Note : Please refer to "GNSS receiver - Complete Logs" to understand various decisions taken in the design and development process including component selection, PCB design, Enclosure design, etc.

Aim

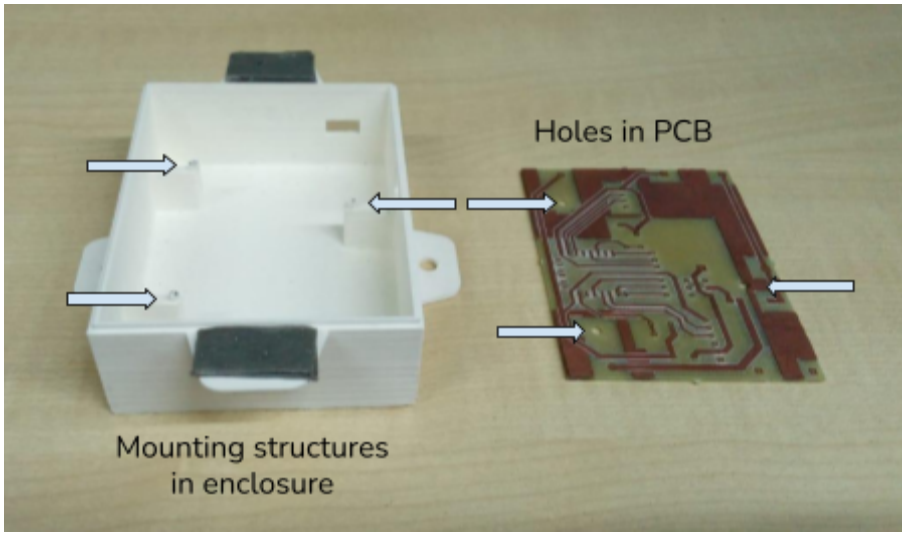
To build a low-cost, power-efficient GNSS receiver for tracking last-mile vehicles like rickshaws, hand-carts, etc.

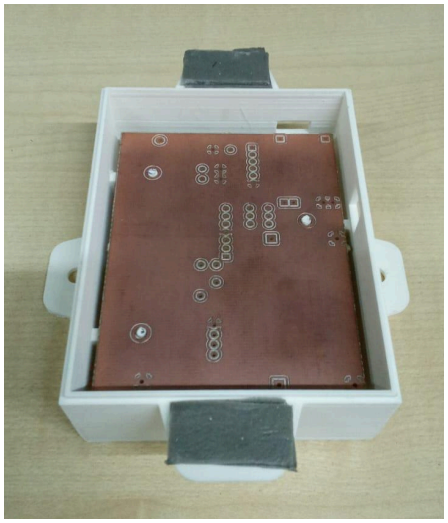
Components

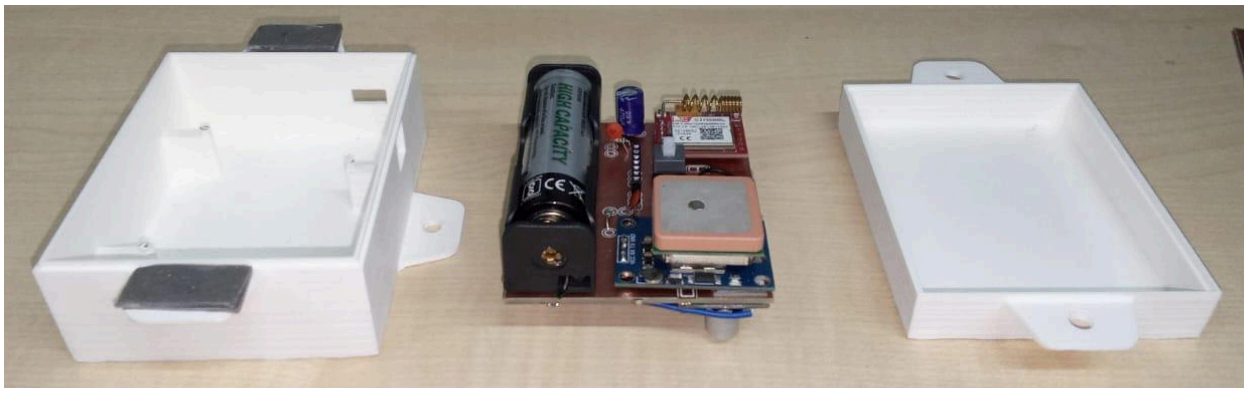
1. Microcontroller : ESP12F
2. GSM : SIM800L
3. GPS : NEO6M
4. Li-Ion charging module : TP4056
5. Voltage regulation : LM2956
6. Resistors
7. Capacitors (Ceramic and Electrolytic) : For ESP12F, SIM800L (to prevent voltage drops across power terminals)
8. Switch
9. Female headers : For programming the microcontroller on PCB in future
10. Li-Ion Battery and Battery Holder
11. PCB : 2-side milled
12. Enclosure : 3D printed

