Journal of Soft Computing Paradigm(ISSN: 2582-2640)
www.irojournals.com/jscp

Machine Learning-Enabled Rover For Autonomous Navigation In Rough Terrain

Parameswari S.[1], Mohammed Daiyaan R.[2], Chella Jayeshwar K.[3], Kannan K.[4]

[1]Associate Professor, [2,3,4]Students, Department of Electronics and Communication Engineering, Sri Sairam Institute of Technology, Tambaram, Chennai - 44, India.

E-mail: [1]parameshwari.ece@sairamit.edu.in,[2]sit23ec114@sairamtap.edu.in,[3]sit23ec061@sairamtap.edu.in, [4]sit23ec062@sairamtap.edu.in

Abstract

A hybrid perception-control framework for navigating rough terrains by remote YOLOv8n-based terrain classification combined with on-board safety-based control of rover is proposed. The system uses ESP32-WROOM-32 module for controlling motors, ultrasonic sensors for detecting obstacles, infrared sensors for detecting cliff edges, and watchdog communication, whereas ESP32-CAM is responsible for sending terrain images to a laptop where the inference algorithm works. Four-class terrain data consisting of 800 images of grass, sand, gravel, and soil is employed for fine-tuning YOLOv8n using stratified dataset. Terrain predictions control movements depending on traversability coefficients, while safety controls take precedence over remote controls when an obstacle or communication link is detected. The evaluation over 60 trials of outdoor navigation has resulted in terrain classification with an accuracy of 78.3% and successful navigation with an 80.0% success rate. The perception-to-action latency was 170±23 ms, fail-safe time was 1.02±0.04 s, and emergency braking was 310±45 ms. The power consumption of the rover was 4.22±0.35 W, and the estimated operational time was around 4.5 h. Such values characterize the performance of the limited resources rover that implements the separation of complex computation-based inference from motion control in embedded systems.

DOI: https://doi.org/10.36548/jscp.2026.4.008
Extended version of article received from **6th International Conference on Artificial Intelligence, 6G Communications and Network Technologies 2026**
Received: 07.09.2026, received in revised form: 05.10.2026, accepted: 19.10.2026, published: 04.11.2026
© 2026 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

Keywords: Autonomous Rover, Terrain Classification, YOLOv8n, ESP32, Hybrid Perception–Control, Obstacle Avoidance, Fail-Safe Navigation.

1. Introduction

Rapid advancement in automation technology has spurred a need for intelligent systems that can work in dangerous and unreachable areas like disaster-hit zones, mining sites, agricultural farms, and unknown territories. These requirements are fulfilled by autonomous rovers by integrating robotics, sensors, and ML to achieve perception and navigation.

The obstacles that prevent the use of autonomous rovers from becoming mainstream is the need for a computational platform like an NVIDIA Jetson board or a computer equivalent to Raspberry Pi. These platforms can execute DL models but increase cost, energy consumption, and complexity. On the other hand, microcontrollers like the ESP32 are cheap and energy-efficient but lack computational power.

To address this trade-off, the proposed architecture for the autonomous rover uses ESP32 and ESP32-CAM with visual inference handled by a remotely stationed laptop base station. A model using YOLOv8n is trained on the basis of a hybrid transfer learning dataset that consists of benchmark data available publicly on GitHub and the custom dataset consisting of video frames specific to the rover. The model recognizes the type of terrain that is currently prevailing from the video frames captured by the camera and feeds the information to the direction planner module.

The fail-safe mechanism in the proposed architecture is based on ultrasound. As opposed to a purely remote-control architecture, the rover is not completely dependent on constant communication between itself and the base station. In cases where there is a break in wireless communication or where there is an obstacle that needs to be addressed immediately, the ESP32 switches to a local fail-safe state using ultrasonic measurement as a means of stopping or changing directions for the rover.

The main contributions of this research are as follows: (1) the creation of a cost-effective ESP32 or ESP32-CAM rover platform with full multi-sensor fusion; (2) the training and utilization of a light-weight YOLOv8n model using domain adaptation based transfer learning; (3) the design of a user interactive IoT dashboard for live telemetry, heads up display and latency

monitoring; and (4) the empirical analysis of quantifiable rough terrain navigation capabilities and local safety reflexes.

2. Related Work

2.1 Terrain-Aware and Obstacle-Avoidance Rover Platforms

An ESP32-based mobile robot with an autonomous rough terrain navigation system incorporating multi-sensor obstacle avoidance and detection of hazards, similar to the objective of the present study, was introduced by Shilpa et al. [1], and reliable obstacle detection with fast response time performance was obtained for both manual and autonomous operation modes. Multi-sensor obstacle avoidance approach, which integrates ultrasonic sensing with road sign recognition using visual sensors and infrared pothole detection, was designed by Zhao et al. [2]. introduced multi-sensor obstacle avoidance using a combination of ultrasonic ranging with vision-based road sign recognition and infrared pothole detection, and their findings proved that fusion of simple sensors increases the reliability of obstacle avoidance compared to the use of just ultrasonic ranging sensors.

2.2 Learning-Based Visual Detection and Terrain Perception

The latest innovations in deep learning technology have allowed robots to process semantic information from camera images for the purpose of terrain comprehension, obstacle detection, and navigation [3], [4], [5], [6]. The object-detection and object-classification systems like the YOLO model are especially suited for real-time robot application due to their high-level feature extraction capacity with low latency.

This framework offers a light-weight architecture that can be used when inference speed is crucial for the implementation. In this case, the Nano version, YOLOv8n [10], is selected as the basic architecture since the computationally complex process of inference can be done on the remote laptop and not on the ESP32-CAM device. Instead of training based on only the initial set of object categories, the model is trained on a carefully selected dataset, which consists of public terrain benchmarks found on GitHub along with field images.

