|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **School of Electronics Engineering (SENSE)** | | | | |
| **PROJECT REPORT** | | | | |
| **COURSE CODE / TITLE** | BECE204L– Microprocessors and Microcontrollers | | | |
| **PROGRAM / YEAR/ SEM** | B.Tech (BEC)/II Year/ WINTER 2023-2024 | | | |
| **DATE OF SUBMISSION** |  | | | |
| **TEAM MEMBERS**  **DETAILS** | **REGISTER NO.** | | **NAME** | |
| **22BEC1222** | | **RALF PAUL VICTOR** | |
| **22BEC1247** | | **VINAY** | |
| **22BEC1258** | | **MRINANK GAUR** | |
| **22BEC1269** | | **KHUSHI JAIN** | |
| **22BEC1187** | | **ABHISHA CHAKRBARTI** | |
| **22BEC1481** | | **ABHISHEK** | |
| **22BEC1282** | | **MANINDRA GOPIREDDY** | |
| **PROJECT TITLE** | **8051 GPS Tracker** | | | |
| **COURSE HANDLING FACULTY** | **Dr. S. REVATHI**  **Associate Professor,**  **SENSE** | **REMARKS** | |  |
| **FACULTY SIGNATURE** |  |

**OBJECTIVE:**

The objective of this project is to design and implement a GPS tracking system using the 8051 micro-controller. The system will utilize a GPS module to accurately determine the real-time location coordinates, which will be processed by the 8051 micro-controller and displayed it on an LCD interfaced to it. The project aims to develop a reliable and efficient tracking solution that can provide precise location information.

**INTRODUCTION:**

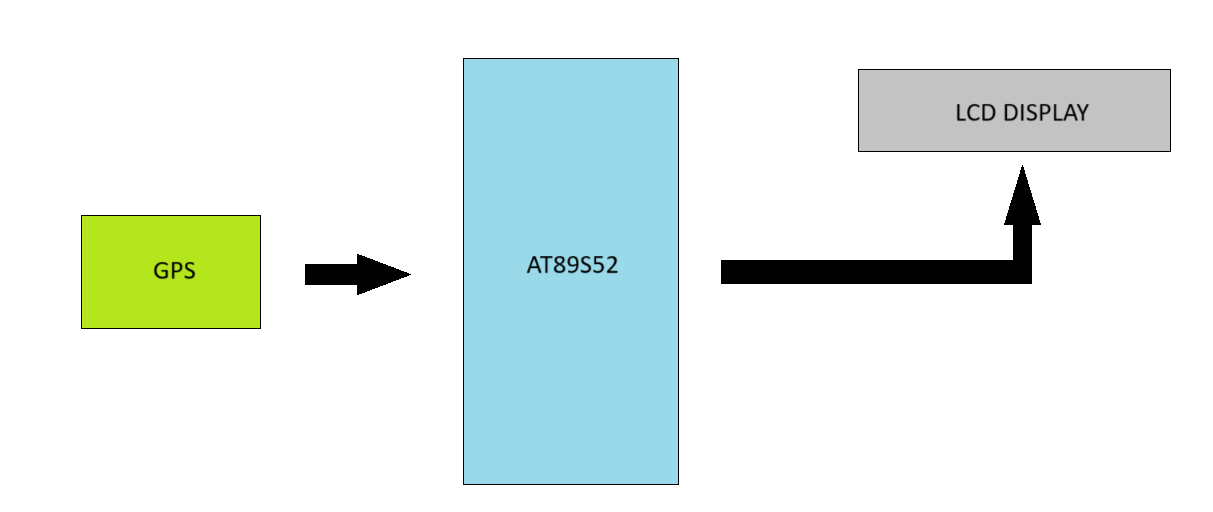
In today's interconnected world, real-time tracking systems have become indispensable across various domains, including navigation, transportation, and logistics. Leveraging the capabilities of microcontrollers and GPS technology, this project aims to develop a GPS tracking system using the 8051 microcontroller. The system is designed to accurately determine and display real-time location coordinates, providing valuable insights for applications such as vehicle tracking, asset monitoring, and personal navigation.