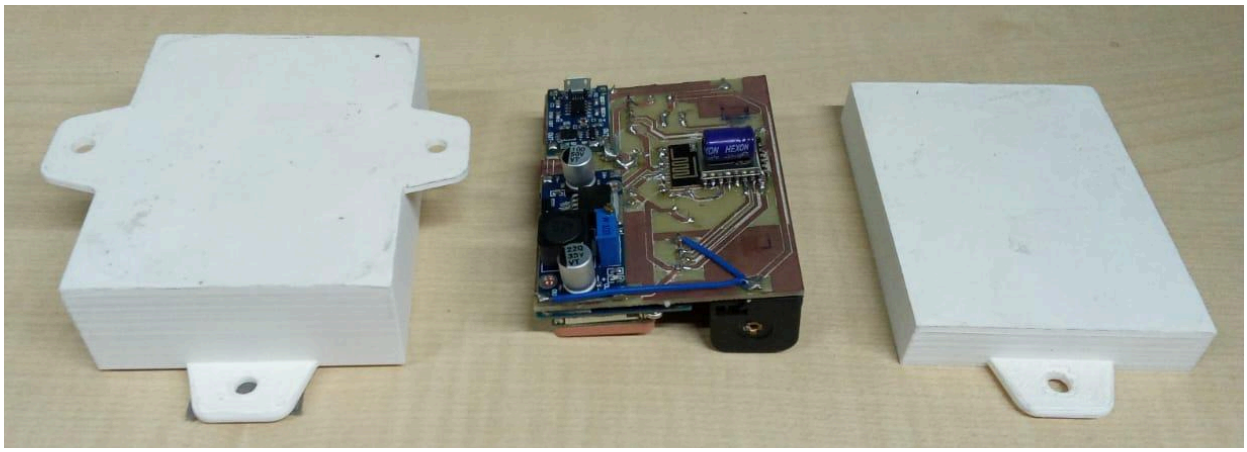
Final Prototype

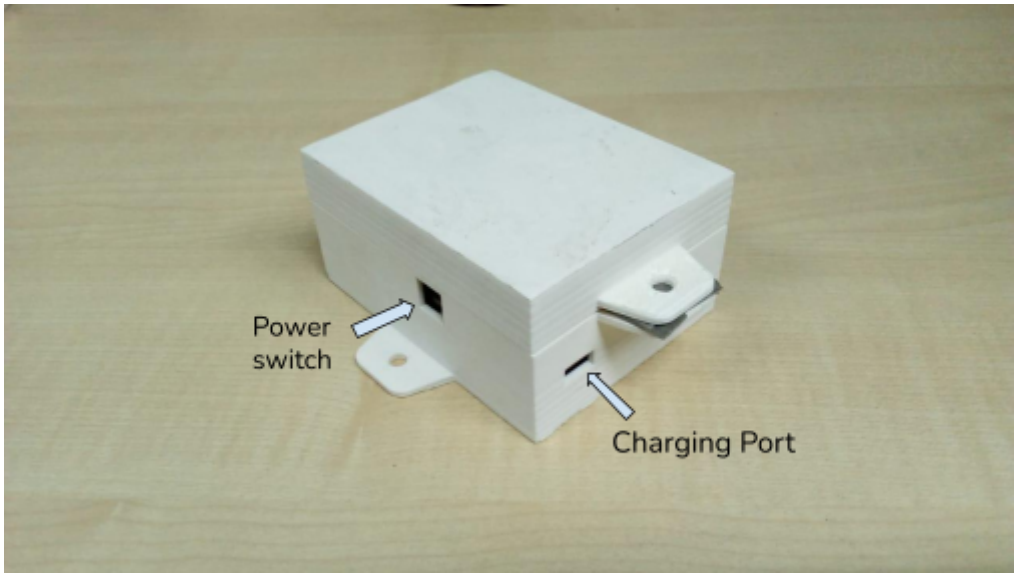
Schematic :
https://github.com/Raghul-PK/GPS_Receiver/tree/main/Final%20Prototype/Schematic
Enclosure (3-D printed) :
https://github.com/Raghul-PK/GPS_Receiver/tree/main/Final%20Prototype/Enclosure%20Design
PCB :
https://github.com/Raghul-PK/GPS_Receiver/tree/main/Final%20Prototype/PCB%20Design
ESP12F code (Go to V2 for Deep-sleep code → More power optimized) :
https://github.com/Raghul-PK/GPS_Receiver/blob/main/Tracker/V1_LibrariesSupport.ino
Backend Location APIs (Flask) - Hosted on Chartr Dev server using gunicorn+nginx :
https://github.com/Raghul-PK/GPS_Receiver/blob/main/Backend_Location_APIs.py


