

REMOTE INTERFACING WITH 8051

Dhiraj Bennadi
Rajat Chaple

Final Project Report
ECEN 5613 Embedded System Design
December 11, 2021

1	INTRODUCTION.....	2
2	TECHNICAL DESCRIPTION	2
2.1	HARDWARE ARCHITECTURE	2
2.1.1	Power Supply Design	3
	<i>Following table contains maximum power considerations for listed components</i>	<i>3</i>
2.2	PCB DESIGN	5
2.2.1	PCB layout	5
2.3	HARDWARE INTERFACING.....	8
2.3.1	Interfacing ESP32 (Wi-Fi module) with MSP432	8
2.3.2	Interfacing 8051 with MSP432.....	8
2.3.3	Interfacing LCD with MSP432	9
2.4	SOFTWARE ARCHITECTURE.....	10
2.4.1	ESP32	11
2.4.2	Programming 8051.....	14
2.4.3	GUI Application for sending Hex file.....	15
2.5	TESTING PROCESS	17
3	RESULTS AND ERROR ANALYSIS.....	19
4	CONCLUSION.....	21
5	WORK DISTRIBUTION	21
6	FUTURE DEVELOPMENT IDEAS	21
7	ACKNOWLEDGEMENTS	21
8	REFERENCES	22
8.1	APPENDIX - BILL OF MATERIALS.....	23
8.2	APPENDIX - SCHEMATICS	23
8.3	APPENDIX - FIRMWARE SOURCE CODE	23
8.4	APPENDIX - SOFTWARE SOURCE CODE	23
8.5	APPENDIX - DATA SHEETS AND APPLICATION NOTES	23

Table 1: Power considerations for master system	3
Table 2 Program stages	20
Table 3 Work Distribution	21
Figure 1: Hardware Architecture.....	2
Figure 2: Power supply circuit	3
Figure 3 Vout Analysis Peak Voltage	3
Figure 4 Vout Analysis.....	4
Figure 5 PCB Layout.....	5
Figure 6 Board Prototype Top	6
Figure 7 Board Prototype Bottom	7
Figure 8 Interfacing ESP32-MSP432.....	8
Figure 9 Interfacing 8051-MSP432.....	8
Figure 10 Interfacing LCD-MSP432.....	9
Figure 11 Software Architecture	10
Figure 12 ESP Flash Download Tool.....	11
Figure 13 Flowchart - ESP32 Wi-Fi Setup.....	12
Figure 14 Flowchart - ESP32 Server Setup.....	13
Figure 15 Programming 8051.....	14
Figure 16 GUI Application.....	15

1 INTRODUCTION

With the emergence of COVID restrictions, the way we perceived our lives changed. Most of the world started living and working remotely. Even, Educational Institutions started with Remote teaching. Though the situations have ameliorated as on today, the possibility of reemergence of restrictions cannot be ruled out. Embedded System Design is a course that is best learnt with hands up experience in the labs. Due to the restriction on the number of people in a closed space, the number of students to access the labs at a particular time might be limited. To resolve one of the issues faced by students for access to labs, we proposed a solution to implement Remote Interfacing with 8051. The project aims to allow the user to flash and test their binaries on the flash memory of 8051. This is achieved by connecting the 8051 to a target microcontroller, MSP432P401R in this case and connect the user to the target controller via a Wi-Fi module, ESP32. The project also aims to design a PCB for the target system consisting of the MSP432 controller, ESP32 module, LCD module and the 8051 microcontroller. The project extends in providing the user with a utility to connect to network and to allow selection of hex files.

2 TECHNICAL DESCRIPTION

2.1 Hardware Architecture

The proposed design for Remote communication with 8051 involves following hardware components as shown in Figure 1.

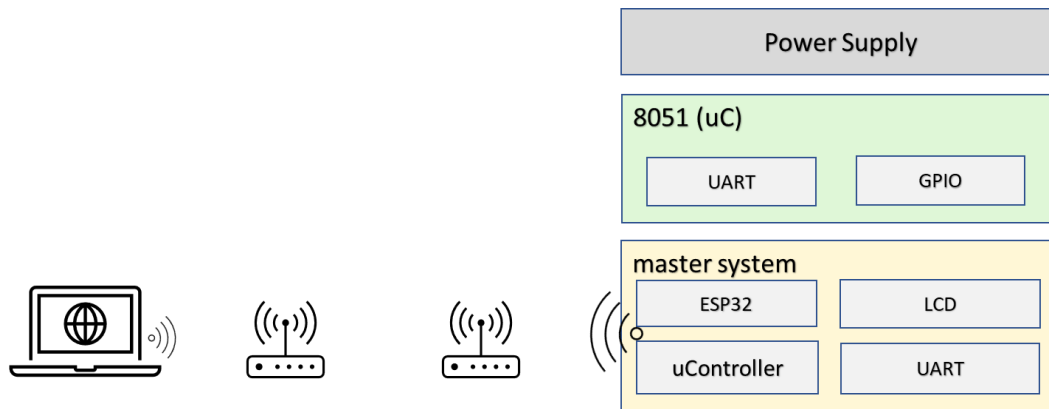


Figure 1: Hardware Architecture

The system involves,

- A power supply to meet the operating voltage and current consumption requirements cumulative of all the components
- A target controller (AT89C51RC2) where binary would be flashed
- A controlling unit which manages data transfers to and from the host PC and the target controller. This unit also capable of setting up control signals for managing configuration of target controller, setting up configuration for wireless communication and displaying current program status of the system to the user.
- An application onto the host system to allow the user to communicate with the master system to transfer the hex file.

2.1.1 Power Supply Design

Following table contains maximum power considerations for listed components

Component	Operating Voltages	Current Consumption
MSP432P401R	1.65V-3.7V	450mA
ESP32	2.3V-3.6V	1200mA
LCD Module	3.3V	140mA

Table 1: Power considerations for master system

To meet the above operating voltages the current requirements, a power supply using LM2576 with a rating of 5V, 3A was designed. The LM2576 is a step-down switching regulator capable of driving 3A load with excellent line and load regulation. The LM2576 requires minimum external components there by reducing the amount of space on the PCB and reducing cost in the BOM.