2.3 TinyML and Hybrid Remote-Processing Architectures

The use of TinyML through compression of ML models on microcontrollers has decreased reliance on cloud or laptop infrastructure in some instances of real-time inference. According to

Zim [7], analysis of the Xtensa LX6 processor, which forms the basis of ESP32, revealed its ability to handle small inference jobs but still with some limitations compared to GPU. Gupta and Shivhare [8] showcased how the ESP32 could be used in a TinyML pipeline for vibration-based predictive maintenance with more than 92% accuracy of fault detection possible onboard. A tactile-enabled deep reinforcement learning solution for avoiding obstacles dynamically in dense crowds was introduced by Ng et al. [9], demonstrating how machine learning-based reflex behaviors are increasingly becoming an essential part of robotics in addition to planning mechanisms. Due to the memory and processing limitations of the ESP32-CAM, a few mixed solutions transmit the camera feed to a better-equipped remote device for analysis, rather than executing the algorithm directly on the microcontroller board.

Collectively, all of this research proves that the low-budget ESP32 or ESP32-CAM rover with the offloaded ML algorithm and sensor fusion is indeed a plausible and justified design choice, not the compromise as compared to other advanced research systems. At the same time, the majority of current DIY and research projects are based on one of the two approaches: entirely utilizing TinyML on board, which significantly constrains the level of model accuracy, or remote control only, which provides no backup solution in case the connection fails.

3. Proposed Methodology

3.1 System Architecture

The overall structure of the rover consists of terrain perception, sensor monitoring, motor control, wireless communication, and power management. The ESP32-WROOM-32 handles the processing of the sensor values and movement of the rover, while the AI-Thinker ESP32-CAM with the OV2640 camera takes pictures of the terrain. Terrain pictures are sent to a base-station computer where the YOLOv8n algorithm executes the terrain classification process. HC-SR04 ultrasonic sensors measure the distance to objects in front of the rover, whereas TCRT5000 infrared sensors detect possible cliffs. A u-blox NEO-6M GPS sensor determines the position of the rover. The six wheels of the rover are controlled using two L298N motor drivers, which are connected to a 2S lithium-ion battery pack.

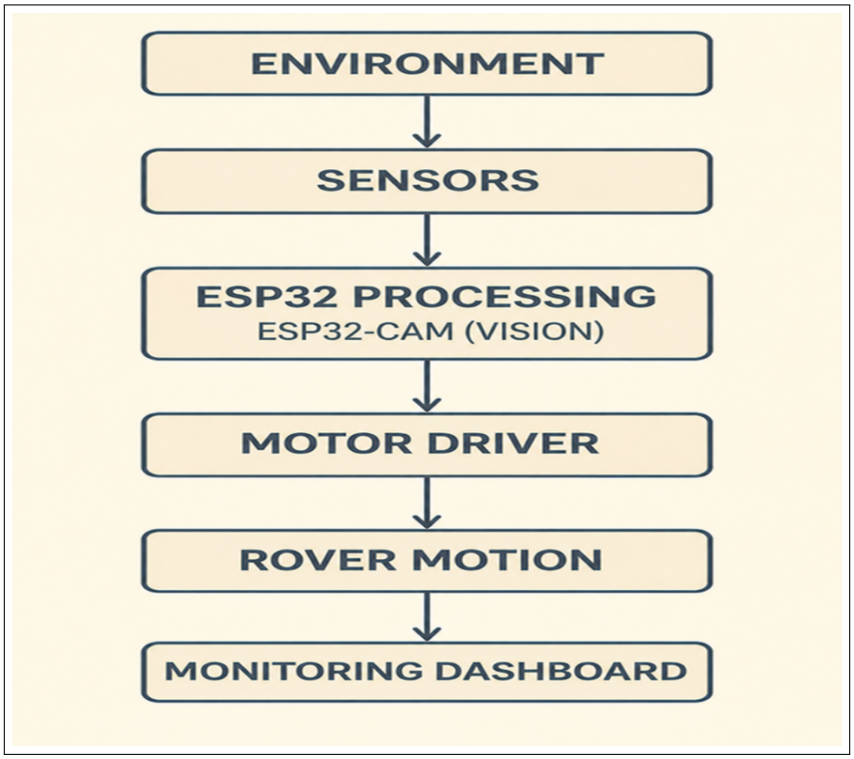


Figure 1. Overall system workflow of the autonomous terrain navigation rover

The work flow of the entire system along with the relationship among all of the subsystems is described in Figure 1. The terrain images are processed in the base station's laptop computer, but the process of sensor monitoring and the motor control is done in the rover. The wireless communication enables video streaming via TCP port 81 and commands via WebSocket port 8080.

The navigation decision flow involved in shifting between the remote machine learning based controls and the local fail-safe control is shown in Figure 2. With an established wireless link, the robot operates using terrain-based navigation commands. In case there is a loss of communication link, the local controller shifts into local fail-safe. Distance from the front obstacle is then evaluated against the set safety limit. In case the distance value is less than the safety limit, then the robot turns. Otherwise, it moves forward at a slower pace.

