DESIGN AND FABRICATION OF MARS ROVER

A thesis

By

S. M. Rakib Hossen	BME2001020458
Md. Jahidul Islam	ME2202027114
Md Belal Hossain	ME2202027113
Nizam Howlader	ME2202027144

Supervisor: Md. Minhaz Uddin

Assistant Professor

Submitted to the

DEPARTMENT OF MECHANICAL ENGINEERING

SONARGAON UNIVERSITY (SU)

In partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING

JANUARY 2026

DECLARATION

We do hereby solemnly declare that, the work presented here in this project report has been carried out by us and has not been previously submitted to any University Organization for award of any degree or certificate

We hereby ensure that the works that has been prevented here does not breach any existing copyright.

We further undertake to indemnify the university against any loss or damage arising from breach of the foregoing obligation.

[Authors]

_ _ _ _ _ _ _ _ _ _ _ _ _ _

S. M. Rakib Hossen

ID: BME2001020458

_ _ _ _ _ _ _ _ _ _ _ _ _ _

Md. Jahidul Islam

ID: ME2202027114

_ _ _ _ _ _ _ _ _ _ _ _ _ _

Md Belal Hossain

ID: ME2202027113

_ _ _ _ _ _ _ _ _ _ _ _ _ _

Nizam Howlader

ID: ME2202027144

ACKNOWLEDGEMENT

First, we started in the name of almighty Allah. This thesis is accomplished under the supervision of Md. Minhaz Uddin, Assistant Professor, Department of Mechanical, Sonargaon University. It is a great pleasure to acknowledge our profound gratitude and respect to our supervisor for this consistent guidance, encouragement, helpful suggestion, constructive criticism and endless patience through the progress of this work. The successful completion of this thesis would not have been possible without his persistent motivation and continuous guidance.

The authors are also grateful to Prof. Md. Mostofa Hossain, Head of the Department of Mechanical Engineering and all respect teachers of the Mechanical Engineering Department for their co-operation and significant help for completing the thesis work successfully.

[Authors]

_ _ _ _ _ _ _ _ _ _ _ _ _ _

S. M. Rakib Hossen

ID: BME2001020458

_ _ _ _ _ _ _ _ _ _ _ _ _ _

Md. Jahidul Islam

ID: ME2202027114

_ _ _ _ _ _ _ _ _ _ _ _ _ _

Md Belal Hossain

ID: ME2202027113

_ _ _ _ _ _ _ _ _ _ _ _ _ _

Nizam Howlader

ID: ME2202027144

ABSTRACT

This project presents the design and development of a Mars Rover–inspired robotic system aimed at exploring harsh and unknown environments. The rover is built with a sturdy multi-leg and wheel-based mechanical structure, allowing it to move smoothly over uneven terrain while maintaining stability. An ESP32 microcontroller acts as the main control unit, managing sensor data, motor control, and wireless communication. The system integrates DC motors driven by an L298N motor driver to achieve precise movement in multiple directions.

To simulate planetary exploration tasks, the rover includes a soil moisture sensor for surface analysis and an ESP32-CAM module for real-time visual monitoring. Rechargeable 12V battery powers the system, while a buck converter ensures stable voltage distribution to sensitive electronic components. The circuit is designed to efficiently manage power, sensing, mobility, and communication in a compact form.

This Mars Rover project demonstrates how embedded systems, sensors, and mechanical design can work together to create an autonomous exploration platform. It serves as a practical learning model for space robotics, terrain navigation, and remote monitoring, while highlighting potential applications in planetary research, disaster inspection, and inaccessible area exploration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

CHAPTER - 1

INTRODUCTION

1.1 Introduction

Robotics technology is emanating at an expeditious pace, contributing new possibilities for automating tasks in many demanding applications, peculiarly in space explorations, military operations, underwater missions, etc. Especially, in space exploration, robotic devices are known as planetary rovers or simply rovers who aim at administering physical analysis of planetary terrains and astronomical bodies. The supervision of data about air pressure, climate, temperature, and other atmospheric aspects can be done by this advanced technology. It is a space exploration vehicle designed to move across the surface of a planet or other heavenly body. Primarily, rovers can be self-governing or can be remotely controlled from the ground stations called Remote Collaboration Center having very definite scientific objectives. Some rovers have been devised in the transportation of members of a human spaceflight crew; others have been partially or fully autonomous robots. Rovers or Unmanned Ground Vehicles (UGV) usually arrive at the planetary surface on a lander-style spacecraft.

The investigation of territories at the microscopic level, investigating the biological aspects of planetary surfaces, analyzing the composition of minerals, rocks, and soils, searching for liquid water in minerals, and measuring the ambient temperature, air pressure, and amount of dust in the landing site are some of its imperative appositeness. Therefore, rovers are eminently computing systems that use complicated embedded software and algorithms to handle computational and processing functionalities.

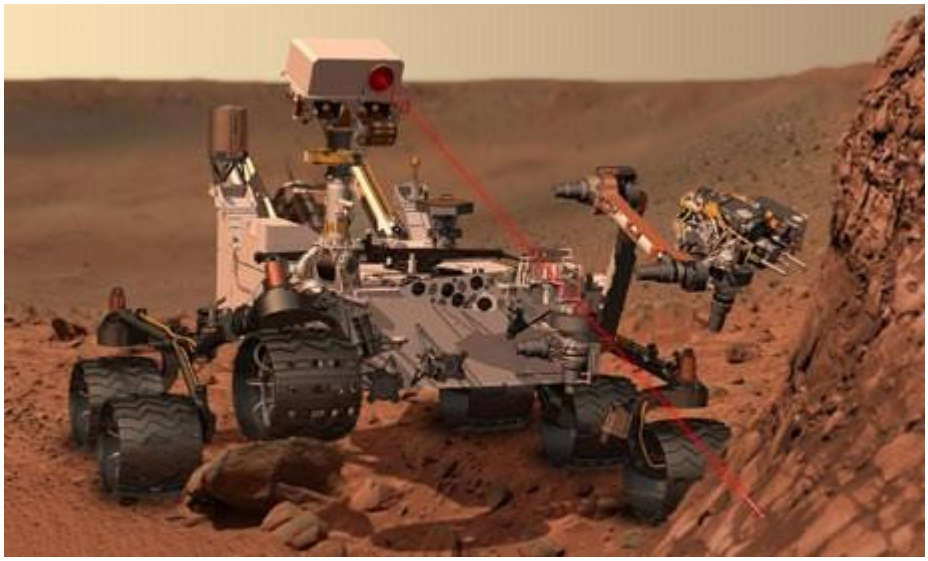

Figure 1.1 : Schematic of Curiosity Rover on Mars

An autonomous robot has the competence to:

- Collect environmental data, such as terrain details, soil moisture, and surface conditions.
- Detect objects of interest, such as rocks, obstacles, or uneven surfaces.
- Travel across rough terrain without direct human intervention using semi-autonomous navigation.
- Perform remote exploration tasks, such as analyzing soil or observing hazardous areas.
- Maintain operational functionality through efficient power management and system monitoring.

In this Mars Rover project, the rover is designed and developed using an ESP32 microcontroller as the main control unit, DC motors controlled via L298N motor drivers for precise navigation, and a rechargeable battery with a buck converter for stable voltage distribution. The system integrates sensors, including a soil moisture sensor, and an ESP32-CAM module for real-time visual monitoring and data collection. The rover is capable of semi-autonomous and wireless-controlled operations, demonstrating practical planetary exploration tasks, including terrain navigation, environmental sensing, and obstacle detection.

The ongoing Mars exploration focuses around surface exploration using satellites and small, autonomous land-exploration rovers, which are used in predilection to manned missions as the cost is substantially lower and the rovers are more suited to the harsh planetary environment. Manned missions are still regarded as the 'holy grail' of topographical exploration; a current rover mission is aiming to scrutinize Mars with a pair of rovers, providing scope for coordination which was hitherto only possible with human exploration. This project studies and demonstrates the design, development, and functionalities of a Mars Rover prototype, integrating autonomous navigation, sensing, and real-time monitoring capabilities.

1.2 Problem Statement

Exploration of distant planets, such as Mars, presents significant challenges due to harsh environmental conditions, uneven terrain, and the inability of humans to safely access these locations. Traditional rovers used in space missions are highly expensive, complex, and difficult to replicate for educational or small-scale research purposes. There is a growing need for a cost-effective, compact, and versatile robotic platform that can simulate planetary exploration tasks, including terrain navigation, environmental monitoring, and data collection.

Current robotic systems often lack integration between mobility, sensing, and real-time monitoring, limiting their practical applications in research, testing, and education. Additionally, remote monitoring of terrain conditions, such as soil properties and obstacles, is essential for autonomous exploration but is often not implemented in low-cost prototypes.

This project addresses these challenges by developing a Mars Rover-inspired system that combines robust mechanical design, sensor integration, camera-based monitoring, and wireless control, providing a functional and accessible solution for planetary exploration simulation, research, and learning purposes.

1.3 Objective

The objectives of this project are:

- To study design and construct a Mars Rover.
- To design and develop a stable rover capable of navigating uneven and rough terrains.

