WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Development of an All-Terrain IoT-Enabled Rover Controlled via ESP32 Microcontroller through an App

[1]Prof. Imran Rafique Sayyed, [2]Rajeev Devender Gaddam, [3]Arshad Esak Shaikh, [4]Vrishin Yogesh Patil, [5]Soham Sunil Katkar

Computer Engineering, Vidyalankar Polytechnic, Mumbai

ABSTRACT

This paper presents the design, implementation, and testing of an all-terrain rover equipped with IoT capabilities, aimed at enhancing versatility and functionality in diverse environments.

The rover employs an ESP32 microcontroller as its core, integrating components such as a GPS module for navigation, a camera module for live video streaming, and sensors for environmental monitoring and obstacle detection.

The development process involved meticulous hardware integration, custom software development, and iterative PCB Design. Testing demonstrated the rover's capability to navigate urban and off-road terrains effectively, with potential applications in exploration, surveillance, and research. Compared to existing designs, this rover provides a cost-effective and compact solution leveraging advanced IoT technologies.

1. Introduction

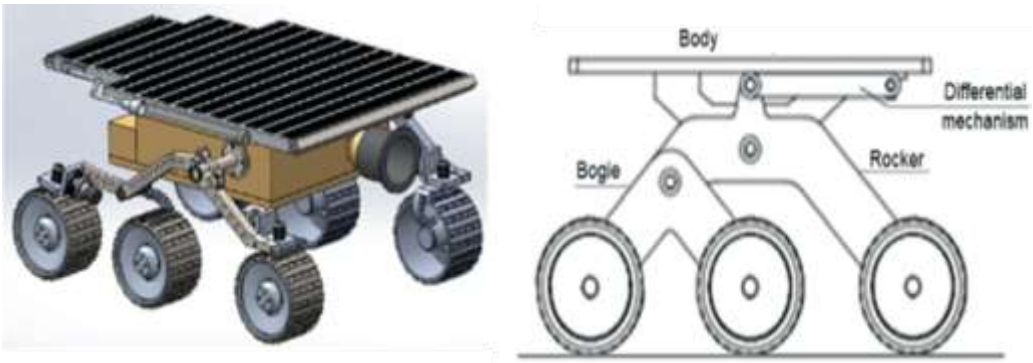
Rovers play a critical role in exploration, automation, and data collection across varied terrains, ranging from urban landscapes to rugged, off-road environments. With advancements in IoT technologies, real-time data acquisition and remote control have become increasingly accessible, enabling more efficient and responsive designs.

Current challenges in rover design include high costs, limited adaptability, and suboptimal integration of sensors and actuators. This paper presents the development of a cost-effective, all-terrain rover using an ESP32 microcontroller, designed to address these issues.

The rover integrates multiple sensors for environmental monitoring and obstacle detection. Its compact PCB Design enhances reliability in harsh conditions.

2. Review of Literature

Fig. 1. Rocker-Bogie Suspension Mechanism.



Planetary exploration missions demand innovative mobility solutions to navigate rugged and unpredictable terrain. The rocker–bogie suspension mechanism has been widely adopted in extraterrestrial rovers, most notably in NASA's Mars Exploration Rovers (Spirit, Opportunity, Curiosity, and

Perseverance). This paper presents a comprehensive overview of the rocker–bogie mechanism, covering its design principles, mechanical structure, kinematics, and advantages over conventional suspension systems. We analyze its impact on rover stability, load distribution, and obstacle negotiation. Additionally, the paper discusses simulation results and real-world performance data to evaluate its effectiveness for future planetary and terrestrial robotic applications.

The success of planetary rovers depends on their ability to traverse highly irregular terrain while maintaining stability and mobility. Traditional suspension systems often fail when confronted with obstacles, steep slopes, or loose surfaces. The rocker–bogie suspension mechanism, originally developed for NASA's Mars Pathfinder mission in the 1990s (Bickler, 1998), has proven to be an effective passive system that enhances rover adaptability and load distribution. The goal of this paper is to analyze the rocker–bogie mechanism in detail, discussing its historical applications in Mars exploration, its kinematic and dynamic behavior, and its potential for future rover applications.