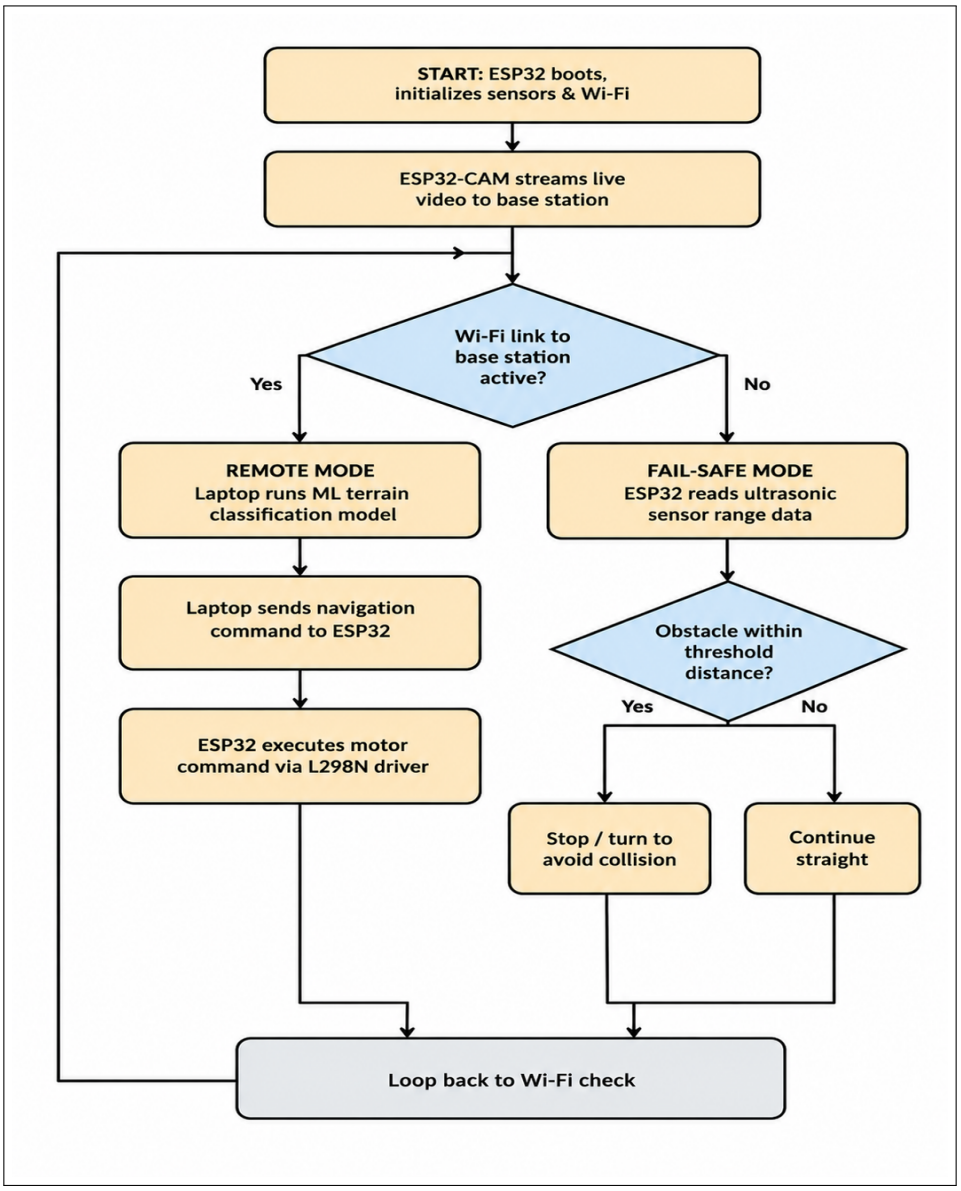


Figure 2. Navigation decision flow for remote control and local fail-safe operation

3.2 Dataset Preparation and Model Training

The terrain classification model was trained using images from the Off-Road Terrain Surface Classification dataset [11], as well as custom images captured by the rover's AI-Thinker OV2640 camera in preliminary field tests. The dataset includes four types of terrain: grass, sand, gravel, and soil, where each terrain type has 200 images in the dataset. For each terrain, 160 images have been collected from the publicly available benchmark, while 40 images have been collected through the rover camera. This gives a combination of public terrain images as well as those from the desired environment. The composition of the dataset is shown in Table 1.

Table 1. Dataset composition and class distribution for terrain classification

Terrain Class	**Public Benchmark [11]**	**Custom ESP32-CAM**	**Total Images**	**Train (70%)**	**Val (15%)**	**Test (15%)**	**Physical Visual Characteristics**
Grass	160	40	200	140	30	30	Green blade textures, vegetated patches, uneven turf
Sand	160	40	200	140	30	30	Fine granular particulate, yellow-tan hue, subject to solar glare
Gravel	160	40	200	140	30	30	Coarse broken stones, irregular high-contrast shadow boundaries, pebbles
Soil	160	40	200	140	30	30	Compacted dirt, dark loam, loose dry earth, subtle textural variations

Total	640	160	800	560	120	120	Balanced 4-class multi-terrain dataset with domain adaptation

Dataset partitioning is an important consideration because images extracted from the same video or recording session may contain highly similar visual patterns. In case such frames are divided into train, validate, and test sets, the evaluation could be overly optimistic about the performance of unseen terrain images. Thus, it would become necessary to identify the recordings first before splitting. Then, images from the same recording should be included in one set only where information about recordings is available. Finally, stratified splitting should be carried out in a ratio of 70:15:15 for all four terrain categories. The final test set should remain separate from model training and hyperparameter selection.

For image-based classification, the YOLOv8n-cls network [10] is applicable, as it will classify a terrain category for a given input image, and object bounding boxes are not needed. This network can be pre-trained on a classification task using pretrained weights and then fine-tuned on the four classes of interest. It was trained for 100 epochs using stochastic gradient descent optimization, starting learning rate 0.01, momentum 0.937, weight decay 0.0005, and cosine annealing of the learning rate. Data augmentation during training will include random horizontal flips, mosaic augmentation, hue, saturation, and value changes, and random scaling. Input image size and data augmentation parameters need to be set based on the actual training configuration.