- To Integrate sensors, such as soil moisture sensors, to monitor environmental conditions.
- To implement a camera module (ESP32-CAM) for real-time visual monitoring and remote observation.
- To enable wireless control and data transmission using ESP32 microcontroller.
- To create a cost-effective and replicable platform for educational and research purposes in robotics and planetary exploration.
- To develop a compact embedded system that integrates mobility, sensing, and communication in one prototype.

1.4 Project Plan

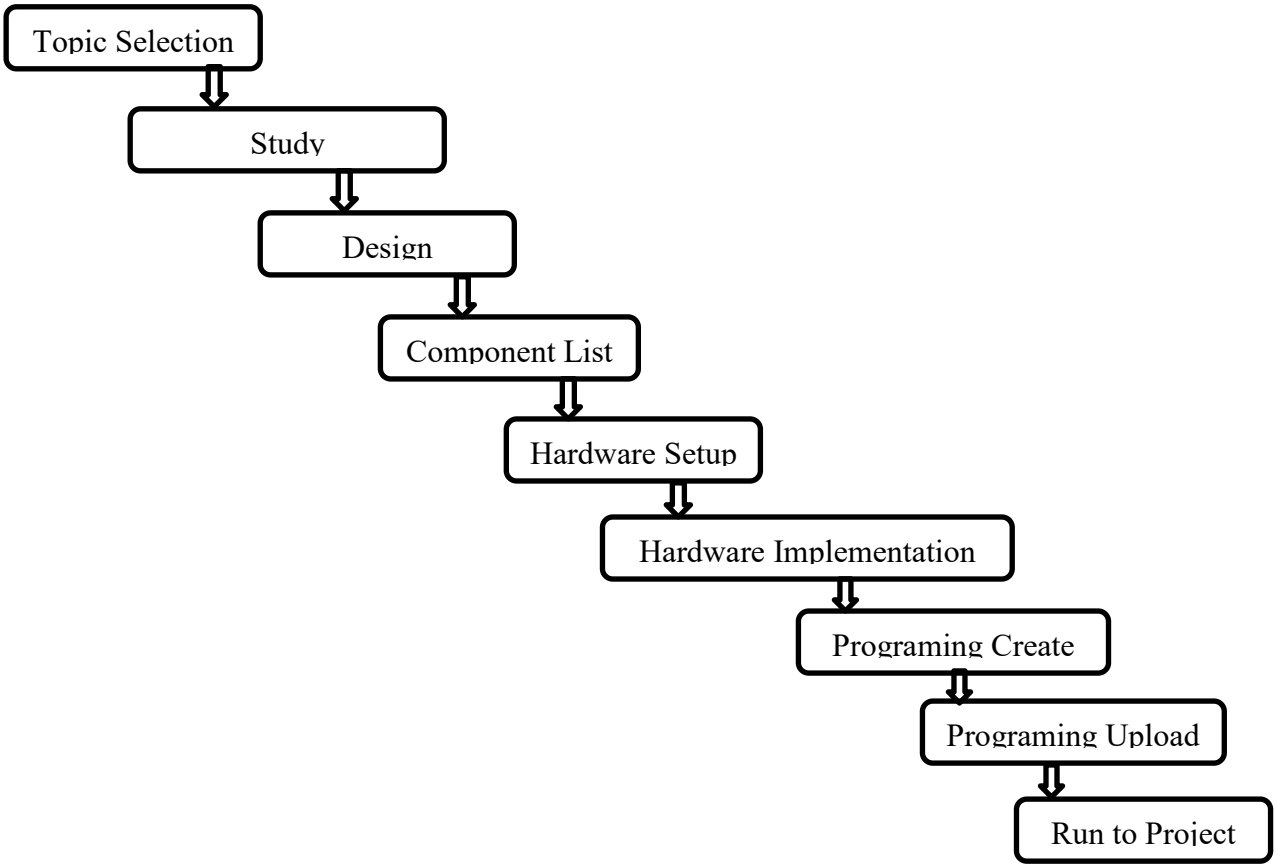

Figure 1.2: Project Implement Chart

1.5 Methodology

Our methodologies for the project:

- Creating an idea for design and construction of an **Mars Rover**. And designing a block diagram & circuit diagram to know which components we need to construct it.

- Collecting all the components and programming the micro controller to control the whole system.
- Setting up all the components in a PCB board & then soldering. Lastly, assembling all the blocks in a board and to run the system & for checking purposes.

1.6 Research Outline

This project book consists of Six chapter. The first chapter contains the statement of the introduction,Overview, Problem Statement, Work Flow Diagram, objectives of the study and Methodology. Chapter two contains system Literatur e Review details. Chapter three Discussed in Block Diagram,Circuit Diagram,Working principle and Cost analysis.Chapter Four describes the hardware implementation with component details and the software which we have used for our work. Chapter Five deals with the result Analysis,shows the complete prototype,Advantage, Application and Limitation of the project that we have built. In the final chapter, we have discussed the , Future Scope, conclusion of the project.

CHAPTER - 2

LITERATURE REVIEW

2.1 Introduction

In this chapter we discuss our system overview. In this paper they also work about Mars Rover. In below section we will describe some previous literature review. From previous literature we gather more knowledge to build this project and we successfully made it .

2.2 Literature Review

The literature survey on multi-terrain remote-operated Mars rovers reveals a comprehensive landscape of advancements and challenges in the field. Researchers have extensively explored sensing technologies, employing cameras, LiDAR, and spectrometers for terrain mapping, obstacle detection, and scientific analysis. Mobility systems have been a focal point, with studies delving into diverse wheel designs, suspension systems, and adaptive locomotion strategies to navigate the varied Martian terrains effectively. Communication protocols have evolved to facilitate reliable remote operation, addressing latency challenges and ensuring seamless data transmission between Earth and the Mars rover. The integration of autonomy and artificial intelligence has been a key research area, enhancing the rover's decision making capabilities in real-time scenarios. Energy management solutions, including solar panels and advanced batteries, have been explored to sustain prolonged missions on the Red Planet. Additionally, the literature highlights advancements in sample collection and analysis mechanisms, human-rover interaction interfaces, and addresses mission challenges posed by Martian conditions. Mission reports from previous rover missions offer valuable insights, and discussions on future developments underscore the continuous pursuit of innovation in multi-terrain remote operated Mars rovers [1].

The literature survey on multi-terrain remote-operated Mars rovers encompasses a comprehensive examination of research papers, academic articles, conference proceedings, and technical reports. It begins with an introduction to Mars exploration objectives,

followed by a review of existing rover designs such as NASA's Spirit, Opportunity, Curiosity, and Perseverance. The survey delves into multi-terrain navigation and mobility techniques, including wheel design, obstacle avoidance algorithms, and autonomous navigation. Remote operation and communication aspects are explored, focusing on tele operation, autonomy levels, and human-robot interaction interfaces. Additionally, scientific instrumentation and payloads are discussed, highlighting their integration with rover mobility. The survey includes literature on AI and machine learning integration for terrain classification and decision-making. Mission case studies provide insights into past and proposed Mars rover missions, evaluating mission architectures, operational challenges, and lessons learned. Finally, future trends and challenges are outlined, including advancements in robotics and prospects for international collaboration. Through this survey, a comprehensive understanding of multi-terrain remote operated Mars rover exploration is attained [2].

Elemental composition of manganese- and phosphorus rich nodules in the Knockfarril Hill member, Gale crater, Mars, The Mars Science Laboratory rover Curiosity encountered nodules rich in manganese and phosphorus while exploring the Knockfarril Hill member of Gale crater on Mars. Deconvolution of X-ray spectroscopy data acquired by the Alpha Particle X-ray Spectrometer (APXS) at the spectral level indicate P2O5 concentrations possibly in excess of 18 wt% and MnO exceeding 8 wt%. The nodules occur intermittently in ~ mm-thick layers concordant with the sedimentary laminae, extending up to ~ 10 cm laterally [3].

Toha and Zainol (2015) researched on an amphibious vehicle during the massive floods in Malaysia in December 2014. This paper contains study of the amphibious vehicle based on rocker bogie mechanism which can be operated in both water and land (terrain surface), that can be used by task Force for carrying aids to the needy. Hari et al. 2017) .The proposed modification increases in the stability margin and proved with valuable and profitable contrasting the static stability factor (SSF) metric with the 3D model simulations done in solid work. In future, if the system installed in heavy vehicles, it will surely decreases the complexity as well as power requirements to retain bumping within it. This is a wide field of study and is very less explored. Yang. et.al, (2014) Dr. Yang specializes in the study of stochastic simulation optimization problems, mobile robotic vehicles and job shop scheduling problems [4].