Numerous studies have examined the performance of the rocker–bogie mechanism in planetary environments. Bickler (1998) first introduced the concept in the Mars Pathfinder mission, demonstrating its effectiveness in maintaining stability on uneven terrain. Subsequent analyses by Lindemann & Voorhees (2005) detailed its deployment on Spirit and Opportunity, emphasizing its passive suspension benefits. More recent studies, such as Guo et al. (2015), have investigated hybrid rocker–bogie systems incorporating active elements for improved terrain adaptation. These studies collectively highlight the advantages of the rocker–bogie system, including its ability to:

• Maintain continuous wheel–ground contact.

• Distribute loads evenly across multiple wheels.

• Passively adapt to varying terrain without requiring additional power consumption.

1. Mechanical Structure and Kinematics

1.1 Rocker and Bogie Components-

The rocker–bogie suspension consists of two main components:

• Rockers: These long pivoting arms are connected to the rover's main chassis. They distribute load across the system and pivot to maintain balance.

• Bogies: Smaller linkages attached to the rockers, allowing for independent wheel movement, thereby maintaining ground contact on uneven surfaces.

1.2 Differential Mechanism-

A critical feature of the rocker–bogie system is the differential mechanism, which ensures that when one rocker rises over an obstacle, the opposite side adjusts accordingly, keeping the rover's chassis relatively level. This design prevents excessive tilting and maintains stability.

1.3 Kinematic Analysis-

The kinematics of the rocker–bogie suspension allows the rover to overcome obstacles up to twice the wheel diameter while minimizing chassis displacement (Lindemann & Voorhees, 2005). The following principles govern its movement:

• Obstacle Climbing: When one wheel encounters an obstacle, the bogie pivots, lifting the wheel while the other wheels remain grounded.

• Load Balancing: Forces are distributed evenly across all six wheels, preventing excessive stress on any single wheel.

• Passive Suspension: No additional motors or actuators are required to adjust wheel height, reducing power consumption.

2. Design Considerations for Rover Applications

2.1 Terrain Adaptability-

The primary advantage of the rocker–bogie system is its ability to traverse uneven and unpredictable terrain. It has been tested extensively in environments simulating Martian conditions, showing exceptional adaptability in rocky and sandy landscapes (Volpe, 2003).

2.2 Stability and Traction-

• Prevention of Tipping: The suspension system prevents excessive tilting, reducing the risk of overturning.

•Maximization of Traction: Continuous wheel–ground contact ensures optimal traction on loose or uneven surfaces.

2.3 Scalability and Customization-

The rocker–bogie design is scalable and can be adapted for different rover sizes by modifying:

• Rocker and bogie length.

• Wheel size and placement.

• Differential mechanism configuration.

2.4 Trade-Offs-

Despite its advantages, the rocker–bogie mechanism has some limitations:

• Mechanical Complexity: The increased number of moving parts introduces potential failure points.

• Speed Limitations: Due to its passive nature, the system limits the rover's speed, as rapid movement can cause excessive rocking.

3. Testing, Simulation, and Performance Analysis

3.1 Simulation Methods-

Simulation tools such as MATLAB, Gazebo, and Adams MSC have been used to evaluate the performance of rocker–bogie suspension in different terrains. Studies by Guo et al. (2015) used Finite Element Analysis (FEA) to optimize structural durability.

3.2 Experimental Results-

• NASA's Mars Rovers: Spirit and Opportunity successfully traversed Martian terrain for over 15 years, validating the rocker–bogie system's reliability.

• Curiosity Rover: Successfully climbed slopes of up to 45 degrees and overcame obstacles of 65 cm (Muirhead, 2014).

4. Implications for Future Rover Projects

The rocker–bogie mechanism remains the gold standard for planetary rovers. However, future adaptations may include:

• Hybrid Systems: Incorporating active suspension elements for enhanced adaptability.

• Advanced Materials: Using lightweight composites to reduce weight and increase durability.

• Autonomous Control: Integrating AI-based real-time terrain analysis for adaptive suspension behavior.

The rocker–bogie suspension mechanism has been instrumental in planetary exploration, proving its effectiveness in Mars missions. Its ability to passively navigate obstacles, maintain stability, and distribute loads efficiently makes it an ideal choice for future rover applications. While improvements in materials and hybrid active-passive systems may further enhance its capabilities, the fundamental design remains a benchmark for rover mobility.