The training and inference were carried out on a base-station laptop which has an Intel Core i7-11800H CPU, 16 GB RAM, and a graphics processing unit NVIDIA GeForce RTX 3050 Laptop GPU with 4 GB of VRAM. The software used included Windows 11 Home, Python 3.10.12, PyTorch 2.1.0, CUDA 11.8, cuDNN 8.7.0, and Ultralytics YOLOv8 version 8.0.227. The inference was carried out in FP16 half precision with a batch size of 1. The trained model was embedded in the rover remote perception pipeline for the prediction of the terrain.

3.3 Hardware Integration and Specifications

The components in the rover include ESP32-WROOM-32, ESP32-CAM, ultrasonic sensor, infrared sensor, GPS, motor driver, and power supply. The ESP32-WROOM-32 is used as the primary controller for sensor acquisition, communication status, telemetry, and motor driver control. The ESP32-CAM is used as the vision module for taking images and transmitting MJPEG video to the base station laptop for classification.

Table 2. Hardware specifications of the autonomous rover platform

Subsystem	Component	Key Specifications	Function
Main Controller	ESP32-WROOM-32	Dual-core LX6, 240 MHz, 520 KB SRAM, Wi-Fi	Sensor acquisition, motor control, and telemetry
Vision Module	AI-Thinker ESP32-CAM, OV2640	2 MP camera, QVGA (320 × 240), 15–20 fps	Image capture and video streaming
Ultrasonic Sensor	HC-SR04	2–400 cm range, 40 kHz	Front obstacle detection
Cliff Sensors	Dual TCRT5000	Infrared sensing, digital output	Ground discontinuity detection
GPS Module	u-blox NEO-6M	1 Hz update rate, UART	Outdoor location tracking
Motor Drive	Dual L298N, six DC gearmotors	1:48 gear ratio, 150 RPM at 6 V	Six-wheel drive and differential steering
Power Supply	2S Li-ion battery, LM2596	7.4 V, 2600 mAh, 19.24 Wh; regulated 5 V output	Motor and electronic power supply

The HC-SR04 ultrasonic sensor detects any obstacle in front of the rover, while the double TCRT5000 sensors detect ground level disruptions that can be a sign of a cliff edge. The u-blox NEO-6M GPS module offers location data for outdoor navigation. There are six DC gear

motors that are driven by two L298N H-bridge driver modules.

A battery pack of 7.4 V and 2600 mAh is used for the power system with two cells of lithium ion. In order to regulate the 5 V supply for the controller, the camera module, and the sensors, an LM2596 buck converter is used; however, for the motor drive section, the battery voltage rail is utilized. The hardware components and their specifications are summarized in Table 2.

The ESP32-WROOM-32 runs the sensor polling loop at 20 Hz, keeps an eye on the wireless communication watchdog, serializes the telemetry data, and creates pulse-width modulation (PWM) signals for motor control. The ESP32-CAM makes use of its external PSRAM for storing images before sending them over the Wi-Fi. Image processing can now be decoupled from the embedded system control, so the laptop can do terrain recognition while the controller still controls the motors and sensors.

3.4 Software Framework and Communication

The software framework coordinates sensor acquisition, terrain classification, motor control, and wireless communication. The ESP32-WROOM-32 reads ultrasonic and infrared sensor data, monitors the communication link, and executes motor commands. The ESP32-CAM captures QVGA images at 320 × 240 pixels and transmits them to the base-station laptop through an HTTP-based Motion JPEG (MJPEG) stream over TCP port 81. The laptop processes the incoming frames using the trained YOLOv8n model and sends navigation commands to the ESP32 through a WebSocket connection on TCP port 8080.

The sensing data and operational state are encoded into JSON packets and sent at 10 Hz. The data comprises the operational mode, communication state, distance measurement via ultrasonics, infrared sensor data, latitude/longitude coordinates, battery level, terrain prediction data, confidence level, and motor PWM data. The packet structure is illustrated in Listing 1.

Listing 1. JSON telemetry payload transmitted through websocket

```
{
"timestamp": 1711200045,
"operating_mode": "REMOTE_ML",
"link_active": true,
```

"distance_us_cm": 42.5,

"ir_status": {

"cliff_left": false,

"cliff_right": false

},

"gps": {

"latitude": 12.96112,

"longitude": 80.05431,

"sats": 8

},

"battery_voltage_v": 7.42,

"terrain_prediction": "GRAVEL",

"confidence_score": 0.84,

"motor_pwm": {

"left": 178,

"right": 178

}

}

The web-based interface runs on the rover from http://192.168.4.1 while the rover is operating in Wi-Fi access point mode. It gives live video feeds, movement controls, an emergency stop control, speed control for the motors, and illumination control. This system is designed such that the rover can be remotely controlled while the telemetry is monitored.

3.5 Navigation and Fail-Safe Control

The navigation module uses terrain classification at the base station along with the sensor-based safety control system on board. During remote operation, the terrain classification

is done using the YOLOv8n model for the frames from the camera, and the traversability factor decides the motor speed depending on the type of terrain predicted. Factors of 0.95, 0.85, 0.70, and 0.50 are used for Grass, Soil, Gravel, and Sand, respectively.

The image of size 320 × 240 pixels is segregated into left, center, and right segments to evaluate the capability of the robot to navigate the environment. In case the center segment is traversable and there are no obstacles ahead on the path, the robot moves forward. However, in case the center segment is not suitable, the other two segments are analyzed to decide the direction to turn.

