INNOVATIVE LAB PROTOTYPE FOR PREDICTING LANDSLIDE RISKS VIA SOIL & TEMPARATURE SENSOR

Submitted by

Abhishek Debnath ***21UEC007***
Santa Sarkar ***21UEC016***

ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

AGARTALA-799055, INDIA

May, 2025

INNOVATIVE LAB PROTOTYPE FOR PREDICTING LANDSLIDE RISKS VIA SOIL & TEMPARATURE SENSOR

Report submitted to

National Institute of Technology, Agartala

For the awarde of degree

of

Bachelor of technology

By

Abhishek Debnath (21UEC007)
Santa Sarkar (21UEC016)

Guided by

Dr. Mitra Barun Sarkar (Assistant Professor of ECE Department, NIT Agartala)

ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY AGARTALA
May,2025

Dedicated To

To our Project Supervisor Dr. Mitra Barun Sarkar, Assistant Professor, ECE Dept, NIT Agartala for sharing hir valuable knowledge, encouragement & showing confidence on us all the time. Each of the faculties of the department to contribute in our development as a professional and help us to achieve this goal. To all those people who have somehow contributed to the creation of this project an who have supported us.

APPROVAL SHEET

This project report entitled "***INNOVATIVE LAB PROTOTYPE FOR PREDICTING LANDSLIDE RISKS VIA SOIL & TEMPARATURE SENSOR***" by **Abhishek Debnath and Santa Sarkar** approved for the degree of **Bachelor of Technology** in **Electronics and Communication Engineering.**

Examiners

_____

_____

_____

_____

_____

Supervisor (s)

_____

_____

_____

_____

Chairman/ HOD

_____

Date: 29/04/2025

Place: NIT Agartala

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Abhishek Debnath

21UEC007

Date: 29/04/2025

(Signature)

Santa Sarkar

21UEC016

Date: 29/04/2025

CERTIFICATE

This is to certify that the project entitled "***INNOVATIVE LAB PROTOTYPE FOR PREDICTING LANDSLIDE RISKS VIA SOIL & TEMPARATURE SENSOR*****"**submittedby"**Abhishek Debnath(2113585), Santa Sarkar(2113790)**", is a record of bonafide workcarried out by the candidate under my supervision in the department of Electronics and Communication Engineering at National Institute of Technology, Agartala. The project report submitted is in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Electronics and communication Engineering from National Institute of Technology, Agartala.

Dr .Mitra Barun Sarkar(Project Supervisor)Assistant Professor
Electronics and communication engineering Dept.
NIT, Agartala

Acknowledgement

We would like to take this opportunity to express our deep sense of gratitude to all who helped us directly or indirectly during this thesis work. Firstly, we would like to thank our supervisor, Dr.Mitra Barun Sarkar for being a great mentor and the best adviser we could ever have. Her advise, encouragement and critics are source of innovative ideas, inspiration and causes behind the successful completion of this report. The confidence shown on us by her was the biggest source of inspiration for us. It has been a privilege working with her from last one year. We are highly obliged to all the faculty members of the Electronics and Communication Engineering Department for their support and encouragement. We also thank Prof.(Dr.) S.K Patra, Director, NIT Agartala and Dr. Atanu Chowdhury, H.O.D, ECE department for providing excellent computing and other facilities without which this work could not achieve its quality goal.

- **Abhishek Debnath (21UEC007)**
- **Santa Sarkar (21UEC016)**

Abstract

Landslides are among the most destructive natural disasters, often resulting in significant loss of life, property, and environmental degradation. In hilly and mountainous regions, timely prediction and early warning are essential to mitigate these risks. This project presents an innovative lab-scale prototype designed to predict landslide risks by monitoring two critical environmental parameters — **soil moisture** and **temperature**. The prototype is built around a microcontroller-based system integrated with multiple sensors to capture real-time data, and employs wireless communication for efficient data transmission and analysis.

The system utilizes a **DHT11 temperature and humidity sensor** and **soil moisture sensor** to simulate early indicators of potential landslide events. These sensors are interfaced with an **Arduino Uno** and **LoRa (Long Range) modules** to enable long-distance wireless communication between the sensor node and the receiver unit. The receiver, built using an **ESP32 microcontroller**, sends the received data to a **google sheet** , where it can be visualized and analyzed through a custom-designed **web-based dashboard**. This dashboard presents the live sensor values and highlights anomalies using predefined threshold values, providing a user-friendly interface for monitoring environmental conditions in real-time.

The prototype demonstrates the feasibility of implementing a low-cost, scalable, and energy-efficient early warning system for landslides. Its modular architecture allows for further integration of additional environmental parameters like rainfall and vibrations, increasing the system's reliability. While this model is tested in a controlled laboratory environment, it lays the foundation for future deployment in real-world field conditions. The data collected from the prototype can help in training predictive models using machine learning to further enhance landslide forecasting accuracy.

Contents

CHAPTER 1

Introduction

Landslides are a recurring natural hazard in various parts of the world, especially in regions with steep terrain, loose soil, and high rainfall. They pose significant threats to human lives, infrastructure, agriculture, and the environment. With increasing urbanization and deforestation in hilly areas, the frequency and impact of landslides have intensified. Traditional landslide monitoring methods often rely on manual observation or expensive equipment that may not be accessible for remote or economically constrained areas. Therefore, the need for a cost-effective, real-time, and scalable landslide prediction system has become more critical than ever.

This project aims to address this challenge by designing and implementing an **innovative lab-scale prototype** that predicts landslide risks by monitoring key environmental factors—**soil moisture** and **temperature**. These parameters are among the most influential in determining the stability of soil. An increase in soil moisture due to prolonged rainfall can reduce soil cohesion, leading to slope failure. Likewise, extreme temperature variations can affect soil strength and water absorption, influencing the likelihood of a landslide.

The system architecture integrates a variety of sensors, including a **soil moisture sensor**, **DHT11 temperature and humidity sensor**, to simulate real-world slope conditions. These sensors are interfaced with an **Arduino Uno**, which acts as the data acquisition unit. The collected sensor data is transmitted using **LoRa (Long Range) wireless modules** to an **ESP32-based receiver**, which then uploads the data to a**google sheet**. After this Machine learning model has been Made to find the risk of the landslide based on the data collected.

The prototype is designed to be modular, low-power, and low-cost, making it ideal for deployment in remote and vulnerable locations. It serves as a foundational step toward a more robust early warning system that can be enhanced with machine learning algorithms, additional sensors, and real-time alerts. Through this project, we aim to demonstrate how modern sensor technology and IoT (Internet of Things) can be leveraged to mitigate landslide risks and contribute to environmental sustainability and disaster resilience.

1.1 **-Objective** –

The primary objective of this project is to **design and develop a real-time Landslide Risk Detection and Alert System** using soil moisture and temperature data. Landslides are hazardous natural events that can cause devastating damage to human lives, property, and the environment. This project aims to build a **lab-scale, cost-effective, and scalable prototype** that can be deployed in landslide-prone areas for early detection and timely response. By leveraging sensor technology and IoT-based communication, the system provides continuous environmental monitoring, ensuring proactive disaster mitigation.

Key Aspects of the Objective

1. **Early Detection of Landslide Riskss**
   The core aim is to identify early signs of slope failure by monitoring soil moisture and temperature levels. Elevated soil moisture due to rainfall and changes in temperature are critical indicators of potential landslide events. This system captures these parameters in real time to provide early warnings before a disaster occurs.

2. **Real-Time Monitoring and Alerts**
   The system operates autonomously and continuously, ensuring uninterrupted environmental monitoring. Using LoRa-based wireless communication, sensor data is transmitted in real time to a central unit, which uploads it to the Firebase Realtime Database. A web dashboard visualizes the data, highlighting anomalies and enabling timely intervention.

3. **Scalability and Modularity**
   The Arduino-based modular design allows easy integration of additional sensors, such as tilt, rainfall, or vibration detectors. This flexibility enables the system to be tailored for various terrains and environmental conditions, ensuring broader applicability and scalability.

4. **Enhancing Disaster Preparedness and Management**
   By providing early warning signals, this system enhances disaster preparedness efforts and supports local authorities in deploying preventive measures. The aim is to reduce casualties, infrastructure loss, and environmental damage by enabling quicker and informed decision-making.

5. **Addressing Limitations of Existing Systems**
   Traditional landslide monitoring relies on manual surveys or complex, expensive equipment, often impractical in remote regions. This prototype overcomes such limitations by offering a low-power, low-cost, and autonomous solution that functions efficiently even in off-grid areas..

6. **Integration of Predictive Features**
   Though the current system focuses on detection, it is structured to allow future incorporation of predictive algorithms. By using historical environmental data and machine learning techniques, the system can evolve into a predictive model for landslide risk forecasting..

7. **Deployment in Remote and Hilly Areas**
   Many high-risk areas lack proper monitoring due to challenging terrain and lack of infrastructure. The lightweight, portable design of this prototype, combined with wireless communication, makes it ideal for deployment in such regions, ensuring widespread and reliable coverage.

This project is a significant step toward making landslide detection smarter, more accessible, and impactful. By combining simple hardware with powerful communication and data visualization tools, it brings practical innovation to the domain of natural disaster risk reduction.

1.2 – **Historical evidences of Landslide Detection -**

1. Kedarnath Flash Flood and Landslide (2013, Uttarakhand, India)

In June 2013, Uttarakhand experienced one of the deadliest natural disasters in India's history. Intense rainfall led to devastating flash floods and landslides, particularly in the Kedarnath region. The unanticipated landslides buried roads, swept away entire villages, and cut off communication in the hilly terrain.

Impacts:

1. **Human Casualties:** Over 5,000 people were estimated to have lost their lives, with many more injured or missing.

2. **Infrastructure Destruction :** Roads, bridges, homes, and public buildings were destroyed, leaving thousands homeless and isolated

3. **Pilgrimage Halt :** The sacred Kedarnath temple became inaccessible, affecting religious tourism and local livelihoods.

4. **Environmental Degradation** : Massive deforestation and unstable slopes triggered repeated landslides even after the initial disaster.

2. Malin Landslide (2014, Maharashtra, India)

In July 2014, the village of Malin in Maharashtra was wiped out by a massive landslide triggered by continuous monsoon rains. The event buried almost the entire village under thick layers of mud and debris.

Impacts:

- **Loss of Life**: Around 151 people were reported dead, with most trapped under debris.
- **Communication Failure**: The remote location delayed rescue operations due to lack of timely alerts and connectivity.
- **Socio-economic Impact**: Families lost homes, agricultural land, and basic infrastructure, severely affecting the rural economy
- Post-Disaster Recovery: Long-term rehabilitation efforts were required to resettle survivors and rebuild the village.

3. Halflong Landslides (2022, Assam, Northeast India)

In May 2022, relentless rainfall in Assam triggered a series of devastating landslides in the Dima Hasao district, particularly in Halflong. The landslides caused widespread destruction in this hilly region of Northeast India, cutting off road and rail connectivity and severely impacting both locals and administration.

