DESIGN OF PORTABLE LAND PARAMETER MEASURING DEVICE ON MACHINE LEARNING USING IoT

CONTENTS

ABSTRACT

ABSTRACT

Agriculture plays a crucial role in sustaining economies and food security, and understanding soil characteristics is vital for optimizing crop growth. This project presents the design and development of a portable land parameter measuring device that combines Internet of Things (IoT) and Machine Learning (ML) technologies to accurately measure and predict key soil parameters. The system integrates multiple sensors such as soil moisture, soil temperature, Ambient humidity and Temperature, and others connected to an ESP32 microcontroller, enabling real-time data collection and wireless transmission. A lightweight machine learning model is trained on environmental sensor data to predict critical parameters such as soil pH, electrical conductivity (EC), soil type, and suitable crops. The model is deployed via a Flask-based web server, accessible locally or through cloud hosting. The ESP32 sends sensor readings to the server, receives predictions, and displays them on an LCD screen embedded in the device, ensuring field usability without a laptop. This system addresses the challenges of traditional soil testing by providing affordable, real-time, and on-site analysis. The combination of IoT and ML not only improves agricultural decision-making but also supports precision farming, reducing costs and improving yield outcomes. The compact design makes it suitable for remote or resource-limited areas, enabling farmers and researchers to make data-driven decisions for sustainable agriculture.

INTRODUCTION

1. INTRODUCTION

1.1 Introduction to Internet of Things

The Internet of things (IoT) refers to the concept of extending Internet connectivity beyond conventional computing platforms such as personal computers and mobile devices, and into any range of traditionally “dumb” or non-internet-enabled physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled. The definition of the Internet of things has evolved due to convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the “smart home”, covering devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smart phones and smart speakers.

1.1.1 Applications

The extensive set of applications for IoT devices is often divided into consumer, commercial, industrial, and infrastructure spaces.

- **Consumer Applications**

A growing portion of IoT devices are created for consumer use, including connected vehicles, home automation, wearable technology (as part of Internet of Wearable Things (IoT), connected health, and appliances with remote monitoring capabilities.

Smart Home: IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems. Long term benefits could include energy savings by automatically ensuring lights and electronics are turned off. A smart home or automated home could be based on a platform or hubs that Control smart devices and appliances. There are also dedicated smart home hubs that are offered as standalone platforms to connect different smart home products and these include Amazon Echo, Google Home.

- **Commercial Applications**

Building and Home Automation: IoT devices can be used to monitor and control the mechanical electrical and electronic systems used in various types of buildings (e.g., public and private, industrial, institutions, or residential) in home automation and building automation systems. In this context, three main areas are being covered in literature:

- The integration of the Internet with building energy management systems in order to create energy efficient and IOT-driven "smart buildings"
- The possible means of real-time monitoring for reducing energy consumption and monitoring occupant behavior.
- The integration of smart devices in the built environment and how they might to know how to use in further applications

- **Industrial Applications**

Manufacturing: The IoT can realize the seamless integration of various manufacturing devices equipped with sensing, identification, processing, communication, actuation, and networking capabilities. Based on such a highly integrated smart cyber physical space, it opens the door to create whole new business and market opportunities for manufacturing. Network control and management of manufacturing equipment, asset and situation management, or manufacturing process control bring the IoT within the realm of industrial applications and smart manufacturing as well. The IoT intelligent systems enable rapid manufacturing of new products, dynamic response to product demands, and

real-time optimization of manufacturing production and supply chain networks, by networking machinery, sensors and control systems together. Digital control systems to automate process controls, operator tools and service information systems to optimize plant safety and security are within the purview of the IoT. But it also extends itself to asset management via predictive maintenance, statistical evaluation and measurements to maximize reliability. Smart industrial management systems can also be integrated with the Smart Grid, thereby enabling real-time energy optimization. Measurements, automated controls, plant optimization, health and safety management, and other functions are provided by a large number of networked sensors. The term industrial Internet of things (IoT) is often encountered in the manufacturing industries, referring to the industrial subset of the IoT. IoT in manufacturing could generate so much business value that it will eventually lead to the Fourth Industrial Revolution, so the so- called Industry 4.0. It is estimated that in the future, successful companies will be able to increase their revenue through Internet of things by creating new business models and improve productivity, exploit analytics for innovation, and transform workforce. The potential of growth by implementing IoT may generate $12 trillion of global GDP by 2030.

- **Agriculture:** There are numerous IoT applications in farming such as collecting data on temperature, rainfall, humidity, wind speed, pest infestation, and soil content. This data can be used to automate farming techniques, take informed decisions to improve quality and quantity, minimize risk and waste, and reduce effort required to manage crops.For example, farmers can now monitor soil temperature and moisture from afar, and evenapply IoT-acquired data to precision fertilization programs.

- **Infrastructure Applications:** Monitoring and controlling operations of sustainable urban and rural infrastructures like bridges, railway tracks and on- and offshore wind-farms is a key application of the IoT. The IoT infrastructure can be used for monitoring any events or changes in structural conditions that can compromise safety and increase risk. The IoT can benefit the construction industry by cost saving, time reduction, better quality workday, paperless workflow and increase in productivity. It can help in taking faster decisions and save money with Real-Time Data Analytics. It can also be used for

scheduling repair and maintenance activities in an efficient manner, by coordinating tasks between different service providers and users of these facilities. IoT devices can also be used to control critical infrastructure like bridges to provide access to ships. Usage of IoT devices for monitoring and operating infrastructure is likely to improve incident management and emergency response coordination, and quality of service, uptimes and reduce costs of operation in all infrastructure related areas.

1.1.2 Advantages of IoT:

- It can assist in the smarter control of homes and cities via mobile phones. It enhances security and offers personal protection.
- By automating activities, it saves us a lot of time.
- Information is easily accessible, even if we are far away from our actual location, and it is updated frequently in real time.
- Electric Devices are directly connected and communicate with a controller computer, such as a cell phone, resulting in efficient electricity use. As a result, there will be no unnecessary use of electricity equipment.
- Personal assistance can be provided by IoT apps, which can alert you to your regular plans.
- It is useful for safety because it senses any potential danger and warns users. For example, GM On Star, is a integrated device that system which identifies a car crash or accident on road. It immediately makes a call if an accident or crash is found.
- It minimizes human effort because IoT devices connect and communicate with one another and perform a variety of tasks without the need for human intervention.
- Patient care can be performed more effectively in real time without the need for a doctor's visit. It gives them the ability to make choices as well as provide evidence-based care.
- Asset tracking, traffic or transportation tracking, inventory control, delivery, surveillance, individual order tracking, and customer management can all be Made **more cost-effective with the right tracking system.**

1.2 Introduction to Machine Learning

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research

in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models to unable for the parameters that can be adapted to observed data in this way the program considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data.

I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

1.2.1 Categories of Machine Learning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modelling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both type of

unsupervised learning in the following section. As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog. Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

1.2.2 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, Al is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn. The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale". Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

1.2.3 Challenges in Machine Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of Al as whole still has a long way to go. The reason behind is that ML. has not been able to overcome number of challenges. The challenges that ML is facing currently are-

Quality of Data - Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming Task - Another challenge faced by ML. models is the consumption of time especially for data acquisition, feature extraction a retrieval.

Lack of Specialist Persons - As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No Clear Objective for Formulating Business Problems - Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of Overfitting & Underfitting - If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of Dimensionality - Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in Deployment - Complexity of the ML model makes it quite difficult to be deployed in real life.

1.2.4 Applications of Machine Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of Al and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention

- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

1.2.5 Machine Learning Techniques

Artificial Intelligence (AI) and Machine Learning (ML) encompass a wide range of techniques and methodologies that enable machines to perform tasks that typically require human intelligence. These techniques are at the heart of many modern applications, from voice assistants and chatbots to advanced medical diagnostics and autonomous vehicles.

One of the fundamental techniques in AI is the use of algorithms to process and analyse data. These algorithms can be broadly categorized into traditional AI algorithms and those used in machine learning. Traditional AI algorithms include search algorithms, which are used to navigate through data to find specific information or solutions to problems. Examples include the A* search algorithm, which is used in pathfinding and graph traversal, and the minimax algorithm, which is used in decision-making processes, particularly in game theory and competitive environments.

Machine learning algorithms, on the other hand, are designed to enable systems to learn from data and improve their performance over time. These algorithms can be divided into three main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on a labeled dataset, meaning that each training example is paired with an output label. The model learns to map inputs to the correct output based on this training data. Common algorithms used in

supervised learning include linear regression, logistic regression, support vector machines (SVM), and decision trees. Neural networks, a more advanced form of supervised learning, are particularly powerful for tasks involving complex and high-dimensional data, such as image and speech recognition.

Unsupervised learning deals with unlabeled data, meaning the algorithm

tries to find patterns and relationships within the data without any explicit instructions on what to look for. Clustering algorithms, such as k-means and hierarchical clustering, are used to group similar data points together. Dimensionality reduction techniques, like principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE), are used to reduce the number of variables under consideration and to visualize high-dimensional data. Unsupervised learning is particularly useful for exploratory data analysis and for finding hidden patterns or intrinsic structures in the data.

Reinforcement learning is a type of machine learning where an agent

learns to make decisions by performing actions in an environment to maximize some notion of cumulative reward. Unlike supervised learning, where the correct answer is provided during training, reinforcement learning relies on a reward signal to evaluate the actions taken by the agent. The agent receives feedback in the form of rewards or penalties and uses this feedback to learn the best strategy or policy to achieve its goals. This approach is highly effective in dynamic environments where the optimal actions are not always apparent and must be discovered through trial and error. Reinforcement learning has been successfully applied to a wide range of problems, including robotics, game playing, and autonomous driving. Another critical technique in AI and ML is the use of neural networks, which are designed to simulate the way the human brain processes information. Neural networks consist of layers of interconnected nodes, or neurons, that process data in a hierarchical manner. The most basic form of a neural network is the feedforward neural network, where information flows in one direction from the input layer to the output layer. More complex architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been developed to handle specific types of data and tasks. CNNs are particularly effective for image processing tasks due to their ability to capture spatial hierarchies in images, while RNNs are suited for sequential data, such as time series or

natural language, because they can maintain information about previous inputs through their recurrent connections.

Deep learning, a subset of machine learning, refers to the use of neural networks with many layers (hence "deep") to model complex patterns in data. The increased depth of these networks allows them to learn more abstract and high-level features from raw data, which has led to significant advancements in fields such as computer vision, natural language processing, and speech recognition. Training deep learning models typically requires large amounts of data and computational resources, but the results have been groundbreaking, enabling the development of AI systems that can outperform humans in certain tasks.

Overall, the techniques in AI and ML are diverse and continually evolving, driven by advances in computational power, availability of data, and innovative research. These techniques form the backbone of intelligent systems that are transforming industries and shaping the future of technology.

1.2.6 Future Directions

Artificial Intelligence (AI) and Machine Learning (ML) have profoundly impacted numerous sectors, revolutionizing how we approach problem-solving and decision-making. Their influence extends across industries such as healthcare, finance, transportation, and entertainment, driving significant advancements and efficiencies. In healthcare, AI and ML are transforming diagnostic and treatment processes. AI-driven diagnostic tools can analyze medical images with remarkable accuracy, often outperforming human radiologists in detecting diseases like cancer. Machine learning models predict patient outcomes and recommend personalized treatment plans based on vast amounts of historical data. These technologies are also accelerating drug discovery by identifying potential drug candidates and predicting their efficacy, significantly reducing the time and cost involved in bringing new medications to market. Moreover, AI-powered wearable devices continuously monitor patients' vital signs, enabling early detection of health issues and timely medical interventions.