Table 3. Operational parameters and control thresholds

Parameter	**Symbol**	**Configured Value**
Communication timeout	T_{timeout}	1000 ms
Emergency stopping distance	d_{stop}	12 cm
Obstacle avoidance threshold	d_{safe}	25 cm
Turning duration	t_{turn}	450 ms
Base motor PWM	PWM_{base}	70% (178/255)
Turning motor PWM	PWM_{turn}	60% (153/255)
Terrain prediction confidence threshold	τ_{conf}	0.60
Ultrasonic sensor polling rate	f_{us}	20 Hz
Telemetry transmission rate	f_{telem}	10 Hz

ESP32 individually senses both ultrasonic distance readings and infrared cliff sensors. In case of any detected obstacle or danger of falling off the edge, the controller ignores the navigation signal and commands the braking motors. In case of loss of contact with the base station, the controller takes control of the rover and relies on ultrasonic readings to decide which direction to follow.

The operation thresholds and control settings have been summarized in Table 3. The parameters control communication loss detection, emergency stoppage, obstacle detection, and

telemetry transmission.

3.6 Physical Prototype

Integration of the constructed rover involves installation of the embedded microcontroller, the camera system, the sensor pack, motor control circuitry, and power supply in the six-wheeled chassis. With a high ground clearance, the six-wheeled chassis allows for the smooth navigation of the surface, while the ESP32-CAM located at the front end takes pictures of the terrain to allow remote terrain classification. The six DC gear motors enable differential steering, while the sensors are used in detecting obstacles and cliffs-edge monitoring. The rover prototype used for hardware integration and outdoor terrain-navigation experiments is shown in Figure 3.

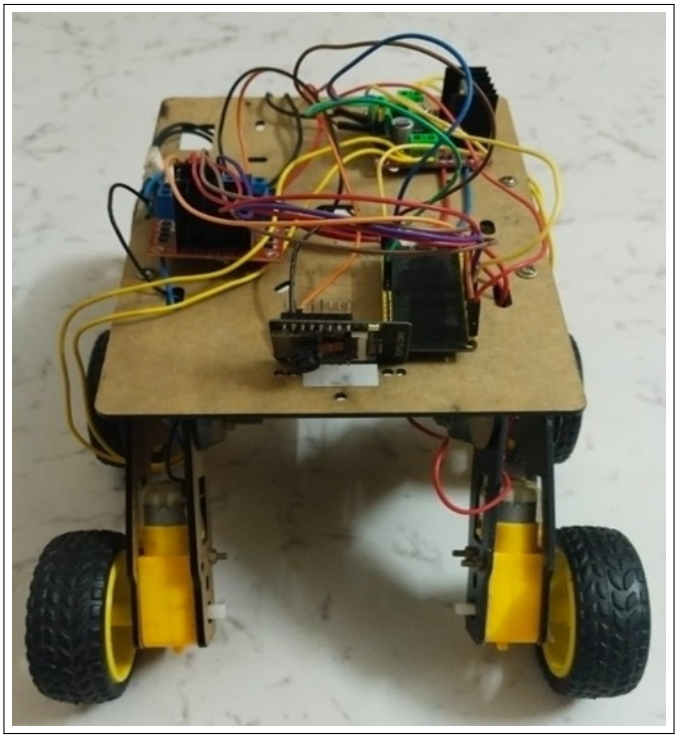


Figure 3. Rover prototype for rough-terrain navigation

Algorithm 1. Hybrid remote navigation and local fail-safe control

Input: Ultrasonic distance d_{front}, infrared cliff-sensor states, terrain prediction, confidence score, and communication status.

Output: Motor-control commands and rover telemetry.

1: Initialize ESP32 controller, sensors, Wi-Fi, motor drivers, and communication watchdog

Set operating mode to `REMOTE_ML`
while Rover is powered ON **do**
    Read ultrasonic distance d_{front}
    Read infrared cliff sensor states
    if $d_{\text{front}} \leq d_{\text{stop}}$ **OR** cliff detected **then**
        Brake motors
        Continue
    end if
    Check communication heartbeat
    if Communication timeout $\geq T_{\text{timeout}}$ **then**
        Set operating mode to `FAIL_SAFE`
    end if
    if Operating mode is `REMOTE_ML` **then**
        Receive terrain prediction and navigation command
        Receive confidence score τ
        if $\tau \geq \tau_{\text{conf}}$ **then**
            Calculate motor speed using terrain traversability factor
            Execute navigation command
        else
            Move forward at half the base motor speed
        end if
    else if Operating mode is `FAIL_SAFE` **then**
        if $d_{\text{front}} \leq d_{\text{safe}}$ **then**
            Turn right for duration t_{turn}
        else
            Move forward at configured fail-safe speed
        end if
    end if
    Transmit JSON telemetry containing:
     Sensor readings, operating status, GPS data,
     Terrain prediction, and motor values

31: Wait for configured telemetry transmission interval

32: **end while**

33: Stop rover when powered OFF

4. Results and Discussion

4.1 Experimental Protocol and Measurement Methodology

Outdoor field trials were carried out using four terrain test sites: grass, sand, gravel, and soil. Each terrain test site was made up of a 15.0 m x 2.0 m straight channel with natural surface variations and at least four to five fixed obstacles like wooden blocks, stones, and curbings. Fifteen separate trials were carried out on each terrain, totaling 60 trials.