The project combines hardware interfacing with software programming to establish communication between the GPS module and the 8051 micro-controller. Additionally, an LCD display is utilized to present the acquired location data in a user-friendly format. The assembly code provided orchestrates the operation of the microcontroller, handling serial communication, data processing, and LCD interfacing.

This project not only demonstrates the practical application of embedded systems but also showcases the efficiency and versatility of the 8051 micro-controller in handling complex tasks with precision. Through meticulous design and implementation, the GPS tracking system presented herein offers a robust solution for tracking and monitoring applications, promising accuracy, reliability, and ease of use.

What sets this project apart is its focus on simplicity, efficiency, and flexibility. By utilizing the 8051 micro-controller, we can create a compact and cost-effective solution that meets the needs of diverse tracking requirements. Additionally, the integration of an LCD display enhances user interaction, providing real-time feedback in a clear and intuitive manner. Ultimately, our work aims to offer a practical and accessible GPS tracking solution that addresses the shortcomings of existing systems while paving the way for future innovations in the field.

**BLOCK DIAGRAM:**



**COMPONENTS/ SOFTWARE REQUIRED:**

1. 8051 DEVELOPMENT KIT
2. GPS NEO-6M
3. AT89S52,
4. 16x2 LCD DISPLAY
5. WIRES
6. LITHIUM ION BATTERY

**PROJECT DESCRIPTION:**

**GPS Module Operation:**

1. **Satellite Communication:** The GPS module continuously receives signals from a network of 24 to 32 orbiting GPS satellites. These signals contain three crucial pieces of information:
   * **Satellite ID:** Identifies the specific satellite transmitting the signal.
   * **Ephemeris Data:** Provides details about the satellite's current position and orbital parameters.
   * **Time Stamp:** Indicates the exact time the signal was sent by the satellite.
2. **Distance Calculation:** The GPS module uses the time difference between the signal's transmission and reception (based on the speed of light) to calculate the distance between itself and each satellite it receives a signal from.
3. **Triangulation:** To determine its precise location (latitude, longitude, and altitude), the GPS module needs data from at least four satellites. With the calculated distances to these satellites and their known positions, the module performs a mathematical process called trilateration (a form of triangulation in 3D space) to pinpoint its location on Earth.

**Interfacing with 8051 Microcontroller:**

1. **Hardware Connection:**
   * **Power Supply:** The GPS module typically requires a separate power supply (often 3.3V or 5V) depending on the specific model.
   * **Serial Communication:** A serial communication protocol like UART (Universal Asynchronous Receiver Transmitter) is used for data exchange between the GPS module and the 8051. The module's TXD (Transmit Data) pin connects to the microcontroller's RXD (Receive Data) pin, and vice versa. Ground (GND) connections are also made.

**Software Programming:**

* **UART Initialization:** The 8051's UART module is configured in the microcontroller's program to establish baud rate, parity, and stop bits that match the communication settings of the GPS module (9600 baud, 8N1 format).
* **Data Reception:** The program continuously reads the serial data stream received from the GPS module.
* **Data Parsing:** The received data typically consists of NMEA (National Marine Electronics Association) sentences, which are formatted strings containing various GPS information. The program parses these sentences to extract specific data fields of interest, such as:
  + **$GNGGA:** Provides latitude, longitude, altitude, time, and fix quality.

**Data Processing and Utilization:** Once extracted, the microcontroller can use the GPS data for various purposes, such as:

* Displaying latitude and longitude coordinates on an LCD

**SIMULATION:**

While Proteus doesn't offer a direct way to simulate real-time GPS functionality due to the lack of actual satellite communication, you can create a setup to test the interaction between the 8051 microcontroller and a simulated GPS module. Here's a detailed explanation:

**Components:**

1. **8051 Microcontroller:** Choose the specific 8051 variant you're working with (e.g., 8051, AT89S52) from the Proteus library.
2. **VGPS:** This will act as a substitute for the real GPS module.
3. **LCD:** You can add an LCD to display the extracted GPS data for visual verification.
4. **Serial Terminal:** This will help to observe the data that is being sent to the microcontroller.