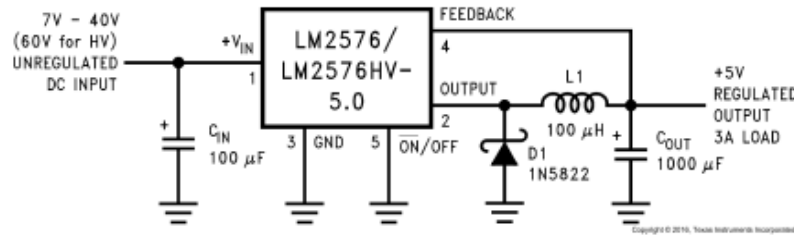


Figure 2: Power supply circuit

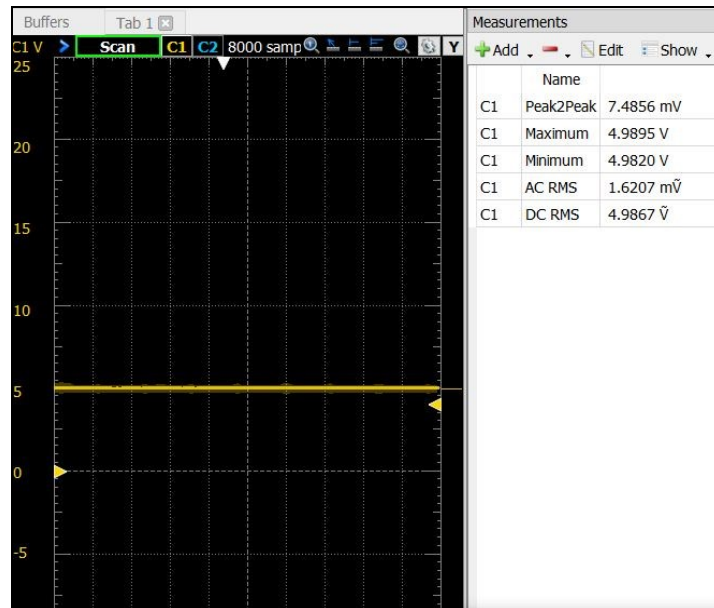
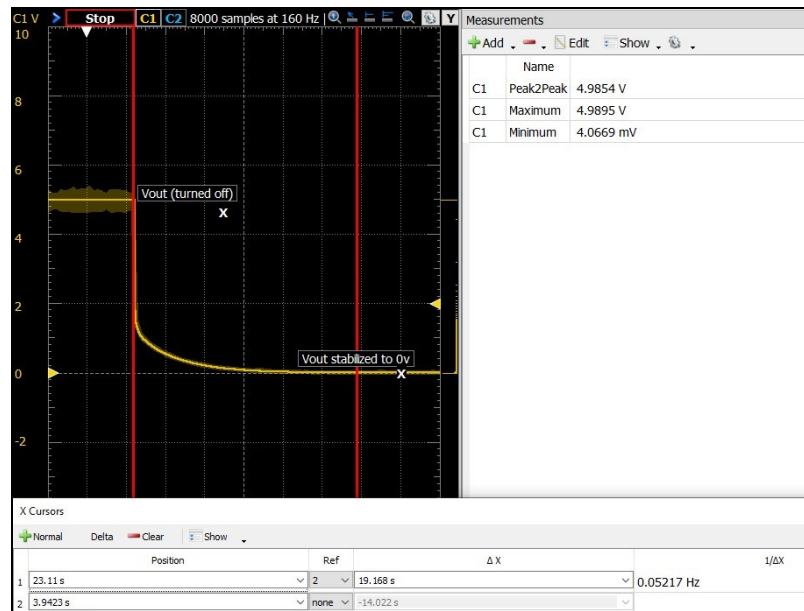


Figure 3 Vout Analysis Peak Voltage

- AC components in Vout: 1.6 mV
- DC component in Vout: 4.9867 V

**Figure 4 Vout Analysis**

- The power supply circuit provides a peak-to-peak voltage of 4.98 V
- Due to output capacitor C_{out} (1000uF), output voltage requires approximately 19 seconds to stable down to 0V. The output capacitor prevents switching of output voltage.

2.2 PCB Design

2.2.1 PCB layout

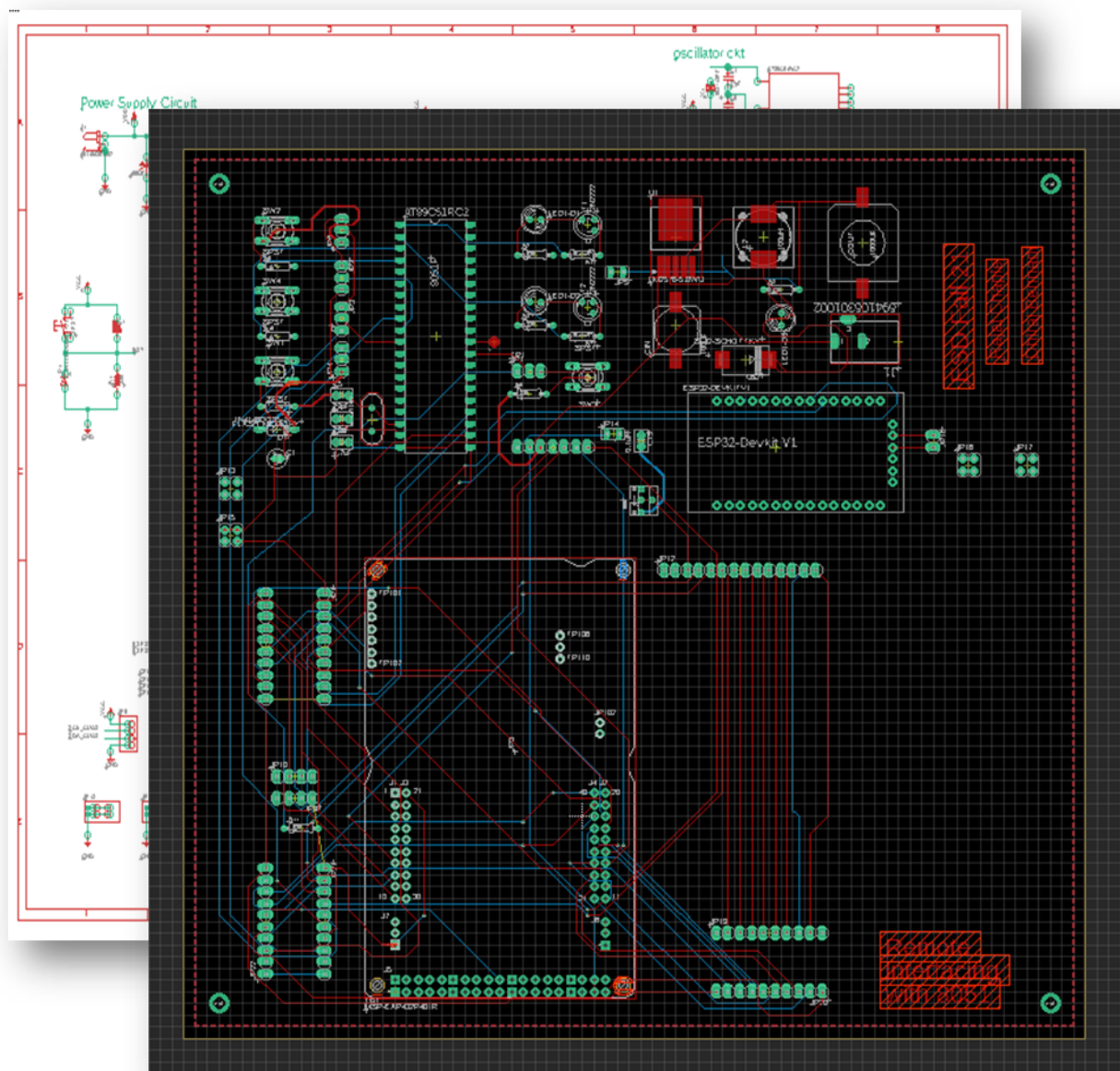


Figure 5 PCB Layout

PCB Properties

Property	Value
Layers	2
Dimensions	170.18 x 195.58 mm
PCB Thickness	1.6
Type	SMD + Through-hole

Due to non-delivery of the PCB board, the same design was implemented on a zero-PCB.