References:

1. Bickler, D. B. (1998). "Rocker-Bogie Suspension System for the Mars Rover." U.S. Patent No. 5,634,667.

2. Lindemann, R. A., & Voorhees, C. J. (2005). "Mars Exploration Rover Mobility Assembly Design, Test, and Performance." IEEE Systems Journal, 41(1), 73–85.

3. Guo, J., Chen, Q., & Yu, H. (2015). "Hybrid Rocker-Bogie Suspension for Improved Obstacle Climbing Performance." Advances in Mechanical Engineering, 7(9), 1–10.

4. Volpe, R. (2003). "Rover Functional Autonomy Development for the Mars Exploration Rovers." IEEE Aerospace Conference Proceedings, 2, 743–752.

5. Muirhead, B. K. (2014). "The Mars Science Laboratory Mission." Acta Astronautica, 94(1), 1–15.

3. Objective:

- Surveillance & Monitoring – Provide real-time video streaming using the ESP32-CAM module.
- Sensor Data Collection – Gather environmental data using BME280, MPU6050, ultrasonic sensors, and GPS.
- Navigation & Positioning – Use GPS for real-time location tracking.
- Efficient Power Management – Optimize battery usage with multiple power sources.

- Modular Design – Allow easy upgrades and modifications.
- Wireless Communication – Control the rover via a custom-built Flutter app.
- Terrain Adaptability – Ensure smooth movement across rough and uneven surface.

4. Advantages:

- Versatility – Can be used for exploration, surveillance, and research in various environments.
- Cost-Effective – Uses ESP32 and other affordable components.
- Remote Accessibility – Controlled via an app, eliminating the need for physical presence.
- Real-Time Data Transmission – Provides instant video feed and sensor data.
- Customizability – Easily upgradeable for additional functionalities.
- Educational Value – Enhances knowledge in IoT, embedded systems, and robotics.

5. System Architecture:

A. Hardware Components

The rover integrates the following components:

- ESP32 Microcontroller: Serves as the central processing unit, managing sensors, motors, and communication.
- HC-SR04 Sensor: Provides accurate obstacle detection through ultrasonic signals.
- Neo-6M GPS Module: Supplies real-time location and navigation data.
- ESP32-CAM Module: Streams live video for remote monitoring and navigation.
- DHT11 Sensor: Monitors environmental parameters such as temperature and humidity.
- MPU-6050 IMU: Measures orientation and motion to enhance stability and control.
- L298N Motor Drivers: Drives six DC motors, ensuring adequate torque for various terrains.
- Power Supply: A single 12V 3A source powers the entire system. The power supply circuit includes a DC jack, IN4007 diode, LM7805_T0220 voltage regulator, two 100μF capacitors, a 470Ω resistor, and an LED indicator for status monitoring.
- The integration of these components was guided by their reliability, availability, and compatibility with the ESP32 platform.

B. Software Design

The rover's software stack is designed to ensure seamless communication and efficient processing:

1. Firmware Development: The ESP32 microcontroller runs custom firmware written in C++, enabling real-time processing of sensor data and motor control. Libraries such as WiFi.h and Adafruit Sensor were utilized to simplify development.
2. Mobile Application: A user-friendly app was developed using Flutter, offering cross-platform compatibility. Key features include:
3. Real-time video streaming from the ESP32-CAM module.
4. GPS-based location visualization on an interactive map.
5. Monitoring of sensor data, including temperature, pressure, and orientation.
6. Control interfaces for navigation and servo motor adjustments.
7. Communication Protocols: Wi-Fi is used for data transmission between the rover and the app, ensuring low latency and reliable connectivity.

6. How to Initialize ESP32-CAM Module:

The ESP32-CAM module is an essential component for real-time image streaming in rover projects. However, many users face challenges in setting up and initializing the module correctly. The following method was used to configure the ESP32-CAM module using the Arduino IDE:

1. Prerequisites

Before proceeding with the setup, ensure you have the following:

• ESP32-CAM Module

• FTDI Programmer (3.3V/5V USB-to-Serial Adapter)

• Jumper Wires

• Arduino IDE Installed

• Wi-Fi Network Credentials (SSID and Password)

2. Step-by-Step Initialization Guide

Step 1: Install Arduino IDE and Required Libraries

1. Open Arduino IDE on your computer.

2. Go to File → Preferences and enter the following URL in the Additional Board Manager URLs field:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

3. Navigate to Tools → Board → Boards Manager and search for ESP32. Install the latest ESP32 by Espressif Systems package.