The human body, with its limited form and senses, can only explore a small portion of the physical world. However, this limitation is a handicap to the infinite potential of the human intellect, which strives to discover the ends of the world. Technology has removed this limitation by allowing man to explore to the deepest Seas and the farthest galaxies. Today most of the missions to deep-sea vents, distant galaxies and neighboring planets are handles using multifunctional Rover bots. The Mars Rover is a vehicle that has been designed to traverse the rugged terrains of Mars and collect samples of various items on Mar's surface. Scientists over the years have tried to explore the possibility of life on Mars. Such explorations have been mostly done using rovers. Hence rovers need to be specially designed to traverse all kinds of terrains and must be equipped with state-of heart technology. A common design element is most rovers over the years is the rocker bogie mechanism. The rocker bogie mechanism has quite a lot of advantages and is hence a well-established mechanism. The main advantage is that it ensures that all the wheels of the rover are in contact with the ground always. This advantage is key to creating a stable all terrain system. Consequently, the traction of the rocker bogie provides is equal and reliable allowing a smooth running even on the uneven terrains. Move over obstacles whose sizes are significantly larger than the wheel diameter because it makes use of an extra set of wheels to provide greater forward thrust. The extra set of wheels also divides the traction force required from each wheel to 1/6th the total value. Moreover, reduced forward thrust is required because the front wheels only need to lift 1/3 of the total weight of the rover. Acting together, the rear four wheels provide enough traction to keep the rover from slipping [5].

Depositional and Diagenetic Processes of Martian Lacustrine Sediments as Revealed at Pahrump Hills by the Mars Hand Lens Imager, Gale Crater, Mars, The Murray formation represents fine-grained sedimentary deposition in lacustrine environments within Gale crater, Mars. Both the overall thickness of the Murray formation and its broad uniformity in sedimentary character suggest the potential for a long-lived, groundwater supported lake system. Rock textures were imaged by the MAHLI camera at the Pahrump Hills location, which represents the lowermost Murray formation. We analyses data from Pahrump Hills to refine earlier estimates of grain size and grain size distribution, as well as to make detailed observations of diagenetic features and modification of primary sediment logical features. These observations and resulting interpretations provide a

detailed look at the dynamic behavior of lake systems on Mars. The lower portions of this exposure are characterized by planar laminated, fine-grained material; the predominant grain size in this region is smaller than very fine sand. Diagenetic mineral precipitation is also prominent in these lower layers, evidenced by likely inset precipitation of lenticular crystals, preferential cementation of lamina in several layers, precipitation of late diagenetic crystal clusters, and secondary modification of previously-deposited crystals [6].

A critical aspect of the literature survey for a multi-terrain remote-operated Mars rover involves investigating how mission planners prioritize scientific objectives and select exploration sites. This process integrates a range of considerations, including geological diversity, potential for past or present habitability, and accessibility for the rover. Scientific instruments and payload constraints play a crucial role in defining mission objectives, with instruments such as spectrometers, cameras, and drills enabling the rover to conduct detailed geological analyses and sample collection. Additionally, mission planners must account for operational factors such as power limitations, communication constraints, and rover mobility capabilities when designing exploration routes and selecting target locations. By analyzing case studies and mission reports from past Mars rover missions, valuable insights can be gained into the challenges and opportunities associated with achieving scientific goals in Martian environments, informing the planning and execution of future missions [7].

2.3 Summary

From the literature discussed above, we gained a lot of knowledge and we inspired to do this project. We were able to do it with everyone tireless work.

CHAPTER - 3

SYSTEM ARCHITECTURE

3.1 Introduction

The project objectives, methods, literature evaluation, and other details were all clarified in the preceding chapter. The block diagram, circuit diagram, Working principle, and final project instrument cost analysis will all be covered in this chapter.

3.2 Block Diagram

Here is the block diagram of the Mars Rover with all the essential components. All of the components are shown in below as a block in this diagram.Here we use Node MCU Micro-controller of our project. Other equipment's are attached with this micro-controller and work together for perform as our desire outcome.

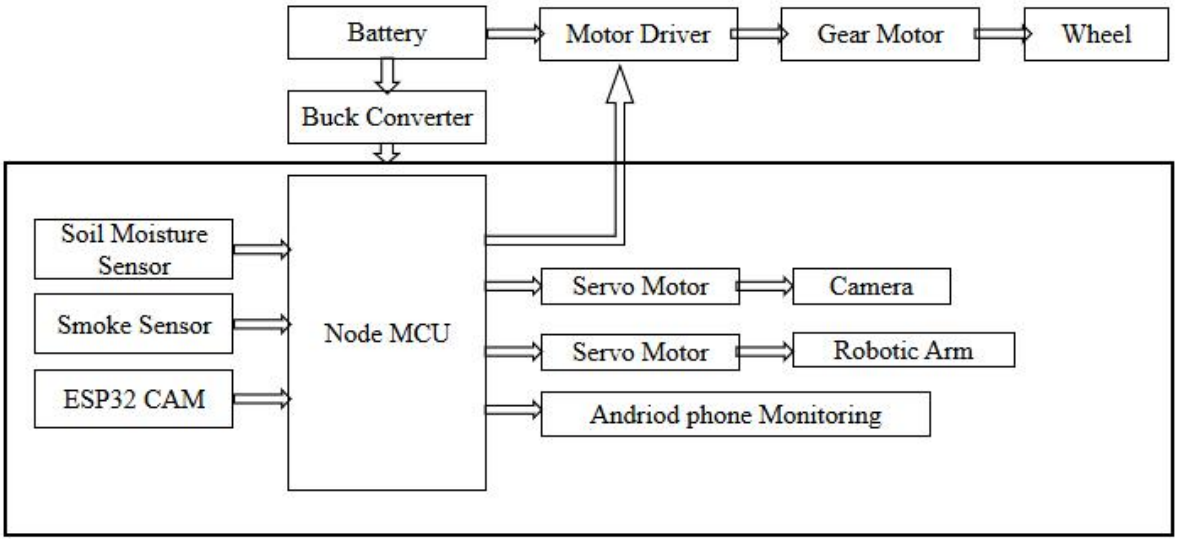

Figure 3.1: Block Diagram of Our System

3.3 Circuit Diagram

In this part we show our project circuit design and connect out instrument through standard wire.

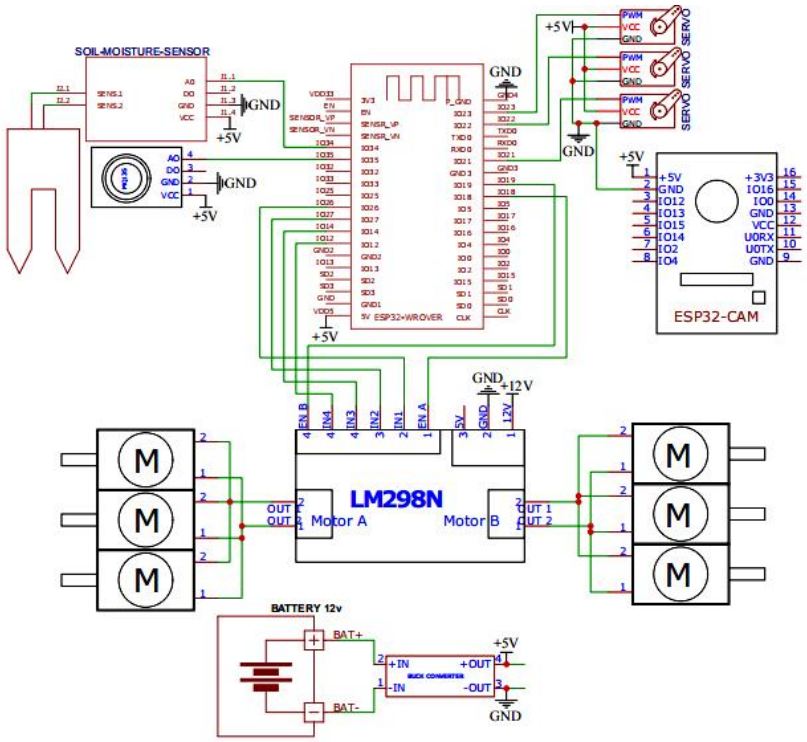

Figure 3.2: Schematic Diagram of Our System

3.4 Working Principle

The Mars Rover project operates using a battery as the main power source, supplying electrical energy to all system components. A buck converter is used to regulate and stabilize the voltage, ensuring safe and reliable power distribution to sensitive devices such as sensors, motors, and the ESP-32 CAM module. The ESP-32 CAM acts as the central control and monitoring unit, capturing live images and transmitting visual data wirelessly for remote observation.

Environmental analysis is performed using a soil moisture sensor to examine surface conditions and a smoke sensor to detect hazardous gases or dust-like conditions, simulating planetary atmosphere monitoring. DC motors connected through a motor driver control the movement of the wheels, enabling forward, reverse, and turning motions over uneven terrain.

A servo motor is used to rotate the camera, allowing adjustable viewing angles, while another servo motor controls the robotic arm for object handling or sample interaction. All components work together to achieve controlled navigation, environmental sensing, and real-time monitoring in a Mars-like exploration environment.

3.5 Cost Analysis

Table 1: List of Component with Price

Sl. no	Particulars	Specification	Qty.	Unit Price (Taka)	Total Price (Taka)
1	Node MCU	ESP 32	1	700	700
2	Battery	3.7v	9	100	900
3	Buck Converter	LM2596	1	200	200
4	Gear Motor	25ga	6	700	4200
5	Camera	ESP-32	1	800	800
6	Motor Driver	LM298N	1	400	400
7	Servo Motor	Mg90s	4	350	1400
8	Smoke Sensor	MQ2	1	250	250
9	Soil Moisture Sensor		1	300	300
10	Wheel		6	400	2400
11	Others				5000
				Total	16,550/=

CHAPTER - 4

HARDWARE & SOFTWARE ANALYSIS

4.1 Introduction

In this section, we will discuss elaborately about " **Multipurpose Agriculture Robot**" and the component description, features, working procedure of our all equipment.The system hardware fabricates composed of power unit, source unit, power store unite, and many more related components.