The financial sector has also witnessed substantial changes due to AI and ML. These technologies enhance fraud detection by identifying unusual patterns in transaction data,

thereby preventing fraudulent activities. In trading, machine learning algorithms analyze market data to forecast stock prices and inform investment strategies, often executing trades at high speeds and with greater precision than human traders. Furthermore, AI-driven chatbots and virtual assistants provide personalized customer service, handling routine inquiries and transactions, which frees up human agents for more complex tasks.

Transportation is another field greatly impacted by AI and ML. Autonomous vehicles, which rely on sophisticated machine learning models to navigate and make real-time decisions, promise to reduce accidents caused by human error and improve traffic flow. AI algorithms optimize logistics and supply chain management by predicting demand, managing inventory, and routing deliveries efficiently. These advancements lead to cost savings and enhanced customer satisfaction through faster and more reliable services.

In the entertainment industry, AI and ML are used to create personalized user experiences. Streaming services like Netflix and Spotify employ machine learning algorithms to analyse users' viewing and listening habits, recommending content tailored to individual preferences. AI-driven tools are also used in content creation, such as generating realistic graphics in video games or producing original music and scripts. Additionally, sentiment analysis tools gauge audience reactions to movies, shows, and advertisements, helping creators and marketers refine their content to better meet audience expectations.

The impact of AI and ML extends beyond these sectors, influencing areas such as education, agriculture, and environmental conservation. In education, adaptive learning platforms use machine learning to tailor educational content to students' individual learning styles and paces, improving learning outcomes. In agriculture, AI-powered systems monitor crop health, optimize irrigation, and predict yields, enhancing productivity and sustainability. Environmental conservation efforts benefit from AI's ability to analyse data from sensors and satellite images, tracking wildlife populations and detecting illegal activities like poaching and deforestation.

Looking to the future, AI and ML are poised to drive further innovations and societal changes. One key area of development is the advancement of explainable AI (XAI), which aims to make AI systems more transparent and understandable to humans. As AI systems become more complex, ensuring that their decision-making processes are interpretable and trustworthy is crucial, particularly in high-stakes domains like healthcare and finance.

Another promising direction is the integration of AI with the Internet of Things (IoT). IoT devices generate vast amounts of data, and AI can analyse this data to derive insights and make intelligent decisions in real-time. This synergy has applications in smart cities, where AI-powered systems manage energy consumption, traffic flow, and public safety, creating more efficient and livable urban environments.

AI and ML will also play a significant role in addressing global challenges such as climate change and pandemics. Machine learning models can predict climate patterns, optimize renewable energy sources, and improve disaster response strategies. In public health, AI can assist in monitoring disease outbreaks, developing vaccines, and managing healthcare resources more effectively. Ethical considerations and regulatory frameworks will be critical as AI and ML continue to evolve. Ensuring that these technologies are developed and deployed responsibly, with attention to issues such as bias, privacy, and job displacement, will be essential to maximizing their benefits while mitigating potential risks. Collaborative efforts between governments, industry, and academia will be necessary to create policies and standards that promote ethical AI development and use.

1.2.7 Challenges and Limitations of Machine Learning

Artificial Intelligence (AI) and Machine Learning (ML) have made significant strides in recent years, but they still face several challenges and limitations. Understanding these issues is crucial for developing more robust and ethical AI systems.

1. Technical Challenges

Overfitting and Underfitting: One of the key challenges in ML is achieving a balance between overfitting and underfitting. Overfitting occurs when a model learns the noise in the training data rather than the underlying pattern, resulting in poor performance on new, unseen data. Underfitting happens when a model is too simple to capture the complexity of the data. Finding the right model complexity and regularization techniques is essential for optimal performance.

Scalability: As datasets grow larger and more complex, scaling ML algorithms becomes challenging. Training models on massive datasets requires substantial computational resources and time. Ensuring that algorithms can efficiently handle large-scale data while maintaining performance is a significant hurdle**.**

1. **Data Privacy and Security**

Privacy Concerns: AI systems often require access to large amounts of personal data, raising concerns about data privacy. Ensuring that sensitive information is protected and used responsibly is a major challenge. Techniques like anonymization and federated learning are being developed to address privacy concerns, but balancing privacy with model effectiveness remains a complex issue.

Data Security: AI systems are vulnerable to security threats such as adversarial attacks, where malicious inputs are designed to fool the model into making incorrect predictions. Ensuring the robustness of AI systems against such attacks and protecting them from data breaches is crucial for maintaining trust and security.

2. **Ethical and Bias Issues**

Algorithmic Bias: AI and ML systems can inadvertently perpetuate or even exacerbate existing biases present in the training data. For instance, biased training data can lead to biased outcomes in areas such as hiring, law enforcement, and credit scoring. Identifying and mitigating biases in AI models is essential for promoting fairness and equity.

Ethical Considerations: The deployment of AI systems raises various ethical concerns, including the potential for misuse, impacts on employment, and decision-making transparency. Ensuring that AI systems are designed and used ethically involves addressing questions about accountability, transparency, and the broader societal impact of AI technologies.

3. **Interpretability and Explainability**

Black-Box Nature: Many advanced ML models, particularly deep learning models, operate as "black boxes," meaning their internal decision-making processes are not easily interpretable. This lack of transparency can hinder trust and make it difficult to understand how decisions are made. Developing techniques for model interpretability and explainability is essential for ensuring that AI systems are transparent and their decisions can be understood and justified.

Model Interpretability: For AI systems to be widely accepted and trusted, it is crucial that their predictions and decision-making processes are interpretable by humans. Efforts to enhance model interpretability involve creating methods and tools that provide insights into

how models arrive at their conclusions, which is especially important in high-stakes domains like healthcare and finance.

4. **Societal Impact and Public Perception**

Job Displacement: The automation of tasks through AI and ML can lead to job displacement, as machines and algorithms increasingly perform tasks previously done by humans. Addressing the impact on employment and developing strategies for workforce retraining and support are important for mitigating the negative effects of automation.

Public Perception: Public perception of AI and ML can vary, with concerns about the potential misuse of technology and its impact on daily life. Building public trust involves transparent communication about how AI systems work, their benefits, and their limitations.

1.3 Introduction to Portable Land Parameter Measuring Device

A Portable Land Parameter Measuring Device powered by IoT and Machine Learning technologies presents a revolutionary approach to modern agriculture and soil management. Traditional soil testing methods are often costly, time-consuming, and require laboratory access, making them less accessible to small-scale farmers and those in rural areas. This device addresses these limitations by providing a compact, real-time, and user-friendly solution for evaluating soil characteristics directly in the field. By integrating multiple sensors with a powerful ESP32 microcontroller, the device collects data such as soil moisture, soil temperature, air humidity, and other environmental parameters. These inputs are then sent wirelessly to a lightweight machine learning model hosted locally or on the cloud using a Flask server, which analyses the data and predicts crucial values like soil pH, electrical conductivity (EC), soil type, and the most suitable crops for the given land.

The system outputs both raw sensor readings and ML-based predictions on a built-in LCD screen, making it truly portable and independent of external devices like laptops or mobile phones. It supports dual power modes using USB or battery (with a TP4056 charging circuit and MT3608 boost converter), ensuring usability in off-grid areas. By enabling smarter decisions about

irrigation, fertilization, and crop planning, this solution empowers farmers with data-driven insights, promoting sustainable and precision agriculture.

1.3.1 Introduction to Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware andsoftware. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers-students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

1.3.2 Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use forbeginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux.Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics Designers and architects build interactive prototypes, musicians and artists use it for installations andto experiment with new musical instruments. Makers, of course, use it to build many of the projects

exhibited at the Maker Faire, for example. Arduino is a key tool to learn newthings. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step-by-step instructions of a kit, or sharing ideas online with other members of the Arduino community. There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Net media's BX-24, Phid gets, MIT's Handy board, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

1.3.3 ESP32

The ESP32 microcontroller has emerged as a cornerstone in the realm of embedded systems and IoT applications. Boasting a dual-core processor and robust wireless connectivity via Wi-Fi and Bluetooth, it offers a powerful platform for developers to create innovative solutions. Its extensive array of peripherals—including GPIO pins, UART, SPI, I2C, ADC, DAC, and more—facilitates versatile interfacing with various sensors, actuators, and other components. This versatility, coupled with low power consumption and advanced security features, makes the ESP32 well-suited for

applications ranging from smart home devices and industrial automation to wearable technology and sensor networks. Supported by a vibrant development ecosystem that includes Express if's ESP-IDF and Arduino IDE, the ESP32 continues to drive advancements in connectivity, efficiency, and functionality across diverse industries.

Different ESP32 modules

- ESP32-WROOM-DA
- ESP32-WROOM-32E
- ESP32-WROOM-32UE
- ESP32-WROOM-32D
- ESP32-WROOM-32U
- ESP32-SOLO-1
- ESP32-WROVER-E
- ESP32-WROVER-IE

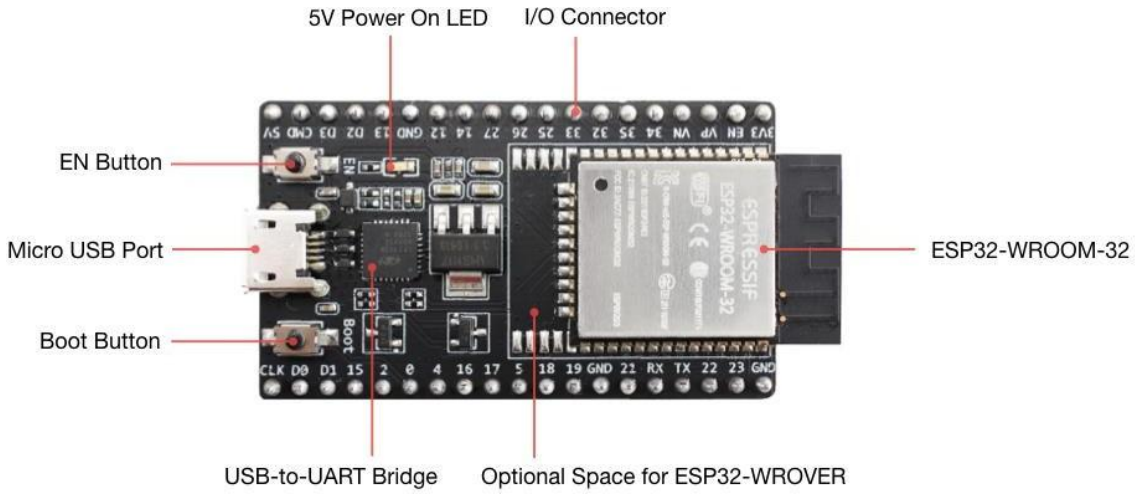

Figure 1.1: ESP32

1.3.4 Pin Configuration of ESP32

- **Power Pins**:

- **VIN**: Input voltage range is 5V. Used to power the ESP32 module.
- **3V3**: Output 3.3V voltage regulator. Used to power external components.
- **GND**: Ground pins for power and signal ground.

- **Analog Pins (ADC)**:

- **ADC1_0** to **ADC1_8**: Analog input pins with 12-bit resolution.
- **ADC2_0** to **ADC2_10**: Additional analog input pins with 12-bit resolution.