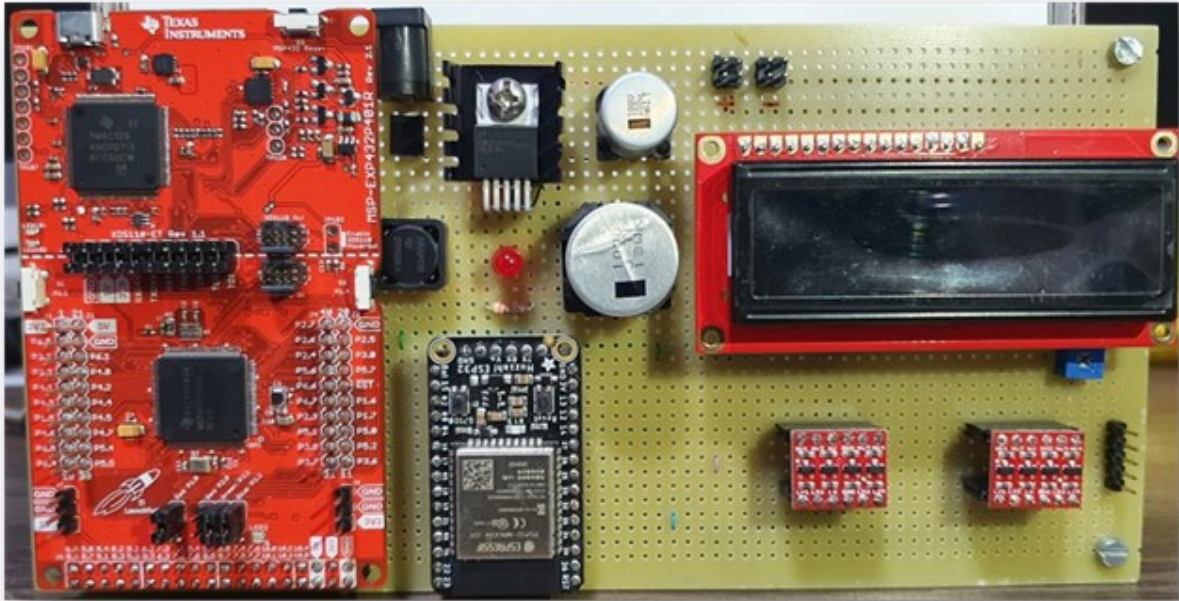


Figure 6 Board Prototype Top

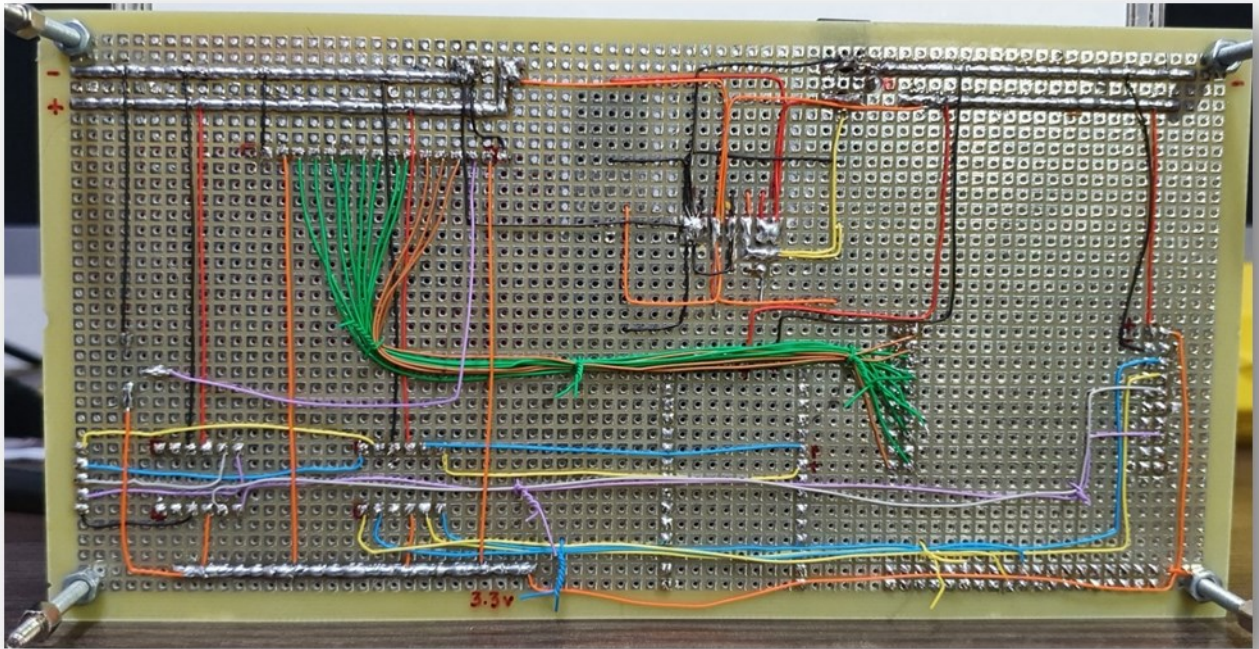


Figure 7 Board Prototype Bottom

2.3 Hardware Interfacing

2.3.1 Interfacing ESP32 (Wi-Fi module) with MSP432

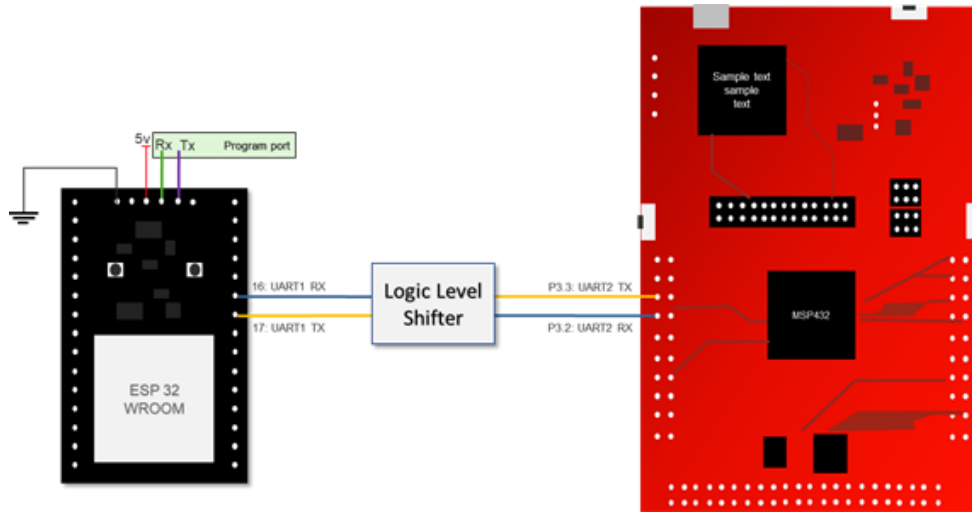


Figure 8 Interfacing ESP32-MSP432

The Wi-Fi module ESP32 is interfaced with MSP432 via serial interface. The ESP32 module operates at 5V logic level, hence a logic level shifter that converts 3.3V output from the MSP432 pins into 5V is used to connect the Tx and Rx pins. UART2 of the MSP432 is used for the above interfacing. UART0 of ESP32 (Program port) is used to program the firmware of AT commands. UART1 of the ESP32 is used as the communication port.