Impacts:

- **Agricultural Losses**: Several tea plantations and farmlands were destroyed, affecting the local economy.
- **Transportation Disruption**: Roads and rail services were suspended, impacting tourism and supply chains.
- **Environmental Damage**: Deforestation and poor land-use planning worsened slope vulnerability.
- **Relocation Needs**: Authorities had to shift families living in vulnerable areas to safer zones.

1.4- **Impact of Technology in Landslide Detection and Prevention –**

1. **Emerging Technologies**: Modern advancements are significantly transforming how landslides are monitored and predicted. Technologies such as **Internet of Things (IoT) sensors**, **Global Positioning Systems (GPS)**, **drone-based aerial mapping**, and **satellite remote sensing** are now being used to detect even minor changes in terrain. Machine learning models are also being trained to predict landslide occurrences based on sensor data such as soil moisture, rainfall intensity, and ground tilt.

   These tools allow continuous monitoring of unstable slopes, which is especially crucial in hilly and high-rainfall regions like Northeast India.

2. **Early Warning Systems**: Real-time warning systems are now using sensor networks embedded in the soil to measure parameters like **soil moisture, rainfall, ground tilt, and vibrations**. These sensors transmit data via **LoRa modules** or cloud-based platforms (such as **Firebase**) to remote servers where anomalies are flagged immediately.

   This allows early alerts to be sent to authorities and local residents, helping in **timely evacuation** and **disaster response**. Mobile applications and SMS-based alert systems are increasingly being used to relay such information to at-risk communities.

3. **Case Studies**:
   - **Sensor-Driven Landslide Alarms in Japan:** Japan uses a dense network of underground sensors and wireless telemetry systems to warn residents living near landslide-prone zones.
   - **DREAM Project (India):** The DST-funded **DREAM** project uses IoT-based wireless sensor nodes in the Western Ghats to monitor soil and hydrological conditions.
   - **Bhutan Early Warning System:** Bhutan has implemented community-based early warning systems with rain gauges and geophones for landslide-prone valleys.
   - **Student Prototypes:** The lab prototype developed in this project demonstrates the feasibility of low-cost landslide prediction systems using **Arduino**, **DHT11**, **soil moisture sensors**, and **LoRa communication** to send real-time data to Firebase.

4. **Challenges and Limitations:**
   Despite technological progress, adoption is limited by several challenges:
   - **Infrastructure Gaps:** Remote and rural regions often lack consistent power supply and internet connectivity.
   - **Cost Barriers:** High costs of commercial-grade sensor systems and satellite subscriptions hinder scalability.
   - **Maintenance Needs:** Sensors require regular calibration and maintenance,

especially in harsh environments.

- **Data Accuracy:** Sensor data can sometimes be affected by environmental noise, leading to false alarms or missed events.

5. **Future Outlook**:
   Research is ongoing into several promising technologies:
   - **Autonomous Drones:** Capable of surveying difficult terrains and identifying early warning signs of slope instability.
   - **AI-Based Prediction Models:** These can process large volumes of sensor data to predict landslide events with higher accuracy.
   - **Community-Integrated Systems:** Projects are exploring participatory sensor networks where citizens help install and monitor low-cost devices.
   - **Integration with National Disaster Platforms:** Linking local IoT systems with national disaster warning databases can improve coordination.

Facts and Figures:

1. **Global Landslide Impact:** Landslides cause over **4,800 deaths annually** and affect thousands of communities, especially in Asia.
2. **Economic Damage:** Landslides cause economic losses exceeding **$20 billion annually**, especially in developing countries.
3. **India's Vulnerability:** More than **15% of India's landmass** is prone to landslides, especially in the Himalayas, Western Ghats, and Northeast states.
4. **Technology Trends:** Countries like **Japan, Switzerland, and the U.S.** lead in sensor-based early warning systems, while India is increasingly promoting **affordable, scalable student innovations** for local-level disaster prediction.

1.4- Ecological and Health Consequences of Landslides -

Landslides are one of the most devastating natural disasters, with profound ecological consequences and serious implications for human health and safety. These events not only reshape landscapes within minutes but also threaten biodiversity, disrupt local ecosystems, and endanger the lives and livelihoods of communities, especially in hilly and forested regions like **Northeast India**.

Ecological Consequences:

1. **Habitat and Biodiversity Loss:**
   Landslides can wipe out entire sections of forests, grasslands, and river ecosystems, instantly destroying habitats for plants, animals, and insects. In biodiversity hotspots, such as the Eastern Himalayas, landslides pose a serious threat to endangered and endemic species, whose population may already be limited. The destruction of vegetation also eliminates food sources and nesting areas, disrupting local ecological balance**.**
2. **Disruption of Ecosystem Services:**
   Natural ecosystems offer critical services like slope stabilization, water regulation, and soil fertility maintenance. Landslides can eliminate tree cover and root systems, reducing the soil's ability to retain water, increasing runoff, and worsening future disasters like floods. This disturbance reduces nature's capacity to mitigate natural risks, making landscapes more vulnerable over time.
3. **Soil and Water Degradation:**
   Landslides typically involve large-scale soil movement, leading to erosion, sedimentation of rivers, and degradation of arable land. Soil layers rich in organic matter are often displaced or buried, reducing land fertility. Rivers and streams become clogged with debris, affecting aquatic life and water quality downstream.
4. **Impact on Climate Resilience:**
   Vegetation loss due to landslides reduces the landscape's ability to act as a carbon sink, weakening the role of forests in climate regulation. Furthermore, exposed slopes are more prone to heat absorption, leading to microclimatic shifts that can affect agriculture and biodiversity patterns in the region.
5. **Hydrological Cycle Alteration:**
   The sudden removal of tree cover and topsoil interrupts natural infiltration processes, decreasing groundwater recharge and altering stream flows. This can reduce water availability for both agriculture and drinking, particularly in communities that rely on springs and shallow wells.

Health Consequences:

1. Direct Fatalities and Injuries:
Landslides cause thousands of deaths and injuries globally each year, often occurring with little or no warning. People caught in a landslide may suffer from crush injuries, fractures, or trauma

due to falling debris or building collapses. In hilly, densely populated regions, landslides can destroy entire villages in minutes.

2. Mental Health Impact:

The psychological toll of landslides on survivors can be immense. The sudden loss of homes, family members, and livelihoods often leads to anxiety, depression, and post-traumatic stress disorder (PTSD). Communities in chronic landslide-prone zones also suffer from persistent fear and emotional distress, which can affect long-term mental wellbeing.

3. Waterborne Diseases:

Landslides can contaminate surface water sources with debris, sewage, and decomposing organic matter, increasing the risk of waterborne illnesses such as cholera, typhoid, and diarrhea, particularly in remote and underdeveloped areas where clean water access is already limited.

4. Displacement and Healthcare Access:

Entire communities may need to be relocated after severe landslides, disrupting access to healthcare, education, and employment. The sudden increase in displaced populations can overwhelm local healthcare infrastructure, especially in rural areas, delaying treatment and recovery for both physical and mental conditions.

5. Nutrition and Food Security:

Agricultural fields and livestock are often buried under landslide debris, leading to loss of crops, income, and local food supplies. In regions where subsistence farming is common, this can trigger malnutrition, food scarcity, and a dependency on aid**.**

Broader Implications:

The ecological and health consequences of landslides underscore the importance of **early detection systems**, **sustainable land use**, and **community-based mitigation strategies**. The Northeast region of India, with its fragile hills, seismic activity, and heavy monsoons, is particularly vulnerable. Addressing these challenges through **IoT-enabled sensor networks**, **rainfall monitoring**, **soil moisture analysis**, and **local awareness programs** can significantly reduce the risks to both ecosystems and human lives.

Effective landslide prevention is not just about technology—it is about **integrating science, local knowledge, and policy** to build resilient communities in harmony with nature.

CHAPTER 2

Literature Survey

***Landslide Detection and Monitoring System*:**

Landslides represent one of the most catastrophic natural disasters, often resulting in significant loss of life, property, and environmental damage. The increasing frequency of landslides, particularly in areas with steep terrains and unpredictable weather patterns, has prompted the development of advanced technologies for early detection, monitoring, and risk assessment. This literature survey explores the methodologies, technologies, and systems designed to address landslide detection and prevention, categorizing the research into landslide detection systems, monitoring technologies, and early warning systems.

---

1. Landslide Detection Systems

1.1 Sensor-Based Systems

Sensor-based systems are fundamental for the real-time monitoring of landslide-prone areas, leveraging various environmental parameters such as soil moisture, tilt, and seismic activity.

- **Soil Moisture Sensors:** Soil moisture is a key indicator of landslide risk, as increased moisture content can destabilize slopes. Various sensor types, including **DHT11** and **capacitive soil moisture sensors**, are used to track soil saturation levels.
    - **Advantages:** Cost-effective, scalable, and offers continuous monitoring.
    - **Challenges:** Sensor calibration, power limitations, and data accuracy in extreme weather conditions.
- **DHT22 Sensors:** The DHT22 sensor measures **temperature** and **humidity**, two environmental factors that significantly influence soil moisture and slope stability.
    - **Advantages:** Low cost, easy integration with microcontrollers, and real-time environmental data.
    - **Challenges:** Limited measurement accuracy and narrower operating range compared to industrial-grade sensors.

1.2 Remote Sensing and Satellite-Based Systems

Satellite-based systems offer large-scale, continuous monitoring of landslide-prone areas, utilizing technologies like radar interferometry and high-resolution imagery.

- **Synthetic Aperture Radar (SAR):** SAR technology is particularly effective in detecting ground displacement, especially in areas with dense vegetation or cloud cover. It can identify changes in the earth's surface, such as those caused by landslides.
    - **Example:** Huser et al. (2020) used **Sentinel-1 SAR** data to monitor surface deformations in **Himalayan** regions and predict potential landslides.
    - **Advantages:** Ability to monitor vast areas in all weather conditions.
    - **Challenges:** High computational cost and delayed data processing.

- **Optical and Thermal Imaging:** High-resolution optical and infrared imagery can be used to detect ground displacement and heat signatures associated with soil movement.
    - **Example:** The **MODIS** (Moderate Resolution Imaging Spectroradiometer) satellite system is utilized for detecting anomalies in vegetation and soil conditions that precede landslides.
    - **Advantages:** Provides real-time data on large-scale geographic areas.
    - **Challenges:** Limited resolution for detecting small-scale landslides and cloud cover interference.

2. Monitoring Technologies

2.1 Geographic Information Systems (GIS)

Geospatial technology plays a critical role in mapping landslide-prone areas, analyzing terrain conditions, and monitoring landslide movement over time.

Risk Mapping and Hazard Assessment: GIS platforms are used to visualize landslide risk zones, integrating environmental parameters such as slope, soil composition, and rainfall patterns.

- **Example:** Wu et al. (2019) demonstrated the use of GIS for creating landslide susceptibility maps in the **Indian Himalayas**.
- **Challenges:** Requires high-quality data and extensive fieldwork.

2.2 Drones and UAVs (Unmanned Aerial Vehicles)

Drones equipped with high-resolution cameras, thermal sensors, and GPS are becoming increasingly popular for monitoring landslides and assessing affected areas in real-time.

Capabilities: Drones can provide aerial views of landslide zones, assess damage, and detect soil movement.