4.2 Components List

Hardware Part

- Node MCU
- Battery
- Buck Converter
- Gear Motor
- Motor Driver
- Soil Moisture Sensor
- Smoke Sensor
- Camera
- Servo Motor

Software Part

- Arduino IDE
- Easy EDA

4.3 Node MCU

Node MCU is an open source firmware for which open source prototyping board designs are available. The name "Node MCU" combines "node" and "MCU" (micro-controller unit). sand built on the Espressif Non-OS SDK for ESP32. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

Figure 4.1: Node MCU

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP32, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications. This an open source IoT platform. It includes firmware which runs on the ESP32 Wi-Fi SoC from Express if Systems, and hardware which is based on the ESP-12 module. The term "Node MCU" by default refers to the firmware rather than the development kits. The firmware uses the Luascripting language. It is based on the eLua project, and built on the Es press if Non-OS SDK for ESP32. Node MCU was created shortly after the ESP32 came out.

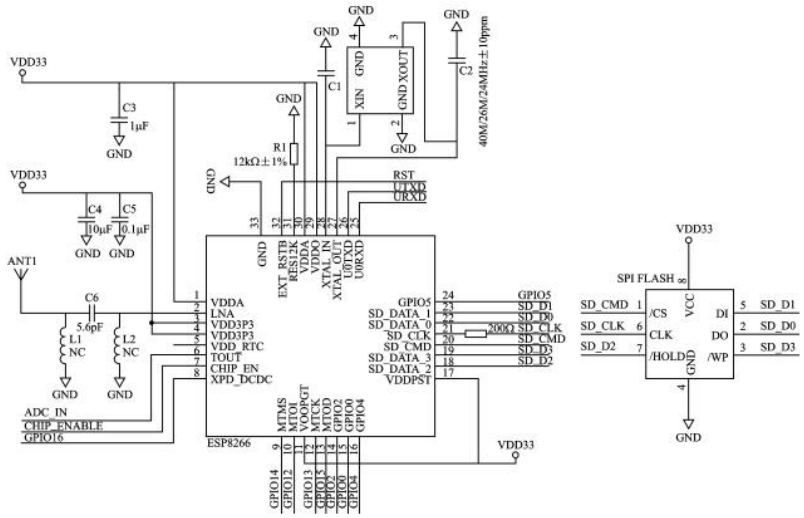

Figure 4.2: Node MCU Schematic Diagram

Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glibto Node MCU project, enabling Node MCU to easily drive LCD, Screen, OLED, even VGA displays. In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over. By summer 2016 the Node MCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

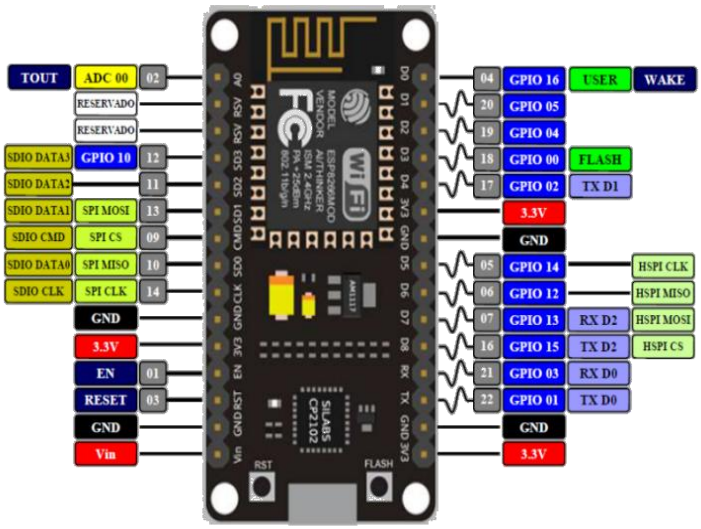

Figure 4.3: Node MCU Pin Out

Node MCU V3 ESP32 ESP-12E is Wi-Fi development board that helps you to prototype your IoT product with few Lua script lines, or through Arduino IDE. The board is based on ESP32 ESP-12E variant, unlike other ESP-12E, you won't need to buy a separate breakout board, usb to serial adapter, or even solder it to a PCB to get started, you will only need a usb cable (Micro USB).

Features

- Communication interface voltage: 3.3V.
- Antenna type: Built-in PCB antenna is available.
- Wireless 802.11 b/g/n standard
- WiFi at 2.4GHz, support WPA / WPA2 security mode
- Support STA/AP/STA + AP three operating modes

- Built-in TCP/IP protocol stack to support multiple TCP Client connections (5 MAX)
- D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA
- AD0: 1 channel ADC
- Power input: 4.5V ~ 9V (10VMAX), USB-powered
- Current: continuous transmission: ≈70mA (200mA MAX), Standby: <200uA
- Transfer rate: 110-460800bps
- Support UART / GPIO data communication interface
- Remote firmware upgrade (OTA)

4.4 Battery

Lithium batteries are primary batteries that have metallic lithium as an anode. These types of batteries are also referred to as lithium-metal batteries. They stand apart from other batteries in their high charge density and high cost per unit. Depending on the design and chemical compounds used, lithium cells can produce voltages from 1.5 V (comparable to a zinc–carbon or alkaline battery) to about 3.7 V.

Disposable primary lithium batteries must be distinguished from secondary lithium-ion or a lithium-polymer, which are rechargeable batteries. Lithium is especially useful, because its ions can be arranged to move between the anode and the cathode, using an intercalated lithium compound as the cathode material but without using lithium metal as the anode material. Pure lithium will instantly react with water, or even moisture in the air; the lithium in lithium ion batteries is in a less reactive compound.

Figure 4.4: 3.7V Battery

Lithium batteries are widely used in portable consumer electronic devices. The term "lithium battery" refers to a family of different lithium-metal chemistry, comprising many types of cathodes and electrolytes but all with metallic lithium as the anode. The battery requires from 0.15 to 0.3 kg of lithium per kWh. As designed these primary systems use a charged cathode, that being an electro-active material with crystallographic vacancies that are filled gradually during discharge.

Product Specification

Table 2: 3.7V Battery Specification

Voltage	3.7 V
Product Type	Lithium-Ion
Battery Capacity	2200mAh
Weight	45 g
Model Number	ICR 18650

4.5 Buck Converter

A buck converter (step-down converter) is a DC-to-DC power converter which steps down voltage (while drawing less average current) from its input (supply) to its output (load). It is a class of switched-mode power supply (SMPS) typically containing at least two semiconductors (a diode and a transistor, although modern buck converters frequently replace the diode with a second transistor used for synchronous rectification) and at least one energy storage element, a capacitor, inductor, or the two in combination. To reduce voltage ripple, filters made of capacitors (sometimes in combination with inductors) are normally added to such a converter's output (load-side filter) and input (supply-side filter). It is called a buck converter because the voltage across the inductor “bucks” or opposes the supply voltage.

Figure 4.5: DC -DC Buck Converter

DC-DC Buck Converter Step Down Module LM2596 Power Supply is a step-down(buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version. The LM2596 series operates at a switching frequency of 150kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators.

Specifications of DC-DC Buck Converter Step Down Module LM2596 Power Supply

- Conversion efficiency: 92%(highest)
- Switching frequency: 150KHz
- Output ripple: 30mA9maximum)
- Load Regulation: ± 0.5%
- Voltage Regulation: ± 0.5%
- Dynamic Response speed: 5% 200uS
- Input voltage:4.75-35V
- Output voltage:1.25-26V(Adjustable)
- Switching Frequency: 150KHz
- Rectifier: Non-Synchronous Rectification
- Module Properties: Non-isolated step-down module (buck)
- Short Circuit Protection: Current limiting, since the recovery
- Operating Temperature: Industrial grade (-40 to +85) (output power 10W or less)

4.6 Soil Moisture Sensor

The soil moisture sensor is one kind of sensor used to gauge the volumetric content of water within the soil. As the straight gravimetric dimension of soil moisture needs eliminating, drying, as well as sample weighting. These sensors measure the volumetric water content not directly with the help of some other rules of soil like dielectric constant, electrical resistance, otherwise interaction with neutrons, and replacement of the moisture content.The relation among the calculated property as well as moisture of soil should be adjusted & may change based on ecological factors like temperature, type of soil, otherwise electric conductivity.

Soil Moisture Sensor Pin Configuration

- VCC pin is used for power.
- A0 pin is an analog output.
- D0 pin is a digital output.
- GND pin is a Ground.

Working Principle

This sensor mainly utilizes capacitance to gauge the water content of the soil (dielectric permittivity). The working of this sensor can be done by inserting this sensor into the earth and the status of the water content in the soil can be reported in the form of a percent.This sensor makes it perfect to execute experiments within science courses like environmental science, agricultural science, biology, soil science, botany, and horticulture.