2.3.2 Interfacing 8051 with MSP432

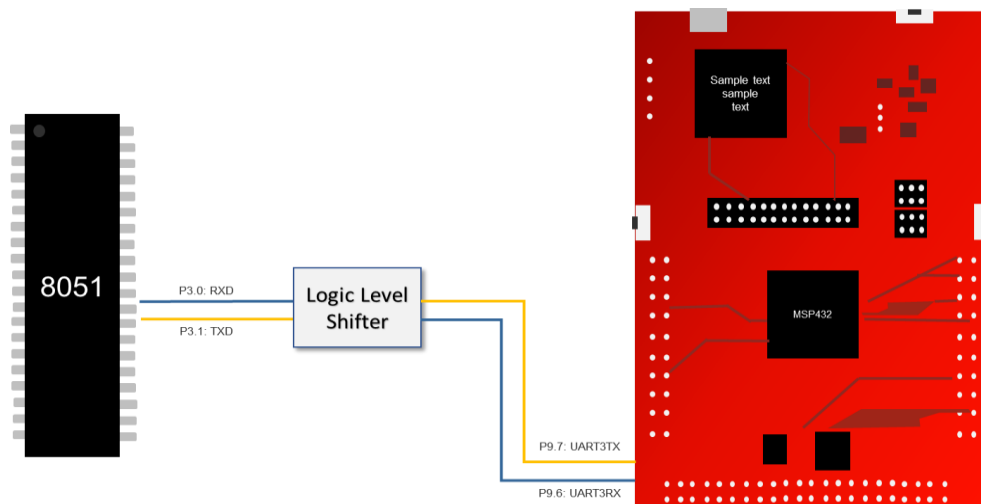


Figure 9 Interfacing 8051-MSP432

The AT89C51RC2 is interfaced with MSP432 via UART protocol. The AT89C51RC2 module operates at 5V, hence a logic level shifter that converts 3.3V output from the MSP432 pins into 5V is used connect the Rx and Tx pins. UART3 of the MSP432 is used in the following configuration.

2.3.3 Interfacing LCD with MSP432

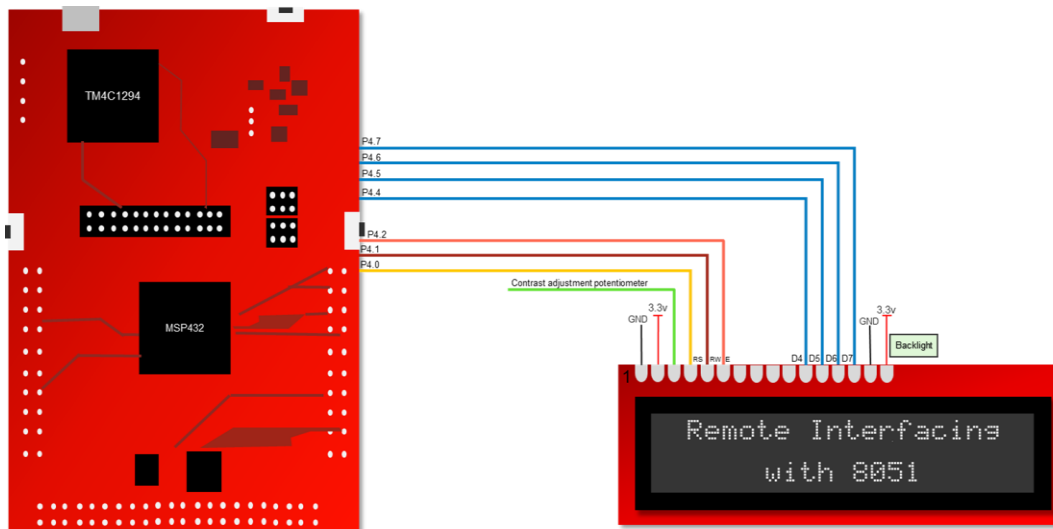


Figure 10 Interfacing LCD-MSP432

The LCD module is interfaced with MSP432 via GPIO. The LCD module can operate at 3.3V, hence a direct connection of the GPIO pins to the data and control lines of the LCD module is implemented. LCD is made to operate in 4-bit mode. This reduced the pin requirements

2.4 Software Architecture

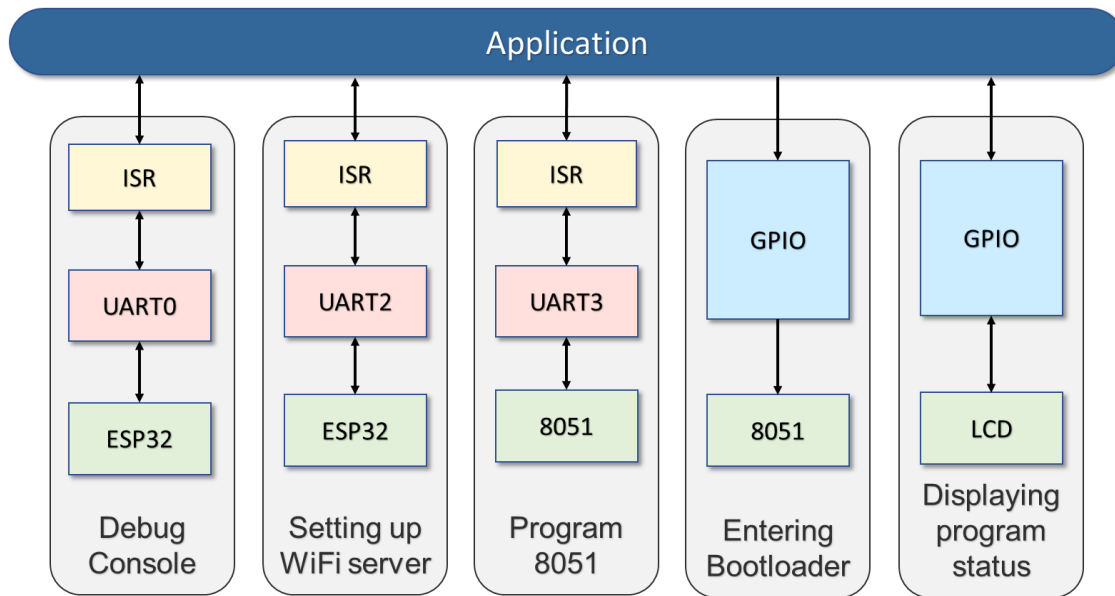


Figure 11 Software Architecture

For the development of software, the application code was written in C to be flashed on MSP432P401R. The following modules were used,

- UART0 to display the debug information of MSP432,
- UART2 to communicate with ESP32
- UART3 to communicate with 8051.
- GPIO Pins were used to enter the bootloader state of the 8051 and show status information of the program on the LCD Module.

The Software Design was built on the principles of non-blocking approach with ISR used for the UART. A command processor was implemented to handle the input stream and output stream of ESP32. The command processor has a look table capable of addition and deletion of AT commands as per the user requirements. The command processor also handled the input of hex files and filtering of the input for additional characters inserted to maintain data control flow.

2.4.1 ESP32

2.4.1.1 Flashing the ESP32

The ESP32 module needs to be flashed with AT firmware for using the AT Commands.

AT Commands are a set of commands to communicate with the ESP32. The commands are categorized into 4 types:

1. Test Command -> Query the Set Commands' internal parameters and their range of values.
2. Query Command -> Return the current value of parameters.
3. Set Command -> Set the value of user-defined parameters in commands and run these commands.
4. Execute Command -> Run commands with no user-defined parameters.