- **Example:** Lin et al. (2021) utilized drones with thermal and optical sensors to monitor landslide-prone areas in **Taiwan**, providing real-time data to emergency response teams.
- **Challenges:** Limited flight time, high operational costs, and regulatory constraints.

3. Early Warning Systems

3.1 IoT and Real-Time Monitoring Systems

Integrating **IoT** sensors with early warning systems allows for continuous monitoring and immediate response in case of landslide threats.

- **Functionality:** IoT-enabled systems collect environmental data and transmit it to central servers, triggering alerts if dangerous conditions are detected (e.g., sudden soil movement, moisture increase).
    - **Example: Nepal's** early warning system integrates IoT sensors for soil moisture and tilt, providing real-time data to local authorities.
    - **Challenges:** Power supply issues in remote areas and sensor calibration.

3.2 Data Integration and Predictive Analytics

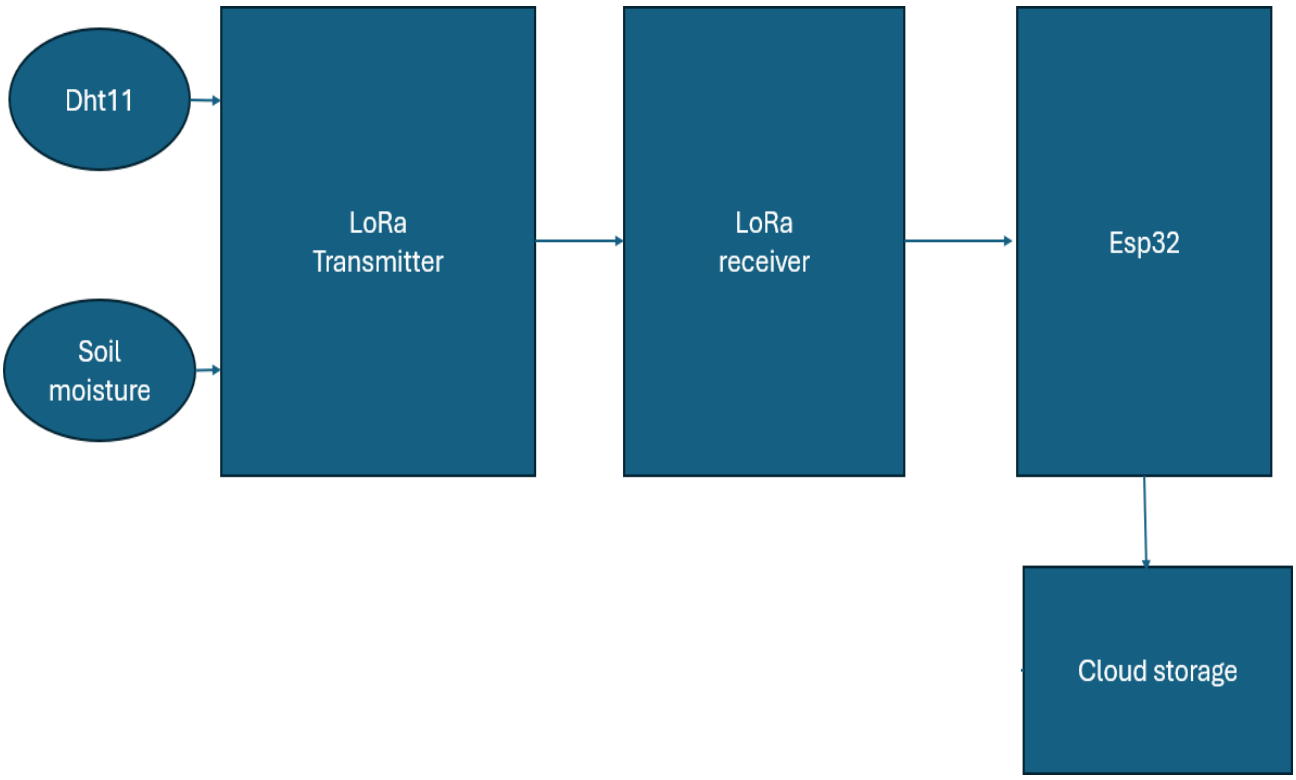
By integrating data from various sources, predictive analytics can be employed to forecast landslides before they occur, enabling timely evacuations and risk management.

- **Example:** Kim et al. (2020) proposed a hybrid system combining **IoT data**, **machine learning**, and **seismic measurements** for early landslide warnings in **South Korea**.
    - **Advantages:** Timely and accurate warnings based on real-time and predictive data.
    - **Challenges:** Computational resources required and model accuracy.

CHAPTER 3

System Archietecture & Equipments

3.1 system Archietecture



This system is a **wireless environmental monitoring and prediction platform** using LoRa communication, an ESP32 microcontroller, Here's a step-by-step explanation:

---

1. Sensor Layer

- **DHT11 Sensor**:
  - Measures **temperature** and **humidity**.
  - Provides basic environmental data important for agricultural or smart garden applications.
- **Soil Moisture Sensor**:
  - Measures the **moisture content** in the soil.
  - Crucial for monitoring irrigation needs and plant health.

Both sensors are connected to the **LoRa Transmitter**.

---

2. Data Transmission Layer

- **LoRa Transmitter**:
  - Collects data from DHT11 and Soil Moisture sensors.
  - **Encodes and wirelessly transmits** the sensor data using **LoRa (Long Range)** protocol, which is ideal for low-power, long-distance communication.
- **LoRa Receiver**:
  - Receives the transmitted data.
  - Acts as a bridge between the remote sensor node and the central processing unit (ESP32).

---

3. Processing Layer

- **ESP32 Microcontroller**:
  - Receives the sensor data from the LoRa Receiver via a UART or Serial interface.
  - **Processes, organizes, and formats** the received data.
  - Responsible for two major tasks:
    - Sending the data to **Cloud Storage**.
    - Feeding the data to the **Prediction Module**.

---

4. Storage Layer

- **Cloud Storage**:
  - The ESP32 sends the real-time data to google sheets
  - Ensures safe, scalable, and remote storage of environmental data.
  - Acts as a centralized database for historical and real-time analysis.

3.2- Materials Required-

Hardware Components:

1. **Arduino Mega:** Microcontroller for processing sensor data.
2. **Dht11 Sensor:** Detects the presence of fire or flame.
3. **Temperature & Humidity Sensor (DHT11):** Monitors temperature variations.
4. **LoRa Transmitter and Receiver :** use for sending data to cloud**.**
5. **Esp32Module:** Sends data to cloud for processing**.**

Software Tools:

1. **Arduino IDE:** For coding and uploading the program to Arduino.

1.1 -Hardware Component Description-

1. Arduino Mega

Arduino is an open-source hardware and software platform designed for building and prototyping electronic projects. It provides an easy-to-use environment for beginners and professionals to create interactive devices and embedded systems. The platform is built around **microcontroller boards** that can read inputs, process data, and control outputs, making it versatile for various applications like robotics, IoT, automation, and education.

Key Features of Arduino

1. **Open-Source**: Arduino's hardware designs, libraries, and development environment are open-source, allowing for customization and adaptation for specific project needs.

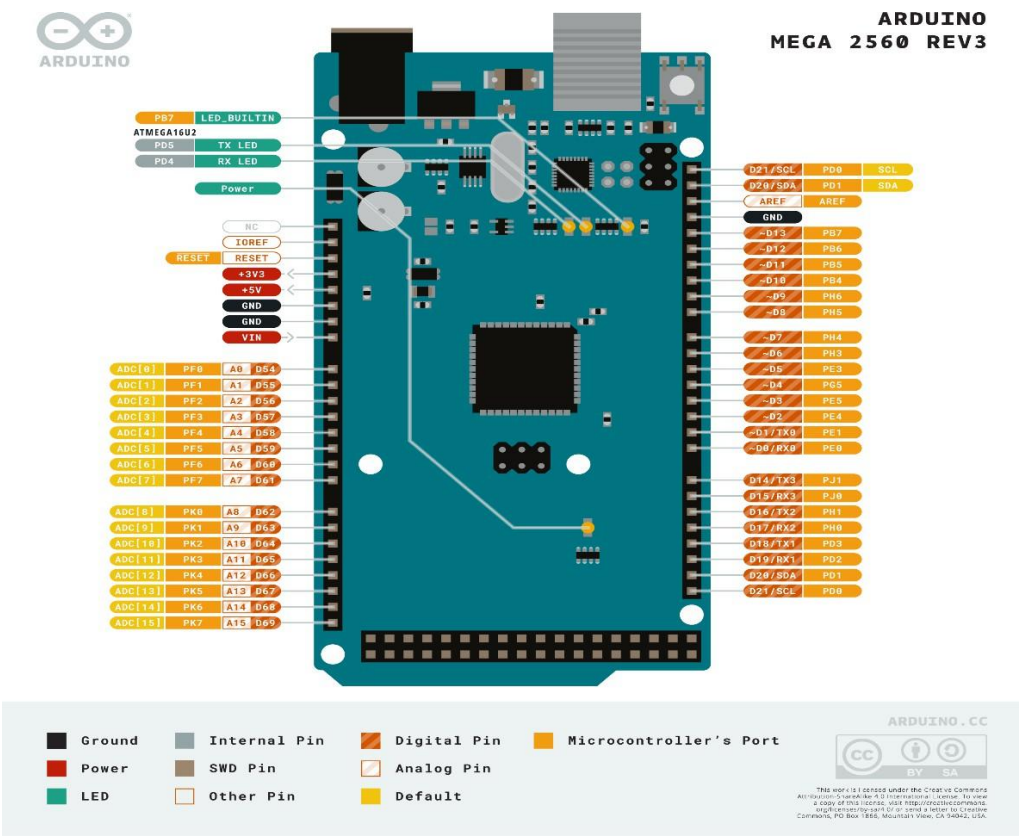
2. **Easy Programming**: The Arduino platform uses a simplified version of C/C++, which is beginner-friendly. It integrates with the **Arduino IDE (Integrated Development Environment)** for writing, uploading, and debugging code.

3. **Wide Range of Boards**: Arduino offers multiple boards, such as the **Arduino UNO**, **Nano**, **Mega**, and **Due**, each suited for different types of projects.

4. **Extensive I/O**: Provides digital and analog pins for interfacing with a variety of components like sensors, motors, LEDs, and displays.

5. **Flexible Power Options**: Can be powered via USB or external DC sources, supporting a range of input voltages for flexibility.

6. **Rich Ecosystem**: Compatible with various shields, libraries, and modules (e.g., Wi-Fi, Bluetooth, GSM), making it ideal for expanding functionality.



1. Power Pins

- Vin: Input voltage pin. Supplies power to the Arduino board when using an external power source (7–12V).
- 5V: Provides a regulated 5V output for powering external devices or sensors.
- 3.3V: Provides a 3.3V output for low-power devices.
- GND (Ground): Multiple ground pins available to complete the circuit.
- IOREF: Provides the operating voltage of the board (typically 5V). Used by shields to ensure compatibility.