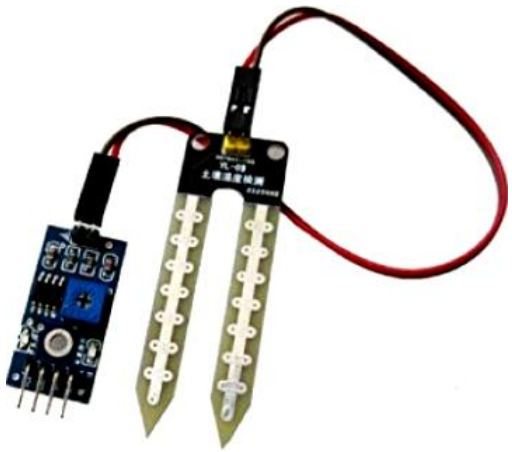

Figure 4.6: Soil Moisture Sensor

Specifications

The specification of this sensor includes the following:

- The required voltage for working is 5V.
- The required current for working is <20mA
- Type of interface is analog.
- The required working temperature of this sensor is 10°C~30°C.

4.7 Gear Motor

A DC motor is any motor within a class of electrical machines whereby direct current electrical power is converted into mechanical power. A 12v DC motor is small and inexpensive, yet powerful enough to be used for many applications.

Figure 4.7: DC Gear Motor

Specification

- Voltage: 12V DC
- Gear ratio: 1/31
- No-load speed: 200 RPM
- Rated Speed: 140 RPM
- Rated torque: 10 kg.cm
- Rated current: 2.5 Amp
- Length of Motor(including spindle): 106 mm/4.17″

- Diameter: 37 mm/1.45″
- Shaft length: 21 mm/0.82″
- Shaft diameter: 6 mm/0.24″

4.8 ESP32- CAM

The ESP32-CAM is a full-featured microcontroller that also has an integrated video camera and micro-SD card socket. It's inexpensive and easy to use, and is perfect for IoT devices requiring a camera with advanced functions like image tracking and recognition.

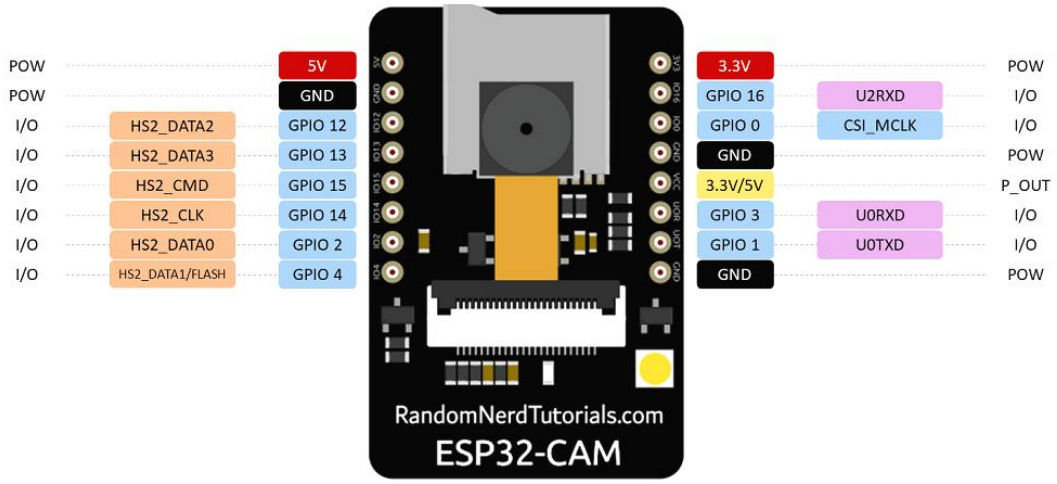

Figure 4.8: ESP-32 CAM Pin out diagram

ESP32-CAM Specifications

The ESP32-CAM is based upon the ESP32-S module, so it shares the same specifications. It has the following features:

- 802.11b/g/n Wi-Fi
- Bluetooth 4.2 with BLE
- UART, SPI, I2C and PWM interfaces
- Clock speed up to 160 MHz
- Computing power up to 600 DMIPS
- 520 KB SRAM plus 4 MB PSRAM
- Supports Wi-Fi Image Upload
- Multiple Sleep modes
- Firmware Over the Air (FOTA) upgrades possible

- 9 GPIO ports
- Built-in Flash LED

4.9 Gas Sensor

The utility model can be used for gas leakage monitoring devices in families and factories, and is suitable for the detection of liquefied petroleum gas, butane, propane, methane, Hydrogen, smoke, etc. This is a very easy to use low cost semiconductor Gas sensor Module with analog and digital output.

Figure 4.9: MQ 2 Gas Sensor

Features

- Adopt high quality double panel design, with power indication and TTL signal output indication.
- It has DO switch signal (TTL) output and AO analog signal output.
- TTL output valid signal is low level. When the output is low, the signal light is on, and the micro-controller or relay module can be directly connected.
- The analog output voltage increases with the concentration, the higher the voltage.
- It has better sensitivity to liquefied petroleum gas, natural gas, urban gas and smoke.
- MQ-2 Gas LPG Butane Methane Sensor Detector Module
- With four screw holes, easy to locate.
- Product size: 32 (L), *20 (W), *22 (H)
- With long service life and reliable stability.
- Fast response recovery features

Specifications

- Input voltage: DC5V
- Power dissipation (current): 150mA
- DO output: TTL, numeric quantities 0 and 1 (0.1 and 5V)
- AO output: 0.1-0.3V (relatively pollution-free), the highest concentration of about 4V voltage
- Special reminder: after the sensor is energized, you need to preheat 20S or so, the data to be stable, sensor heating is a normal phenomenon, because the internal heating wire, if hot, it is not normal.

Connection mode

- VCC: power supply positive (5V)
- GND: power supply negative pole
- DO:TTL switch signal output
- AO: analog signal output
- Functions: This version supporting test procedures
- Using chips: AT89S52
- Crystal oscillator: 11.0592MHZ
- Since this Gas Sensor module is sensitive to smoke it can be used in for fire detection. MQ135 Gas Sensor is also sensitive to flammable/combustible gasses like LPG, Propane & Hydrogen.
- Baud rate: 9600

4.10 Motor Driver

The LM298N is a popular 16-Pin Motor Driver IC. As the name suggests it is mainly used to drive motors. A single LM298N IC is capable of running two DC motors at the same time; also the direction of these two motors can be controlled independently.

Figure 4.10: Motor driver

Working Process

LM298N IC is a typical Motor Driver IC which allows the DC motor to drive on any direction. This IC consists of 16-pins which are used to control a set of two DC motors instantaneously in any direction. It means, by using a LM298N IC we can control two DC motors.

Features

- Can be used to run Two DC motors with the same IC.
- Speed and Direction control is possible
- Motor voltage Vcc2 (Vs): 4.5V to 36V
- Maximum Peak motor current: 1.2A
- Maximum Continuous Motor Current: 600mA
- Supply Voltage to Vcc1(vss): 4.5V to 7V
- Transition time: 300ns (at 5Vand 24V)
- Automatic Thermal shutdown is available

LM298N Pin Configuration

Table 3: Motor Driver Pinout

Pin Number	Pin Name	Description
1	Enable 1,2	This pin enables the input pin Input 1(2) and Input 2(7).
2	Input 1	Directly controls the Output 1 pin, Controlled by digital circuits.
3	Output 1	Connected to one end of Motor 1.
4	Ground	Ground pins are connected to ground of circuit (0V).
5	Ground	Ground pins are connected to ground of circuit (0V).
6	Output 2	Connected to another end of Motor 1.
7	Input 2	Directly controls the Output 2 pin, Controlled by digital circuits.
8	Vcc2 (Vs)	Connected to Voltage pin for running motors (4.5V to 36V).
9	Enable 3,4	This pin enables the input pin Input 3(10) and Input 4(15).
10	Input 3	Directly controls the Output 3 pin, Controlled by digital circuits.
11	Output 3	Connected to one end of Motor 2.
12	Ground	Ground pins are connected to ground of circuit (0V).
13	Ground	Ground pins are connected to ground of circuit (0V).
14	Output 4	Connected to another end of Motor 2.
15	Input 4	Directly controls the Output 4 pin, Controlled by digital circuits.
16	Vcc2 (Vss)	Connected to +5V to enable IC function.

Use of a L298N Motor Driver IC

Using this L298N motor driver IC is very simple. The IC works on the principle of Half H-Bridge, let us not go too deep into what H-Bridge means, but for now just know that H bridge is a set up which is used to run motors both in clock wise and anti clockwise direction. As said earlier this IC is capable of running two motors at the any direction at the same time, the circuit to achieve the same is shown below.