We used the following firmware version to flash the ESP32: ESP32-WROOM-32_AT_Bin_V2.1

Flashing Steps using tool *Flash Download Tool from Espressif*:

1. Open the ESP Flash Download Tool
2. Select the Mode, we used the Developer Mode
3. Select the device, we used ESP32 as our device
4. Select the default binary file factort_WROOM-32.bin and check the option DoNotChgBin and click on start

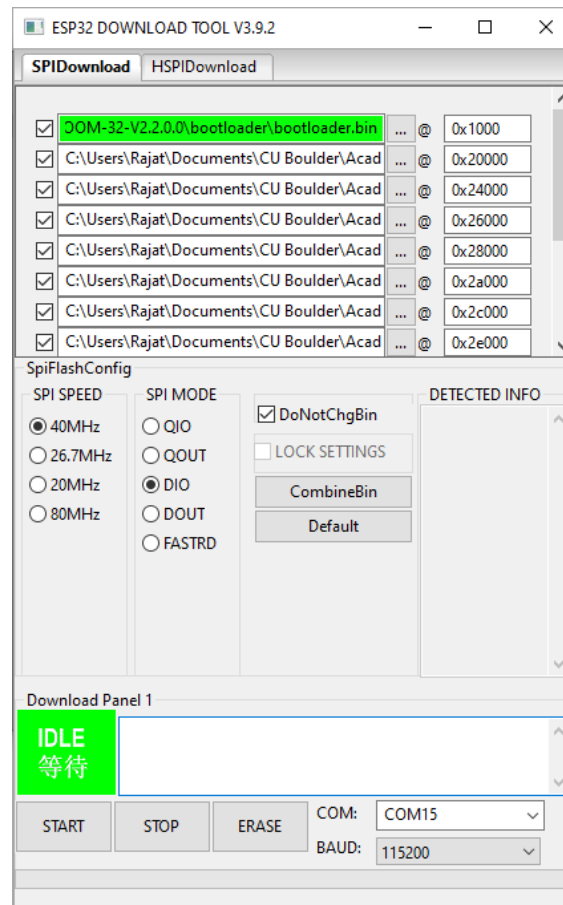


Figure 12 ESP Flash Download Tool

2.4.1.2 Setting up Wi-Fi connection

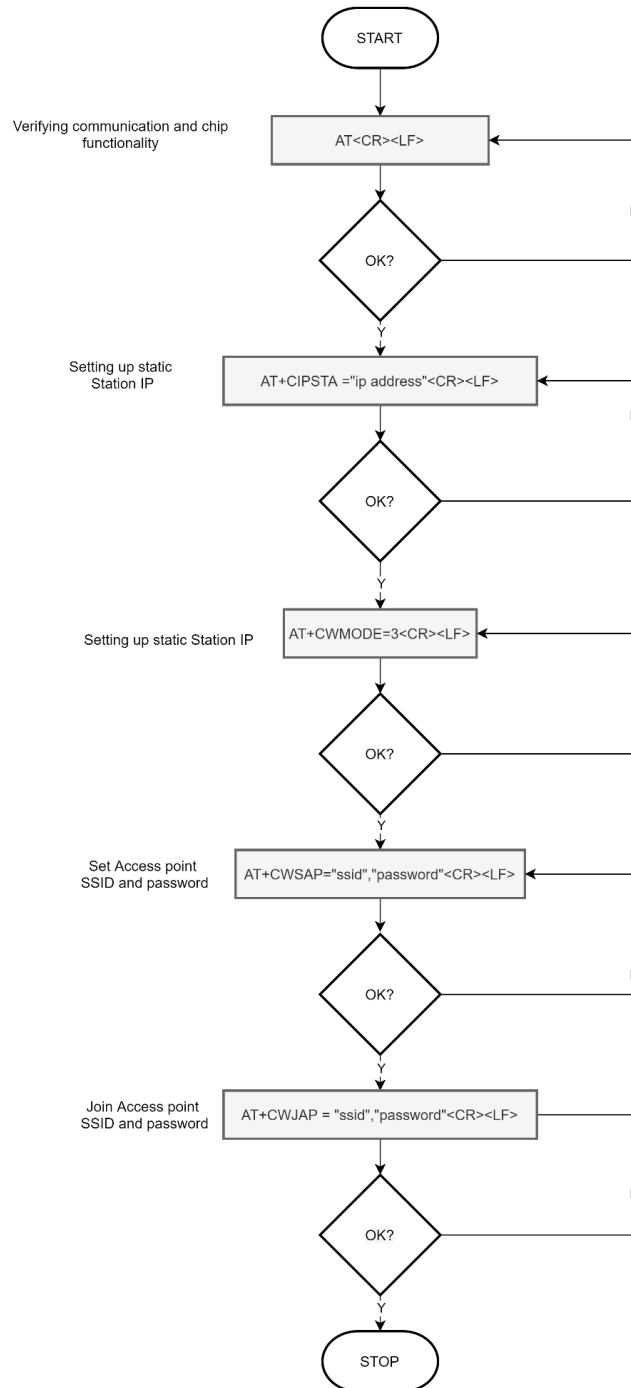


Figure 13 Flowchart - ESP32 Wi-Fi Setup

The ESP32 Wi-Fi module is connected to a local network to enable the sending of data remotely using AT Commands. The ESP32 can be either set to Station mode or Access Point mode. Application demands it to operate in Station mode. In this mode, ESP32 connects to a Wi-Fi router with specified SSID and Password.