---

2. Digital Pins (0–53)

- Function: Used for digital input or output operations (HIGH/LOW signals).
- Special Features:

o Pins 2–13, 44–46: Support PWM (Pulse Width Modulation), useful for controlling motors, LEDs, etc.

o Pins 0 (RX) and 1 (TX): Serial communication (UART0).

o Pins 2, 3, 18, 19, 20, 21: External interrupt capability.

---

3. Analog Input Pins (A0–A15)

- Function: Used to read analog signals (e.g., from temperature or light sensors).
- Resolution: 10 bits, providing values between 0 and 1023.
- Voltage Range: 0V to 5V (default).

• Can Be Used as Digital Pins: All analog pins can also be used as digital I/O.

---

4. Communication Pins

- ***• Serial (UART):***
  - o Serial0: RX (Pin 0), TX (Pin 1)
  - o Serial1: RX1 (Pin 19), TX1 (Pin 18)
  - o Serial2: RX2 (Pin 17), TX2 (Pin 16)
  - o Serial3: RX3 (Pin 15), TX3 (Pin 14)
- o Used for communication with computers, Bluetooth modules, GPS, etc.

• I2C (Wire Library):

o SDA (Pin 20): Serial Data Line
o SCL (Pin 21): Serial Clock Line
o Used for communicating with I2C-compatible devices like OLEDs or RTCs.

• SPI (Serial Peripheral Interface):

o MISO (Pin 50): Master In Slave Out
o MOSI (Pin 51): Master Out Slave In
o SCK (Pin 52): Serial Clock
o SS (Pin 53): Slave Select
o Used for high-speed communication with SD cards, displays, etc.

---

5. Reset Pin

• RESET: Resets the microcontroller. Can be used to restart the program on the board. Pulling this pin LOW resets the board**.**

---

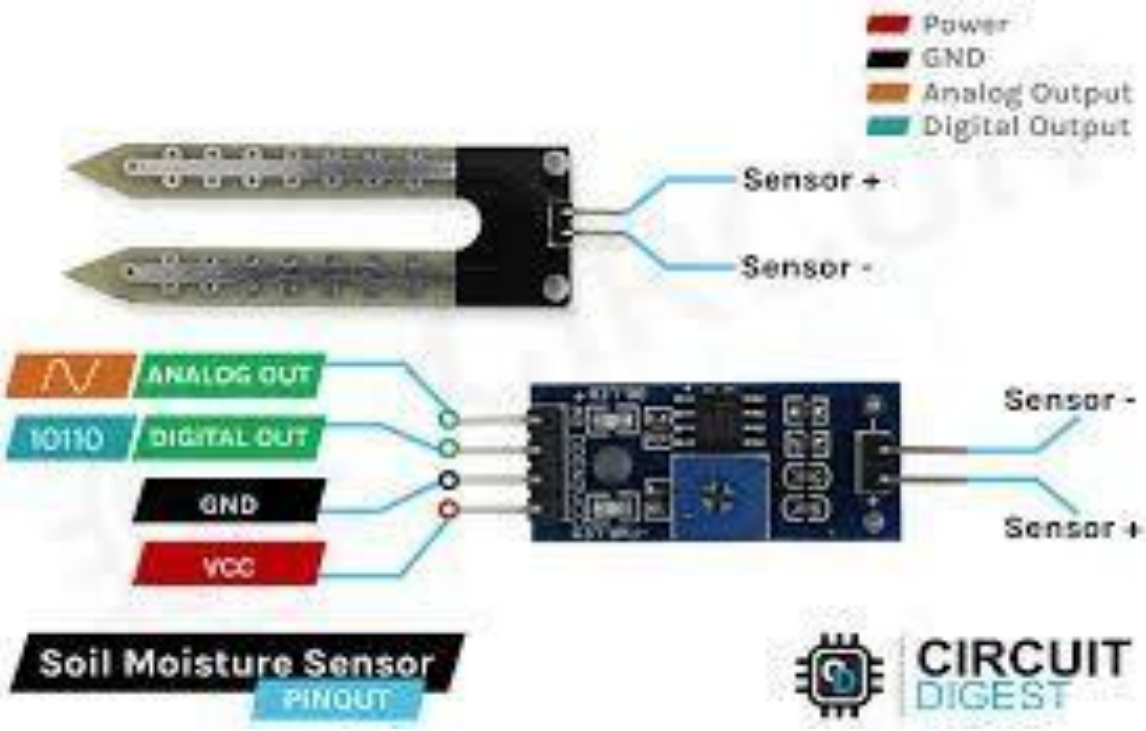
6. Other Pins

• AREF (Analog Reference): Used to set an external reference voltage for analog inputs (default is 5V).

• IOREF: Provides the voltage level (5V or 3.3V) at which the board operates. Helps shields detect the correct logic level.

2. Soil Moisture Sensor:

A soil moisture sensor measures the water content in the soil, helping determine whether the soil is dry, moist, or wet. It is commonly used in smart irrigation systems, greenhouse monitoring, and agricultural projects.



Working Principle:

The sensor works by measuring the resistance between two probes inserted into the soil. Moist soil conducts electricity better than dry soil. The sensor outputs either an analog voltage (for moisture level) or a digital HIGH/LOW signal based on a threshold.

Pin Description:

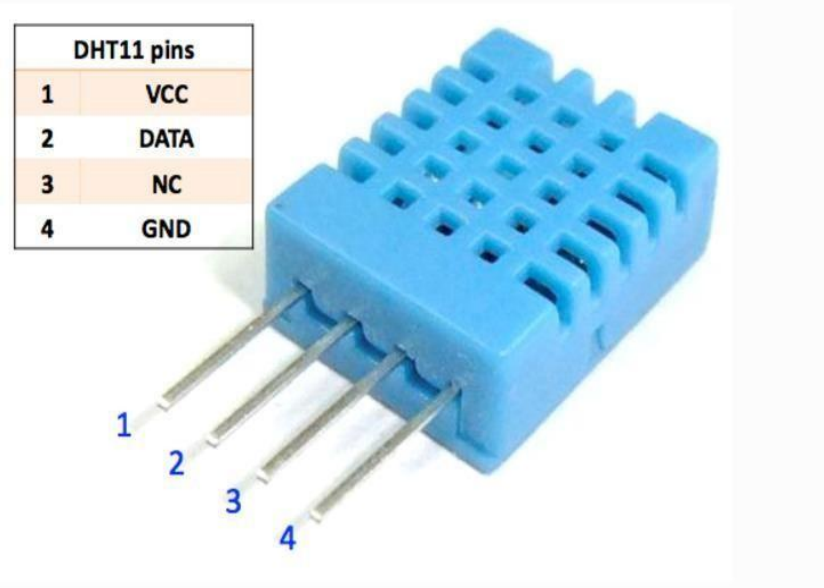
- VCC: Connect to 3.3V or 5V power supply.
- GND: Connect to ground.
- AO: Analog output pin; gives a variable voltage based on soil moisture level.
- DO: Digital output pin; gives HIGH or LOW based on the preset moisture threshold (can be adjusted via the onboard potentiometer).

3. *Temperature & Humidity Sensor (DHT11)*

The DHT11 is a digital sensor used for measuring temperature and humidity. It integrates a capacitive humidity sensor and a thermistor to monitor environmental conditions. The sensor outputs data as a digital signal, making it easy to interface with microcontrollers like Arduino and Raspberry Pi.

Features:

- Low-cost and simple to use.
- Measures temperature (in °C) and relative humidity (%).
- Ideal for environmental monitoring applications.



Pin Description:

- **VCC**: Connect to 5V.
- **GND**: Connect to ground.
- **DATA**: Outputs digital data to a microcontroller pin (e.g., Arduino A0).

LoRa RYLR896 Module:

The LoRa RYLR896 is a long-range, low-power wireless communication module based on Semtech's SX1276 LoRa transceiver. It is widely used for IoT applications, remote sensing, and telemetry where long-distance communication is essential.

Working Principle:

The RYLR896 uses LoRa modulation technology to achieve long-range communication (up to 10 km in open areas) with low power consumption. It operates in the sub-GHz frequency bands (typically 433 MHz, 868 MHz, or 915 MHz depending on region) and communicates via AT commands through a UART (serial) interface.

Pin Description:

- VCC: Connect to 3.3V power supply (Do not connect to 5V directly).
- GND: Connect to ground.
- TX: Transmit pin; sends serial data to the microcontroller's RX pin.
- RX: Receive pin; receives serial data from the microcontroller's TX pin.
- RST (optional): Reset pin; can be used to manually reset the module (usually not required).

ESP32 Module:

The ESP32 is a powerful Wi-Fi and Bluetooth-enabled microcontroller module developed by Espressif. It is widely used in IoT applications due to its dual-core processor, rich peripheral set, and wireless capabilities, making it ideal for smart home devices, wearable tech, and sensor networks.

Working Principle:

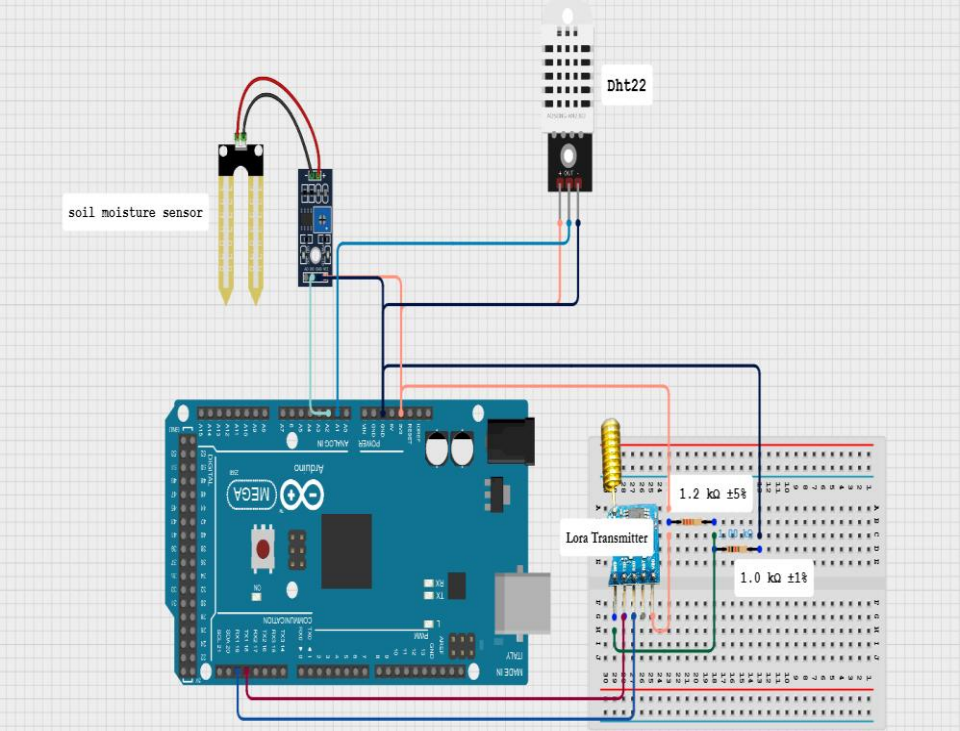
The ESP32 integrates Wi-Fi (802.11 b/g/n) and Bluetooth (Classic and BLE) on a single chip, allowing it to connect to the internet or other devices wirelessly. It operates as both a microcontroller and a communication module, capable of running embedded applications while handling network tasks simultaneously.