4. Next, install the necessary library:

- Go to Sketch → Include Library → Manage Libraries.
- Search for "ESP32" and install the latest version of the ESP32 Camera library.

Step 2: Connect ESP32-CAM to FTDI Programmer

To upload code to the ESP32-CAM, you need to connect it to an FTDI programmer in FLASH mode. Use the following connections:

ESP32-CAM	**FTDI Programmer**
5V	VCC (5V)
GND	GND
U0R (RX)	TX
U0T (TX)	RX
IO0	GND (For Flashing)

Step 3: Load Camera Web Server Example

1. In Arduino IDE, go to File → Examples → ESP32 → Camera → CameraWebServer.

2. Edit the following line in the code:

#define CAMERA_MODEL_AI_THINKER

and define the AI-THINKER model by uncommenting.

3.comment the default code:

#define CAMERA_MODEL_ESP_EYE

Step 4: Enter Wi-Fi Credentials

Find and modify the following lines in the code to enter your Wi-Fi SSID and password:

Note wifi credentials should be off wifi device such as wifi router or mobile phone wifi.

const char* ssid = "Your_WIFI_Name";

const char* password = "Your_WIFI_Password";

Step 5: Select Board and Port

1. Go to Tools → Board and select AI Thinker ESP32-CAM.

2. Set the following parameters:

- Upload Speed: 115200
- Flash Mode: QIO
- Partition Scheme: Huge APP

3. Select the correct COM port under Tools → Port.

Step 6: Upload Code to ESP32-CAM

1. Press and hold the BOOT(IO0) button on the ESP32-CAM module.

2. Click the Upload button in Arduino IDE.

3. Once uploading is complete, disconnect IO0 from GND and press the RESET button.

Step 7: Get the ESP32-CAM IP Address

1. Open the Serial Monitor (Baud Rate: 115200).

2. Once the ESP32-CAM connects to Wi-Fi, it will display an IP Address in the Serial Monitor.

3. Copy this IP address and enter it into any web browser to access the ESP32-CAM web server.

3. Troubleshooting Common Issues

• Error: Camera Not Detected → Ensure that all wiring is correct, and the ESP32-CAM has sufficient power (5V).

• Continuous Rebooting (Brownout Error) → Use a better power supply (5V 2A recommended).

• Upload Fails → Double-check that IO0 is connected to GND while flashing.

• No IP Address in Serial Monitor → Ensure the correct Wi-Fi credentials are entered.

• JPEG format is not supported in this module → In the code of camera web server just change the code as given below:

config.frame_size = FRAMESIZE_SVGA;

config.pixel_format = PIXFORMAT_RGB565;

Then upload the code and the error will be fixed.

By following this step-by-step guide, users can successfully initialize the ESP32-CAM for real-time image streaming in their rover project. Proper setup ensures smooth operation and prevents common errors that arise during initialization

7. How to Initialize NEO-6M GPS Module

The NEO-6M GPS module is widely used in rover applications for location tracking and navigation. However, users often face difficulties in initializing and obtaining GPS data. The following method was used to configure the NEO-6M GPS module:

1. Prerequisites

Before proceeding with the setup, ensure you have the following:

- NEO-6M GPS Module
- ESP32 Microcontroller
- Arduino IDE Installed
- Jumper Wires for Connections

2. Step-by-Step Initialization Guide

Step 1: Install Arduino IDE and Required Libraries

1. Open Arduino IDE on your computer.

2. Install the required libraries to communicate with the NEO-6M module:

- Go to Sketch → Include Library → Manage Libraries.
- Search for TinyGPS++ and install it.
- Search for SoftwareSerial (if using an Arduino instead of ESP32) and install it.

Step 2: Connect NEO-6M GPS Module to ESP32

Connect the NEO-6M module to the ESP32 using the following wiring:

NEO-6M GPS Module	**ESP32**
VCC	3.3V or 5V (Check module specs)
NEO-6M GPS Module	**ESP32**
GND	GND
TX	GPIO16
RX	GPIO17

