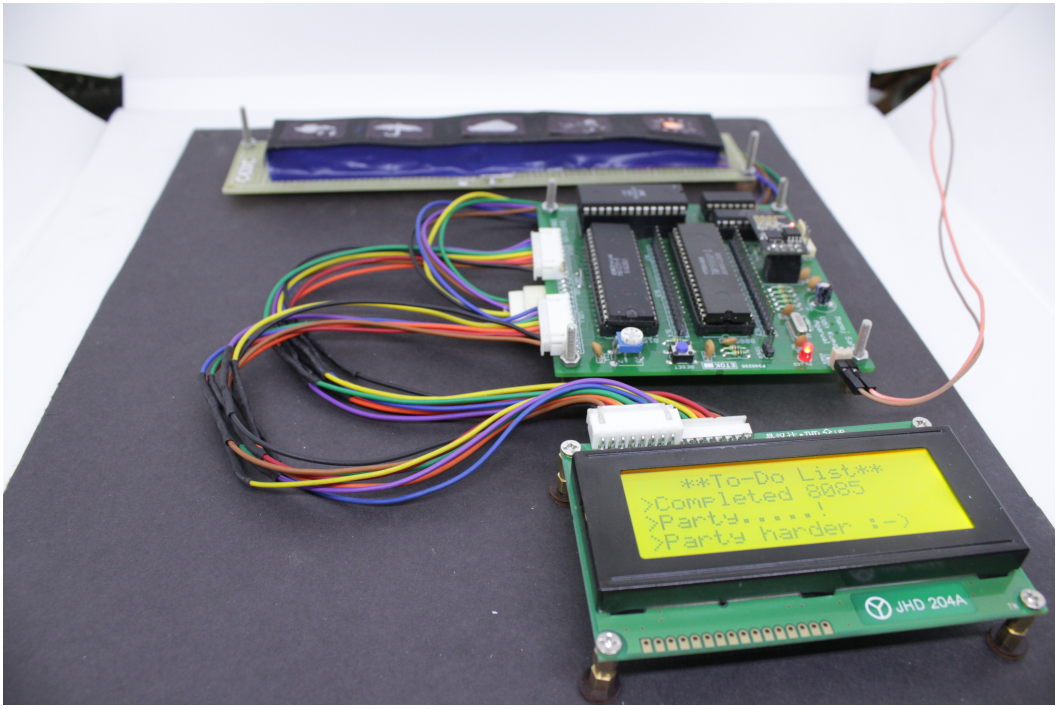
20X4 LCD interfacing

7.3 Configuring ESP8266

ESP is programmed using a FTDI programmer , such that GPIO0 is used to supply interrupt to 8085's RST 7.5. ESP fetches the data from the cloud and preformats it for removing erroneous data. SSID and password of the wifi to which ESP is connected can be changed by issuing a command on android app, along with new SSID and password. ESP is used in master mode, which instructs 8085 to display data after receiving it's interrupt. Major difficulty faced in this section was, providing a stable internet connection to esp. Unstable connection is leading to corrupted data.

7.4 Hardware Assembly

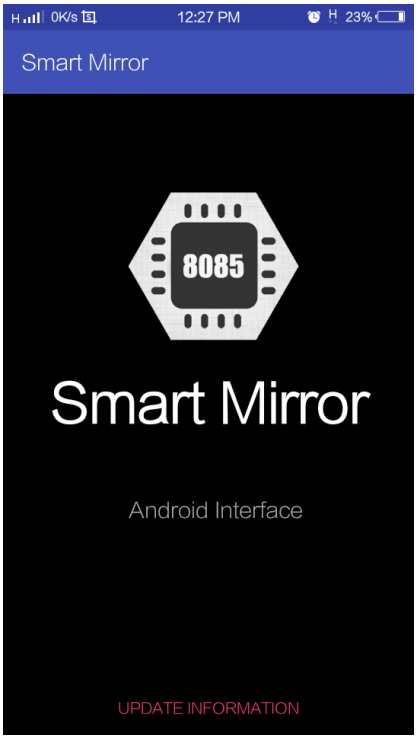
As our hardware assembly, we have deployed a 2 way mirror fitted in a wooden frame with sufficient space inside to carry different electronics components. At the back of the two way mirror, we have kept our 20X4 LCD along with weather icons.