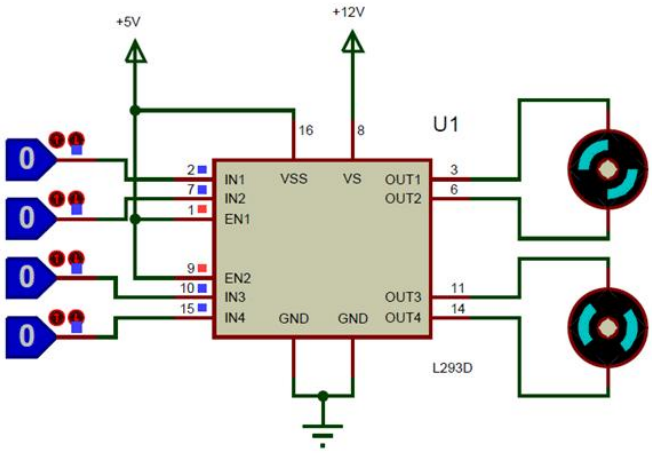

Figure 4.11: LM298N circuit Diagram

All the Ground pins should be grounded. There are two power pins for this IC, one is the Vss(Vcc1) which provides the voltage for the IC to work, this must be connected to +5V. The other is Vs(Vcc2) which provides voltage for the motors to run, based on the specification of your motor you can connect this pin to anywhere between 4.5V to 36V, here I have connected to +12V.

The Enable pins (Enable 1,2 and Enable 3,4) are used to Enable Input pins for Motor 1 and Motor 2 respectively. Since in most cases we will be using both the motors both the pins are held high by default by connecting to +5V supply. The input pins Input 1,2 are used to control the motor 1 and Input pins 3,4 are used to control the Motor 2. The input pins are connected to the any Digital circuit or micro-controller to control the speed and direction of the motor.

4.11 Servo Motor

A servo motor is an electrical device that can push or rotate an object with great precision. If you want to rotate and object at certain angles or distances, you use servo motors. It is made by a simple motor which is driven by a servo mechanism. If the motor is DC driven then it is called DC servo motor and if it is AC driven motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages. These are being used in various applications like toy cars, RC helicopters and planes, robotics, machines etc.

Figure 4.12: Servo Motor

Servo motors are rated at kg / cm (centimeters per kilogram). Most hobby servo motors are rated at 3 kg / cm or 6 kg / cm or 12 kg / cm. This kg / centimeter tells you how much weight your servo motor can lift at a certain distance. For example: A 6 kg / cm servo motor should be able to lift 6 kg if the load is suspended 1 cm away from the motor shaft, to be less than the carrying capacity. The position of the servo motor is determined by the electric pulse and its circuitry is placed next to the motor.

Servo Mechanism

It consists of three parts:

i. Controlled device
ii. Output sensor
iii. Feedback system

All motors have three wires. Of which two will be used for the supply (positive and negative) and one will be used for the signal transmitted from the MCU. The servo motor is controlled by PWM (Pulse with Modulation) which provides the control wires. Has minimum pulse, maximum pulse and repetition rate. The servo motor can rotate 90 degrees in two directions, creating its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the pulse length will determine how far the motor rotates. For example, a 1.5 mm pulse will turn the motor to 90 °, for example, if the branch is less than 1.5 mm shaft, it will move to 0 to, and if it is longer than 1.5 mm, the sardo will turn to 180 to.

The servo motor operates on the PWM (pulse width modulation) principle, meaning that the angle of rotation is controlled by the duration of the pulse applied to its control pin. Basically the servo motor is made up of a DC motor which is controlled by a variable resistor (potentiometer) and some gears. The high speed force of the DC motor is converted to torque by the gears. We know that work = force x DISTANCE, less in DC motor force and more in distance (speed) and in servo, forces are higher and distance is less. The pentiometer server is connected to the output trench to calculate the angle and stop the DC motor at the required angle.

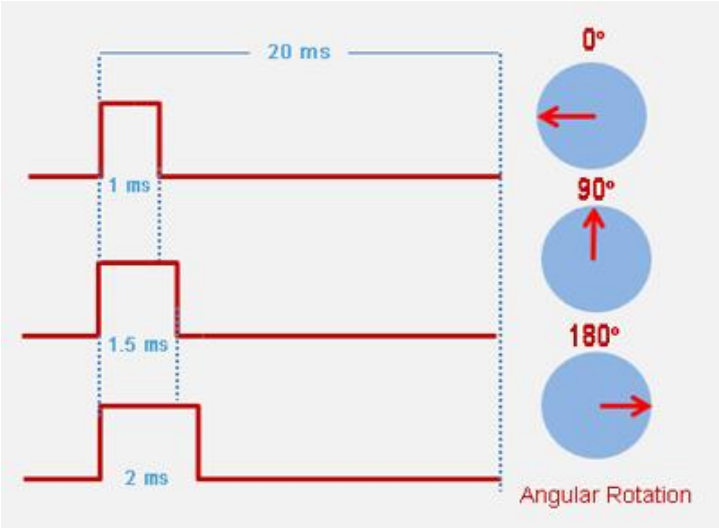

Figure 4.13: Schematic Diagram of servo motor

The servo motor can be rotated from 0 to 180 degrees but depending on the output it can go up to 210 degrees. This degree of rotation can be controlled by applying an electric pulse of appropriate width to its control pin. Servo examines pulses every 20 milliseconds. The 1 ms (1 millisecond) width pulse servo can rotate 0 degrees, 1.5 mm 90 degrees (neutral position) and 2 ms pulse it can rotate 180 degrees. All servo motors work directly with your + 5V supply rails but we need to be careful about how much current the motor will use. If you plan to use more than two servo motors, you should design a suitable servo solder

SOFTWARE

4.12 Arduino IDE

The digital microcontroller unit named as Arduino UNO can be programmed with the Arduino software IDE. There is no any requirement for installing other software rather than Arduino. Firstly, Select "Arduino UNO from the Tools, Board menu (according to the microcontroller on our board). The IC used named as ATmega328 on the Arduino UNO comes pre burned with a boot loader that allows us to upload new code to it without the use of an external hardware programmer.

Communication is using the original STK500 protocol (reference, C header files). We can also bypass the boot loader and programs the microcontroller through the ICSP (In Circuit Serial Programming) header. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.The Arduino UNO is one of the latest digital microcontroller units and has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL at (5V) with serial communication, which is available on digital pins 0 -(RX) for receive the data and pin no.1 (TX) for transmit the data. An

ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an .in file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board.

The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial Communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. Arduino programs are written in C or C++ and the program code written for Arduino is called sketch. The Arduino IDE uses the GNU tool chain and AVR Lab to compile programs, and for uploading the programs it uses argued. As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

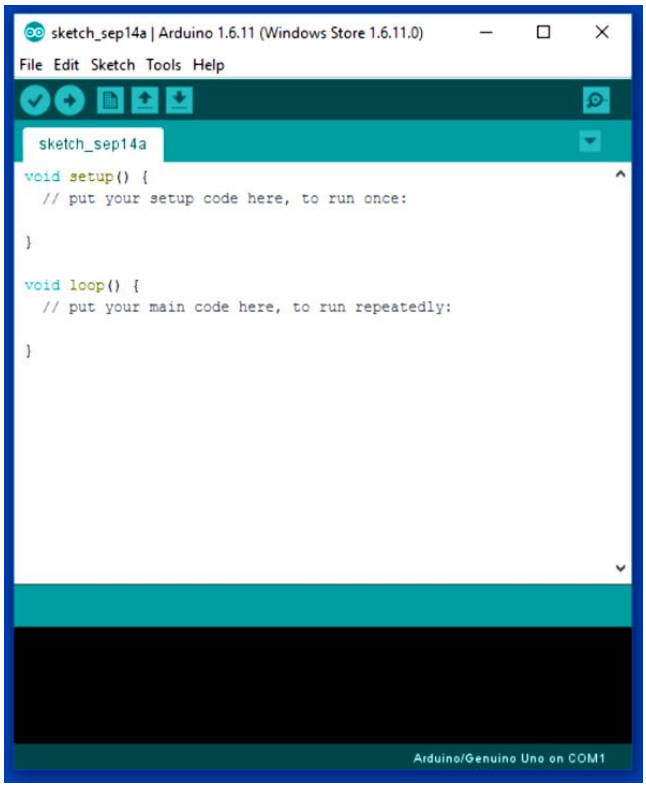

Figure 4.14: Arduino Software Interface IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog. Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is

probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino boot loader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The boot loader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

4.13 Easy EDA

Easy EDA is a web-based EDA tool suite that enables hardware engineers to design, simulate, share-publicly and privately-and discuss schematics, simulations and printed circuit boards. Other features include the creation of a bill of materials, Gerber files and pick and place files and documentary outputs in PDF, PNG and SVG formats. Easy EDA allows the creation and editing of schematic diagrams, SPICE simulation of mixed analogue and digital circuits and the creation and editing of printed circuit board layouts and, optionally, the manufacture of printed circuit boards.