**Schematic Design:**

1. Connect the 8051's RXD pin to the TXD pin of the Serial Terminal and VGPS.
2. Provide necessary power and ground connections for all components based on their specifications.
3. Connect the LCD to the 8051's data and control pins according to the LCD's interface.

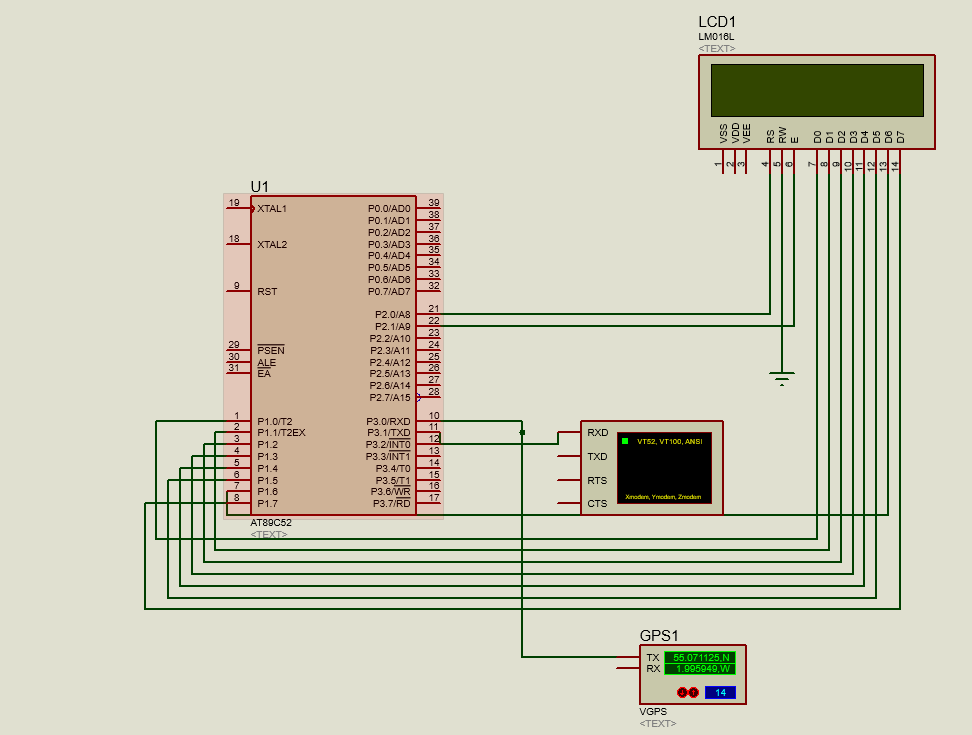


Fig. Proteus Diagram

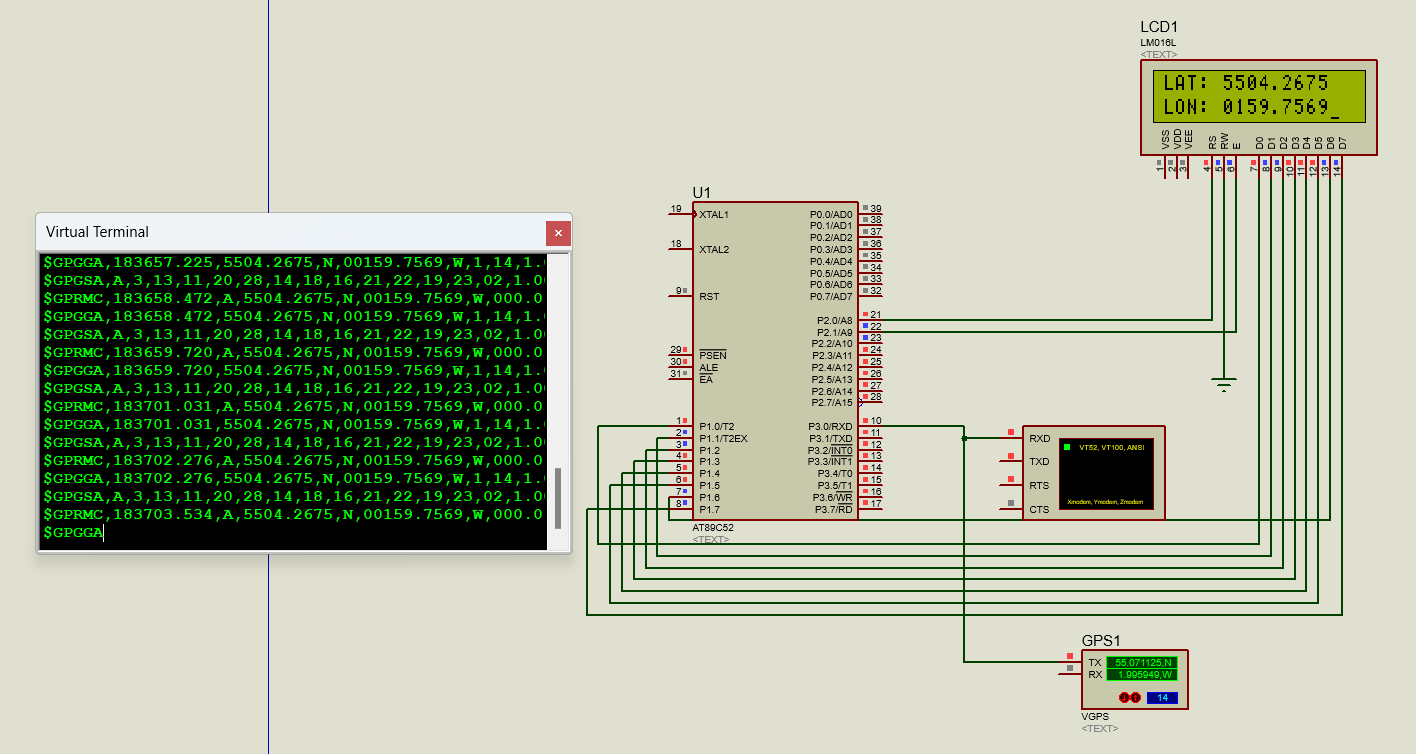


Fig. Proteus Simulation

**Simulated VGPS Data:**

1. Configure the VGPS to transmit NMEA sentences at a regular interval.
2. Here are some examples:

$GPGGA,000000.0000,00000.0000,N,000000.0000,W,1,08,0.9,454.5,M,-45.8,M,2.0,0000\*5F (Sample GPGGA sentence)

**Simulation and Testing:**

1. Simulate the Proteus design.
2. Observe the data on the serial terminal to ensure it's being transmitted correctly.
3. Verify that the microcontroller code is receiving and parsing the NMEA sentences as expected.
4. Check if the extracted GPS data (e.g., latitude, longitude) is displayed in the LCD correctly.

**CONCEPT LEARNED:**

The 8051 microcontroller is an 8-bit microcontroller introduced by Intel in 1980. It is based on Harvard architecture, which means it has separate memory spaces for program and data. The 8051 has a separate address space for storing the program in ROM and data memory in RAM. The program memory is typically 4 KB in size, while the data memory can range from 128 bytes to several kilobytes. It typically consists of a CPU, RAM, ROM, I/O ports, timers/counters, serial communication ports, and interrupt control circuits.

Interfacing 8051 with GPS and LCD: Connect the GPS module to the 8051-microcontroller using serial communication (UART) and connect the LCD to the 8051 using parallel communication. Initialize UART communication with the GPS module, allowing the microcontroller to receive GPS data such as latitude, longitude, and time. Then write the code to process GPS data and extract information. Initialise the LCD module and write the code to display on LCD screen. Integrate the hardware for GPS communication and LCD display and write the code to format GPS data and display it on LCD screen in readable format. Finally the GPS tracker will display the longitude and latitude on LCD screen.