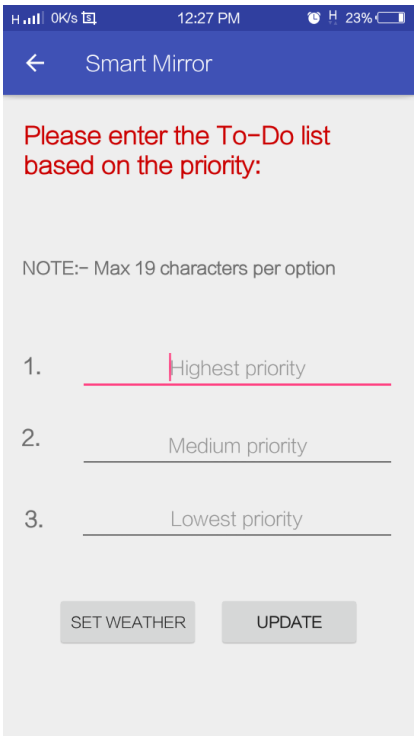
Hardware assembly

7.5 Our Android interface

An android application was made to upload user data to the cloud database using REST API's. Application can also be used to change SSID and password of the ESP.

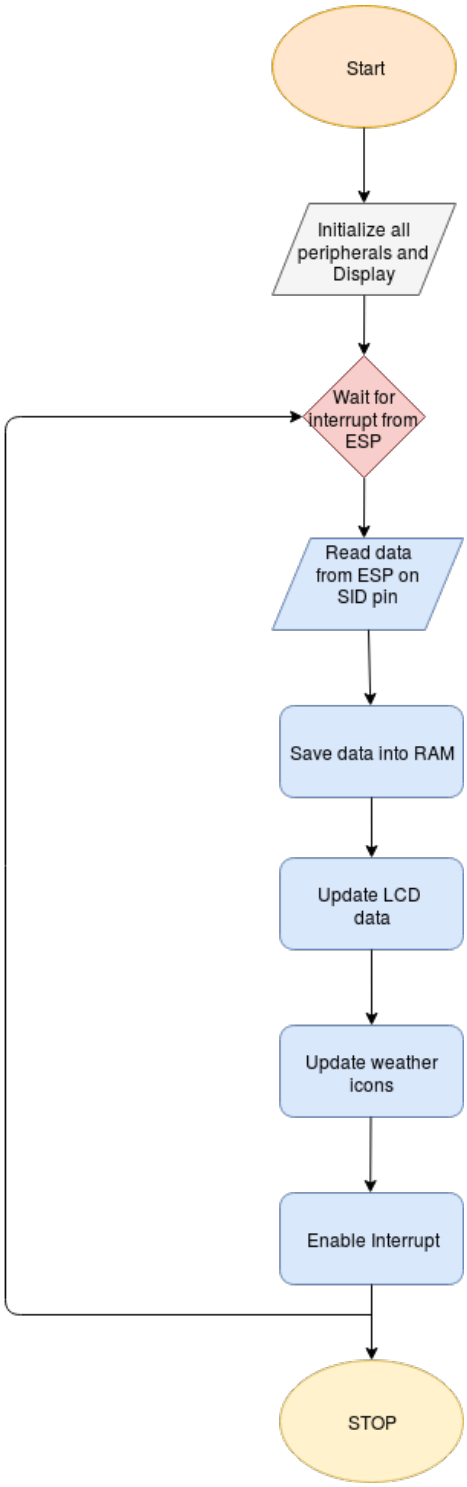