Step 3: Upload the Test Code

1. Open Arduino IDE and create a new sketch.

2. Copy and paste the following test code to check if the NEO-6M module is working:

```
#include <TinyGPS++.h>
#include <HardwareSerial.h>
static const int RXPin = 16, TXPin = 17; // Define GPS RX and TX pins
static const uint32_t GPSBaud = 9600; // Default baud rate for NEO-6M
  TinyGPSPlus gps;
HardwareSerial GPS_Serial(1);
  void setup() {
    Serial.begin(115200);
    GPS_Serial.begin(GPSBaud, SERIAL_8N1, RXPin, TXPin);
    Serial.println("GPS Module Initialized. Waiting for data...");
}
   void loop() {
    while (GPS_Serial.available() > 0) {
       gps.encode(GPS_Serial.read());

       if (gps.location.isUpdated()) {
          Serial.print("Latitude: ");
          Serial.println(gps.location.lat(), 6);
          Serial.print("Longitude: ");
          Serial.println(gps.location.lng(), 6);
       }
    }
}
```

Step 4: Select Board and Port

1. Go to Tools → Board and select DOIT ESP32 DEVKIT V1 Module (or your ESP32 variant).

2. Select the correct COM port under Tools → Port.

3. Click Upload and wait for the process to be complete.

Step 5: Open Serial Monitor to View GPS Data

1. After uploading, open Serial Monitor (Baud Rate: 115200).

2. Wait for the NEO-6M module to obtain a GPS fix (this may take a few minutes).

3. Once the fix is acquired, you will see real-time latitude and longitude coordinates displayed on the Serial Monitor.

3. Troubleshooting Common Issues

• No GPS Data Displayed? → Ensure the GPS module has a clear view of the sky (outdoors or near a window).

• Baud Rate Mismatch? → Verify that the module is communicating at 9600 baud. If not, try 38400 baud.

• Incorrect Wiring? → Double-check TX and RX connections (TX from GPS should go to RX on ESP32 and vice versa).

• Weak Satellite Signal? → The module may take up to 5-10 minutes to acquire a fix the first time

By following this step-by-step guide, users can successfully initialize the NEO-6M GPS module and integrate real-time location tracking into their rover project. A properly configured GPS module ensures accurate navigation and positioning for autonomous exploration.

8. Flutter App Integration using flutter_mjpeg:

To stream the **ESP32-CAM video** inside a Flutter application:

1. Add the **flutter_mjpeg** package in pubspec.yaml:

```
dependencies:
  flutter:
    sdk: flutter
  flutter_mjpeg: ^2.1.0
```

2. Use the following Flutter widget to display the ESP32-CAM stream:

```
import 'package:flutter/material.dart';
import 'package:flutter_mjpeg/flutter_mjpeg.dart';

class CameraStreamPage extends StatelessWidget {
  final String esp32CamIP = "http://192.168.1.100"; // Replace with your ESP32-CAM IP

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("ESP32-CAM Live Stream")),
      body: Center(
        child: Mjpeg(
          stream: "$esp32CamIP:81/stream",
          isLive: true,
          fit: BoxFit.cover,
        ),
      ),
```

```
  );
 }
}
```

- **Replace** "http://192.168.1.100" with the actual ESP32-CAM IP.
- The **Mjpeg widget** streams the **ESP32-CAM feed in real-time**.

4. Final Steps

- Run flutter pub get to install dependencies.
- Launch the app, and the **ESP32-CAM live stream** will be displayed.
- Ensure that both the **ESP32-CAM and mobile device are on the same WiFi network**.

This method provides a **low-latency video stream** directly inside a Flutter app, making it ideal for **robotics, surveillance, and IoT applications**.

9. Results:

- Successful Operation – The rover functions smoothly in both manual and autonomous modes.
- Real-Time Data & Video Streaming – ESP32-CAM provides a live feed, and sensor data is transmitted to the app.
- Accurate GPS Tracking – The rover can send real-time coordinates to the Flutter app.
- Stable Connectivity – The ESP32 in AP mode ensures uninterrupted communication between the rover and the app.
- Effective Terrain Handling – The 6-wheel drive and motor controllers enable smooth movement on different surfaces.
- Optimized Power Consumption – Multiple power sources effectively sustain long operational hours.
- User-Friendly App Interface – The Flutter app provides an intuitive UI/UX for controlling the rover.
- Successful Exhibition Presentation – The project was well-received and successfully demonstrated at the Final Year Project Exhibition at Vidyalankar Polytechnic, Mumbai.