- **Digital GPIO Pins**:

- **GPIO0** to **GPIO39**: General purpose digital input/output pins. These can be configured as input or output, with support for internal pull-up or pull-down resistors.

- **Special Function Pins**:

- **UART Pins (TX, RX)**: Typically GPIO1 and GPIO3 or GPIO17 and GPIO16.
- **SPI Pins (SCK, MOSI, MISO, CS)**: Typically GPIO18, GPIO23, GPIO19, and GPIO5 respectively.
- **I2C Pins (SDA, SCL)**: Typically GPIO21 (SDA) and GPIO22 (SCL).
- **PWM Output Pins**: Available on GPIO0, GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO15, GPIO25, GPIO26, GPIO27, etc.
- **Touch Sensor Pins**: Capacitive touch sensor inputs on GPIO4, GPIO0, GPIO2, GPIO15, GPIO13, GPIO12, GPIO14, GPIO27, GPIO33, GPIO32.

➢ **Other Pins**:

- **EN**: Enable pin to power on the ESP32 module.
- **BOOT**: Boot mode selection pin.
- **IO0**: Input/output pin used for programming and boot mode selection.
- **IO33, IO32**: Additional digital input/output pins.

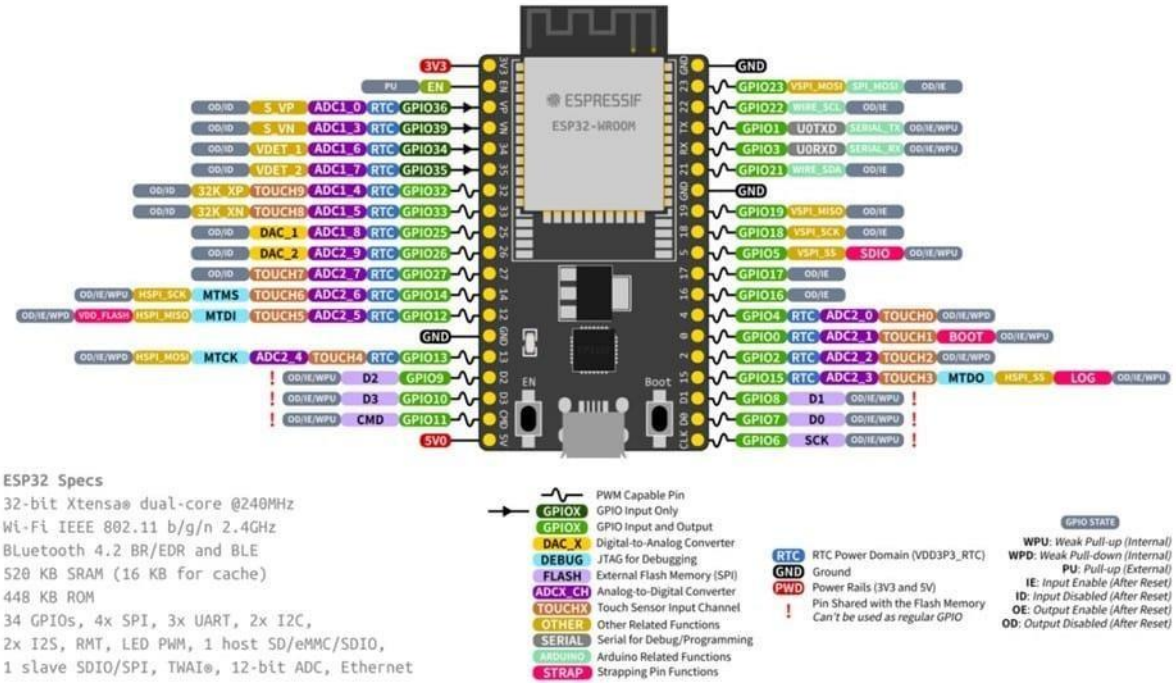

Figure 1.2: Pin Configuration ESP32

1.3.5 Soil Moisture Sensor

The soil moisture sensor is an analog sensor used to measure the volumetric water content of the soil. It consists of two conductive probes that are inserted into the soil. The electrical resistance between the two probes changes depending on the water content in the soil water being a good conductor, increases the conductivity and reduces resistance. This change is converted into an analog signal which can be read by a microcontroller like the ESP32.

1. **Operating Voltage**: 3.3V to 5V
2. **Output Type**: Analog (AO) and Digital (DO)
3. **Interface with ESP32**: Analog Output (AO) connected to any analog-capable GPIO pin (e.g., GPIO 34)

Technical Specifications:

1. **Analog output voltage:** 0V (wet) to ~4.2V (dry)
2. **Working current:** ~5 mA
3. Corrosion-prone if exposed long term to moisture — often used with coating or in temporary measurements

Working Principle:

When inserted into the soil:

1. Dry soil has high resistance, which reduces the flow of current between the probes. The sensor gives a higher voltage output.
2. Wet soil has low resistance, allowing more current to flow. The sensor gives a lower voltage output.

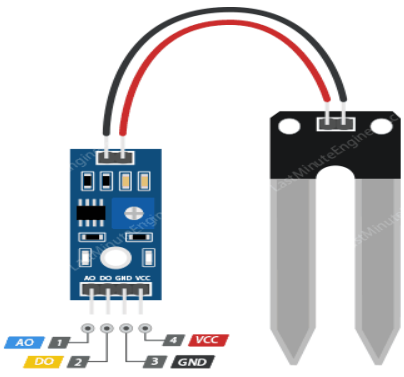

Figure 1.3: SOIL MOISTURE SENSOR

What is the use of Soil Moisture Sensor?

The soil moisture sensor is used to determine the water content present in the soil. Moisture plays a vital role in plant growth, as both excess and deficit water levels can negatively affect crop yield. By measuring the moisture level, farmers can make informed decisions about irrigation timing and frequency.

In this project, the soil moisture sensor provides real-time analog data to the ESP32 microcontroller, which maps the sensor value to a readable moisture percentage. This value is displayed on the LCD screen and also fed into the machine learning model for further prediction of soil parameters such as pH, EC, and soil type. The sensor is easy to use and install in various soil conditions and is essential for smart irrigation and water conservation efforts.

1.3.6 Soil Temperature Sensor – DS18B20

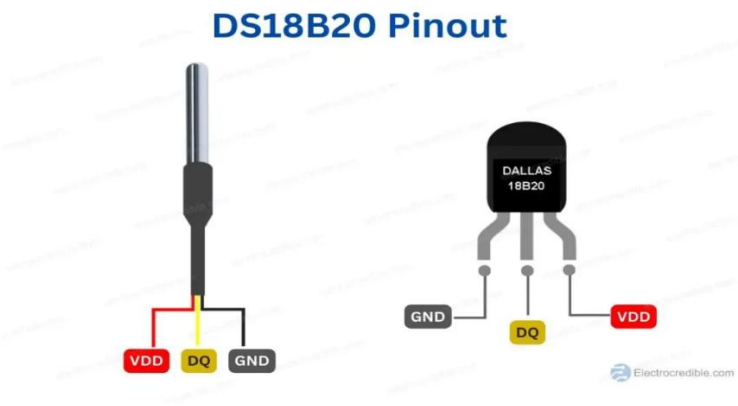

Figure 1.4: Soil Temperature Sensor

The DS18B20 is a digital temperature sensor commonly used for accurate and reliable measurement of soil temperature. It operates over a wide voltage range and communicates using the 1-Wire protocol, which allows data transmission and power delivery over a single data line. This makes it highly efficient and ideal for low-pin-count microcontrollers like the ESP32.

- **Operating Voltage**: 3.0V to 5.5V
- **Output Type**: Digital
- **Interface**: 1-Wire Protocol (requires only one data pin)
- **Temperature Range**: -55°C to +125°C
- **Accuracy**: ±0.5°C (from -10°C to +85°C)

Working Principle:

The DS18B20 sensor contains a built-in temperature sensor and internal memory to store calibration data. Once initialized, it sends temperature readings in digital format through a single GPIO pin. It requires a 4.7kΩ pull-up resistor on the data line to ensure stable communication.

This sensor is ideal for soil temperature measurement due to its waterproof version, which can be directly buried in the soil or inserted into water-diluted soil without damage.

What is the use Soil Temperature Sensor?

The soil temperature sensor (DS18B20) is a critical component in the land parameter measuring device as it provides accurate readings of the ground temperature, which is essential for agricultural planning. Soil temperature affects seed germination, root development, water absorption, and microbial activity in the soil. Monitoring this parameter helps farmers determine the best time for sowing, irrigation, and fertilization.

In this project, the DS18B20 sensor is used to continuously monitor the soil temperature and send digital readings to the ESP32 microcontroller. The data is displayed on the LCD and also serves as an input to the machine learning model, which predicts soil characteristics and suitable crops. Since the DS18B20 is waterproof and durable, it is ideal for direct insertion into the soil, ensuring reliable and consistent measurements in various environmental conditions.

1.3.7 LCD Display

A Liquid Crystal Display (LCD) is an electronic display module that uses liquid crystal technology to present information clearly and efficiently. In my project, the LCD is essential for providing real-time information and enhancing user interaction. It displays the current time and date, helping patients keep track of their medication schedule accurately. The LCD offers visual reminders and alerts when it's time to take medication, ensuring doses are not missed. Furthermore, after a dose is taken, the LCD confirms the action by displaying a message that the medication was taken successfully, providing reassurance to both patients and caregivers. Integrating an LCD into the project significantly enhances its usability and effectiveness in managing medication adherence.

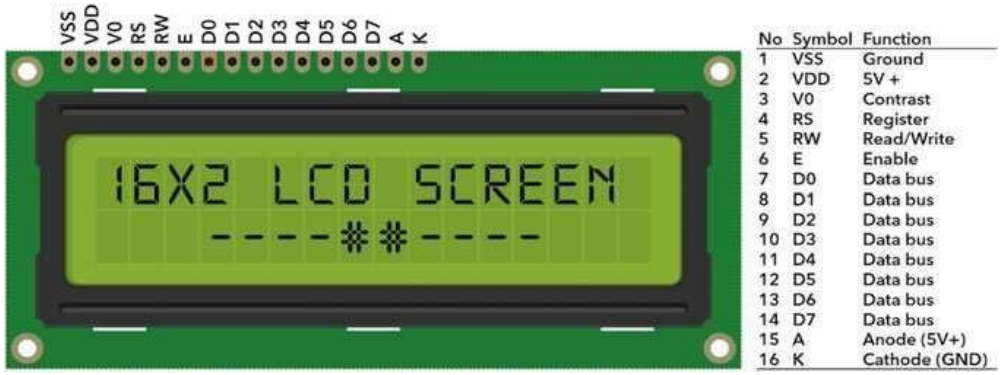

Figure 1.5: 16x2 LCD Display

1.3.8 Ambient Temperature and Humidity Sensor

The DHT11 is a basic, low-cost digital sensor used to measure ambient air temperature and relative humidity. It combines a thermistor and a capacitive humidity sensor with a microcontroller to output a calibrated digital signal. The sensor is widely used in environmental monitoring systems due to its simplicity, affordability, and acceptable accuracy for general applications.

Operating Voltage: 3V to 5.5V

Output Type: Digital

Temperature Range: 0°C to 50°C (±2°C accuracy)

Humidity Range: 20% to 90% RH (±5% accuracy)

Interface: Single-wire digital communication

Working Principle:

The DHT11 uses a capacitive humidity sensor to measure air moisture and a thermistor to measure temperature. An internal microcontroller processes the signal and sends the final calibrated values as a digital output. It communicates through a single data pin, reducing wiring complexity and saving GPIO pins on the ESP32.