Android application



Android application

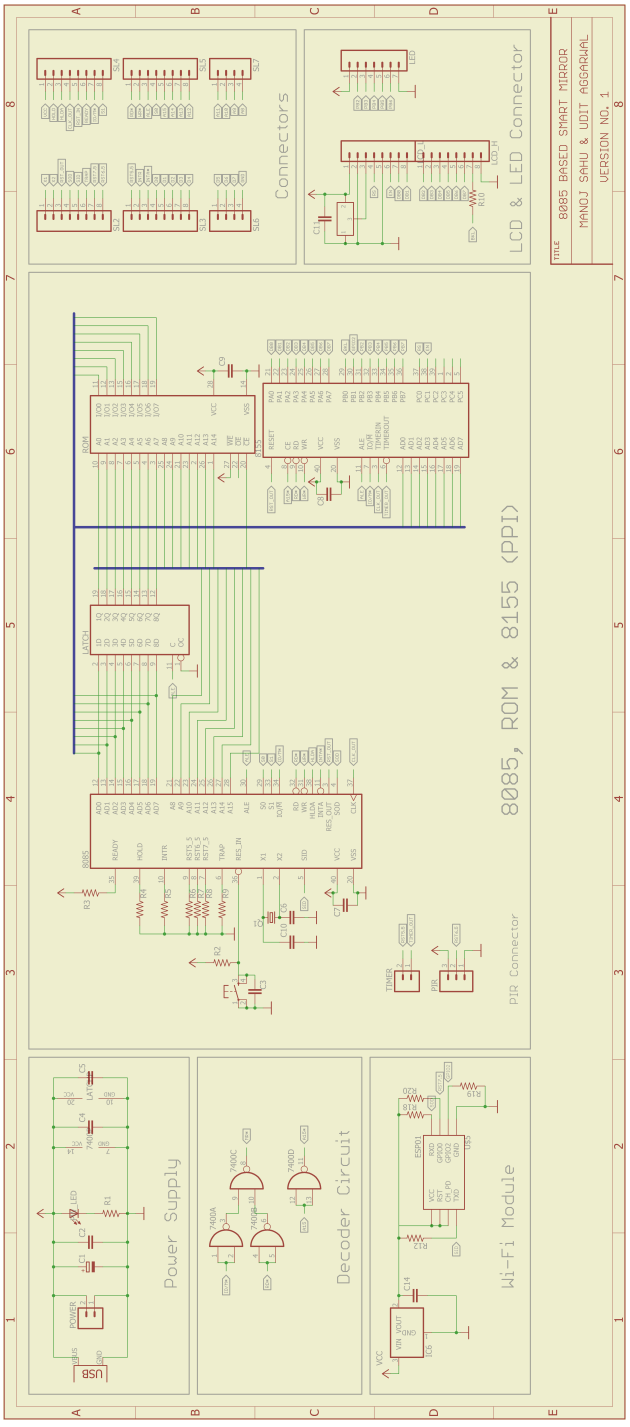
8 Flow Chart


Start
Initialize all peripherals and Display
Wait for interrupt from ESP
Read data from ESP on SID pin
Save data into RAM
Update LCD data
Update weather icons
Enable Interrupt
STOP