10. Outputs:

1. System Overview Diagram (Introduction section): The block diagram of the entire rover system.

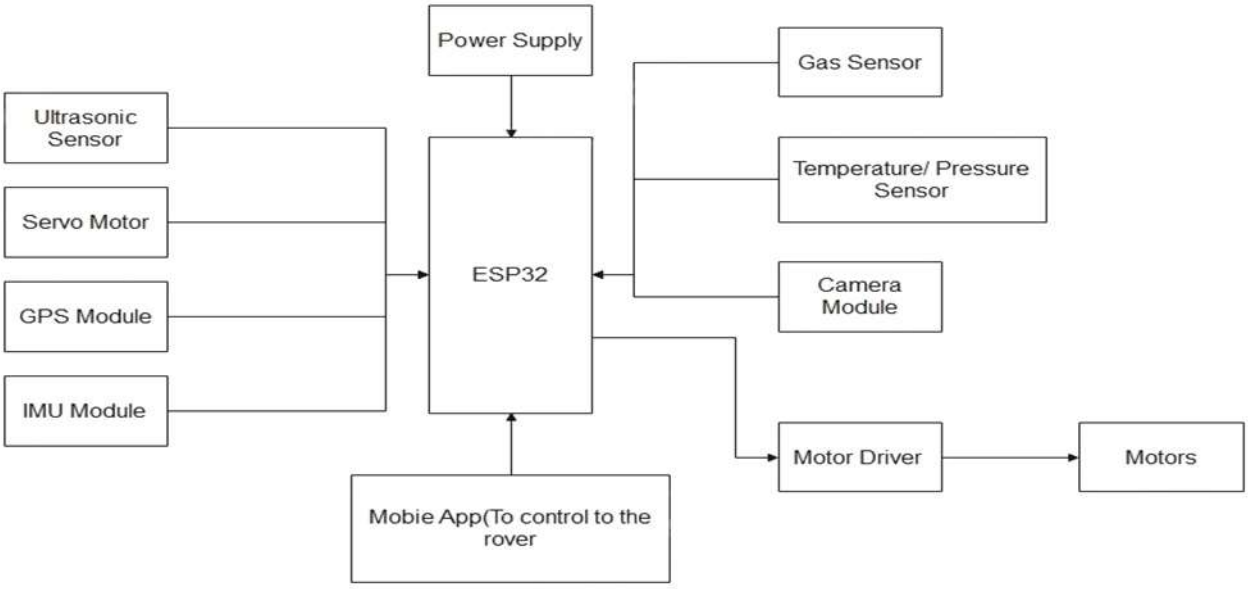


Fig. 2. The block diagram of the entire rover system.