Pin Description:

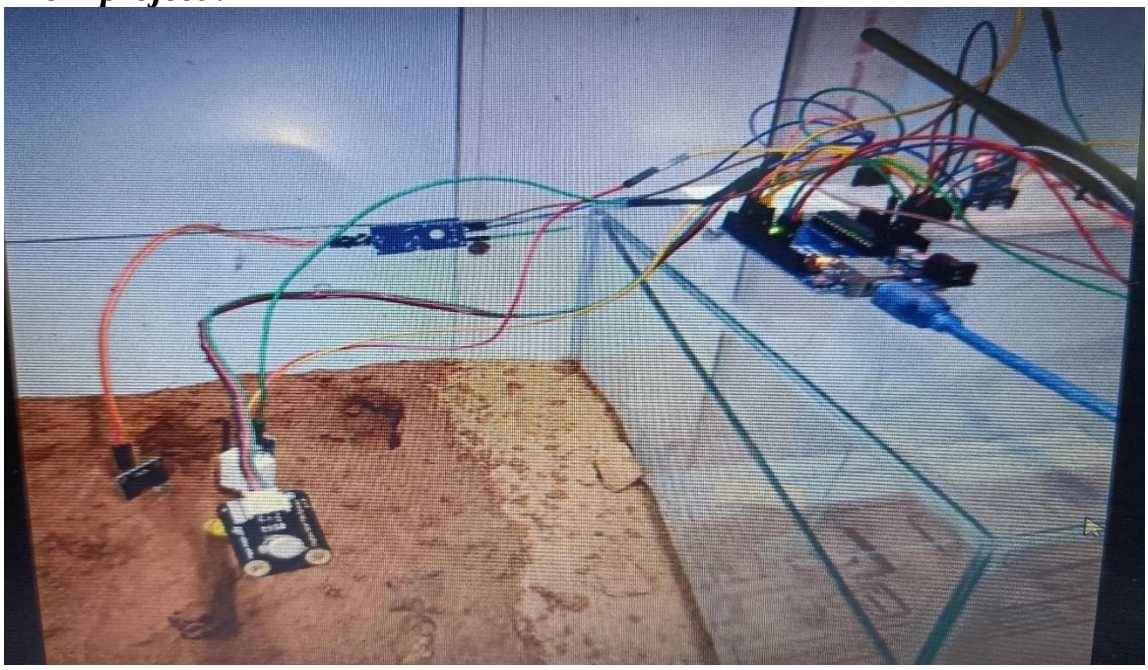
- VIN (or V5/VUSB): Input voltage pin (5V from USB or external power source).
- 3.3V: Regulated 3.3V output; used to power sensors and other peripherals.
- GND: Ground pin.
- EN (Enable): Reset or enable pin. Pulling LOW resets the module.
- GPIO0–GPIO39: General-purpose I/O pins used for digital/analog I/O, PWM, I2C, SPI, UART, etc.
- TX0 (GPIO1): Default serial transmit pin (UART0).
- RX0 (GPIO3): Default serial receive pin (UART0).
- ADC (e.g., GPIO32–39): Analog input pins for reading sensor values (12-bit resolution).
- DAC (e.g., GPIO25, GPIO26): Digital-to-analog output pins.
- I2C (SDA/SCL): Can be assigned to any GPIO via software.
- SPI/UART/PWM: Multiple peripherals can be assigned to various GPIOs as needed.

2.4 – Circuit :

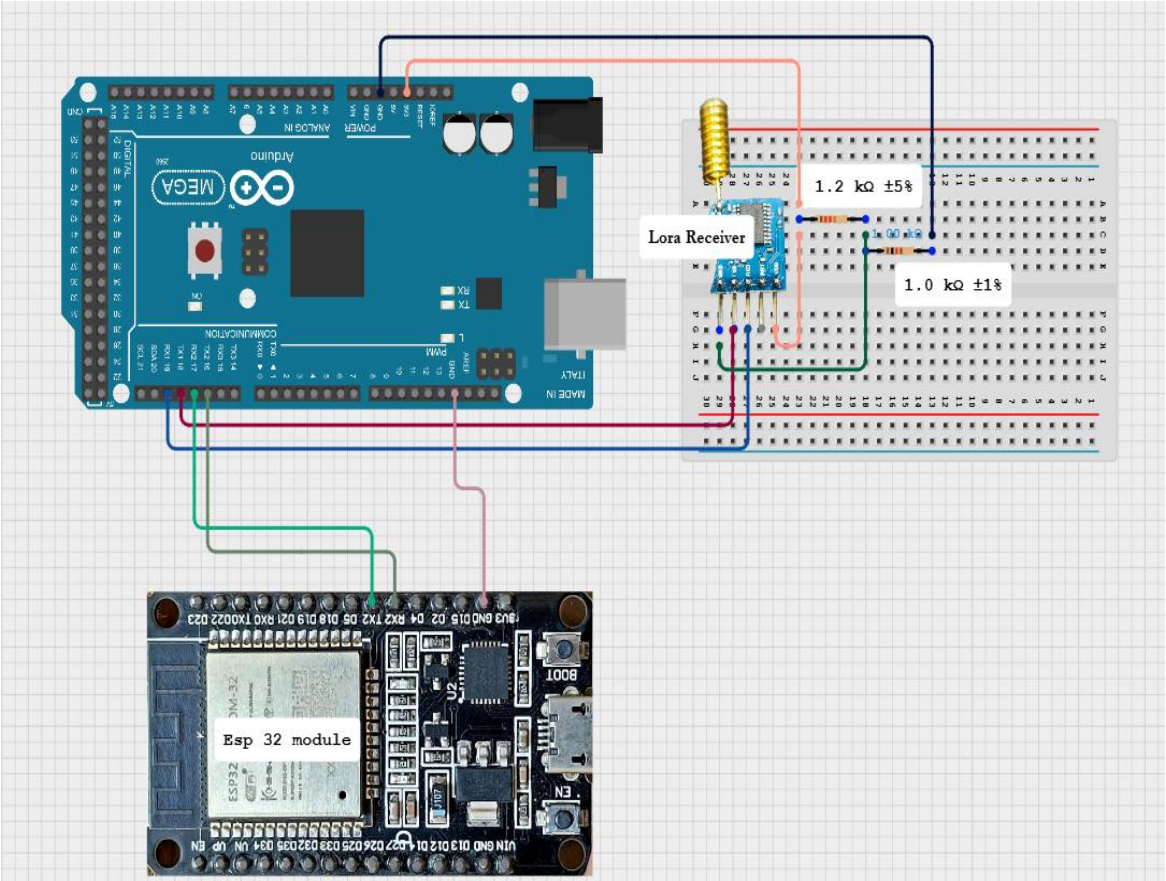
1. LoRa Transmitter with sensor Integrated :



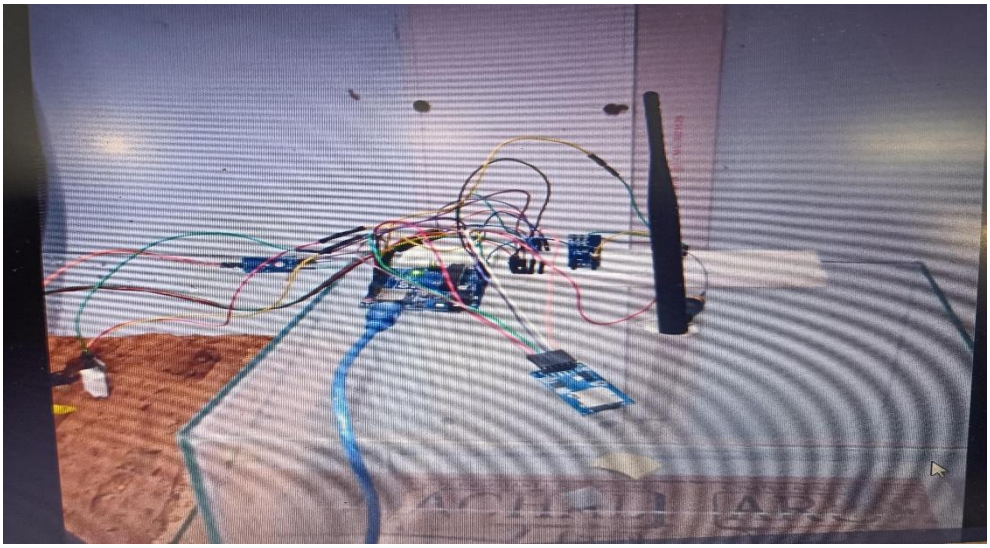
From project :



2. LoRa Receiverr with Esp32 Integrated :



From project :



Connection Layout Description:

Pin Connection Table – Arduino Mega (LoRa Transmitter) with Components:

Component Name	Pin on Component	Pin on Arduino Mega	Description
Arduino Mega (LoRa Transmitter)	--	--	Microcontroller for processing sensor data and transmitting over LoRa.
DHT11 Sensor	VCC	5V	Power supply to the DHT11 sensor.
	GND	GND	Ground connection for the DHT11 sensor.
	DO	A1	Digital output for temperature and humidity readings.
	VCC	5V	Power supply to the temperature & humidity sensor.
Soil moisture sensor	GND	GND	Ground connection for the sensor.
	DO	A2	Digital output for temperature and humidity readings.
LoRa Transmitter	VCC	3.3V	Power supply to the LoRa module (ensure 3.3V for proper operation).
	GND	GND	Ground connection for the LoRa module.
	TXD	10	Transmit data pin from LoRa transmitter to LoRa receiver.
	RXD	11	Receive data pin from LoRa receiver (optional depending on use case).