Pin Configuration:

VCC – 3.3V or 5V

GND – Ground

DATA – Connected to any digital GPIO pin (e.g., GPIO 15 on ESP32)

Figure 1.6: Temperature and Humidity Sensor

What is the use of DHT11?

The DHT11 sensor is used in this project to monitor environmental air conditions, including temperature and humidity. These parameters are essential in understanding the microclimate around the soil, which affects soil moisture retention, evapotranspiration, and plant comfort. In agriculture, humidity and ambient temperature influence disease risk, crop growth, and water needs.

In the land parameter measuring device, DHT11 readings are sent to the ESP32 and displayed on the LCD. The data also contributes to the machine learning model, aiding in predicting soil characteristics and suitable crops more accurately. Its ease of use and digital interface make it an efficient choice for embedded IoT systems.

1.3.9 Jumper Wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as need though jumper wires come in a variety of colors, the colors don't actually mean anything. This means that a red jumper wire is technically the same as a black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power.

Figure 1.7: Jumper Wires

1.3.10 3.7V Li-ion Battery

The 3.7V Li-ion battery is used as the portable power source for the land parameter measuring device. It provides the necessary voltage to operate the ESP32 microcontroller and all connected sensors when the system is used in the field. The battery is rechargeable and lightweight, making it ideal for outdoor applications. It is connected through a TP4056 charging module for safe charging, and its output is boosted to 5V using an MT3608 module to meet the voltage requirements of the components.

Figure 1.8: 3.7V Battery

LITERATURE SURVEY

2 LITERATURE SURVEY

2.1 INTRODUCTION:

A portable land parameter measuring device is developed to provide farmers and agricultural professionals with accurate, real-time information about soil conditions directly from the field. Agriculture, being the backbone of many economies, faces major challenges due to changing climate, inefficient land management, and lack of timely information about soil health. Traditionally, soil analysis requires lab-based testing which is time-consuming, costly, and impractical for regular monitoring. The introduction of low-cost sensors, microcontrollers, and machine learning techniques has opened up possibilities for smart farming.

This project aims to address these challenges by designing a system that integrates IoT sensors with a machine learning model to predict soil properties such as pH, EC, and soil type, along with recommending suitable crops. When the user places the device into the soil, it instantly collects data through sensors and sends it to a trained ML model hosted on a local or cloud server. The predictions are then displayed on an LCD screen attached to the device, enabling on-the-spot decision-making without needing any external system. This intelligent device not only saves time but also helps in precision agriculture by reducing unnecessary resource usage, thereby benefiting both farmers and the environment.

Block Diagram:

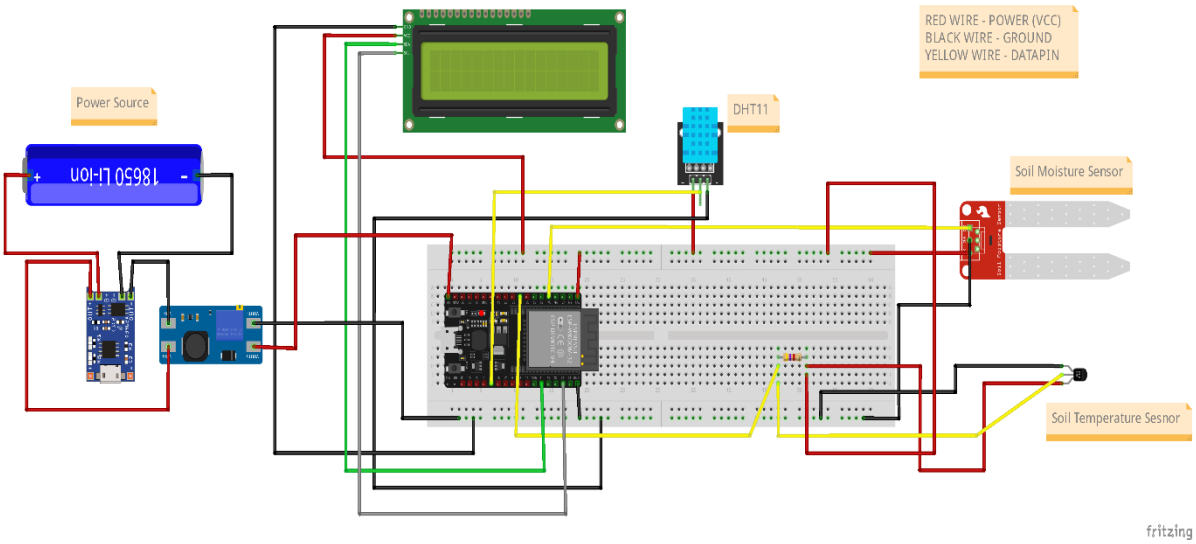

Figure 2.1: Block Diagram

2.2 History

In earlier times, assessing land or soil quality involved traditional methods that were largely manual, observational, and often unreliable. Farmers relied on seasonal patterns, physical soil appearance, and trial-and-error approaches to decide crop types and irrigation needs. With the advancement of agricultural science, laboratory testing of soil samples became the standard practice for measuring parameters like pH, moisture, and nutrient levels. However, these lab-based methods required significant time, infrastructure, and recurring costs, making them less accessible to small or remote farmers.

The emergence of Internet of Things (IoT) brought a shift in soil monitoring practices, enabling real-time sensing of environmental parameters using embedded systems and wireless communication. Simultaneously, Machine Learning (ML) emerged as a powerful tool to analyse large datasets and generate predictions. Over the past few years, researchers started integrating these two technologies to automate soil monitoring and crop recommendation systems.

Initial systems were mostly lab prototypes with bulky hardware or required cloud-dependent software. But now, with the availability of compact microcontrollers like the ESP32, low-cost sensors, and lightweight ML models, the vision of a truly portable, real-time land parameter device is becoming a practical reality. Our project builds upon this evolution by combining IoT and ML into a single, low-power device capable of functioning directly in the field, making smart agriculture more accessible and effective.

2.3 Survey

Several research works have been conducted over the years focusing on precision agriculture through the use of modern technologies like IoT, sensor networks, and machine learning. Early systems were mainly centered around sensing soil moisture or temperature and displaying the data on mobile or web interfaces. While these solutions helped in remote monitoring, they lacked the ability to interpret the data or provide actionable recommendations to farmers.

Recent advancements have brought about the integration of machine learning with sensor-based IoT systems. Studies have shown the use of classification algorithms to predict soil type, pH levels, and suitable crops based on sensor inputs such as temperature, humidity,

and moisture. Some systems have used cloud platforms for real-time monitoring, but required continuous internet connectivity, which is not always feasible in remote farming regions.

A few works have explored smart soil testing kits or portable devices, but they either relied on mobile apps for prediction or involved external hardware and lacked built-in displays. These limitations made them less user-friendly for on-field farmers.

Based on the review of existing literature, there is a need for a simple, low-cost, portable device that not only senses real-time soil parameters but also gives instant predictions using machine learning, without requiring continuous internet or complex setup. This project addresses those limitations by providing a compact, self-contained system using ESP32, sensors, a rechargeable battery, and a machine learning model that delivers predictions directly on an LCD screen.

2.4 Problem Statement

Traditional soil testing methods are often time-consuming, expensive, and require laboratory facilities, making them inaccessible for small-scale farmers and those in remote areas. These methods also do not provide immediate feedback, which delays critical decisions related to crop selection, irrigation, and fertilizer application. While some digital solutions exist, they either depend heavily on internet connectivity, mobile applications, or require bulky and non-portable equipment.

Moreover, farmers often lack a simple and affordable tool that can measure essential soil parameters like moisture, temperature, pH, and EC on the spot and provide intelligent insights such as soil type classification and suitable crop recommendations. There is a clear need for a portable, real-time, low-cost device that combines sensor technology with machine learning to provide accurate predictions directly in the field without requiring advanced technical knowledge or infrastructure.

This project aims to solve this gap by developing a portable land parameter measuring device that integrates IoT-based sensors with a machine learning model, displays instant results on an LCD screen, and operates on a rechargeable battery, making it a practical tool for smart agriculture.

2.5 Existing System

In the existing systems for soil analysis, most solutions rely on laboratory testing or app-based digital tools. Traditional methods involve collecting soil samples and sending them to agricultural labs for analysis of parameters such as pH, electrical conductivity, and nutrient levels. These tests, although accurate, are time-consuming, costly, and impractical for frequent or immediate use, especially in rural or remote areas. Some modern systems use mobile applications connected to external soil testing devices via Bluetooth or USB. These systems require smartphones and often depend on continuous internet connectivity to send data to cloud servers for processing. While such systems provide digital access to soil health data, they are limited by their dependency on external devices, app support, mobile network coverage, and technical knowledge for operation. There are also systems using IoT technology to monitor parameters like soil moisture and temperature. However, most of these are designed for fixed installations in farms or greenhouses and are not easily portable. Additionally, they often lack built-in intelligence to analyse data or provide recommendations without manual interpretation. These limitations highlight the need for a standalone, portable, and easy-to-use solution that not only senses real-time data but also intelligently interprets it without relying on mobile apps or cloud platforms.

2.6 Proposed System

The proposed system is a portable, battery-powered device designed to measure soil and environmental conditions and intelligently predict key soil characteristics using a machine learning model. It combines IoT-based sensor technology with artificial intelligence to assist farmers in assessing land conditions without the need for laboratory testing or complex mobile applications.

The system consists of an ESP32 microcontroller connected to sensors that collect real-time data, including soil moisture, soil temperature (via DS18B20), and ambient temperature and humidity (via DHT11). These input values are sent to a machine learning model, which has been trained on agricultural soil datasets. Based on these inputs, the model predicts critical outputs such as soil pH, electrical conductivity (EC), soil type, and suitable crop for cultivation.

The results are displayed instantly on a 16x2 LCD screen, making the device completely standalone and user-friendly for field use. The system is powered by a 3.7V rechargeable

battery connected through a TP4056 charging module and an MT3608 voltage booster, enabling full portability. This integrated solution addresses the limitations of existing systems by eliminating the need for external apps, cloud dependency, or laboratory facilities, while still providing accurate and real-time decision support for precision farming.

Advantages:

 The system is fully portable and battery-operated, making it ideal for use in remote agricultural fields without the need for constant power supply.

 It provides real-time measurement of environmental parameters like soil moisture, soil temperature, and air humidity, enabling immediate field-level analysis.

 Machine learning is used to intelligently predict soil pH, EC, soil type, and suitable crop based on real-time sensor inputs, eliminating the need for manual interpretation or lab testing.

 The device displays results directly on a built-in LCD screen, removing the dependency on mobile apps, internet access, or external devices.

 Cost-effective solution suitable for small-scale and marginal farmers, as it uses low-cost sensors and open-source technologies.

2.7 About Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

2.7.1 Advantages of Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a must for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

2.7.2 Characteristics of Python

- Following are important characteristics of Python Programming
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

2.7.3 New Approach for building window Software

The Python Framework simplifies Windows development. It provides developers with a single approach to build both desktop applications sometimes called smart client applications and Web-Based applications. It also developers to use the same tools and skills to develop software for a verity of system ranging from handled smart phones to large server installations.

2.7.4 Applications of Python

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read − Python code is more clearly defined and visible to the eyes.

Easy-to-maintain − Python's source code is fairly easy-to-maintain.