2. Circuit Diagram of the rover:

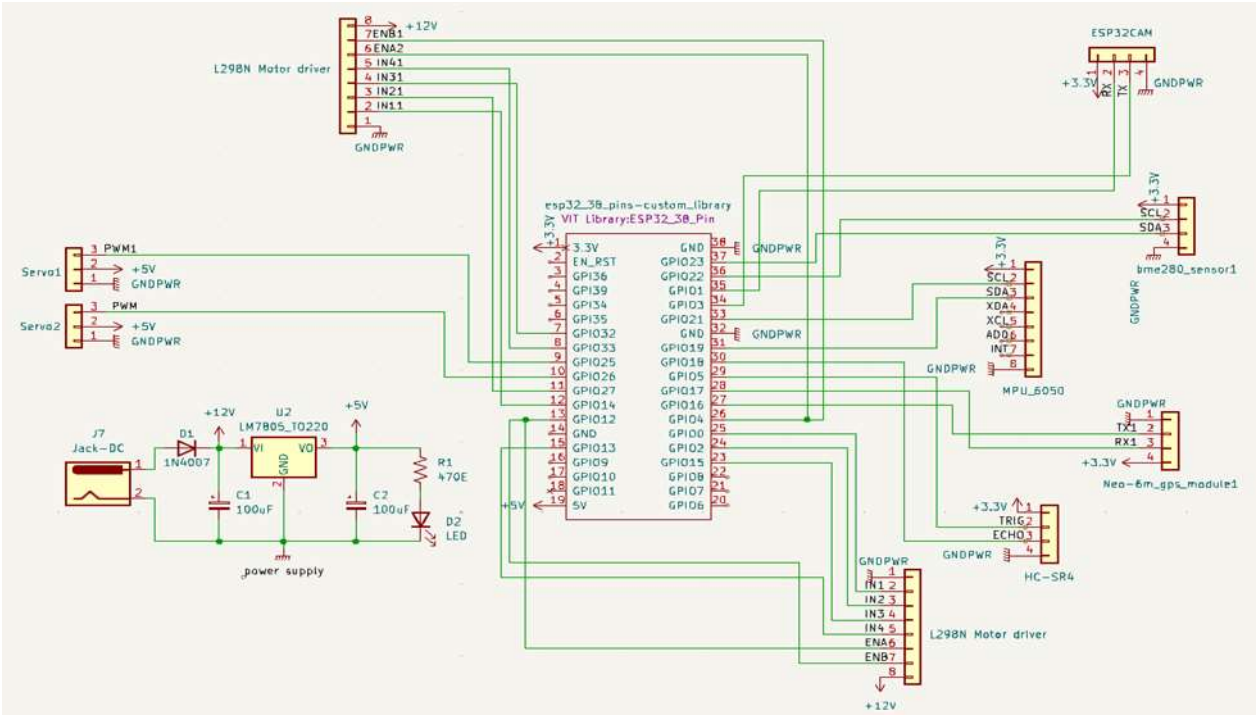


Fig. 3. The final Circuit diagram of the rover.