The travel distance was taken by a calibrated mechanical surveyor wheel whose error range is ±0.05 m and checked by displacement using the GPS signal from the u-blox NEO-6M module. The time of traversal was recorded by the base station timer from the initiation of the movement till reaching the 15 m target line or getting immobilized. Collision occurred when contact with the obstacles led to chassis displacement or reached the threshold of 12 cm for emergency stop.

Communication delay was tested by the use of WebSocket ping-pong packets sent at 10 Hz using TCP port 8080. The delay for video transmission was computed by considering the time delay between the capture of the camera frame to its display on the base station. The performance test included the accuracy of terrain classification, success rate of navigation, distance covered, time to traverse, speed, collision rate, communication delay, time taken for fail-safe operation, and power consumption. Quantitative measurements from the 15 trials per terrain were reported as the sample mean and standard deviation.

4.2 Terrain Classification Performance

Performance of the terrain classifier was tested using the hold-out testing data set by calculating the accuracy, precision, recall, and F1 score for each terrain type. The results obtained are shown in Table 4.

Grass performed best in terms of classification because of its unique color distribution and plant texture. Sand was also able to be classified successfully because of its relatively

consistent visual appearance, although high levels of sunlight may have lessened the contrast of the surface. Gravel had more variety in terms of the color and shadows of its stones, which sometimes led to misclassification. Soil was the hardest class to classify because of its similarity to dry sand visually.

Table 4. Per-class terrain classification performance on the held-out test set

Terrain Category	**Test Samples**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Grass	30	86.7% (26/30)	86.7%	86.7%	86.7%
Sand	30	83.3% (25/30)	83.3%	83.3%	83.3%
Gravel	30	76.7% (23/30)	74.2%	76.7%	75.4%
Soil	30	66.7% (20/30)	69.0%	66.7%	67.8%
Overall (Macro Average)	120	78.3% (94/120)	78.3%	78.3%	78.3%

The results of classification are also shown in the confusion matrix given in Figure 4, which provides information about the classification result with respect to both correct and wrong prediction for all four types of terrain. The elements along the diagonal show correct classifications, whereas the remaining elements in the matrix correspond to incorrect prediction of the terrain. The pattern observed indicates that it is tough to distinguish visually similar surfaces like gravel and soil among other types of terrain.

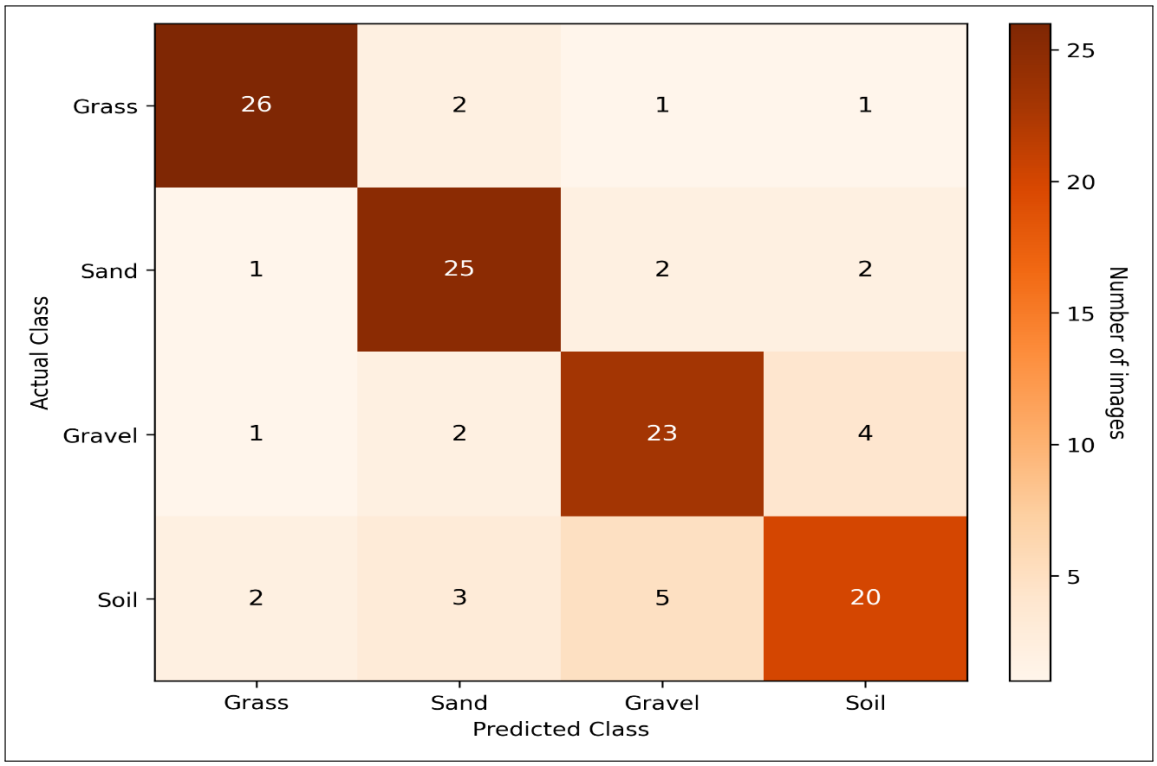


Figure 4. Confusion matrix for terrain classification on the held-out test set

4.3 Autonomous Navigation Performance

Table 5. Quantitative rover navigation performance across field test terrains

Terrain Testbed	Trials	Success Rate (%)	Mean Distance (m)	Mean Time (s)	Speed (m/s)	Collisions / Trial
Grass	15	86.7% (13/15)	14.2 +/- 0.8 m	48.6 +/- 4.2 s	0.29 m/s	0.13 +/- 0.35 (2)
Sand	15	73.3% (11/15)	12.8 +/- 1.6 m	62.4 +/- 6.8 s	0.21 m/s	0.33 +/- 0.49 (5)
Gravel	15	80.0% (12/15)	13.6 +/- 1.1 m	54.1 +/- 5.1 s	0.25 m/s	0.20 +/- 0.41 (3)
Soil	15	80.0% (12/15)	13.9 +/- 0.9 m	51.8 +/- 4.7 s	0.27 m/s	0.20 +/- 0.41 (3)

Overall / Mean	60	80.0% (48/60)	13.6 +/- 1.1 m	54.2 +/- 5.2 s	0.25 m/s	0.22 +/- 0.42 (13)

The performance of autonomous navigation is assessed in each of the four terrain testbeds based on traversal success, traversed distance, time taken, average speed, and number of collisions. The field trial results are presented in Table 5.