2.4.1.3 Setting up Server

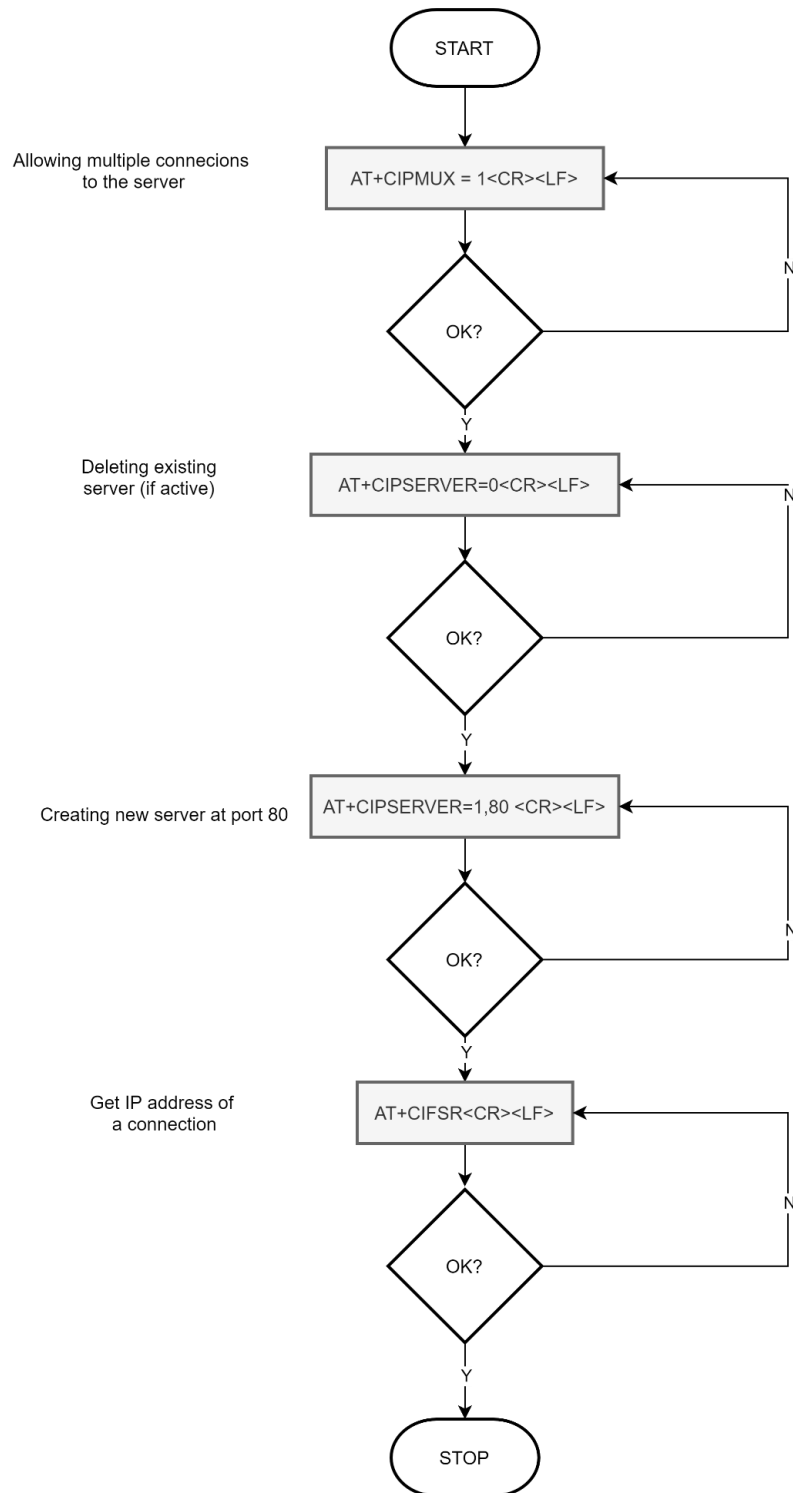


Figure 14 Flowchart - ESP32 Server Setup

In order to data transfer the ESP module has to function as a server/client. Once the host system and the ESP32 are connected to a server, the ESP module acts as a server to the client(host utility) to allow data transmission. Above configuration allows multiple clients to be connected to this server. Though, the application does not demand multiple clients, this configuration was made keeping future scope of the project.

2.4.2 Programming 8051

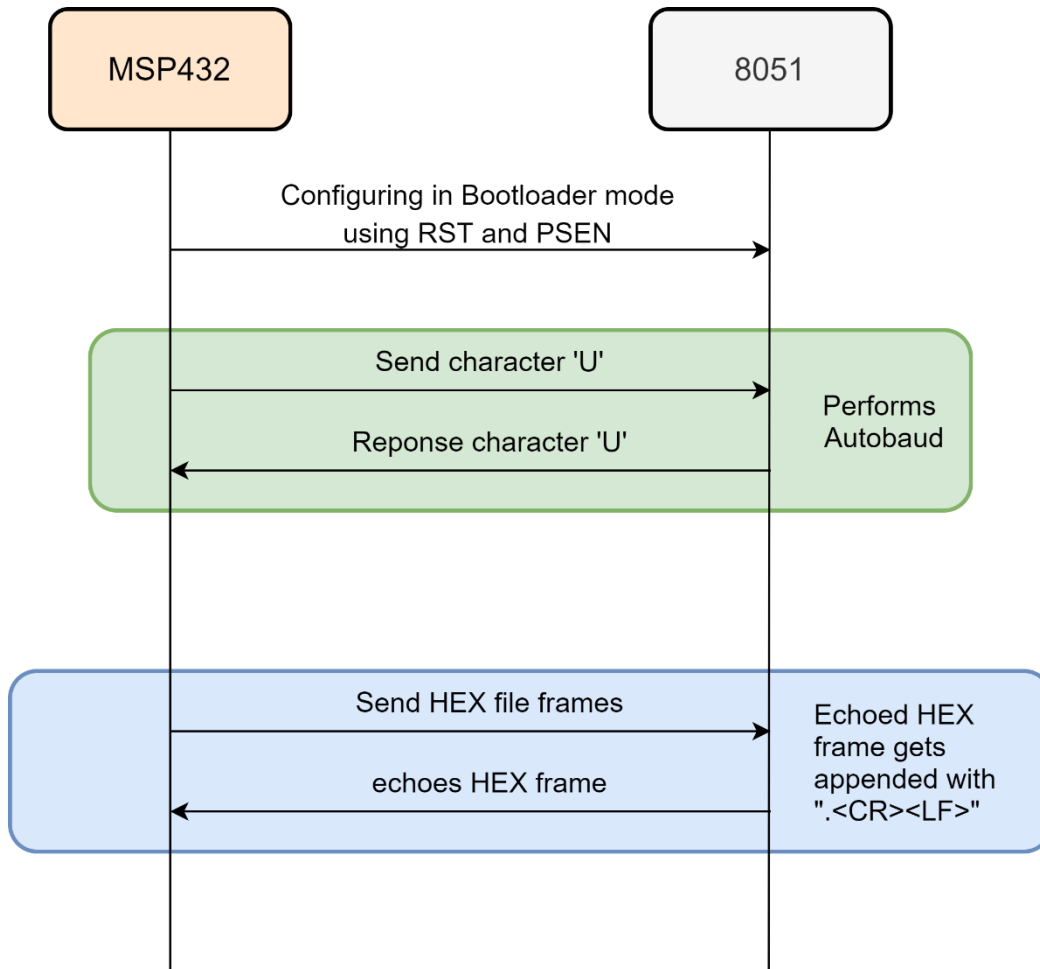


Figure 15 Programming 8051

The AT89C51RD2 Bootloader facilitates In-System Programming and In-Application Programming. In-System Programming allows the user to program or reprogram a microcontroller on-chip Flash memory without removing it from the system and without the need of a pre-programmed application. The UART bootloader can manage a communication with a host through the serial network. It can also access and perform requested operations on the on-chip Flash Memory.

In-Application Programming (IAP) allows the reprogramming of a microcontroller on-chip Flash memory without removing it from the system and while the embedded application is running. The UART bootloader contains some Application Programming Interface routines named API routines allowing IAP by using the user's firmware.

For our application we have used In-System Programming to program the flash memory by configuring the microcontroller in bootloader mode.

The bootloader can be activated in 2 ways:

1. Hardware Conditions
2. Regular boot process

We have used the hardware condition to activate the bootloader. The hardware conditions PSEN low during a reset cycle forces the 8051 in bootloader mode.

UART is used to communicate with the 80C51 bootloader in the following configuration.

8-bit data, Parity: None, stop bits: 1, Flow Control: None, Baud Rate: Autobaud is performed by the bootloader to compute the baud rate chosen by the host.

Initialization Sequence:

An initialization step must be performed after each Reset. After microcontroller reset, the bootloader waits for an autobaud sequence. The host initiates the communication by sending a 'U' character to help the bootloader to compute the baud rate (autobaud).

Command Data Stream Protocol

Each frame sent by the host is echoed by the bootloader.