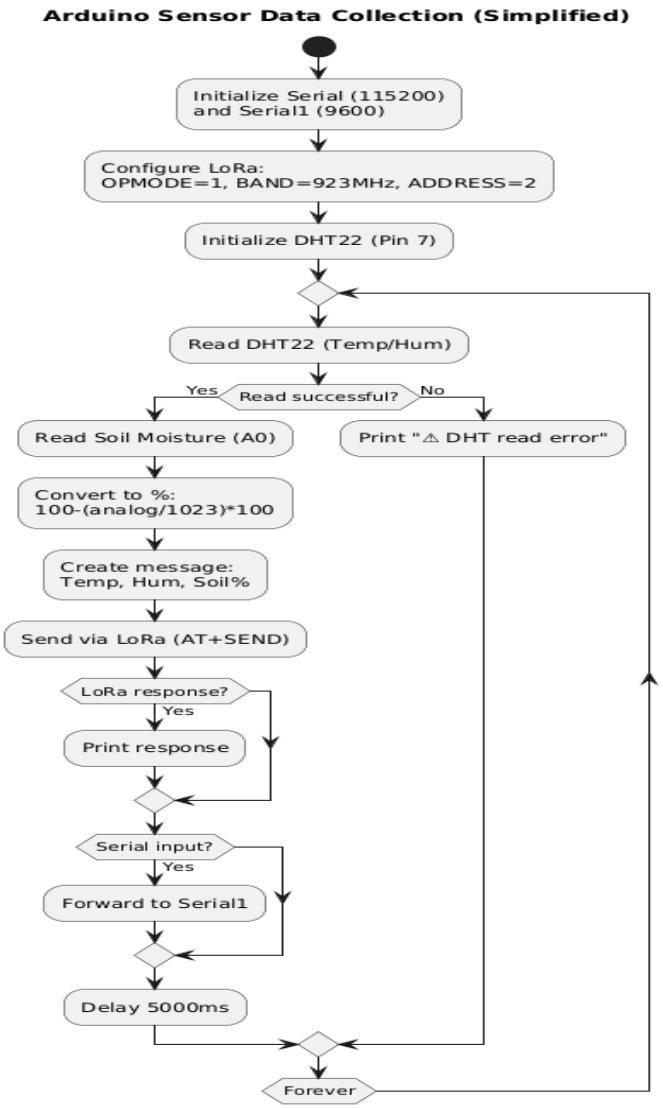
Pin Connection Table – Arduino Mega (LoRa Receiver) with ESP32:

Component Name	Pin on Component	Pin on Arduino Mega	Description
Arduino Mega (LoRa Receiver)	-	-	Microcontroller for receiving data via LoRa and sending it to ESP32.
LoRa Receiver	VCC	3.3V	Power supply to the LoRa module (ensure 3.3V for proper operation).
	GND	GND	Ground connection for the LoRa module.
	TX	D10	Receive data from LoRa transmitter (RX on LoRa → TX on Arduino).
	RX	D11	Transmit data to LoRa transmitter (TX on LoRa ← RX on Arduino).
ESP32 Module	VCC	3.3V	Power supply to the ESP32 module (3.3V).
	GND	GND	Ground connection for the ESP32 module.
	TX	D2	Transmit data from ESP32 to cloud or other devices.
	RX	D3	

CHAPTER 4

Workflow and Program code

1.1 -Work Flow-



Steps :

1. Data Collection (Using Arduino Mega and Sensors)

- **Arduino Mega** is connected to multiple **sensors** like:
    - Soil Moisture Sensor
    - Temperature and Humidity Sensor (like DHT11/DHT22)
- Arduino Mega **reads** the sensor data at regular intervals.
- The **raw sensor readings** (e.g., soil moisture = 40%, temp = 30°C) are collected and **prepared** for transmission.

---

2. Data Transferring to Cloud (Using LoRa Transmitter and LoRa Receiver)

- The **Arduino Mega** is connected to a **LoRa transmitter module** (like SX1278).
- The collected data is **sent wirelessly** using LoRa technology from the Arduino Mega to a **LoRa Receiver**.
    - **LoRa** allows **long-range** (several kilometers) and **low-power** data transmission, perfect for remote landslide-prone areas.
- **At the receiving side**, the **LoRa Receiver** is connected to an **ESP32** microcontroller.
- The **ESP32** receives the sensor data from the LoRa receiver via UART (Serial Communication).

---

3. Storing of Data in Cloud (Using ESP32)

- The **ESP32** connects to the **Internet** via **Wi-Fi**.
- It **uploads** the received sensor data to a **cloud server** or a **Firebase Realtime Database**.
    - The data is stored securely.
    - The cloud acts as a **database** where historical and live sensor readings are available.

Hardware device Roles:

Device	Role
Arduino Mega + Sensors	Collects raw environmental data.
LoRa Transmitter	Sends data wirelessly from Arduino to receiver.
LoRa Receiver	Receives data and passes it to ESP32.
ESP32	Connects to Wi-Fi, uploads data to cloud server.
Cloud Server	Stores data

4.2-Program code---

TRAMSMITTER CODE :

```
#include <DHT.h>
#include <SPI.h>

// LoRa Serial: Serial1 (Pins 18 TX1, 19 RX1)

#define DHTPIN 7
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define SOIL_MOIST A0
                                // Tilt sensor on digital pin 5

// Variables
int tilt = 0;
float hum = 0.0, temp = 0.0;
float moisture_percentage = 0.0;

String outgoingData;
int stringLength;

void setup() {
  Serial.begin(115200);
  Serial1.begin(9600);
  delay(2000);


  Serial.println("     Initializing LoRa Configuration...");

  Serial1.print("AT\n");
  delay(100);
  Serial.print("Interface: "); Serial.println(Serial1.readString());

  Serial1.print("AT+OPMODE=1\n");
  delay(100);
  Serial.print("OPMODE: "); Serial.println(Serial1.readString());
```

```
  Serial1.print("AT+BAND=923000000\n");
  delay(100);
  Serial.print("BAND: "); Serial.println(Serial1.readString());

  Serial1.print("AT+ADDRESS=2\n"); // This is the sender node
  delay(100);
  Serial.print("ADDRESS: "); Serial.println(Serial1.readString());

  Serial.println(" LoRa Node Configured");
  Serial.println("------------------------------------------------");
  Serial.println(" Sensor Pins:");
  Serial.println(" DHT22      : Pin " + String(DHTPIN));
  Serial.println("ç Soil Moisture : Pin " + String(SOIL_MOIST));

        dht.begin();
}

                  ------------------------------------------------
void loop() {
  // Reading from DHT22
  hum = dht.readHumidity();
  temp = dht.readTemperature();

  if (isnan(hum) || isnan(temp)) {
    Serial.println("  Failed to read from DHT sensor!");
    return;
  }

  // Soil Moisture
  sensor_analog = analogRead(SOIL_MOIST);
  moisture_percentage = (100 - ((sensor_analog / 1023.00) * 100));

  // Prepare the outgoing message
  outgoingData =
              ", Temp: " + String(temp, 2) +
              ", Hum: " + String(hum, 2) +
              ", Soil_%moist: " + String(moisture_percentage, 2) +
```

```
  stringLength = outgoingData.length();

  // Print to serial
  Serial.println(" Sending to LoRa Receiver:");
  Serial.println("Message: " + outgoingData);
  Serial.println("Length : " + String(stringLength));
  Serial.println("-------------------------------------------------");

  // Transmit to LoRa Receiver (Address 1)
  Serial1.print("AT+SEND=1," + String(stringLength) + "," + outgoingData + "\n");

  // Check LoRa response
  if (Serial1.available()) {
    Serial.println(" LoRa Response:");
    Serial.println(Serial1.readString());
  }

  // Serial passthrough
  if (Serial.available()) {
    Serial1.print(Serial.readString());
  }

  delay(5000); // Adjust delay as needed
}
```

Code Explanation:

---

1. Setup: Hardware and Libraries

- Sensors connected:
  - DHT22 (Temperature & Humidity) on pin D7
  - Soil Moisture on A0
- LoRa module is connected via Serial1 (pins 18 TX1 and 19 RX1 on Arduino Mega).
- DHT sensor library is used to interact with DHT22.
- SPI and SD libraries are included but SD is not yet used in this code (it might be for future extension like logging to SD card**).**

---

2. Setup Function (setup())

- Initialize Serial (115200 baud) for printing to the computer.
- Initialize Serial1 (9600 baud) for communicating with the LoRa module.
- Configure LoRa module via AT commands:
  - AT → Check if LoRa is responding.
  - AT+OPMODE=1 → Set mode to transmit.
  - AT+BAND=923000000 → Set LoRa frequency (923 MHz).
  - AT+ADDRESS=2 → Set sender node address as 2.
- Print all sensor pins and confirm LoRa configuration.
- Start DHT22 sensor.

---

3. Loop Function (loop())

a) Read Sensor Data

- Read temperature and humidity from DHT22.
- Read soil moisture (analog voltage → converted to % wetness).

---

b) Prepare Message

- Create a formatted string like:

Temp: 28.50, Hum: 65.30, Soil_%moist: 70.10

Calculate the length of the outgoing message.

---

c) Transmit Data to LoRa Receiver

- Use:

Serial1.print("AT+SEND=1,<length>,<message>\n");

  - Send to Address 1 (receiver address).
- Print confirmation messages on Serial Monitor.

---

d) Optional: Manual Pass-through

- If you type anything into Serial Monitor manually, it forwards it to LoRa (Serial1).

---

4. Delay

- Wait for 5 seconds before next reading and transmission.

RECEIVER CODE :

```
#include <SPI.h>

void setup() {
  Serial.begin(115200);      // USB Monitor
  Serial1.begin(9600);       // LoRa Module (TX = 18, RX = 19)
  Serial2.begin(9600);       // To ESP32 (TX2 = 16, RX2 = 17)

  delay(2000);
  Serial.println("      Configuring LoRa Receiver...");

  Serial1.print("AT\n");
  delay(100);
  Serial.println("Interface: " + Serial1.readString());

  Serial1.print("AT+OPMODE=1\n");
  delay(100);
  Serial.println("OPMODE: " + Serial1.readString());

  Serial1.print("AT+BAND=923000000\n");
  delay(100);
  Serial.println("BAND: " + Serial1.readString());

  Serial1.print("AT+ADDRESS=1\n");
  delay(100);
  Serial.println("ADDRESS: " + Serial1.readString());

  Serial.println("  LoRa Receiver Ready. Waiting for data...\n");
}

void loop() {
  if (Serial1.available()) {
    String incoming = Serial1.readStringUntil('\n');
    incoming.trim();
    Serial.println("  Raw LoRa Data: " + incoming);

    int payloadStart = incoming.indexOf("Tilt:");
    if (payloadStart >= 0) {
      String payload = incoming.substring(payloadStart);
      int endIdx = payload.indexOf(",-", payloadStart);
```

```
    if (endIdx >= 0) {
      payload = payload.substring(0, endIdx);
    }

    int tempIdx = payload.indexOf("Temp:");
    int humIdx  = payload.indexOf("Hum:");
    int soilIdx = payload.indexOf("Soil_%moist:");

    if (tiltIdx >= 0 && tempIdx >= 0 && humIdx >= 0 && soilIdx >= 0 && rainIdx >= 0) {
      String temp = payload.substring(tempIdx + 5, humIdx - 1);
      String hum  = payload.substring(humIdx + 4, soilIdx - 1);
      String soil = payload.substring(soilIdx + 13, rainIdx - 1);

      String cleaned =  "Temp:" + temp + ",Hum:" + hum + ",Soil:" + Serial.println(" ■ Data Received: " +
      cleaned);

      Serial2.println(cleaned);  // Send to ESP32
    } else {
      Serial.println(" ■ Parsing error: Some values missing.");
    }
   }
  }

  if (Serial.available()) {
    Serial1.print(Serial.readString());
  }
}
```

1. setup()

- Initialize Serial ports:
    - Serial: To see messages on your computer via USB (115200 baud).
    - Serial1: Communication with the LoRa module (9600 baud).
    - Serial2: Sending cleaned data to an ESP32 (9600 baud).
- Configure the LoRa module:
    - AT check.
    - Set to OPMODE 1 (LoRa mode).
    - Set Frequency Band (923 MHz).
    - Set Address 1 (receiver node address).

After setup:

LoRa receiver is ready and waiting for incoming LoRa data.