3. PCB Design Section:

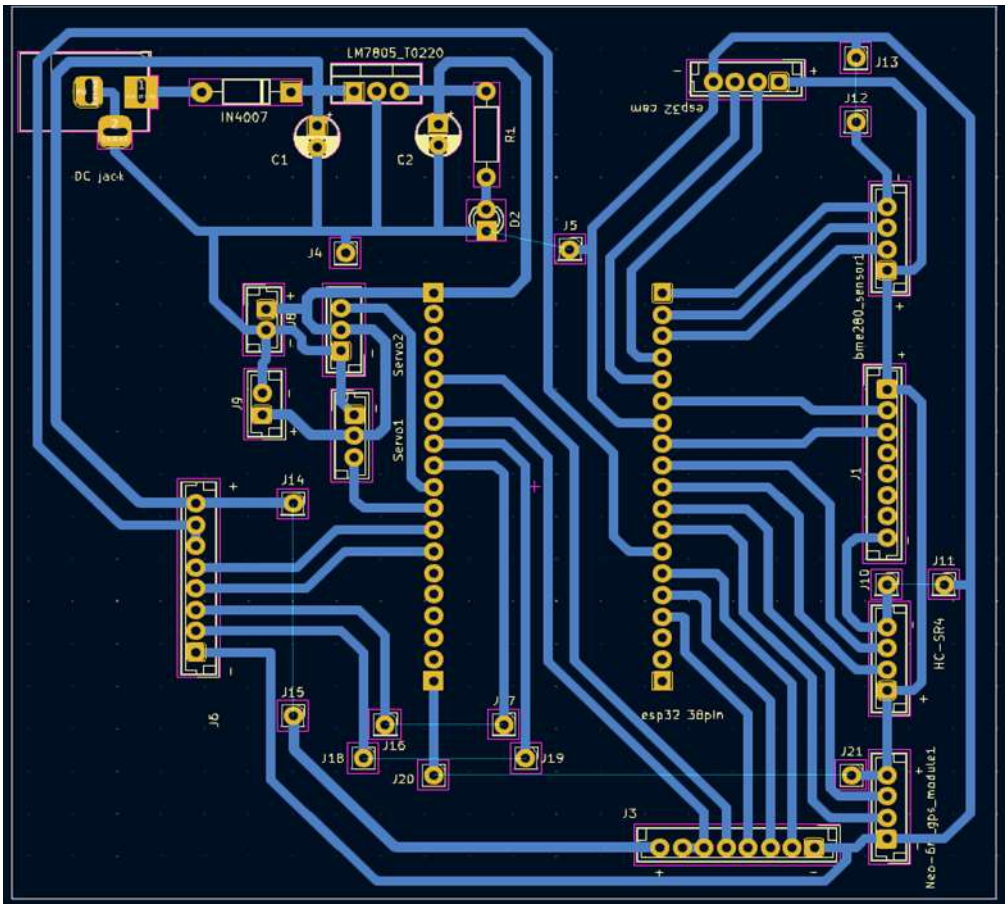


Fig. 4. Final PCB Design

4. App Interface Software:

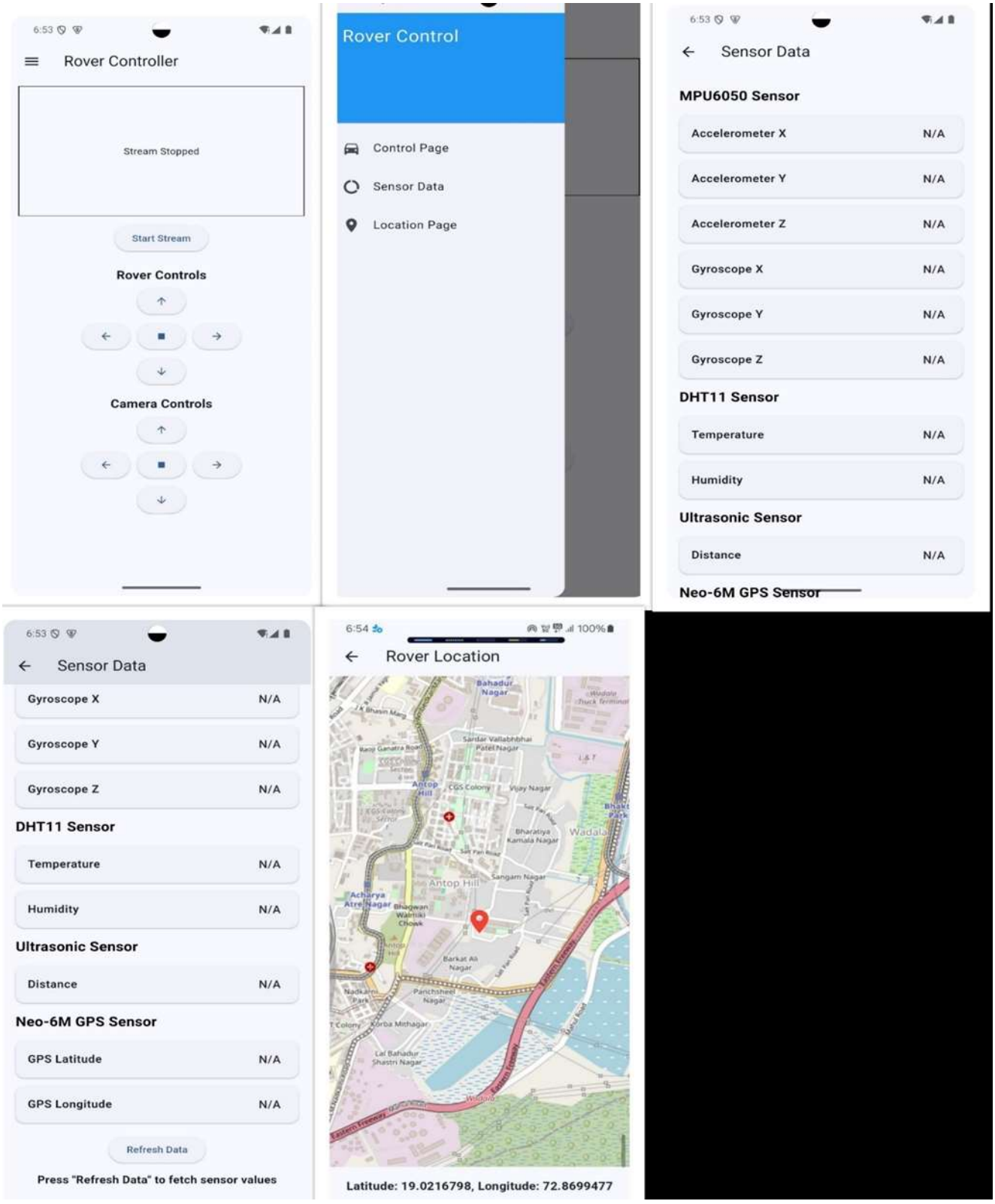


Fig. 5. Completed app software interfaces

11. References

[1] IJRPR, "International Journal of Research Publication and Reviews".

Available: https://www.ijrpr.com/editorialboard.php

[2] Espressif Systems, "ESP32 Technical Reference Manual",

Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[3] Adafruit, “DHT11 Sensor Module Datasheet”,

Available: https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf

[4] SparkFun, “Neo-6M GPS Module Documentation”,

Available: https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/

[5] Flutter, “Flutter: Beautiful Native Apps”,

Available: https://flutter.dev