A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases − Python provides interfaces to all major commercial databases.

GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable − Python provides a better structure and support for large programs than shell scripting.

2.7.5 Python Programming for Machine Learning

Python is used in this project to implement machine learning for predicting soil parameters such as pH, electrical conductivity (EC), soil type, and suitable crop. It provides a simple, flexible, and powerful ecosystem of libraries for data processing, model training, and deployment via a web API.

2.7.6 Machine Learning Model Development

The machine learning model was built using the scikit-learn library in Python. A dataset containing soil moisture, temperature, and humidity values was used to train the model to predict four outputs: soil pH, EC, soil type, and crop recommendation. The data was preprocessed using pandas and numpy to handle missing values, scale features, and format input for the model.

2.7.7 Flask-based Web API

To make the machine learning model accessible to the ESP32 microcontroller, a web server was created using Flask. This server exposes a /predict endpoint that receives sensor values from the device through a POST request, processes the inputs using the

trained ML model, and sends the predictions back in JSON format. This API acts as a bridge between the hardware and the Python-based ML backend.

2.7.8 Model Deployment using Joblib

After training, the model was saved in .pkl format using the joblib library. This allows the Flask application to load the model quickly during API runtime without retraining. Joblib is ideal for saving and loading large machine learning models efficiently in Python.

2.7.9 Used Python Libraries

pandas: For data analysis and preprocessing of the dataset.
numpy: For handling numerical operations and array manipulations.
scikit-learn: Used to build and train the classification model.
joblib: Used to serialize (save) and deserialize (load) the trained ML model.
Flask: A micro web framework used to deploy the model as a REST API.
jsonify: From Flask, used to format and send API responses in JSON.
CORS (flask_cors): Allows the ESP32 to send cross-origin requests to the Flask API.

2.8 Requirement Analysis

The system requires sensors to collect real-time soil and environmental data, an ESP32 microcontroller for processing and communication, and an LCD for displaying results. A rechargeable battery setup ensures portability. On the software side, Arduino IDE is used for programming the ESP32, and a machine learning model is hosted using Python and Flask to predict soil parameters and suitable crops.

2.8.1 Functional Requirements

The functional requirements of the Portable Land Parameter Measuring Device define the complete behaviour of the IoT and Machine Learning system. It involves sensor-based data collection, wireless communication, ML-based prediction, and real-time display via both onboard LCD and a web frontend interface.

INPUT:

Sensor Data Acquisition:

Real-time measurement of soil moisture, soil temperature (DS18B20), and ambient temperature & humidity (DHT11) through ESP32-connected sensors.

Wi-Fi Communication:

The ESP32 connects to a Wi-Fi network and sends the sensor data to a Flask-based machine learning backend using a POST request to the /predict endpoint.

OUTPUT:

Machine Learning Predictions:

The Flask server receives sensor input and returns four predicted values: soil pH, EC, soil type, and suitable crop.

LCD Display Output:

The ESP32 displays both the live sensor values and predicted results directly on a 16x2 LCD screen for immediate access in the field.

/predict Endpoint (Flask API):

Receives sensor input via POST request, processes it through the ML model, and returns predictions in JSON format.

/latest Endpoint (Flask API):

Stores the most recent prediction so it can be accessed later (e.g., by a browser frontend) via a GET request.

Frontend Web Interface:

A webpage built using HTML, CSS, and JavaScript fetches the latest predicted results from the /latest endpoint and displays them in a user-friendly format for mobile or desktop viewing.

HARDWARE REQUIREMENTS:

- ESP32 NodeMCU microcontroller
- Soil Moisture Sensor
- DS18B20 Soil Temperature Sensor
- DHT11 Temperature & Humidity Sensor
- 16×2 LCD Display with I2C module
- TP4056 Charging Module
- MT3608 Boost Converter

- 3.7V Rechargeable Battery
- Jumper Wires and Breadboard or PCB

SOFTWARE REQUIREMENTS:

Embedded Programming:

Arduino IDE and Embedded C (for ESP32 control and communication)

Python & Machine Learning:

Python 3, Flask web framework, pandas, numpy, scikit-learn, joblib, flask_cors

Frontend Web Development:

HTML, CSS, JavaScript (to display predictions by calling /latest endpoint)

API Endpoints:

/predict: Receives sensor data and returns prediction (POST)

/latest: Returns most recent prediction result (GET)

Hosting:

PythonAnywhere or local server for Flask API and model hosting

2.8.2 Non-Functional Requirements

Performance: The system must provide real-time predictions with minimal delay (1–2 seconds).

Usability: Simple and user-friendly interface via LCD and web display for non-technical users.

Portability: Fully battery-powered and compact for outdoor field use.

Reliability: Consistent and accurate sensor readings and ML predictions.

Scalability: Backend can support multiple devices or requests in future expansions.

Maintainability: Code and model should be easy to update, debug, and retrain.

Security: API must be protected against unauthorized access; ensure secure communication.

Availability: Always accessible when powered; server should stay online during use.

UML MODELING

3 UML MODELING

3.1 Introduction to UML

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software system. UML was created by the Object Management Group (OMG) and UML 1.0specification drift was proposed to the OMG in January 1997.OMG is continuously making efforts to create a truly industry standard.

- UML stands for Unified Modeling Language.

- UML is different from the other common programming language such as C++, java, etc.
- UML is a pictorial language used to make software blueprints.

- UML can be described as a general-purpose visual modeling language to visualize, specify, construct and document software system.
- Although UML is generally used to model software system, it is not limited within this boundary. It is generally used to model software system as well. For example, the process flows in a manufacturing unit, etc.
- UML is not a programming language, but tools can be used to generate code in various language using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard.

3.2 Goals of UML

- UML aimed to create a universal modeling language for all types of systems, emphasizing simplicity and widespread applicability.
- UML diagrams cater not only to developers but also to business users, general audiences, and anyone interested in understanding complex systems.
- UML integrates with development processes rather than being a standalone methodology, enhancing system development and understanding.

- UML provides a straightforward approach to model and visualize both software and non-software systems in modern, intricate environments.

3.3 UML Standard Diagrams:

The elements are like components which can be associated in diverse ways to make a complete UML picture, which is known as diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real-life system. Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagram is not a new concept but it is used widely in different forms in different industries. We prepare UML diagram to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. There are two broad categories of diagram and they are again divided into subcategories:

- Structural Diagrams
- Behavioral Diagrams

3.3.1 Structural Diagrams

The structural diagram represents the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. These static parts are represented by classes, interfaces, object, components, and nodes. The four structural diagrams are:

- Class Diagram
- Object Diagram
- Component Diagram
- Deployment Diagram

3.3.2 Behavioral Diagrams

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagram captures the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. UML has the following five types of behavioral diagrams:

- Use case Diagram
- Sequence Diagram
- Activity Diagram

3.3.2.1 Use Case Diagram:

Use case describes the behavior of the system as seen from the actor's point of view. A use case diagram can portray then different types of users of a system and the many ways that they interact with system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well. Actors initiate the use cases for accessing system's functionality. When actors and use cases exchange information, they are said to Communicate. To describe a use case, we use a template composed of six fields:

- **Use Case Name:** The name of the use case.
- **Participating Actors:** The actors participating in the particular use case.
- **Entry Condition:** Condition for initiating the use case.
- **Flow of events:** Sequence of steps describing the functioning of Use case.
- **Exit Condition:** Condition for terminating the use case.
- **Quality Requirements:** Requirements that do not belong to the use case but constraint the functionality of the system.

Use Case Diagram:

Use Case Diagram for Portable Land Parameter Measuring Device

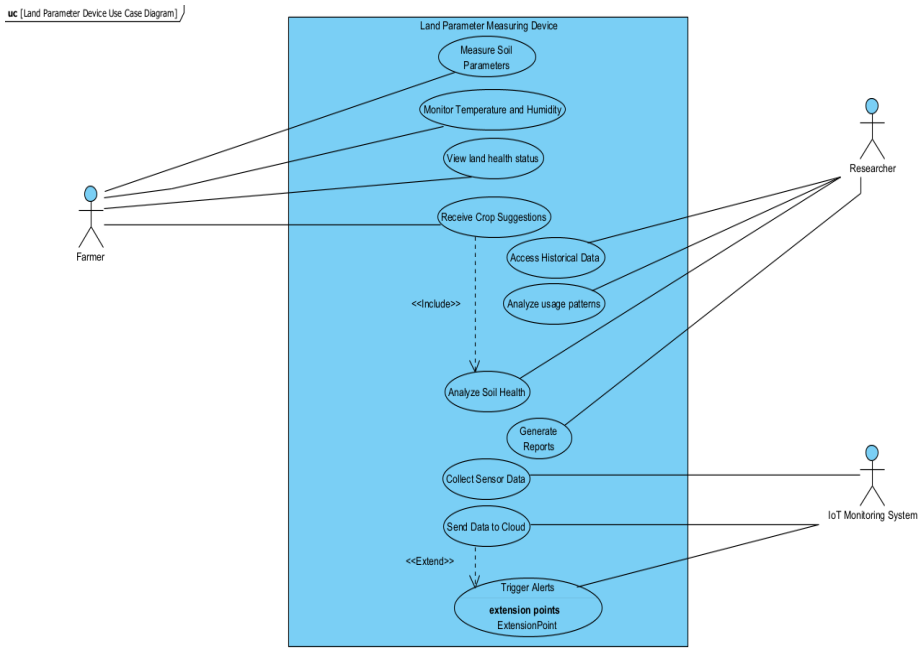

Figure 3.1: Use Case Diagram

Description:

The Portable Land Parameter Measuring Device illustrated in the use case diagram encompasses functionalities for collecting soil data, monitoring environmental conditions, analyzing land health, and receiving crop suggestions using IoT and machine learning technologies. It facilitates interactions primarily between three types of actors: Farmer, Researcher, and IoT Monitoring System.

Actors:

There are three main actors in the Land Parameter Measuring Device project:

1. Farmer:

The Farmer uses the device to measure soil parameters, monitor land conditions, and receive crop suggestions. This actor benefits from real-time insights into land health and suitable crop predictions based on current soil status.

2. Researcher:

The Researcher accesses historical soil data, analyzes usage patterns, and generates reports to support agricultural research and development. They rely on accurate and consistent data collection.

3. IoT Monitoring System:

This system interacts with the device for data collection, cloud transmission, and triggering alerts if abnormal conditions are detected. It ensures backend data flow and system health monitoring.

Use Case 1: Measure Soil Parameters

Use Case ID	UC01
Use case Name	Measure Soil Parameters
Participating Actors	Farmer
Entry Condition	The device is powered on and inserted into soil.
Flow of Events	• Sensor values (moisture, temperature, humidity) are read via ESP32. • Values are displayed on the LCD and sent to the server.
Exit Condition	Soil data is collected and displayed.
Target Requirements	Sensors should deliver accurate and stable readings with minimal calibration.

Table 3.1: Measure Soil Parameters

Use Case 2: Monitor Temperature and Humidity

Use Case ID	UC02
Use case Name	Monitor Temperature and Humidity
Participating Actors	Farmer
Entry Condition	The system starts a sensor reading cycle.

Flow of Events	• DHT11 and DS18B20 sensors provide environmental readings. • Data is shown on LCD and sent to cloud/server.
Exit Condition	Temperature and humidity data is recorded.
Target Requirements	Sensors should work reliably outdoors under varying conditions.