The robot showed its best navigation ability on grass because of the good surface traction that allowed it to move stably. Navigation on sand was difficult because of the lack of traction due to sinkage and slipping. The loose stones in gravel caused vibrations on the chassis and some interference to the sensors. Compacted soil offered enough traction, although there were some deviations in path on wet soil.

4.4 Communication, Fail-Safe Response, and Power Performance

Performance of communication, responsiveness to control signals, fail-safe behavior, and electrical power consumption were analyzed during trial operations. These analyses included video transmission, base station inference, command delivery, telemetry update, and on-board safety reactions. The outcomes are summarized in Table 6.

Closed-loop latency measurement reveals that the perception to action pipeline operates within the control limit set. Telemetry was sent in a constant stream, and any communication loss caused the on-board watchdog to initiate fail-safe autonomous operation mode. The ultrasonic braking system also acted as a local obstacle avoidance system independent of continuous wireless connection.

The power analysis indicated that the electronics used up less power than the electronics-motor combination during an actual crossing. From the measured operating power and the size of the battery, it is estimated that the continuous running time would be approximately 4.5 hours. The findings indicate the compromise that exists between responsiveness of control, local safety, and power usage.

Table 6. Communication, latency, fail-safe, and power performance

Performance Metric	**Experimental Result**
Video Streaming Latency	118 ± 14 ms
Inference Latency	34 ± 4 ms
Control Command Latency	18 ± 5 ms
Closed-Loop Latency	170 ± 23 ms
Telemetry Update Rate	10 Hz
Telemetry Packet Loss	0.65%
Fail-Safe Activation Time	1.02 ± 0.04 s
Emergency Braking Time	310 ± 45 ms
Electronics Power Consumption	1.42–1.85 W
Total Traversal Power	4.22 ± 0.35 W
Estimated Battery Runtime	4.5 hours

4.5 Browser-Based Interface and Integrated System Assessment

The browser-based UI is a platform where the user can control the rover, view its state and monitor it. The dashboard features the video stream, movement control, an emergency stop button, the ability to change the speed of the motors and control illumination. Information related to telemetry and system events is logged for post-flight analysis.

The overall system performance is summarized in Table 7, including terrain classification, autonomous navigation, fail-safe response, communication delay, and power consumption. The design involves remote visual processing and local obstacle detection that allows for terrain-guided navigation while providing a fail-safe response in case of wireless communication loss.

Table 7. Overall rover system performance summary

Performance Dimension	**Evaluated Metric**	**Observed Experimental Value**
Visual Perception	Terrain Classification Accuracy (Top-1)	65%–85% (Mean: 78.3%)
Visual Perception	Inference Processing Rate	29.4 fps (34 ms latency)
Autonomous Navigation	Corridor Traversal Success Rate	80.0% (48/60 trials)
System Robustness	Fail-Safe Activation on Link Severance	1.02 ± 0.04 s
System Robustness	Emergency Braking Response Time	310 ± 45 ms
Communication	Closed-Loop Latency (Camera $\rightarrow$ Motor)	170 ± 23 ms
Communication	Telemetry Packet Delivery Rate	99.35% (Packet loss 0.65%)
Energy Efficiency	Electronics Payload Power Draw	1.42 W – 1.85 W
Energy Efficiency	Total Operating Power & Battery Life	4.22 W / $\sim$4.5 hours runtime

4.6 Interface Implementation and Operational Demonstration

This GUI is the graphical user interface created for monitoring and control of the rover in action. Figure 5 shows the interface created that consists of the live camera view, movement controls, speed controls, and illumination control. This GUI enables the remote monitoring of the surrounding area of the rover and control of its operations.

The live camera feed provides visual observation capabilities during navigation, whereas the directional keys provide commands to move the rover. Adjusting motor speed allows for controlling the speed at which the commands are executed, and illumination control is useful for

operating the rover under different lighting conditions.