**IMPLEMENTATION:**

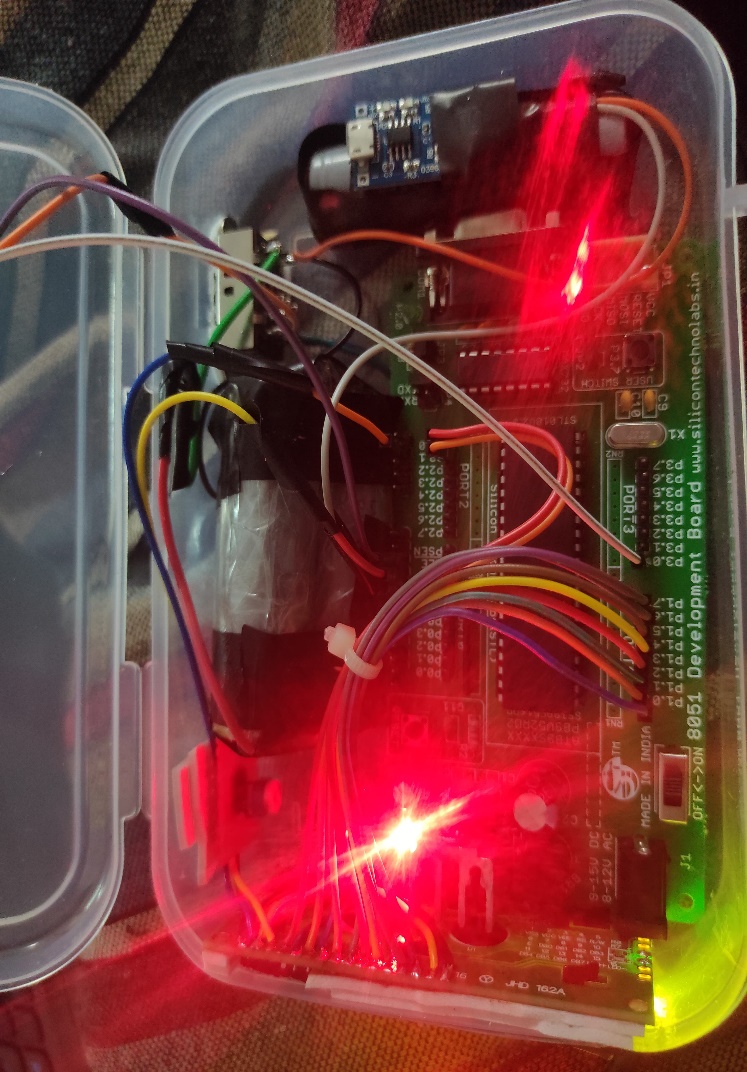


Fig. Hardware switched on by giving power supply



Fig. Connection of 8051 with GPS and LCD in process



Fig. Connection of 8051 with GPS and LCD confirmed



Fig. Output

YouTube Link: <https://www.youtube.com/watch?v=erZ1EfOQrkE>

**CHALLENGES FACED:**

**1. Voltage Level Matching:** Connecting a real GPS module require a USB-to-serial converter to translate communication protocols and ensure voltage levels between the module and the Proteus environment match to avoid damage.

**2. Microcontroller vs. GPS Module:** The 8051 microcontroller and the GPS module operates on different voltage levels (commonly 3.3V or 5V). Connecting them directly could damage one or both components.

**3. Communication Protocol:** Standard GPS modules typically use a serial communication protocol like UART (Universal Asynchronous Receiver Transmitter) to transmit NMEA sentences. However, the specific pin configuration and connection requirements might vary depending on the GPS module you choose. You'll need to consult the module's datasheet for proper wiring and ensure the microcontroller's UART pins are connected to the corresponding pins on the GPS module.