Table 3.2: Monitor Temperature and Humidity

Use Case 3: Receive Crop Suggestions

Use Case ID	UC03
Use case Name	Receive Crop Suggestions
Participating Actors	Farmer
Entry Condition	Sensor values are successfully sent to the server.
Flow of Events	• Data is posted to /predict endpoint. • ML model predicts pH, EC, soil type, and crop. • ESP32 receives and displays results.
Exit Condition	Suggestions are shown on LCD and/or web.
Target Requirements	Predictions must be accurate and relevant to the current soil condition.

Table 3.3: Receive Crop Suggestions

Use Case 4: Analyze Soil Health

Use Case ID	UC04
Use case Name	Analyze Soil Health
Participating Actors	Researcher
Entry Condition	Historical data is accessed.
Flow of Events	• Researcher uses backend data or /latest endpoint. • Trends are analyzed for land improvement.
Exit Condition	Reports or conclusions are generated.

Target Requirements	Data history must be clean, reliable, and timestamped.

Table 3.4: Analyze Soil Health

Use Case 5: Send Data to Cloud

Use Case ID	UC05
Use case Name	Send Data to Cloud
Participating Actors	IoT Monitoring System
Entry Condition	ESP32 has collected data.
Flow of Events	• ESP32 sends POST request to /predict. • Server processes and stores data.
Exit Condition	Data is stored and response is sent back.
Target Requirements	Server should handle requests quickly and without downtime.

Table 3.5: Send Data to Cloud

3.3.2.2 Sequence Diagram:

Sequence diagrams depict interactions between objects in a system through a timeline of messages exchanged. Each message triggers operations within receiving objects, influencing their behavior and interactions with other objects. Arguments accompanying messages provide data necessary for executing operations, binding parameters to the receiving object's functions. These diagrams help visualize the flow of operations and communication paths, aiding in the understanding and design of complex system behaviors. Sequence diagrams are invaluable for clarifying the order of operations, dependencies between objects, and the overall dynamics of object-oriented systems during development and analysis phases.

Sequence diagram for Portable Land Parameter Measuring Device:

Sequence Diagram for Farmer (User)

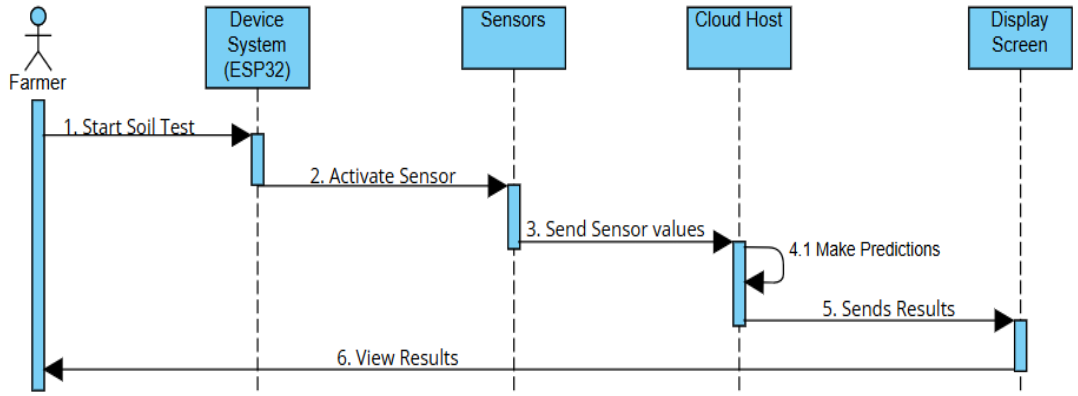

Figure 3.2: Sequence Diagram for Farmer (User)

Description:

The sequence diagram illustrates the step-by-step process of how the Portable Land Parameter Measuring Device operates using IoT and Machine Learning. The process begins when the farmer initiates the soil testing by interacting with the device. This

command activates the ESP32 microcontroller, which in turn powers and controls the connected sensors. These sensors then collect various soil and environmental parameters such as soil moisture, soil temperature, and ambient humidity. Once the sensor values are captured, they are transmitted by the ESP32 to a cloud-hosted server via a Wi-Fi connection. The cloud host contains a pre-trained machine learning model, which processes the incoming data to predict soil pH, electrical conductivity (EC), soil type, and the most suitable crop. After processing, the server sends the prediction results back to the device. The ESP32 receives the results and displays them on the onboard LCD screen, allowing the farmer to easily view and understand the current condition of the land. This sequence ensures real-time feedback and decision-making support for agricultural activities.

Sequence Diagram for Researcher

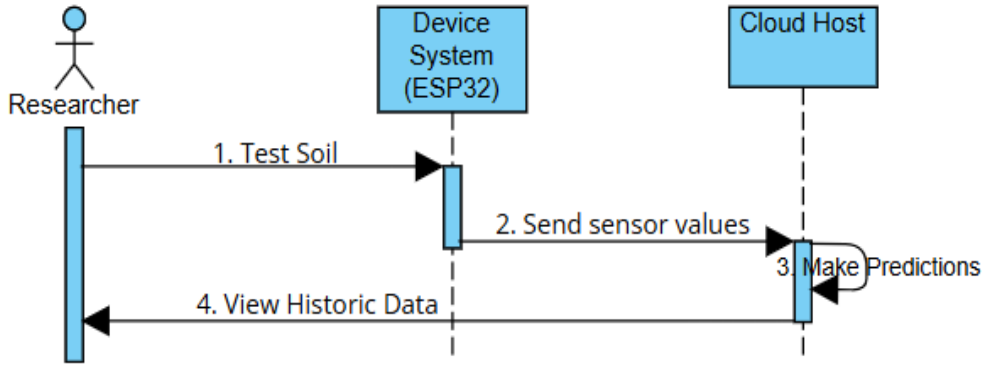

Fig 3.3: Sequence diagram for Researcher

Description:

This sequence diagram represents the interaction between the researcher and the land parameter measuring device system for analyzing soil data using IoT and Machine Learning. The process begins when the researcher initiates a soil test through the ESP32-based device. Once triggered, the ESP32 gathers sensor data related to soil and

environmental conditions such as moisture, temperature, and humidity. These sensor values are then transmitted to the cloud host, where a pre-trained machine learning model processes the data to make predictions regarding parameters like soil pH, electrical conductivity (EC), soil type, and crop suitability. In addition to generating real-time predictions, the system provides the researcher with access to historical data stored in the cloud. This enables the researcher to analyze long-term patterns and trends in soil health, aiding in agricultural research, decision-making, and land management strategies.

Activity Diagram:

An activity diagram is a graphical representation that models the workflow or activities of a system. It is similar to a flowchart and is used to depict the dynamic aspects of the system, detailing the sequence of activities and the flow of control from one activity to another.

3.3.2.3 Activity diagram:

Activity Diagram for Portable Land Parameter Measuring Device

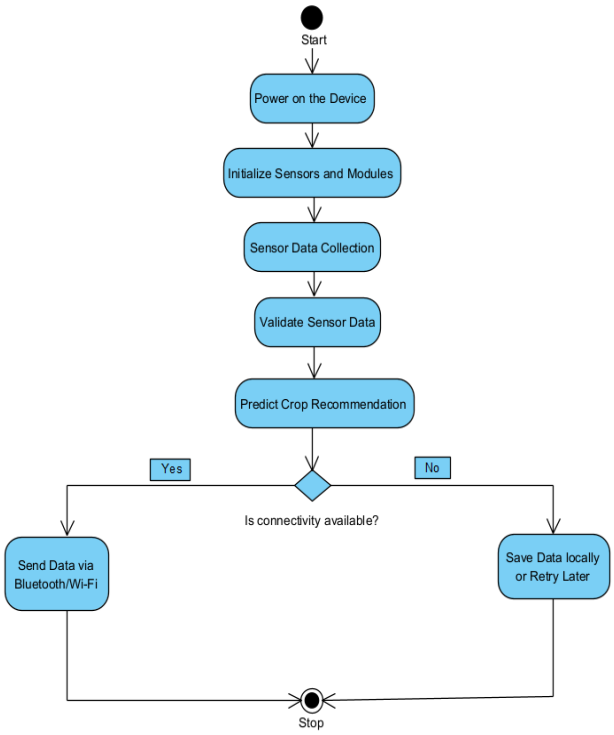

Figure 3.4: Activity Diagram

Description:

The activity diagram illustrates the operational workflow of the Portable Land Parameter Measuring Device, which integrates IoT and Machine Learning to provide real-time soil analysis and crop recommendations. The process begins with powering on the device, followed by initializing all connected sensors and modules. Once the sensors are ready, they begin collecting soil and environmental data, which is then validated for accuracy. The validated data is sent to a cloud-based machine learning model that analyzes it and predicts the most suitable crop for the given land condition. After prediction, the system checks for network connectivity. If connectivity is available, the data is sent to the cloud or external systems via Bluetooth or Wi-Fi. If connectivity is not available, the system saves the data locally or retries the upload at a later time. This ensures continuous data capture and decision support even in areas with unstable internet access, making the device reliable and effective for field use.

DESIGN

4 DESIGN

System Design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. In System design, developers:

- Define design goals of the project
- Decompose the system into smaller sub systems
- Design hardware/software strategies
- Design persistent data management strategies
- Design global control flow strategies
- Design access control policies and
- Design strategies for handling boundary conditions. System design is not algorithmic. It is decomposed of several activities. They are:
- Identify Design Goals
- Design the initial subsystem decomposition
- Refine the subsystem decomposition to address the design goals.

System Design is the transformation and analyzes a model into a system design model. Developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams. Developers also select strategies for building the system, such as the hardware/software platform on which the system runs, the

result of the system design is model that includes a clear description of each of these strategies, subsystem decomposition.

4.1 Design Goals

Design goals are the qualities that the system should focus on. Many design goals can be inferred from the non-functional requirements or from the application domain.

- **Readable**: The arrangement of fonts and words in order to make written content flow in a simple and easy to read manner.
- **Usability:** This targets for the percentage of users who clearly understands the interface of the application
- **Reliable:** Ensures the system perform a specified function within a given system for and expected lifecycle.
- **Reusable**: Reusing several components previously used makes it quicker and easier to design a new project.
- **Extendable**: It is a measure of the ability to extend a system and the level of effort required to implement extension.
- **Flexible:** The capacity to support multiple functions without altering the architecture of the system.
- **Efficient:** Efficiency means that the project does not waste any resources like timeand memory in order to process the application.

4.2 System Design and Architecture

The Land Parameter Measuring Device system is designed to seamlessly integrate Internet of Things (IoT) and Machine Learning (ML) technologies to monitor soil and environmental parameters and predict soil characteristics, suitable crops, and overall land health. The system is composed of interconnected components that operate together to collect, process, predict, and display critical information for farmers and researchers.

4.2.1 System Architecture Overview

The system architecture consists of the following main modules:

1. Sensor Hardware (IoT Layer - ESP32)
2. Cloud Backend (Flask + PythonAnywhere)
3. Machine Learning Model (Local and Cloud Deployment)
4. Display Interface (16×2 LCD + HTML/CSS/JS Web Dashboard)
5. Frontend Communication (Wi-Fi/Bluetooth + REST API)

Each component interacts with others through a clearly defined communication protocol and logical workflow as shown in the system block diagram (refer Fig. 4.3).