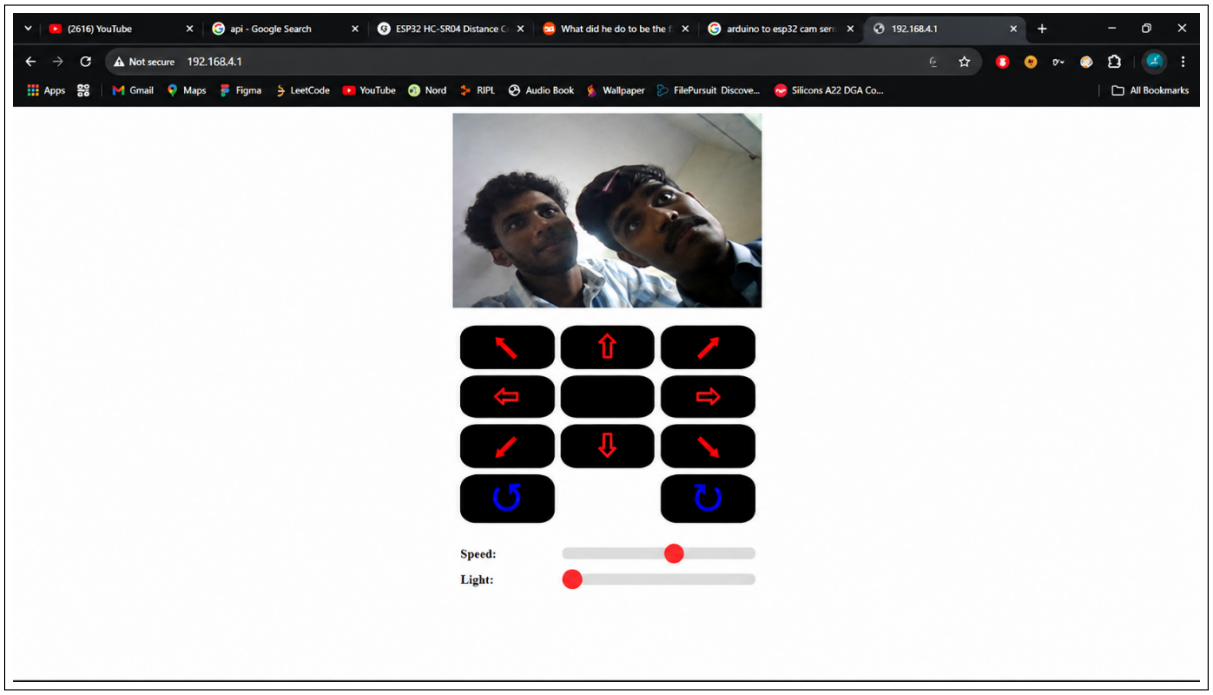


Figure 5. Browser-based rover interface for real-time monitoring and remote control

5. Conclusion and Future Work

This design consists of an autonomous rover that incorporates ML based terrain classification, motion control through embedded algorithms, and sensor-based safety features for autonomous navigation in complex terrains. The architecture has been designed such that the vision-based inference algorithm has been placed at the base station computer level, but the obstacle detection, cliff avoidance, and safety actions have been kept onboard, especially when there is a breakdown in communication. Future work will address enhancing the ability of terrain classification in different lighting conditions, increasing image variety used in training, and depth estimation for traversability assessment. Future research will explore adaptive path planning, onboard inference, better cliff detection, and efficient motor control. Extensive testing on varied terrain types and under different environmental conditions will allow thorough performance evaluation and practical application.

References

[1] Shilpa, D., S. Pavani, M. Triveni, P. Nandini, G. Venkateswarao, and M. Navya. "An ESP32-Based Autonomous Mobile Robot for Rough Terrain Navigation Using Multi-Sensor

Obstacle Avoidance and Hazard Detection." International Journal of AI Electronics and Nexus Energy, 2026, vol. 2, no. 2: 77-85.

[2] Zhao, Jianwei, Jianhua Fang, Shouzhong Wang, Kun Wang, Chengxiang Liu, and Tao Han. "Obstacle avoidance of multi-sensor intelligent robot based on road sign detection." Sensors, 2021, vol. 21, no. 20: 6777.

[3] Zhou, Lanfeng, Ziwei Liu, and Wenfeng Wang. "Terrain Classification Algorithm for Lunar Rover Using a Deep Ensemble Network with High-Resolution Features and Interdependencies between Channels." Wireless Communications and Mobile Computing, 2020, vol. 2020, no. 1: 8842227.

[4] Egan, Rebecca, and Ali Haydar Göktogan. "Deep learning-based terrain classification for traversability analysis, path planning and control of a Mars rover." In Australasian Conference on Robotics and Automation, 2021, 1-10.

[5] Zhang, Bo, Guobin Li, Qixin Zheng, Xiaoshan Bai, Yu Ding, and Awais Khan. "Path planning for wheeled mobile robot in partially known uneven terrain." Sensors, 2022, vol. 22, no. 14: 5217.

[6] Zhang, Qingfeng, Wenpeng Ma, Qingchun Zheng, Xiaofan Zhai, Wenqian Zhang, Tianchang Zhang, and Shuo Wang. "Path planning of mobile robot in dynamic obstacle avoidance environment based on deep reinforcement learning." IEEE Access, 2024, vol. 12: 189136-189152.

[7] Zim, Md Ziaul Haque. "TinyML: analysis of Xtensa LX6 microprocessor for neural network applications by ESP32 SoC." arXiv preprint arXiv:2106.10652, 2021.

[8] Gupta, Shubbham, and Shiv Naresh Shivhare. "Embedded tinyml for predictive maintenance: Vibration analysis on esp32 with real-time fault detection in industrial equipment." International Journal on Computational Modelling Applications, 2025, vol. 2, no. 2: 1-17.

[9] Ng, Yung Chuen, Qi Wen Shervina Lim, Chun Ye Tan, Zhen Hao Gan, and Meng Yee Michael Chuah. "Tactile Aware Dynamic Obstacle Avoidance in Crowded Environment with Deep Reinforcement Learning." arXiv preprint arXiv:2406.13434, 2024.

[10] Jocher, Glenn, Ayush Chaurasia, and Jing Qiu. "Ultralytics YOLOv8." Version 8.0.0, Ultralytics, 2023. Available: https://github.com/ultralytics/ultralytics.

[11] Off-Road Robotics Research Group. "Unstructured Rough Terrain Surface Classification Benchmark Dataset." GitHub Repository, 2023. Available: https://github.com/offroad-robotics