**4. Separate Power Source:** The GPS module requires a separate power supply (often 3.3V or 5V) depending on its specifications.

**5**. **Physical Integration:** Depending on the form factor of the GPS module and the project setup, we needed to consider how to physically mount and secure the module within our design.

**6.** **Antenna Placement:** For optimal GPS signal reception, the GPS module's antenna needs to be placed in a location with a clear view of the sky.

**APPLICATIONS:**

The project aims to develop a reliable and efficient tracking solution that can provide precise location information, enabling applications such as:

1. vehicle tracking
2. asset monitoring
3. personal navigation.

**CONCLUSION:**

In conclusion, this project successfully demonstrated the development and implementation of a GPS tracking system utilizing the 8051 microcontroller, with LCD interfacing for user interaction. Through the integration of a GPS module and efficient programming, the system was able to accurately track and report real-time location coordinates while providing visual feedback through an LCD display.

The project achieved its objectives of providing a reliable and efficient tracking solution suitable for various applications including vehicle tracking, asset monitoring, and personal navigation. Throughout the project, several key challenges were addressed, including power optimization, accuracy enhancement, and user interface design.

Future enhancements could include the integration of additional sensors for enhanced functionality, such as temperature or motion sensors, as well as the incorporation of wireless communication modules for real-time data transmission. Overall, this project serves as a foundation for further research and development in the field of GPS-based tracking systems, showcasing the capabilities of the 8051 microcontroller in embedded applications with LCD interfacing for user interaction.