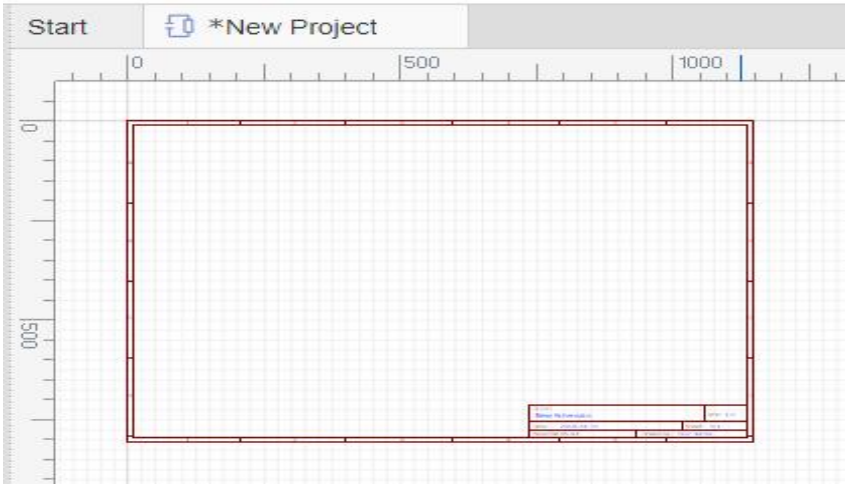

Figure 4.15: Easy EDA Software Interface

Subscription-free membership is offered for public plus a limited number of private projects. The number of private projects can be increased by contributing high quality public projects, schematic symbols, and PCB footprints and/or by paying a monthly subscription.Registered users can download Gerber files from the tool free of charge; but for a fee, Easy EDA offers a PCB fabrication service. This service is also able to accept Gerber file inputs from third party tools.

CHAPTER - 5

RESULT ANALYSIS

5.1 Project Outcome

After finally completing this project, we ran it & we observed the output of this project. We can see that it is working well as expected. After making our project we observe it very careful. It works as we desire. Our project give output perfectly and all equipment are work perfectly. We check how much it works and we get perfect output from this project.

- The rover successfully navigated uneven terrain using motor-driven wheels with stable control.
- Soil moisture sensor effectively monitored surface conditions during exploration.
- Smoke sensor accurately detected hazardous gas or dust-like environments.
- ESP-32 CAM provided real-time visual monitoring of the surroundings.
- Camera movement through a servo motor enabled adjustable viewing angles.
- The robotic arm servo motor performed smooth and controlled object handling actions.
- IoT-based monitoring was successfully achieved using ESP32 AP Mode without internet dependency.
- Live camera feed and sensor data were accessed wirelessly through a local network.
- The system demonstrated reliable remote monitoring and control in Mars-like conditions.

5.2 Complete Project Prototype

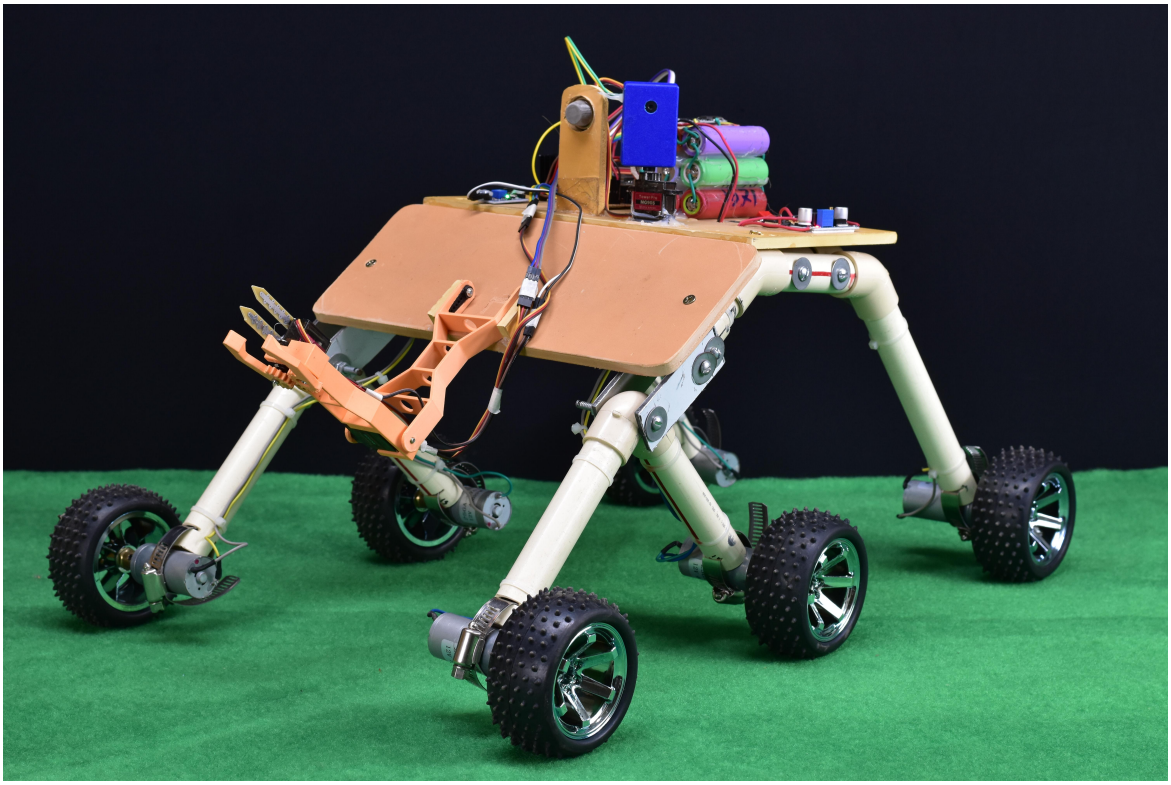

Figure 5.1 : Our Final project

5.3 Advantage

There are many advantages of the project. Some of these are given below:

- Easy to use.
- Lowered Operation Costs
- Efficient and Saves Time
- Enables safe exploration of harsh and inaccessible environments.
- Provides real-time visual feedback through ESP-32 CAM.
- Allows environmental analysis using soil moisture and smoke sensors.
- Offers flexible camera movement and object handling using servo motors.
- Demonstrates effective integration of robotics, sensors, and IoT technology.
- Cost-effective and suitable for educational and research applications.

5.4 Limitation

Through the project has many advantages it has also some limitation, they are given below:

- The system needs to be more efficient to produce more energy.
- Used cheap Chinese products for the prototype so there's some processing delay present in the circuit
- To get all notification mobile and project must be connected with the internet.
- This project can now be only used for small scale purposes.

5.5 Application

This project has applications in many fields due its necessity. We have selected a few of them and they are given below:

- Planetary surface exploration and research simulation.
- Remote monitoring of hazardous or inaccessible environments.
- Disaster management and search operations in risky areas.
- Military and defense surveillance in dangerous zones.

5.6 Discussion

While working on our project, we did face some difficulties as it is a very complex system but the end results, we came up with were quite satisfactory. We have put the whole system through several tasks to validate our work and also have taken necessary notes for future improvements. Some future recommendations that we have involves improvement in system design and wiring, adding features for more efficient.

This Mars Rover project demonstrates the practical integration of robotics, sensors, and IoT technology for exploration in challenging environments. The system effectively combines mobility, environmental sensing, and real-time visual monitoring using the ESP32 platform. AP Mode–based IoT monitoring enables wireless operation without internet dependency, making the rover suitable for remote and isolated locations. Stable power management through a buck converter ensures reliable performance of all components. The use of servo motors for camera movement and robotic arm control enhances observation and interaction capabilities. Overall, the project serves as a cost-effective and educational prototype for understanding planetary exploration concepts, autonomous systems, and embedded control, with potential applications in research, disaster response, and hazardous environment monitoring.

CHAPTER - 6

CONCLUSION

6.1 Conclusion

The Mars Rover project successfully demonstrates the design and implementation of a robotic exploration system capable of operating in harsh and unknown environments. By integrating mobility, environmental sensing, and IoT-based monitoring, the rover provides a practical simulation of planetary exploration missions. The use of an ESP32 CAM enables real-time visual monitoring, while sensors such as soil moisture and smoke sensors allow effective analysis of surface and atmospheric conditions. Stable power distribution through a buck converter ensures reliable performance of all electronic components.

The rover's movement is efficiently controlled using motor drivers and wheels, and servo motors enhance functionality by enabling camera rotation and robotic arm operation. AP Mode–based IoT monitoring allows wireless control without relying on internet connectivity, making the system suitable for remote locations. Overall, this project serves as a cost-effective, educational, and scalable prototype that highlights the potential of robotics and embedded systems in space exploration, disaster management, and hazardous environment monitoring.

6.2 Future Work

As we have already discussed about the advantages, applications of our project so definitely there's room for improvement. Some of these are listed below:

- In future development this project can be develop by more sensor & alarm system.
- In Future we will add extra power system (Include Renewable Energy).
- In the future, add a GSM module to send an SMS alert system.
- In future, implementation of obstacle detection and avoidance using ultrasonic or LiDAR sensors.

REFERENCES

[1] J. Biesiadecki, et al, “Mars Exploration Rover Surface Operations: Opportunity at Meridian Planum,” submitted to Proc. IEEE SMC 2005, Waikoloa, HI.

[2] https://en.m.wikipedia.org/wiki/Mars_rover#cite_ref-42

[3] Arias, Francisco. J (2018). "A Method of Attaining High Pressurized Vessels in Space, the Moon and With Particular Reference to Mars". 2018 International Energy Conversion Engineering Conference. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2018-4488. ISBN 978-1-62410-571-5. S2CID 240369235.

[4] https://www.researchgate.net/publication/224626247

[5] Reid Simmons, Lars Henriksen, Lonnie Chrisman and Greg Whelan School of Computer Science/Robotics Institute Carnegie Mellon University Pittsburgh, PA 15213. www.cs.cmu.edu .