---

2. loop()

- If data is received from the LoRa module:
    - Read the incoming LoRa message (Serial1.readStringUntil('\n')).
    - Trim whitespace.
    - Print the raw received LoRa data.
- Parse useful parts of the message:
    - Find where Tilt:, Temp:, Hum:, Soil_%moist.
    - Extract values between these labels.
    - Create a cleaned string like:

      Temp:30.25,Hum:58.50,Soil:39.00
- Print the cleaned string to Serial Monitor.
- Send the cleaned string to ESP32 over Serial2.println().

Esp32 code :

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "wifi user name";
const char* password = "password";

// Replace with your actual Google Apps Script Web App URL
const char* scriptURL = "url script link";

void setup() {
  Serial.begin(115200);               // USB Serial Monitor
  Serial2.begin(9600, SERIAL_8N1, 17, 16);  // RX=17, TX=16 from Mega

  WiFi.begin(ssid, password);
  Serial.println("  Connecting to WiFi...");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\n WiFi connected.");
}

void loop() {
  if (Serial2.available()) {
    String rawData = Serial2.readStringUntil('\n');
    rawData.trim();
    Serial.println(" Received from Mega: " + rawData);

    // Example: Tilt:12,Temp:25.3,Hum:68,Soil:40,Rain:2
    int tempIdx = rawData.indexOf("Temp:");
    int humIdx  = rawData.indexOf("Hum:");
    int soilIdx = rawData.indexOf("Soil:");

    if (tiltIdx >= 0 && tempIdx >= 0 && humIdx >= 0 && soilIdx >= 0 && rainIdx >= 0) {
      String temp = rawData.substring(tempIdx + 5, humIdx - 1);
      String hum  = rawData.substring(humIdx + 4, soilIdx - 1);
      String soil = rawData.substring(soilIdx + 5, rainIdx - 1);
```

```
    // Build JSON
    String json = "{";
    json += "\"Temp\":\"" + temp + "\",";
    json += "\"Hum\":\"" + hum + "\",";
    json += "\"Soil\":\"" + soil + "\",";
    json += "}";

    Serial.println("  Sending JSON to Google Sheets: " + json);

    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(scriptURL);
      http.addHeader("Content-Type",  "application/json");

      int httpResponseCode = http.POST(json);
      if (httpResponseCode > 0) {
        String response = http.getString();
        Serial.println("  Response: " + response);
      } else {
        Serial.println("+ Error sending data. Code: " + String(httpResponseCode));
      }
      http.end();
    } else {
      Serial.println("ı WiFi not connected.");
    }
   } else {
    Serial.println("ı Data format invalid or incomplete.");
   }
  }

  delay(500); // Slight delay to avoid rapid looping
}
```

This ESP32 code does **3 main things**:

1. Connects to **WiFi**.
2. Receives **sensor data** from Arduino Mega via **Serial2**.
3. Sends that data as **JSON** to a **Google Apps Script Web App**, which then probably logs it into a **Google Sheet**.

Included Libraries

#include <WiFi.h>
#include <HTTPClient.h>

- WiFi.h: For ESP32 to connect to WiFi.
- HTTPClient.h: Lets you send HTTP POST requests (to Google Apps Script URL in this case).

WiFi Credentials and URL

const char* ssid = "wifi user name";
const char* password = "password";
const char* scriptURL = "https://script.google.com/macros/s/.../exec";

- These are your:
  - **WiFi SSID and Password** (to connect to the internet).
  - **Google Apps Script Web App URL** where the ESP32 will send sensor data.

---

setup() Function

Serial.begin(115200);
Serial2.begin(9600, SERIAL_8N1, 17, 16);

- Serial: Communicates with **your computer** (for debugging).
- Serial2: Communicates with **Arduino Mega** using pins **RX=17** and **TX=16** at **9600 baud**.

WiFi.begin(ssid, password);
Serial.println(" Connecting to WiFi...");

- Start connecting to the given WiFi.

cpp
CopyEdit
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("\n WiFi connected.");

- Wait in a loop until WiFi is **successfully connected**, printing ... every 0.5 seconds.

---

loop() Function

if (Serial2.available()) {
  String rawData = Serial2.readStringUntil('\n');
  rawData.trim();

- If ESP32 receives **a full line of data** from Arduino Mega, it stores it in rawData.
- trim() removes unwanted newlines or spaces.

Example rawData might look like:
Temp:31.40,Hum:59.70,Soil:0.39

Parsing the Data

```
int tempIdx = rawData.indexOf("Temp:");
int humIdx  = rawData.indexOf("Hum:");
int soilIdx = rawData.indexOf("Soil:");
```

Finds **where** each value starts in the string using .indexOf().

Extracting Values

```
String temp = rawData.substring(tempIdx + 5, humIdx - 1);
String hum  = rawData.substring(humIdx + 4, soilIdx - 1);
String soil = rawData.substring(soilIdx + 5, rainIdx - 1);
```

- Uses .substring() to extract just the **numbers**, leaving out the labels like "Temp:".

For example, from this string:
Tilt:0,Temp:31.40,Hum:59.70,Soil:0.39
extracts:
temp = "31.40"
hum = "59.70"
soil = "0.39"

Making a JSON string

```
String json = "{";
json += "\"Temp\":\"" + temp + "\",";
json += "\"Hum\":\"" + hum + "\",";
json += "\"Soil\":\"" + soil + "\",";
json += "}";
```

- Combines all the values into a **JSON format** string.

Example:
json

```
{
  "Temp": "31.40",
  "Hum": "59.70",
  "Soil": "0.39"
}
```

Send JSON to Google Apps Script

```
if (WiFi.status() == WL_CONNECTED) {
```

```
HTTPClient http;
http.begin(scriptURL);
http.addHeader("Content-Type", "application/json");

int httpResponseCode = http.POST(json);
```

- If ESP32 is online:
  - It opens a **connection** to your **Google Script URL**.
  - Adds a **JSON header**.
  - Sends the **JSON data** using .POST(json).

Response Handling

```
if (httpResponseCode > 0) {
  String response = http.getString();
  Serial.println("Response: " + response);
} else {
  Serial.println("+ Error sending data. Code: " + String(httpResponseCode));
}
http.end();
```

- If Google Sheet received it:
  - Print the server response (Success, usually).
- If failed:
  - Show the HTTP error code.

Invalid Format Handling

```
} else {
  Serial.println("ι Data format invalid or incomplete.");
}
```

- If the input string doesn't have all required parts (Tilt, Temp, etc.), it skips sending.

Delay

```
delay(500);
```

- Prevents flooding the Google server with too many requests too quickly.

CHAPTER 5

Result and Outcome

5.1 –Result

1. Monitoring Environmental Conditions:

- The system continuously monitors soil conditions and atmospheric parameters in real-time.
- The sensors provide accurate readings:
  - Tilt Sensor: Detects ground tilt, indicating possible land movement.
  - Soil Moisture Sensor: Measures the moisture content in the soil.
  - DHT11 Sensor: Measures atmospheric temperature and humidity.

---

2. Landslide Risk Detection:

- The system detects potential landslide conditions when:
  - Tilt sensor detects unusual inclination (tilt value deviates from safe range).
  - Soil moisture exceeds the saturation threshold (e.g., soil moisture > 0.6).
  - Temperature rises or falls abnormally affecting soil stability (e.g., > 40°C or < 5°C).
  - Humidity drops drastically, indicating dry unstable soil.

---

3. Alert Activation:

- When critical conditions are detected:
  - A buzzer is activated to emit a loud sound for 2 seconds, warning people nearby of possible ground instability.

---

4. Sending Alerts to cloud storage:

- Upon detecting a risky situation, the system:
  - Transmits the sensor data via LoRa to an ESP32 receiver.
  - The ESP32 collects the data and uploads it to Google Firebase Realtime Database for storage and remote monitoring.
- The data includes:
  - Tilt status (Normal/Tilted)
  - Temperature and Humidity readings
  - Soil moisture percentage
  - Timestamp (in IST)

---

5. Real-Time Data Updates:

- The ESP32 ensures continuous real-time data updates to Firebase.
- The system checks for new sensor readings every 2 seconds to maintain constant monitoring.

6. Continuous Monitoring and Reset:

- After sending data:
  - The system resets internally and continues monitoring for the next environmental changes.
- This cycle ensures continuous detection without needing manual resets.

Real-Life Application Results

Scenario 1: Landslide Risk Detected

- When critical soil and weather conditions are observed:
  - Buzzer sounds locally to warn people nearby.
  - Data is uploaded to Firebase, allowing authorities to be notified in real-time.
  - Early action can be taken to evacuate and prepare for possible landslide events.

Scenario 2: No Risk Detected

- If no abnormal conditions are present:
  - The system silently continues monitoring.
  - No unnecessary alerts are triggered, ensuring energy efficiency and false alarm reduction.

Impact of Results

- Early Warning System:
  - Allows detection of unstable ground conditions early, minimizing potential disasters.
- Accurate Real-Time Monitoring:
  - Continuous environment sensing and immediate updates to Firebase enable fast remote decisions.
- Community Safety:
  - Local buzzer alerts and remote notifications help safeguard communities in landslide-prone areas.
- Scalability:
  - The LoRa-based architecture allows monitoring over large terrains without relying on heavy internet infrastructure.

Serial monitor output of Lora Transmitter section :