Holes in PCB
Mounting structures
in enclosure




HIGH CAPACITY





Quick guide for setting up the Prototype

(Note : I have used Airtel SIM as SIM800L supports only 2G and have used Firebase as my DB to store Location+SNR data)

1. Insert SIM card in SIM800L
2. Create a new Google Firebase's Realtime Database
3. Goto ESP12F code and make the following changes;
    a. Change the APN username and password according to your Data provider (SIM used)

```
18      char apn[]  = "airtelgprs.com";
19      char user[] = "";
20      char pass[] = "";
```

b. Write your corresponding DB url (If you are using Firebase as your Database, only do the following changes)

```
13      const char FIREBASE_HOST[]  = "gps-tracker2-13e30-default-rtdb.firebaseio.com";
14      const String FIREBASE_AUTH  = "zhw6Zkn3OQXjojhr8dDPn3OTJFdzhNk27c828puQ";
15      const String FIREBASE_PATH  = "/";
16      const int SSL_PORT          = 443;
```

4. Goto Backend_Location_APIs.py code
   a. Change the URL to read the data from. Incase, you are using Firebase change your URL and the corresponding AUTH key as well (lines 11 and 41)

```
10  def getLatestLocation(gps_id):
11      url = "https://gps-tracker2-13e30-default-rtdb.firebaseio.com/" + gps_id + ".json?auth=zhw6Zkn3OQXjojhr8dDPn3OTJFdzhNk27c828puQ"

40  def getDateJourney(gps_id, date):
41      url = "https://gps-tracker2-13e30-default-rtdb.firebaseio.com/" + gps_id + ".json?auth=zhw6Zkn3OQXjojhr8dDPn3OTJFdzhNk27c828puQ"
```

How does ESP12F code work ?

In void setup(), the modem is started. Notice, (line 58) where setHttpResponseTimeout is used to control the HTTP response wait-time after every request (or DB update) we do.

```
65      if (!modem.gprsConnect(apn, user, pass))
```

Connecting the modem to GPRS to enable data connection (connecting with the internet).

```
73      http_client.connect(FIREBASE_HOST, SSL_PORT);
```

Connecting with Firebase Database on the given port.

```
77      while (true) {
```

The while loop in line 77 is used for transferring data to DB. If HTTP connection breaks, the modem object is restarted.

```
135  void create_data()
```

The logic for reading GPS data (Serial data comes to ESP12F from NEO-6M GPS) and extracting the necessary info (Lat, Long, SNR, etc) is handled in create_data() function (line 135).

```
93  void PostToFirebase(const String & path , const String & data, HttpClient* http)
```

The logic of sending data to Database is handled in PostToFirebase function (line 93).

Here is how the Google Firebase Realtime Database is structured...

Backend Location APIs (Built in Flask - Click Link above to see code)

1. Get Last known location of a particular vehicle
2. Get All coordinates (journey) of a particular vehicle on a particular data