**APPENDIX:**

|  |  |
| --- | --- |
| **CODE LINES** | **COMMENTS** |
| ORG 0000H  SJMP 0030H | Skipping to Main Program |
| ORG 0500H  MYCOM:DB 38H,01H,0EH,06H,80H,0  FIRST:DB"PROJECT DEMO",0  SECOND:DB"CONNECTING TO",0  THIRD:DB"GPS",0  FOURTH:DB"CONNECTION",0  FIFTH:DB"SUCCESFULL",0  LAT:DB"LAT: ",0  LON:DB"LON: ",0 | All the Required Strings are saved Here in 0500H of Code Memory. These are accessed in the code using DPTR. |
| COMSUB: CLR P2.0  CLR P2.2  SETB P2.1  MOV P1,A  CLR P2.1  ACALL DELAY  RET | The Command Subroutine for LCD |
| DATASUB: SETB P2.0  CLR P2.2  SETB P2.1  MOV P1,A  CLR P2.1  ACALL DELAY  RET | The Data Subroutine for LCD |
| DELAY: MOV R3,#0DDH  L1:MOV R0,#0FFH  HERED:DJNZ R0,HERED  DJNZ R3,L1  RET | Delay Subroutine |
| PRINT: NOP  PRINTLOOP:CLR A  MOVC A,@A+DPTR  JZ EXITPRINT  ACALL DATASUB  INC DPTR  SJMP PRINTLOOP  EXITPRINT:NOP  RET | This subroutine accesses the Strings saved in 0500H of code memory and prints it in the LCD |
| ORG 0030H  MOV TMOD,#20H MOV TH1,#-03D  MOV SCON,#50H | Setting Tmod, TH1 and Scon values For Serial Communication |
| MOV DPTR,#MYCOM  COMLOOP:CLR A  MOVC A,@A+DPTR  JZ EXITCOM  ACALL COMSUB  INC DPTR  SJMP COMLOOP | Initialising all the required Commands for LCD Interfacing |
| EXITCOM:    MOV A,#82H  ACALL COMSUB  MOV DPTR,#FIRST  ACALL PRINT  ACALL DELAY  MOV A,#01H  ACALL COMSUB  MOV A,#82H  ACALL COMSUB  MOV DPTR,#SECOND  ACALL PRINT  MOV A,#0C7H  ACALL COMSUB  MOV DPTR,#THIRD  ACALL PRINT | The 1st Stages of The project is printed in the LCD using these Lines of Code. |
| MAINLOOP:  AGAIN:SETB TR1  CLR RI  HERE:JNB RI,HERE  MOV A,SBUF  CLR TR1  CJNE A,#'$',AGAIN | Serial Communiction to recive the required "$" value |
| MOV R0,#50H  MOV R5,#50D  AGAIN1:SETB TR1  CLR RI  HERE1:JNB RI,HERE1  MOV A,SBUF  MOV @R0,A  INC R0  CLR TR1  DJNZ R5,AGAIN1 | Saving 50 values after "$" in Memory |
| MOV R0,#50H  CJNE @R0,#'G',AGAIN  INC R0  CJNE @R0,#'N',AGAIN  INC R0  CJNE @R0,#'G',AGAIN  INC R0  CJNE @R0,#'G',AGAIN  INC R0  CJNE @R0,#'A',AGAIN  CJNE R7,#01H,NORMAL  SJMP SKIP | Checking the first 5 values that contain the Coordinates. (Checking GPGGA) |
| NORMAL:  MOV A,#01H  ACALL COMSUB  MOV A,#83H  ACALL COMSUB  MOV DPTR,#FOURTH  ACALL PRINT  MOV A,#0C3H  ACALL COMSUB  MOV DPTR,#FIFTH  ACALL PRINT | "Normal" label Only Executes in first run. This Labe; doesn’t run on the second loop of the program. |
| SKIP:  ACALL DELAY  MOV A,#01H  ACALL COMSUB | Clears the LCD |
| LATITUDE:  MOV A ,#80H  ACAll COMSUB  MOV DPTR,#LAT  LALOOP:CLR A  MOVC A,@A+DPTR  JZ LATVALUE  ACALL DATASUB  INC DPTR  SJMP LALOOP | Prints the Latitude Screen |
| LATVALUE:  MOV R1,#61H  LATNUM:CLR A  MOV A,@R1  ACALL DATASUB  INC R1  CJNE R1,#6AH,LATNUM  MOV A,#0C0H  ACALL COMSUB | Prints the required Latitude value which is saved in Memory |
| LONGITUDE:  MOV DPTR,#LON  LONLOOP:CLR A  MOVC A,@A+DPTR  JZ LONVALUE  ACALL DATASUB  INC DPTR  SJMP LONLOOP | Prints Longitude Screen |
| LONVALUE:  MOV R1,#6EH  LONNUM:CLR A  MOV A,@R1  ACALL DATASUB  INC R1  CJNE R1,#78H,LONNUM  CJNE R7,#01H,FIRSTLOOP  SJMP NEXTLOOPS  FIRSTLOOP:INC R7  NEXTLOOPS:ACALL DELAY  MOV A,#01H  ACALL COMSUB  AJMP MAINLOOP | Prints the required Longitude value which is saved in Memory |
| END | End of progran |