[6] Chen, X., & Patel, S. R. (2019, July 20). Autonomous Control System for Mars Rovers in Challenging Terrains. Proceedings of the International Conference on Space Robotics, 45-56. URL: https://www.space-robotics conf.org/proceedings/2019/papers/123456789/123456789.pdf

[7] X. Wu, L. Yang and M. Xu, "Speed following control for differential steering of 4WID electric vehicle," IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society, 2014, pp. 3054- 3059, doi: 10.1109/IECON.2014.7048945.

[8] Randel A. Lindemann and Chris J. Voorhees, "Mars Exploration Rover mobility assembly design test and performance", Systems Man and Cybernetics 2005 IEEE International Conference on, vol. 1, pp. 450- 455, 2005.

APPENDIX

Programming Code:

//**Transmitter** //

```
#include <WiFi.h>
#include <WebServer.h>
#include <Arduino.h>
#include <ESP32Servo.h>

// -------------------- AP CONFIG --------------------
const char* ssid = "ESP32_Rover_AP";
const char* password = "12345678";

WebServer server(80);

// -------------------- SENSOR PINS --------------------
#define SOIL_PIN 34
#define GAS_PIN  35

int soilPercent = 0;
int gasPercent = 0;

// -------------------- MOTOR DRIVER (L298N) --------------------
#define IN1 26
#define IN2 27
#define ENA 18

#define IN3 14
#define IN4 12
#define ENB 19

int speedValue = 150;
String direction = "STOP";

// -------------------- PWM CONFIG --------------------
const int freq = 20000;
const int pwmChannelA = 0;
const int pwmChannelB = 1;
const int resolution = 8;

// -------------------- SERVO CONFIG --------------------
#define SERVO1_PIN 23
#define SERVO2_PIN 22
```

```
#define SERVO3_PIN 21

Servo servo1, servo2, servo3;

int target1 = 0, target2 = 0, target3 = 0;
int pos1 = 0, pos2 = 0, pos3 = 0;

// -------------------- MOTOR FUNCTIONS --------------------
void stopMotors(){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  ledcWrite(pwmChannelA, 0);
  ledcWrite(pwmChannelB, 0);
  direction = "STOP";
}

void forward(){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  ledcWrite(pwmChannelA, speedValue);
  ledcWrite(pwmChannelB, speedValue);
  direction = "FORWARD";
}

void backward(){
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  ledcWrite(pwmChannelA, speedValue);
  ledcWrite(pwmChannelB, speedValue);
  direction = "BACKWARD";
}

void left(){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  ledcWrite(pwmChannelA, speedValue);
  ledcWrite(pwmChannelB, speedValue);
  direction = "LEFT";
}

void right(){
```

```
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  ledcWrite(pwmChannelA, speedValue);
  ledcWrite(pwmChannelB, speedValue);
  direction = "RIGHT";
}

void forwardLeft(){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  ledcWrite(pwmChannelA, speedValue/2);
  ledcWrite(pwmChannelB, speedValue);
  direction = "FORWARD-LEFT";
}

void forwardRight(){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  ledcWrite(pwmChannelA, speedValue);
  ledcWrite(pwmChannelB, speedValue/2);
  direction = "FORWARD-RIGHT";
}

// ---------------------- SMOOTH SERVO MOVEMENT ----------------------
void smoothServoMove() {
  if (pos1 != target1) {
    pos1 += (pos1 < target1) ? 1 : -1;
    servo1.write(pos1);
  }
  if (pos2 != target2) {
    pos2 += (pos2 < target2) ? 1 : -1;
    servo2.write(pos2);
  }
  if (pos3 != target3) {
    pos3 += (pos3 < target3) ? 1 : -1;
    servo3.write(pos3);
  }
}

// ---------------------- HTML UI ----------------------
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
```

```
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
  body{font-family: Arial; background:#111; color:white; text-align:center;}
  h2{margin-top:20px;}
  .grid{
    display:grid;
    grid-template-columns:repeat(3,1fr);
    gap:10px;
    width:90%;
    margin:auto;
  }
  button{
    padding:25px;
    font-size:22px;
    border:none;
    border-radius:12px;
    background:#222;
    color:white;
    box-shadow:0 0 8px #00d4ff;
  }
  button:active{background:#00d4ff;color:black;}
  .box{
    width:90%;
    background:#222;
    padding:15px;
    border-radius:12px;
    margin:20px auto;
  }
</style>

<script>
function send(cmd){ fetch(`/cmd?move=${cmd}`); }
function setSpeed(){
  let s=document.getElementById('spd').value;
  fetch(`/speed?val=${s}`);
  document.getElementById('sVal').innerHTML=s;
}

function setServo(id){
  let v = document.getElementById('sv'+id).value;
  document.getElementById('sv'+id+'val').innerHTML = v;
  fetch(`/servo?id=${id}&val=${v}`);
}

setInterval(()=>{
 fetch('/status').then(r=>r.json()).then(j=>{
   document.getElementById('st').textContent = `DIR: ${j.direction} | SPEED: ${j.speed}`;
   document.getElementById('soil').textContent = j.soil + "%";
```

```
  document.getElementById('gas').textContent = j.gas + "%";
 });
},500);
</script>

</head>
<body>
<h2>ESP32 Rover Controller</h2>
<h3 id="st">DIR: STOP | SPEED: 150</h3>

<div class="grid">
  <div></div>
  <button onclick="send('F')">  </button>
  <div></div>

  <button onclick="send('L')">  </button>
  <button onclick="send('S')">■ </button>
  <button onclick="send('R')">  </button>

  <div></div>
  <button onclick="send('B')">  </button>
  <div></div>
</div>

<div class="grid">
  <button onclick="send('FL')">↖</button>
  <div></div>
  <button onclick="send('FR')">↗</button>
</div>

<div class="box">
  <h3>  Moisture: <span id="soil">0%</span></h3>
  <h3>  Gas Level: <span id="gas">0%</span></h3>
</div>

<div class="box">
  <h3>Speed: <span id="sVal">150</span></h3>
  <input type="range" min="50" max="255" value="150" id="spd" oninput="setSpeed()">
</div>

<div class="box">
  <h3>Servo Controller</h3>

  <h4>Servo 1: <span id="sv1val">0</span>°</h4>
  <input type="range" min="0" max="180" value="0" id="sv1" oninput="setServo(1)">

  <h4>Servo 2: <span id="sv2val">0</span>°</h4>
  <input type="range" min="0" max="180" value="0" id="sv2" oninput="setServo(2)">
```

```
  <h4>Servo 3: <span id="sv3val">0</span>°</h4>
  <input type="range" min="0" max="180" value="0" id="sv3" oninput="setServo(3)">
</div>

</body>
</html>
)=====";

// -------------------- SERVER HANDLERS --------------------
void handleRoot(){ server.send(200, "text/html", MAIN_page); }

void handleCommand(){
  String move = server.arg("move");
  if(move == "F") forward();
  else if(move == "B") backward();
  else if(move == "L") left();
  else if(move == "R") right();
  else if(move == "FL") forwardLeft();
  else if(move == "FR") forwardRight();
  else stopMotors();
  server.send(200, "text/plain", "OK");
}

void handleSpeed(){
  speedValue = server.arg("val").toInt();
  server.send(200, "text/plain", "OK");
}

void handleServo(){
  int id = server.arg("id").toInt();
  int val = server.arg("val").toInt();

  if(id == 1) target1 = val;
  if(id == 2) target2 = val;
  if(id == 3) target3 = val;

  server.send(200, "text/plain", "OK");
}

void handleStatus(){

  int soilRaw = analogRead(SOIL_PIN);
  int gasRaw = analogRead(GAS_PIN);

  soilPercent = map(soilRaw, 0, 4095, 0, 100);
  soilPercent = constrain(soilPercent, 100, 0);

  gasPercent = map(gasRaw, 0, 4095, 0, 100);
  gasPercent = constrain(gasPercent, 0, 100);
```

```
  String json = "{";
  json += "\"direction\":\"" + direction + "\",";
  json += "\"speed\":" + String(speedValue) + ",";
  json += "\"soil\":" + String(soilPercent) + ",";
  json += "\"gas\":" + String(gasPercent);
  json += "}";

  server.send(200, "application/json", json);
}

// -------------------- SETUP --------------------
void setup(){
  Serial.begin(115200);

  pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);

  ledcSetup(pwmChannelA, freq, resolution);
  ledcSetup(pwmChannelB, freq, resolution);
  ledcAttachPin(ENA, pwmChannelA);
  ledcAttachPin(ENB, pwmChannelB);

  servo1.attach(SERVO1_PIN);
  servo2.attach(SERVO2_PIN);
  servo3.attach(SERVO3_PIN);

  pos1 = 0; pos2 = 0; pos3 = 0;
  servo1.write(0); servo2.write(0); servo3.write(0);

  WiFi.softAP(ssid, password);
  Serial.println(WiFi.softAPIP());

  server.on("/", handleRoot);
  server.on("/cmd", handleCommand);
  server.on("/speed", handleSpeed);
  server.on("/servo", handleServo);
  server.on("/status", handleStatus);
  server.begin();
}

// -------------------- LOOP --------------------
void loop(){
  server.handleClient();
  smoothServoMove(); // Smooth non-blocking servo movement
}
```