4.2.2 Input Design

The input design includes both **sensor data inputs** and manual user triggers. It focuses on capturing real-world values such as:

- Soil Moisture
- Soil Temperature (DS18B20)
- Air Temperature and Humidity (DHT11)

Inputs are gathered using sensors connected to an **ESP32 microcontroller** and are formatted into a structured data array. This data is then:

- Validated in real-time (basic thresholding and range checks)
- Prepared for transmission to the cloud or local ML model

Key considerations in input design:

- Sensor initialization and error handling
- Ensuring valid ranges of data before model prediction
- Minimal user interaction: one-button testing for ease of use
- Lightweight data packets for fast transmission

4.2.3 Output Design

Outputs of the system are generated through the **machine learning model** and displayed using:

- A **16×2 I2C LCD screen** (for real-time portable usage)
- A **cloud-based web interface** (built using HTML, CSS, and JavaScript)

Prediction results include:

- **Soil pH level**
- **Electrical Conductivity (EC)**

- **Soil Type**
- **Recommended Crop**

The system ensures that results are:

- Displayed quickly after testing
- Clearly visible in both local LCD and mobile/web interface
- Logged to cloud or local storage for future reference

Output characteristics:

- Fast loading
- User-friendly format
- Real-time updates
- Includes alerts or highlights for critical results (e.g., low pH warning)

4.2.4 Machine Learning Model Design

The ML component is designed to predict multiple outputs using **supervised learning algorithms**, specifically trained on labeled datasets.

Model Workflow:

- **Input:** Preprocessed sensor readings (features)
- **Training Algorithm:** Random Forest Classifier, Linear Regression
- **Output:** Soil Type, pH, EC, Suitable Crop
- **Serving Method:**
- Locally using Python (offline testing)
- On cloud using **Flask API** (hosted on PythonAnywhere)

Targeted Requirements:

- Fast inference time
- High prediction accuracy on noisy, real-world data
- Easy integration with ESP32 via HTTP requests

4.2.5 Algorithms Used:

Linear Regression

Purpose: Predict **pH** and **Electrical Conductivity (EC)** (Continuous values).

Why used: Linear Regression is simple, interpretable, and effective for predicting continuous numerical data like pH and EC based on environmental sensor readings.

Inputs:

- Soil moisture
- Soil temperature
- Ambient temperature
- Humidity

Outputs:

- Predicted pH value
- Predicted EC value

Random Forest Classifier

Purpose: Predict **Soil Type** and **Recommended Crop** (Categorical values).

Why used: Handles complex, nonlinear relationships well and reduces overfitting by using multiple decision trees.

Inputs:

- Sensor data + Predicted pH and EC

Outputs:

- Predicted soil type (e.g., Clay, Loam, Sandy)
- Recommended crop (e.g., Rice, Wheat, Tomato)

4.2.6 Data Flow and Communication Design

The data flow is designed to support both **real-time operation** and **delayed communication** (in case of connectivity issues).

1. ESP32 activates sensors and collects values
2. Sends sensor data to Flask /predict endpoint
3. Flask loads the trained ML model and returns prediction
4. Prediction results are stored at /latest endpoint and displayed locally
5. Frontend dashboard fetches from /latest using AJAX (every 3–5 sec)

Backup Logic:

- If Wi-Fi is not available, ESP32 saves data locally and retries later.

4.2.7 Frontend Design

The frontend is designed to:

- Fetch latest predicted results from /latest endpoint
- Display in structured HTML dashboard
- Refresh periodically for real-time updates
- Work across desktops and mobile phones

Technologies Used:

- HTML + CSS (for styling)
- JavaScript + Fetch API (for dynamic updates)

4.2.8 Cloud Hosting

- Flask app and ML model hosted on **PythonAnywhere**
- Endpoints:
- /predict (POST): Receives sensor data, runs prediction
- /latest (GET): Returns latest results for frontend display
- Ensures 24/7 accessibility, making the device **portable and wireless**

4.2.9 System Characteristics and Features

- **Modularity:** IoT, ML, and UI components are loosely coupled and independently testable
- **Real-time Capability:** Near-instant prediction and display
- **Portability:** Powered by 3.7V battery + TP4056 + MT3608 step-up module
- **User-Centric Design:** One-button operation, no technical knowledge needed
- **Scalability:** Additional sensors (e.g., NPK, light) can be integrated easily
- **Cloud Integration:** Enables remote monitoring and updates
- **ML Flexibility:** Models can be retrained with better data for higher accuracy

Overview Screen of Portable Land Parameter Measuring Device using IoT

Figure 4.1: Overview Screen

4.3 What makes up an Arduino?

ARDUINO IDE

You decided to go and buy yourself an Arduino, but once it arrived, you realized you have no idea what to do with it. Do not panic, for help is at hand! In this how-to, we will look at how to get started with Arduino microcontroller boards. We'll cover software installation, as well as connecting and configuring the Arduino IDE.

You Will Need

- USB B Cable
- Windows 10, Windows 8, Windows 7, Mac, or Linux OS
- Arduino IDE

- About 15 minutes

4.3.1 Download the IDE

First, you must download the IDE and install it. Start by visiting Arduino's software page. The IDE is available for most common operating systems, including Windows, Mac OS X, and Linux, so be sure to download the correct version for your OS. If you are using Windows 7 or older, do not download the Windows app version, as this requires Windows 8.1 or Windows 10.

Once the installer has downloaded, go ahead and install the IDE. Chances are you will want to enable all options on the installer, including any USB drivers and libraries, but do make sure to read the EULA!

The Arduino IDE

The Arduino IDE is incredibly minimalistic, yet it provides a near-complete environment for most Arduino-based projects. The top menu bar has the standard options, including "File" (new, load save, etc.), "Edit" (font, copy, paste, etc.), "Sketch" (for compiling and programming), "Tools" (useful options for testing projects), and "Help". The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.

Figure 4.2: Arduino IDE

Projects made using the Arduino are called sketches, and such sketches are usually written ina cut-down version of C++ (a number of C++ features are not included). Because programming a microcontroller is somewhat different from programming a computer, there are a number of device-specific libraries (e.g., changing pin modes, output data on pins, reading analog values, and timers). This sometimes confuses users who think Arduino is programmed in an "Arduino language." However, the Arduino is, in fact, programmed in C++.It just uses unique libraries for the device. The 6 Buttons while more advanced projects will take advantage of the built-in tools in the IDE, most projects will rely on the six buttons found below the menu bar.

The button bar

The check mark is used to verify your code. Click this once you have written your code.

1. The arrow uploads your code to the Arduino to run.

2. The dotted paper will create a new file.

3. The upward arrow is used to open an existing Arduino project.

4. The downward arrow is used to save the current file.

5. The far right button is a **serial monitor**, which is useful for sending data from the Arduino to the PC for debugging purposes.

There are plenty of other features available to consider on the IDE. But, having used many different types of microcontrollers and having been involved in multiple programming environments, it is shocking how simple the Arduino and its IDE is! In lessthan two minutes, you can get a simple C++ program uploaded onto the Arduino and have it.

CODING

5.CODING

The goal of coding or programming phase is to translate the design of the system produced during the phase into code in a given programming language, which can be executed by a computer and the performs the computation specified by the design.
The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost, but the goal should be to reduce the cost of later phase. In other words, the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer.

5.1 Coding Approach:

There are two major approaches for coding any software system. They are top-Down approach and bottom-up approach.
Bottom-up approach can suit for developing the object-oriented systems. During system design phase of reduce the complexity. We decompose the system into appropriate number of subsystems, for which objects can be modelled independently. These objects exhibit the way the subsystems perform their operations.
Once object have been modeled they are implemented by means of coding. Even though related to the same system as the objects are implemented of each other the Bottom-Up approach is more suitable for coding these objects.
In this approach, we first do the coding of objects independently and then we integrate these modules into one system to which they belong. In this project, top-Down approach is followed.

5.2 Information Handling:

Any software system requires some amount of information during its operation selection of appropriate data structures can help us to produce the code so that objects of the system can better operate with the available information decreased complexity.

The system ensures accurate processing of environmental data and meaningful predictions without involving complex encryption or authentication, making it lightweight, fast, and practical for agricultural use.

5.3 Programming Style:

Programming style deals with act of rules that a programmer must follow so that the characteristics of coding such as Traceability, Understands the ability, Modifiability, and Extensibility can be satisfied.
In the current system, we followed the coding rules for naming the variables and methods. The system is developed in a very interactive and users friendly manner.

5.4 Verification and Validation:

Verification is the process of checking the product built is right. Validation is the process of checking whether the right product is built. During the Development of the system coding for the object as been thoroughly verified from various aspects regarding their design in the way they are integrated and etc. The various techniques that have been followed for validation discussed in testing the current system. Validations applied to the entire system at two levels.

5.5 Input level validation:

Although the system doesn't use form-based user inputs like traditional desktop software, it performs data-level validations:
On the **ESP32**, sensor readings are validated for range and stability before sending to the server.
On the **Flask server**, incoming sensor data is checked for missing or invalid values before processing.
On the **frontend**, display logic ensures only valid, updated results are shown to users, avoiding outdated or incomplete data.

5.6 SOURCE CODE:

5.6.1 Arduino Source Code:

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal_I2C.h>
// #include <WiFiManager.h>

#define DHTPIN 4
#define DHTTYPE DHT11
#define SOIL_MOISTURE_PIN 34
#define SOIL_TEMP_PIN 25

DHT dht(DHTPIN, DHTTYPE);

OneWire oneWire(SOIL_TEMP_PIN);
DallasTemperature soilTemperature(&oneWire);

LiquidCrystal_I2C lcd(0x27, 16, 2);

const char* ssid = "realme 12 Pro 5G";
const char* password = "server error";

String serverUrl = "http://Prasanth23.pythonanywhere.com/predict";
```

```
String lines[9];
int currentLine = 0;

String extractStringValue(String json, String key) {
  int keyStart = json.indexOf("\"" + key + "\":");
  if (keyStart == -1) return "";

  int valueStart = json.indexOf("\"", keyStart + key.length() + 3);
  int valueEnd = json.indexOf("\"", valueStart + 1);

  if (valueStart == -1 || valueEnd == -1) return "";

  return json.substring(valueStart + 1, valueEnd);
}

void setup() {
  Serial.begin(115200);
  dht.begin();
  soilTemperature.begin();
  lcd.init();
  lcd.backlight();

  WiFi.begin(ssid, password);
  lcd.setCursor(0, 0);
  lcd.print("Connecting WiFi");

  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }

  lcd.clear();
  lcd.print("WiFi Connected");
  delay(1000);

}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  int soilMoistureRaw = analogRead(SOIL_MOISTURE_PIN);
  float soilMoisture = soilMoistureRaw;
  soilTemperature.requestTemperatures();
  float soilTemp = soilTemperature.getTempCByIndex(0);

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Sensor read error!");
    lcd.clear();
    lcd.print("Sensor Error!");
    delay(2000);
    return;
  }

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
```

```
  WiFiClient client;

  http.begin(client, serverUrl);
  http.addHeader("Content-Type", "application/json");

  String json = "{\"soil_moisture\":" + String(soilMoisture, 2) +
          ",\"ambient_temp\":" + String(temperature, 2) +
          ",\"humidity\":" + String(humidity, 2) +
          ",\"soil_temp\":" + String(soilTemp, 2) + "}";

  int httpResponseCode = http.POST(json);