**HARDWARE:**

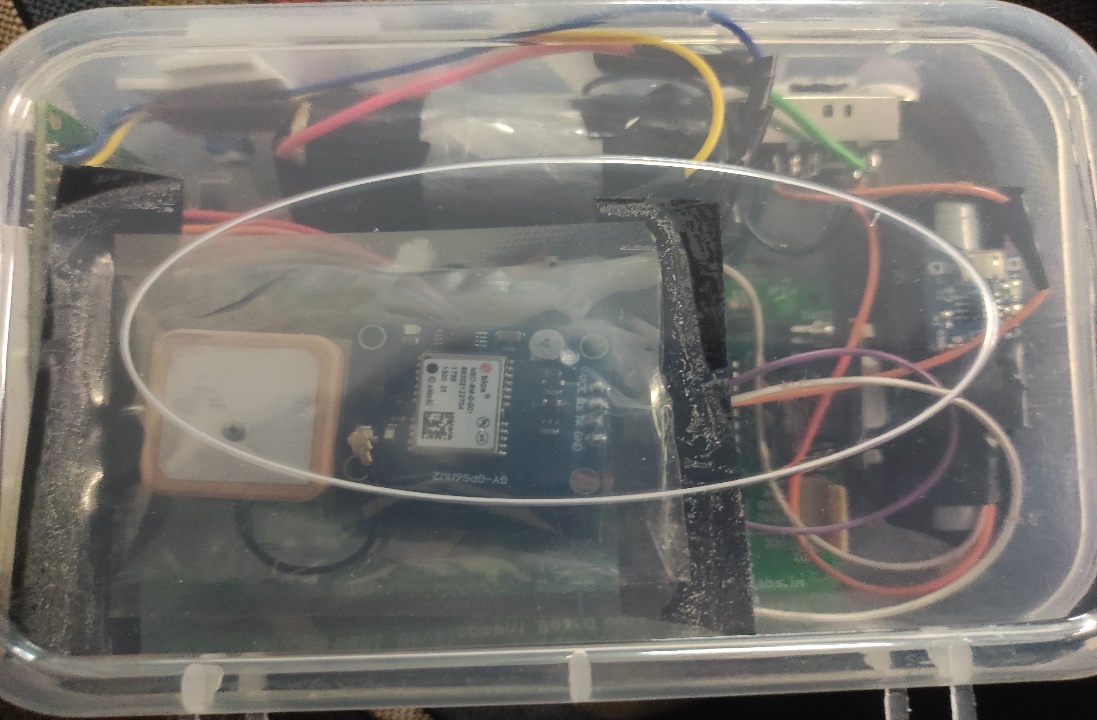


Fig. Hardware overall set up

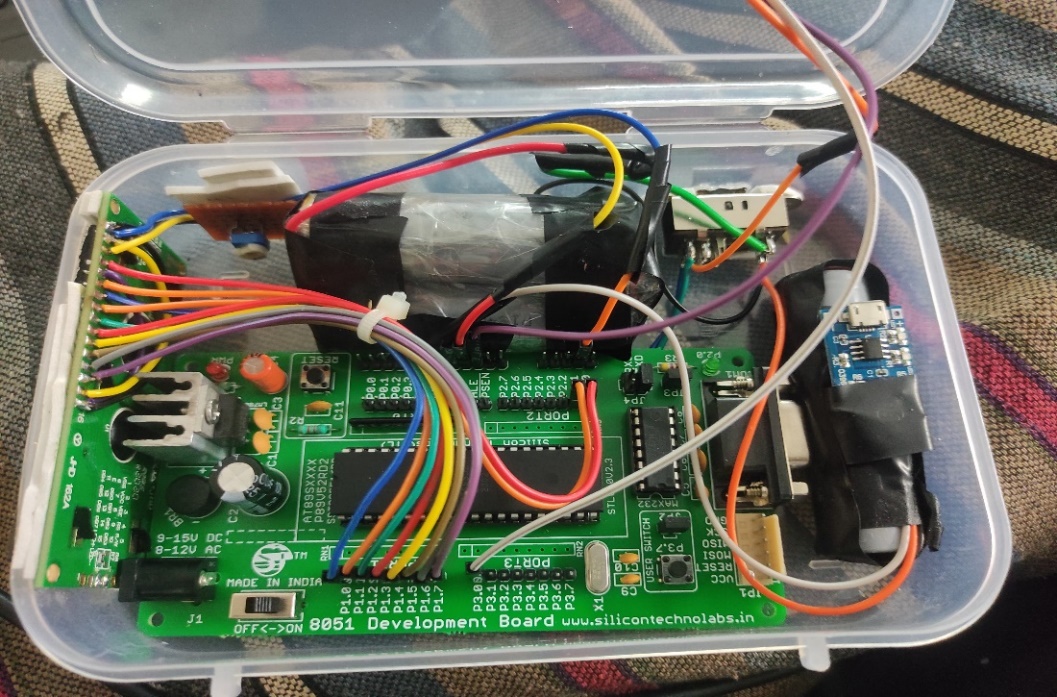


Fig. Hardware connected- 8051 interfaced with LCD and GPS