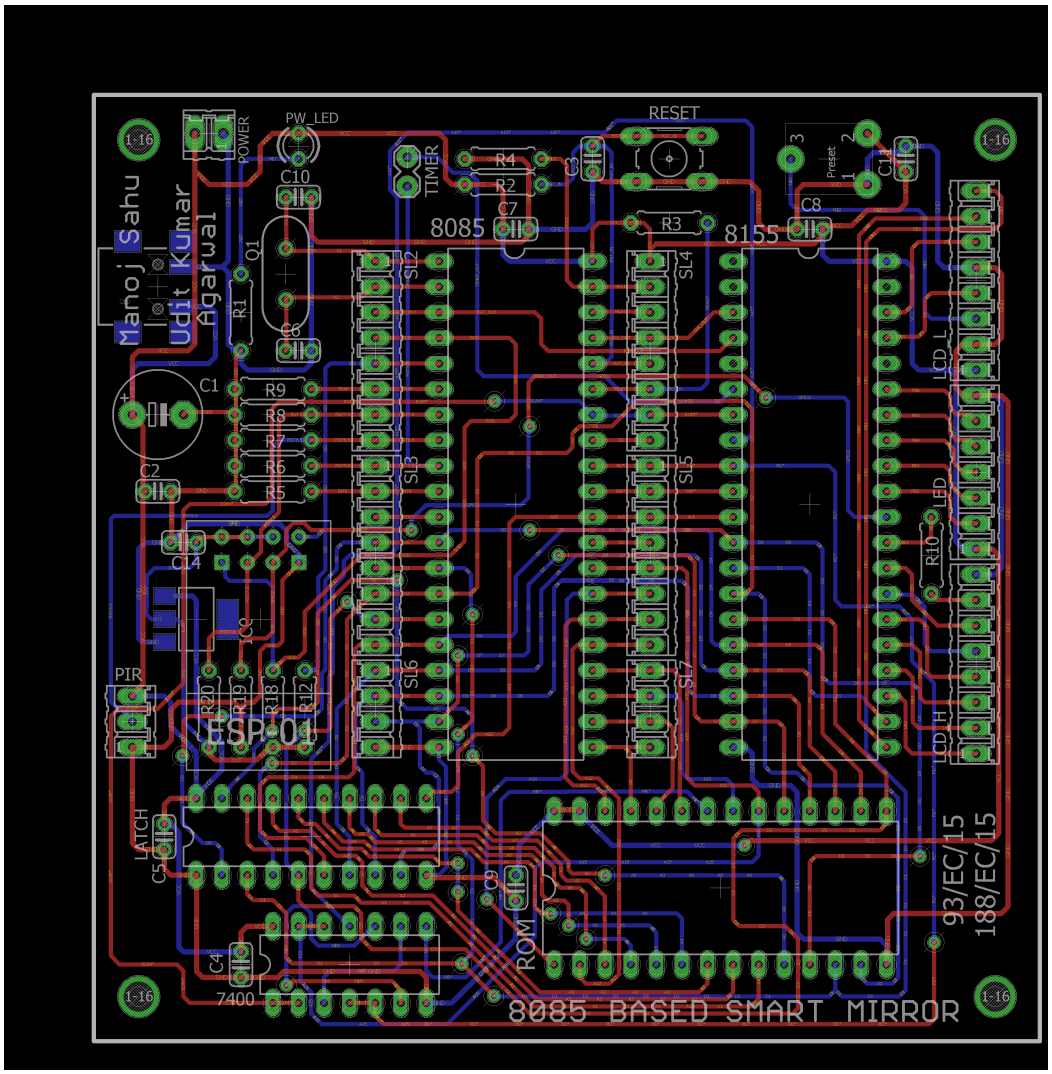
9 Supporting Materials

9.1 Project Schematic And Boards

9.1.1 Board 1



Schematic of Board 1



Board Layout of Board 1

9.2 Bill of Materials

Part	Value	Device	Package	Quantity
C1	10uf	CPOL-EUE5-8.5	E5-8,5	1
C2	0.1uf	C-EU025-024X044	C025-024X044	7
C4	10uf	C-EU025-024X044	C025-024X044	1
C6	22pf	C-EU025-024X044	C025-024X044	2
IC1		8085	DIL40	1
IC2		8155	DIL40	1
IC3	74HC573N	74HC573N	DIL20	1
IC4		58C256P	DIL28-6	1
IC5	7400N	7400N	DIL14	1
IC6	REG1117	REG1117	SOT223	1
LCD_H		M08	08P	2
POWER		M02	02P	1
PW_LED	LED3MM	LED3MM	LED3MM	5
Q1	6MHZ	CRYSTALHC49US	HC49US	1
R1	1k	RESISTOR0207/7	0207/7	1
R2	10k	RESISTOR0207/7	0207/7	12
R10	470	RESISTOR0207/7	0207/7	1
RESET	OMRON_SWITCH_10-XX	OMRON_SWITCH_10-XX	B3F-10XX	1
SL2		M08	08P	4
SL6		M04	04P	2
TIMER	M02LOCK	M02LOCK	1X02_LOCK	
U$2	USB(POWER)	USB(POWER)	USB(POWER)	
U$3	PRESET_LR	PRESET_LR	PRESET_LR	
U$5	ESP01	ESP01	ESP01	

10 References

[1] Microprocessor Programming Architecture and Application with 8085 by Ramesh S. Gaonkar.
[2] http://www.8085projects.in/
[3] https://www.electronicshub.org/interfacing-20x4-lcd/