The bootloader echoes with: <.> <CR><LF> appended to the transmitted frame when programmed.

2.4.3 GUI Application for sending Hex file

Developed GUI connects with ESP32 over Wi-Fi and establishes TCP IP socket communication.

Utility allows user to select HEX file to be transmitted. Clicking on Program button initiates transfer of Hex file to remote device.

Remote device then flashes this software to 8051 once transfer is completed.

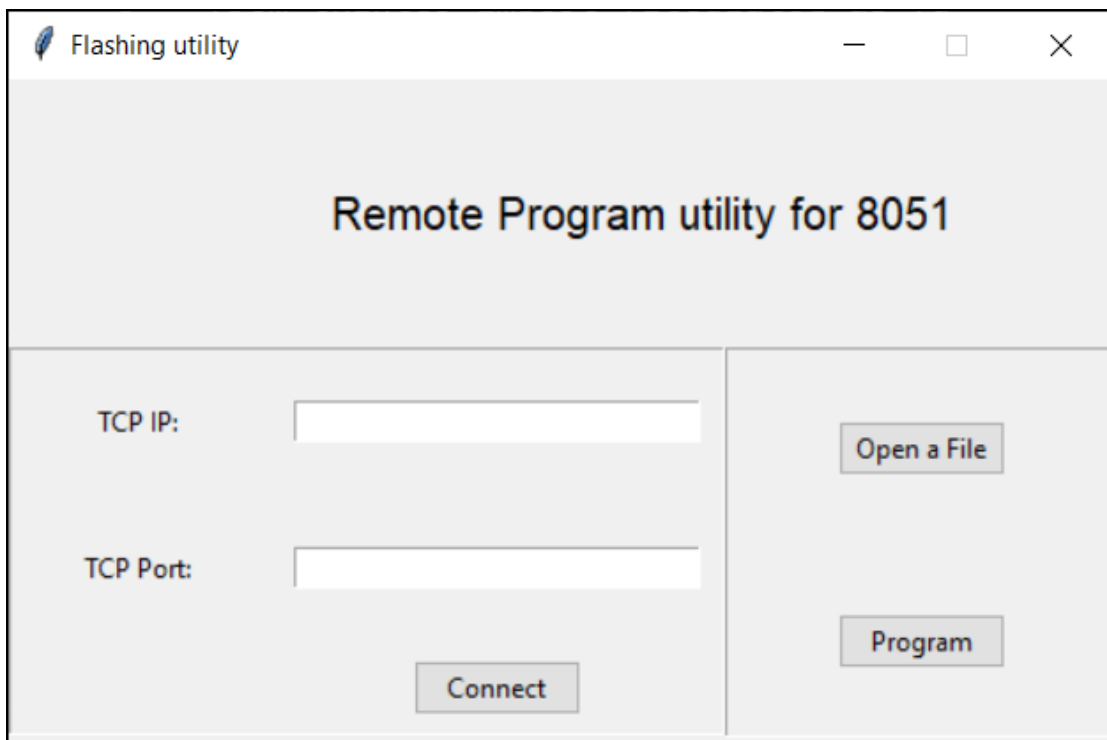
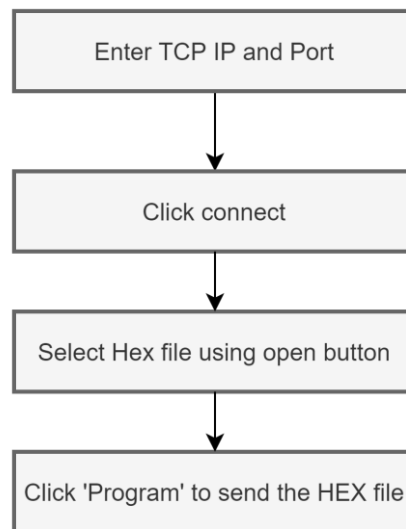


Figure 16 GUI Application

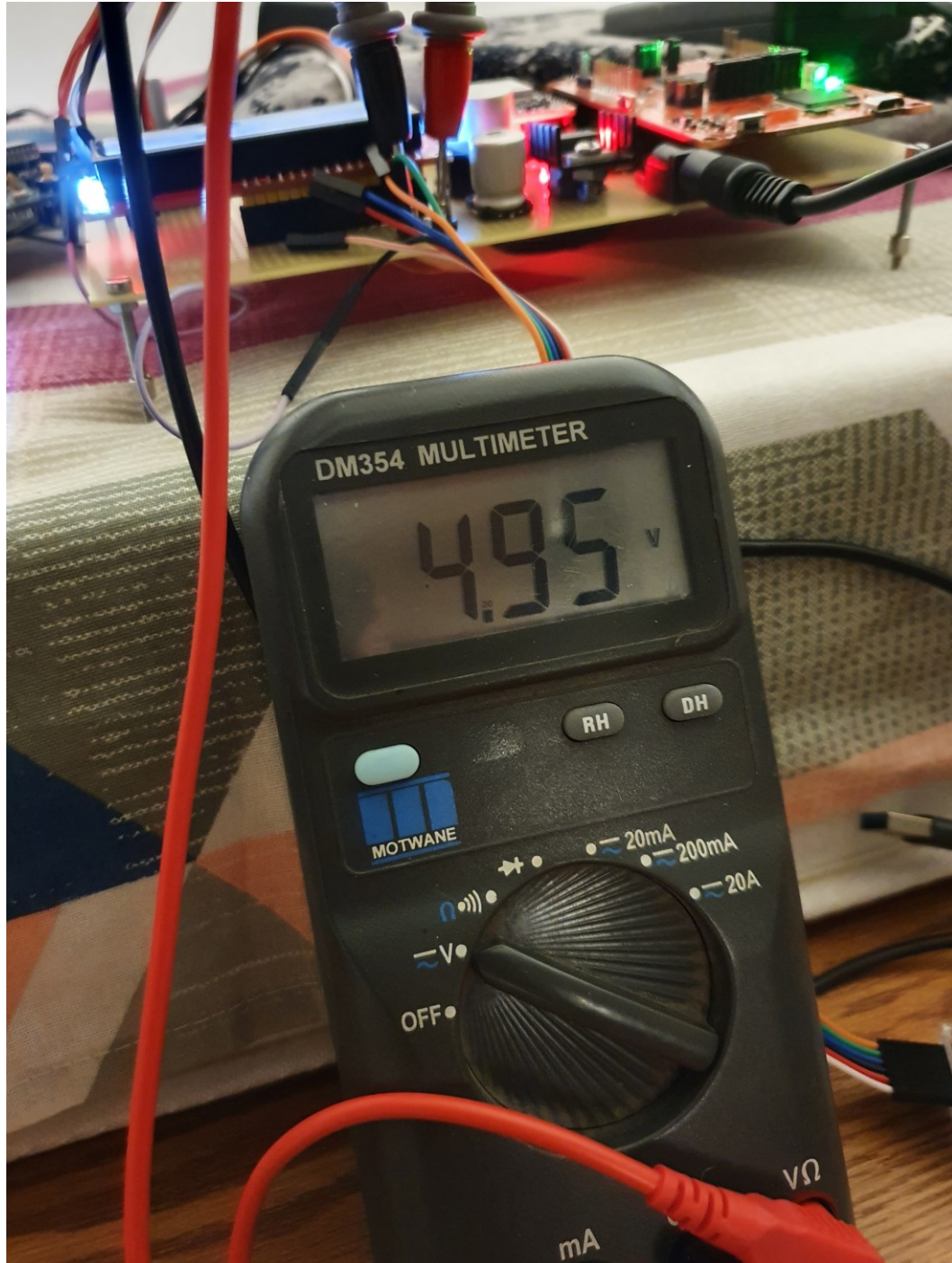
Follow following steps in order to program 8051 remotely using GUI application.