```
Initializing LoRa Configuration...
Interface: +OK
OPMODE: +OK
BAND: +OK
ADDRESS: +OK
LoRa Node Configured
------------------------------------------------
Sensor Pins:
DHT22       : Pin 7
Soil Moisture : Pin A0
```

Sending to LoRa Receiver:
Message: Temp: 29.42, Hum: 58.73, Soil_%moist: 65.00
Length : 68

LoRa Response:
+OK

Then after 5 seconds:

Sending to LoRa Receiver:
Message: Temp: 29.50, Hum: 58.50, Soil_%moist: 64.00
Length : 68

LoRa Response:
+OK

Serial monitor output of Lora Receiver section :

Configuring LoRa Receiver...
Interface: +OK
OPMODE: +OK
BAND: +OK
ADDRESS: +OK
LoRa Receiver Ready. Waiting for data...

Raw LoRa Data: +RCV=1,62,Tilt:0, Temp:30.25, Hum:58.50, Soil_%moist:39.00, Rain_%:5,-29,9
Data Received: Temp:30.25,Hum:58.50,Soil:39.00

Serial Monitor output of Esp 32 sending data to cloud :

Connecting to WiFi...
.....
WiFi connected.
Received from Mega: Tilt:0,Temp:31.40,Hum:59.70,Soil:0.39,Rain:0
Sending JSON to Google Sheets: {"Temp":"31.40","Hum":"59.70","Soil":"0.39"}
Response: Success

5.2 – Challenges Faced:

Designing and deploying a real-time landslide prediction system involves multiple technical and environmental challenges. These issues must be carefully managed to ensure system reliability, scalability, and practical usefulness.

---

1. Sensor Reliability and Environmental Noise

Ensuring reliable and consistent data collection from sensors like soil moisture, temperature, humidity, and tilt sensors presents significant challenges:

- **Fluctuations** in readings due to weather changes (e.g., rain, sunlight, fog).
- **Sensor drift** over time that affects accuracy.
- **Ground disturbances** (e.g., animals or human activity) causing temporary false readings.

Regular calibration and robust hardware integration are essential to minimize these errors.

---

2. Data Transmission and LoRa Communication

The system relies on LoRa-based wireless transmission to send data from remote areas to a receiver, which introduces challenges:

- **Signal loss or interference** due to terrain (hills, trees, buildings).
- **Limited range** in dense forested areas or valleys.
- **Packet collisions or partial data** from multiple nodes in a network.

Optimizing transmission frequency, packet formatting, and antenna placement is necessary for stable performance.

---

3. Real-Time Prediction and False Alerts

Running machine learning models in near real-time requires fast, accurate decisions. However:

- Incomplete or inconsistent sensor data can lead to **false positives** or **missed detections**.
- Threshold-based decisions (e.g., humidity < 30%) may not work in all geographic locations.

Combining rule-based detection with LSTM-based learning reduces false alerts but still requires fine-tuning.

---

4. Power and Hardware Limitations

Operating in remote, landslide-prone regions introduces infrastructure constraints:

- **Power supply limitations**, especially in forested or hilly areas.
- **Weatherproofing** the hardware to handle extreme rain, humidity, or landslide debris.
- **Cost-effectiveness** while using durable components and long-range communication modules.

Designing a low-power, rugged system that's still affordable is crucial for real-world deployment.

CHAPTER 6

Limitations and Future Improvements

6.1 Limaitation :

Although the landslide prediction system demonstrates promising results, it still faces several limitations that affect its performance, scalability, and reliability, especially when deployed in rugged or remote terrains.

1. Limited Geographical Coverage

The system's prediction accuracy depends on localized environmental data, making wide-area implementation complex:

- Localized Sensor Range:
  - Sensors such as soil moisture, tilt, temperature, and humidity can only monitor small regions (typically within a few meters).
  - Large-scale deployment would require numerous sensor nodes across a wide terrain.
- Scaling Challenges:
  - Installation Difficulty:
    - Deploying sensors across hilly and forested landslide-prone zones is physically challenging.
  - Maintenance Overhead:
    - Each node requires periodic battery replacement or calibration, which is labor-intensive and impractical in hard-to-reach areas.
  - High Cost of Expansion:
    - A dense network of robust sensors and communication hardware significantly increases setup and maintenance costs.

2. Communication Dependency on LoRa and Internet Gateways

The system uses LoRa-based communication and internet connectivity for transmitting real-time data to cloud platforms like Firebase:

- Limitations of LoRa:
  - Effective only under line-of-sight conditions and limited range in mountainous or heavily forested areas.
  - Susceptible to signal degradation due to obstacles or weather changes.
- Cloud Dependency:
  - Requires stable internet access at the receiver node (ESP32).
  - Outages in internet connectivity or server-side issues (e.g., Firebase downtime) can halt

data transmission and prediction updates.

- Limited Two-Way Communication:
  - Current design may lack robust acknowledgment or feedback from the server, leading to potential data loss in case of communication failure.

---

3. Sensor Noise and False Triggers

Environmental sensors are prone to inconsistent data due to dynamic conditions in real-world settings:

- Inaccurate Readings:
  - Sudden environmental changes (e.g., heavy rain, animal movement) may cause spikes in sensor values.
  - Tilt sensors might register ground vibration not related to landslides.
- Consequences of False Alerts:
  - Unnecessary warnings reduce trust in the system.
  - Alert fatigue can lead users to ignore genuine high-risk predictions.
- Mitigation Challenges:
  - Filtering algorithms or ML models need more real-world data to effectively distinguish noise from real threats.

---

4. Power Supply Constraints in Remote Areas

Operating in isolated, off-grid areas poses serious energy challenges:

- Power-Hungry Components:
  - LoRa modules, GPS, and ESP32 require continuous power for operation and transmission.
- Battery Dependency:
  - Current system relies on rechargeable batteries, which may not last long in remote field deployment without regular human intervention.
- Solar Panel Limitations:
  - In hilly or densely vegetated regions, sunlight is inconsistent due to shade and weather.
- Future Energy Considerations:
  - Integration of ultra-low-power microcontrollers, efficient solar charge controllers, or hybrid energy sources is essential for sustainable long-term operation.

6.2—Future Improments ---

The landslide prediction system has significant potential for expansion and enhancement through technological innovation, better infrastructure, and integration with broader disaster management frameworks. The following areas highlight key directions for future improvement:

---

1. Integration of More Advanced Machine Learning Models other than LSTM :

To improve accuracy and reduce false predictions, machine learning (ML) algorithms can be introduced:

- **Data-Driven Prediction:**
  - Train ML models using historical sensor data (soil moisture, tilt, temperature, etc.) to predict landslide probability more precisely.
- **Pattern Recognition:**
  - ML models can detect patterns or anomalies that traditional threshold-based systems may miss.
- **Adaptive Systems:**
  - Models can auto-adjust to environmental changes, learning and adapting over time for improved reliability.
- **Potential Models:**
  - Use of Random Forest, SVM, or lightweight neural networks deployable on microcontrollers.

---

2. Real-Time Mobile App for Public Alerting

Creating a mobile application would significantly enhance system usability and public engagement:

- **Instant Alerts:**
  - Deliver real-time SMS or push notifications to nearby residents, government authorities, or disaster response teams.
- **Live Dashboard:**
  - Show live sensor data, risk level, and historical trends on a map-based interface.
- **Two-Way Communication:**
  - Allow users to report visible cracks, soil movement, or recent rain events, enhancing data accuracy.

---

3. Integration with Government Disaster Management Portals

The system can be aligned with national and local disaster alert systems for more coordinated response:

- **Automated Data Sharing:**
  - Real-time transmission of landslide warnings to government platforms (e.g., NDMA, IMD).
- **Collaborative Mapping:**
  - Integrate with GIS databases and satellite data to visualize landslide-prone zones dynamically.
- **Policy Support:**
  - Contribute to early warning systems and policy-making decisions related to construction or deforestation in vulnerable areas.

---

4. Expansion Using IoT and Edge Computing

Upgrading the system with IoT and edge computing can reduce latency and dependence on cloud:

- **Local Decision-Making:**
  - Edge processors like ESP32-S3 or Raspberry Pi can locally process sensor data and trigger alerts without relying entirely on the cloud.
- **IoT Mesh Networks:**
  - Create decentralized networks of nodes communicating with each other for wider coverage and resilience.
- **Remote Configuration:**
  - Remotely update firmware and threshold parameters over-the-air (OTA) without physical access.

---

5. Multi-Sensor Fusion for Enhanced Accuracy

Combining data from various types of sensors can offer a more holistic and reliable prediction:

- **Additional Sensors:**
  - Incorporate rainfall sensors, vibration sensors, and barometric pressure sensors.
- **Cross-Validation:**
  - Use sensor fusion algorithms to confirm landslide indicators across multiple sources.
- **Redundancy:**
  - Ensure backup readings in case a single sensor fails or gives anomalous data.

---

6. Use of Renewable and Sustainable Energy Solutions

Long-term deployment in remote areas requires energy autonomy:

- **Improved Solar Solutions:**
  - Use MPPT solar charging and higher-efficiency panels.
- **Hybrid Power Systems:**
  - Combine solar with piezoelectric generators or mini wind turbines.
- **Smart Power Management:**
  - Automatically switch to low-power mode during inactivity or nighttime to preserve energy.

---

7. Deployment of Autonomous Drone Units

To monitor difficult terrains or inaccessible regions, drone-based support can be introduced:

- **Periodic Aerial Scans:**
  - Use drones to gather terrain images and sensor data from multiple points.
- **Rapid Emergency Inspection:**
  - Deploy drones to quickly inspect areas flagged by the ground system as high-risk.
- **Drone-to-Cloud Sync:**
  - Relay data from sensors to cloud via drone networks in areas with zero network coverage.

Conclusion

The **Landslide Prediction and Alert System** project represents a vital step toward enhancing disaster preparedness in landslide-prone regions. Landslides continue to pose a serious threat to human life, infrastructure, and the environment—particularly in hilly terrains and areas with high rainfall. This project demonstrates how modern sensing and communication technologies can be effectively harnessed to provide a real-time, low-cost solution for early landslide detection and response.

Key Achievements

1. **Multi-Sensor Data Acquisition**
    - The system integrates multiple environmental sensors including **Soil Moisture**, **Temperature**, **Humidity**, **Rainfall Detection**, and **Tilt** sensors.
    - Collectively, these sensors provide critical insights into the conditions leading up to a potential landslide event.

2. **LoRa and IoT-Enabled Communication**
    - Sensor data is transmitted using **LoRa modules**, enabling long-range, low-power communication suitable for remote areas.
    - The data is received by an **ESP32**, which processes and uploads it to **Firebase Realtime Database**.
    - This ensures centralized and remote access to real-time environmental parameters.

3. **Real-Time Monitoring and Alerts**
    - The system provides instant feedback through **local buzzers** and **virtual terminal output** in simulations, with potential for real-world alerting mechanisms (e.g., sirens, SMS, mobile apps).
    - Critical for enabling timely evacuation and risk mitigation actions.

4. **Low-Cost, Scalable Solution**
    - Built using cost-effective components such as **Arduino Uno**, **ESP32**, and **off-the-shelf sensors**, the system is suitable for large-scale deployment in resource-constrained settings.

References

[1] R. Subramanian, S. Muthuraj and K. Maran, "Real-Time Landslide Detection System Using Wireless Sensor Networks," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 8, no. 6, pp. 3342–3347, 2019.

[2] A. Sharma and M. Bhardwaj, "Design and Implementation of Landslide Monitoring System Using IoT," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 11, pp. 3119–3124, 2019.

[3] J. K. Sundararajan and A. Arivudainambi, "Wireless Sensor Network-Based Early Warning System for Landslides," *Journal of Earth System Science*, vol. 129, article 137, 2020, doi:10.1007/s12040-020-01403-w.

[4] H. K. Sahu, S. Panda, and B. P. Swain, "Internet of Things (IoT) Based Smart Landslide Monitoring and Alert System," *Procedia Computer Science*, vol. 167, pp. 2261–2268, 2020.

[5] R. Rawat, R. Singh and N. Rawat, "Development of a Landslide Early Warning System Using Low-Cost Sensors and IoT," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 4, pp. 3167–3172, 2019.

[6] K. K. Kumar, R. Srinivasan, and S. Natarajan, "IoT-Based Landslide Detection System Using Smart Sensors and Cloud Computing," *Materials Today: Proceedings*, vol. 45, pp. 4314–4319, 2021. doi:10.1016/j.matpr.2020.10.766.

[7] B. Prakash, M. R. Kiran, and G. Vinodhini, "Landslide Monitoring Using Wireless Sensor Networks and Machine Learning Techniques," *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 5, pp. 117–120, 2021.

[8] M. H. Anbazhagan and R. Kumaravel, "Wireless Sensor Networks for Real-Time Landslide Monitoring: A Survey," *Geotechnical and Geological Engineering*, vol. 40, no. 4, pp. 1865–1880, 2022. doi:10.1007/s10706-022-02005-2.