2.5 Testing Process

The system was tested through a series of manual tests verifying the functionality of individual component.

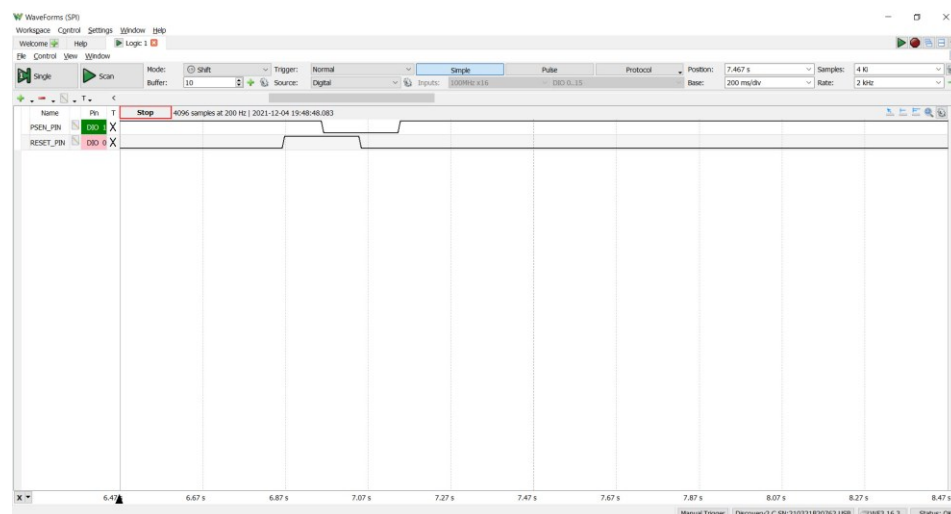
1. The output of the power supply was tested on the Digital Multimeter and Oscilloscope.



2. The board was tested for proper voltages at test point.
3. The ESP32 module was tested for basic AT commands after flashing the AT firmware.



4. The ESP32 module was tested for connecting to Wi-Fi by connecting it to a mobile hotspot.
5. Entering bootloader mode sequence was verified using Logic analyzer



6. The bootloader mode of the 8051 was tested by sending a character 'U' and receiving back the same character on the UART terminal.

```
U:02000004100BD.
:20020000758901758AF9758CF375881075A882C3029379077A019020007B007C80004007B6.
:13022000BA0004790E7A0100BB7F037BFFD30BEBF080EA31.
:20000B00D292BCFF027C7F0CECF019B90013B2917A007907C3758800758AF9758CF3758806.
:07002B001000A293C2923203.
:00000001F
```

7. The flashed binary was visually tested for blinking of the LED connected to a port of the 8051 microcontroller.





8.

3 RESULTS AND ERROR ANALYSIS

The project was successfully implemented as proposed. We were able to develop a utility which would transfer the HEX file from host PC to master system over a network. The application software on the master device was successful in establishing a connection with the host PC via ESP32 for data transfer.

The software was successful in configuring, programming, and rebooting 8051 in-order to perform In-System Programming. However, we faced difficulty while getting our PCB fabricated from vendor which affected our timelines. As a result, added functionality of Memory backup was not implemented.

The following table shows the stages of execution of the Remote Interfacing with 8051.

Images displaying program stages	Description
	Program started
	Device trying to connect to Wi-Fi
	Device has connected to a Wi-Fi
	Device has set up a server and waiting for client







	Client connected (utility I our case)
	Device receiving HEX file over WiFi
	Once Hex file has received, device has configure 8051 into bootloader
	Device transferring hex file to 8051 memory
	Once Hex file is transferred, Reset 8051 to exit bootloader
	Firmware successfully flashed

Table 2 Program stages

4 CONCLUSION

The project gave us exposure to technologies such as PCB Design, Interrupt based UART Programming, Command Processor, Hardware and Software Architecture Design, Socket Programming, Wi-Fi, Developing GUI. This further inspires us to extend the functionalities of the project by implementing memory interfacing and allowing flashing of programs stored in the memory.

5 WORK DISTRIBUTION

Task	Assigned to
Power Circuit design	Rajat Chaple
Ordering of parts	Dhiraj Bennadi
Schematic	Rajat Chaple
PCB Layout	Rajat Chaple
Interfacing UART (ESP32 to 8051)	Dhiraj Bennadi
Interfacing SPI (ESP32 to Memory card)	Dhiraj Bennadi
Socket communication	Dhiraj Bennadi
GUI Utility	Rajat Chaple
Testing	Both
Report	Both

Table 3 Work Distribution

6 FUTURE DEVELOPMENT IDEAS

Few of the learnings with the experience gained while working on this project would help us to provide suggestions to be implemented in the future. Designing and fabrication of PCB should be done very early in the development phase. This allows the users to rectify any mistakes in the design which would save a lot of debugging time later. The timelines should be managed such that multiple revisions of the PCB be incorporated in the project.

Additionally the ability to connect wirelessly to any network and implement process of flashing and testing the firmware would be encouraged.

7 ACKNOWLEDGEMENTS

We would like to thank the course instructor, Dr. Linden McClure for encouraging us to take up the project. The professor also contributed to the project with his valuable suggestions.

We would also like to extend our gratitude to the course teaching assistants Mr. Sundar Krishnakumar, Mr. Alex Fritz and Mr. Venkat Tata for assisting us throughout the development of the project.

8 REFERENCES

1. Power supply design: <https://www.ti.com/lit/gpn/LM2576>
2. ESP32 AT Firmware flashing: https://docs.espressif.com/projects/esp-at/en/latest/Get_Started/Hardware_connection.html
3. ESP32 AT command set: https://docs.espressif.com/projects/esp-at/en/latest/AT_Command_Set/index.html
4. MSP432 Reference manual: https://schaumont.dyn.wpi.edu/ece4703b21/_downloads/8bf98313124444641502c686bb90dbaa/msp432p401r-trm.pdf
5. MSP432 interfacing with 16x2 LCD: <https://microdigisoft.com/ti-launchpad-interfacing-lcd-16x2-with-msp432-microcontroller/>
6. AT89C51RC2 bootloader: <http://ww1.microchip.com/downloads/en/devicedoc/doc4180.pdf>

8.1 Appendix - Bill of Materials



BOM_esd_fall_final_
project.xls

Major components are listed below (for detailed BOM, kindly refer embedded excel sheet)

Part Description	Source	Cost
AT89C51RD2	Digi-Key www.digikey.com	\$20
ESP32 WROOM	Adafruit www.adafruit.com	\$11.95
MSP432	TI www.ti.com	\$19.90
Level shifters	Sparkfun www.sparkfun.com	\$4.50
TOTAL		\$56.35

Note: All the below sections are added in separate file

8.2 Appendix - Schematics

8.3 Appendix - Firmware Source Code

8.4 Appendix - Software Source Code

8.5 Appendix - Data Sheets and Application Notes