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Response:");
    Serial.println(response);

    int pH_idx = response.indexOf("\"pH\":");
    int EC_idx = response.indexOf("\"EC\":");
    int soil_idx = response.indexOf("\"soil_type\":\"");
    int crop_idx = response.indexOf("\"suitable_crop\":\"");

        float pH = response.substring(response.indexOf("\"pH\":") + 5,
response.indexOf(",", response.indexOf("\"pH\":"))).toFloat();
        float EC = response.substring(response.indexOf("\"EC\":") + 5,
response.indexOf(",", response.indexOf("\"EC\":"))).toFloat();
    String soilType = extractStringValue(response, "soil_type");
    String crop = extractStringValue(response, "suitable_crop");
```

```
    lines[0] = "Moisture: " + String(soilMoisture, 1);
    lines[1] = "Amb Temp: " + String(temperature, 1) + " C";
    lines[2] = "Humidity: " + String(humidity, 1) + " %";
    lines[3] = "Soil Temp: " + String(soilTemp, 1) + "C";
    lines[4] = "pH: " + String(pH, 2);
    lines[5] = "EC: " + String(EC, 0);
    lines[6] = "Soil: " + soilType;
    lines[7] = "Crop: " + crop;
    lines[8] = "Reading next...";

  } else {
    Serial.print("HTTP Error: ");
    Serial.println(httpResponseCode);
    lines[0] = "Server Error";
  }

  http.end();
} else {
  Serial.println("WiFi Disconnected");
  lines[0] = "WiFi Lost!";
}

for (int i = 0; i < 9; i++) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(lines[i].substring(0, 16));
  delay(2500);
```

```
  }

  delay(1000);
}
```

5.6.2 Data Preprocessing and Model Training – Python Source Code:

```
import pandas as pd
import joblib
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier

df = pd.read_csv("Soil_Parameter_ML_Dataset_200.csv")

X = df[['soil_moisture', 'ambient_temp', 'humidity', 'soil_temp']]

# Outputs to predict
y_ph = df['pH']
y_ec = df['EC']
y_soil_type = df['soil_type']
y_crop = df['suitable_crop']

# Train regression models
ph_model = LinearRegression()
ec_model = LinearRegression()

ph_model.fit(X, y_ph)
ec_model.fit(X, y_ec)
```

```
joblib.dump(ph_model, 'pH_model.pkl')
joblib.dump(ec_model, 'EC_model.pkl')

# Train classification models for soil type and crop
soil_type_model = RandomForestClassifier()
crop_model = RandomForestClassifier()

soil_type_model.fit(X, y_soil_type)
crop_model.fit(X, y_crop)

joblib.dump(soil_type_model, 'soil_type_model.pkl')
joblib.dump(crop_model, 'crop_model.pkl')
```

5.6.3 Flask Server Backend – Python Source Code:

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import joblib
import numpy as np
import pandas as pd
import os

# Start Flask app
app = Flask(__name__)
CORS(app)

# Load trained models
ph_model = joblib.load('pH_model.pkl')
ec_model = joblib.load('EC_model.pkl')
```

```
soil_type_model = joblib.load('soil_type_model.pkl')
crop_model = joblib.load('crop_model.pkl')

# Global dictionary to store last prediction
latest_data = {}

# Path to live log CSV file
LOG_FILE = "live_data_log.csv"

@app.route('/predict', methods=['POST'])
def predict():
    global latest_data
    data = request.get_json()

    try:
        # Extract input sensor values
        soil_moisture = float(data['soil_moisture'])
        ambient_temp = float(data['ambient_temp'])
        humidity = float(data['humidity'])
        soil_temp = float(data['soil_temp'])

        input_data = [[soil_moisture, ambient_temp, humidity, soil_temp]]

        # Predict
        pH = ph_model.predict(input_data)[0]
        EC = ec_model.predict(input_data)[0]
        soil_type = soil_type_model.predict(input_data)[0]
        crop = crop_model.predict(input_data)[0]
```

```
        latest_data = {
            'soil_moisture': soil_moisture,
            'ambient_temp': ambient_temp,
            'humidity': humidity,
            'soil_temp': soil_temp,
            'pH': round(pH, 2),
            'EC': round(EC, 2),
            'soil_type': soil_type,
            'suitable_crop': crop
        }

        # ---- Logging part ----
        df_log = pd.DataFrame([latest_data])
        if os.path.exists(LOG_FILE):
            df_log.to_csv(LOG_FILE, mode='a', header=False, index=False)
        else:
            df_log.to_csv(LOG_FILE, index=False)

        return jsonify(latest_data)

    except Exception as e:
        return jsonify({'error': str(e)})

@app.route('/latest', methods=['GET'])
def get_latest():
    if latest_data:
        return jsonify(latest_data)
```

```
    else:
        return jsonify({'message': 'No data received yet'}), 404

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

5.6.4 Frontend (Web Interface):

HTML Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Monitoring Dashboard</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Monitoring Dashboard</h1>
        <div class="card">
            <h2>Sensor Readings</h2>
            <p><strong>Soil Moisture :</strong> <span id="moisture">--</span></p>
              <p><strong>Ambient Temperature :</strong> <span id="temperature">--
</span></p>
            <p><strong>Humidity :</strong> <span id="humidity">--</span></p>
                  <p><strong>Soil Temperature :</strong> <span id="soil_temp">--
</span></p>
        </div>
```

```
      <div class="card">
        <h2>Predicted Parameters</h2>
        <p><strong>pH :</strong> <span id="ph">--</span></p>
        <p><strong>EC :</strong> <span id="ec">--</span></p>
        <p><strong>Soil Type :</strong> <span id="soil_type">--</span></p>
        <p><strong>Suitable Crop :</strong> <span id="crop">--</span></p>
      </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

CSS Source Code:

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background: #f0f4f7;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: flex-start;
  min-height: 100vh;
}

.container {
  width: 90%;
```

```
  max-width: 600px;
  margin-top: 30px;
  background: #ffffff;
  padding: 20px 30px;
  border-radius: 10px;
  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.1);
}

h1 {
  text-align: center;
  margin-bottom: 25px;
  color: #2e7d32;
}

.card {
  background: #e8f5e9;
  padding: 15px 20px;
  margin-bottom: 20px;
  border-radius: 8px;
  box-shadow: inset 0 0 5px rgba(0, 0, 0, 0.05);
}

.card h2 {
  margin-top: 0;
  color: #1b5e20;
}

.card p {
```

```
  font-size: 16px;
  margin: 8px 0;
}
```

JavaScript Source Code:

```
async function fetchData() {
  try {
    const response = await fetch("https://Prasanth23.pythonanywhere.com/latest");
    const data = await response.json();

    document.getElementById("moisture").innerText = data.soil_moisture;
     document.getElementById("temperature").innerText = data.ambient_temp + "
°C";
    document.getElementById("humidity").innerText = data.humidity + " %";
    document.getElementById("soil_temp").innerText = data.soil_temp + " °C";

    document.getElementById("ph").innerText = data.pH;
    document.getElementById("ec").innerText = data.EC;
    document.getElementById("soil_type").innerText = data.soil_type;
    document.getElementById("crop").innerText = data.suitable_crop;
  } catch (err) {
    console.error("Error fetching data:", err);
  }
}

setInterval(fetchData, 10000);
fetchData();
```

TESTING

6 .TESTING

Testing is the process of finding differences between the expected behavior specifiedby system models and the observed behavior of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the codes the application should be thoroughly tested and validated.

Unit testing finds the differences between the object design model and its corresponding components. Structural testing finds differences between the system design model and a subset of integrated subsystems. Functional testing finds differences between the use case model and the system.

Finally, performance testing, finds differences between non-functional requirements and actual system performances. Form modeling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

6.1 Testing Activities

Testing a large system is a complex system is a complex activity and like any complex activity. It must be breaking into smaller activities. Thus, incremental testing was performed on the project i.e., components and subsystems of the system were tested separately before integrating them to from the subsystem for system testing.

6.2 Types of testing

Unit Testing:

Unit testing ensures that each individual functional block of the system performs accurately and reliably in isolation. Each module is tested to confirm that it produces the expected outputs for a given set of valid inputs. The data flow within modules is validated for correctness, and logical checks are applied to ensure consistent behavior. All core computations and operations are verified against known scenarios to ensure accuracy.

Additionally, interfaces responsible for displaying or transmitting results are tested to confirm they present correct values under normal and edge-case conditions.

Integration Testing :

Integration testing focuses on validating the smooth and accurate interaction between the different modules of the system. It ensures that data flows correctly between the individual components and that the integrated units function as expected when combined. Communication pathways are tested for consistency and synchronization, ensuring that updates are processed and reflected across all relevant layers. The system's ability to detect and recover from disruptions or failures is also verified to maintain stability and robustness during real-time operation.

Functional Testing:

Functional testing ensures that every major feature of the system performs as specified and delivers accurate, meaningful results to the user. It verifies that data collected from various sources is correctly processed, transmitted, and interpreted. The testing focuses on the logical correctness of the input-to-output flow, accuracy of predictions generated, and timely delivery of information to the output interfaces. It also ensures that all functionalities such as data acquisition, prediction, storage, and display work cohesively to provide a seamless and reliable user experience.

Performance Testing:

Performance testing is essential to ensure the system operates efficiently under various environmental and operational conditions. This includes evaluating the system's response time from data acquisition to prediction output, and ensuring that data processing and transmission occur without noticeable delays. The system is tested for scalability by simulating increased data input volumes and verifying that performance remains stable. Stress tests are also performed to assess system reliability during continuous operation or under limited connectivity conditions, ensuring consistent functionality even under resource constraints.

Accuracy Testing:

Accuracy testing focuses on verifying that the sensor readings accurately reflect the true environmental and soil conditions. This includes comparing the collected sensor data with known reference values to ensure high precision. The test also involves checking for anomalies, false readings, or sensor drift over time. Accurate data transmission to the processing unit or cloud server is crucial to ensure reliable predictions of soil parameters and suitable crops.

Usability Testing:

Usability testing ensures that the system is user-friendly, intuitive, and accessible to non-technical users such as farmers. This involves evaluating the clarity and readability of output on both the LCD screen and the web interface. The goal is to ensure that the system presents prediction results in a straightforward and meaningful way, with minimal steps required for operation. Feedback from users is collected to identify any difficulties in operating the device or understanding the displayed results, and to improve the overall user experience.

Security Testing:

Security testing involves ensuring the protection of data and system integrity throughout the soil monitoring and prediction process. This includes verifying secure communication between hardware components, local processing units, and cloud endpoints to prevent unauthorized access or data breaches. The system is tested for vulnerabilities such as data tampering, injection attacks, or interception of transmitted sensor values. Robust validation, access control, and encryption mechanisms are essential to maintain the confidentiality and reliability of both sensor data and machine learning predictions.

Black box Testing:

Black-box testing involves evaluating the system's functionality without examining its internal code or structure. Testers focus on verifying that the system correctly handles sensor inputs such as soil moisture, temperature, and humidity, and produces the expected outputs—like pH, EC, soil type, and suitable crop predictions. The aim is to ensure the system meets its functional requirements and performs accurately from a user's perspective, regardless of the internal implementation.

Whitebox Testing:

White-box testing involves examining the internal logic and source code of the system. This includes validating the correctness of the machine learning algorithms used for predicting soil pH, EC, soil type, and suitable crops, as well as testing data preprocessing and decision logic. It ensures that internal workflows, such as sensor data handling, model integration, and result generation, function as intended. This type of testing helps identify logical errors, optimize performance, and verify that each code component behaves correctly under various conditions.

6.3 Testing Plan:

A comprehensive test plan for the Land Parameter Measuring Device involves several essential phases. Initially, unit testing focuses on validating individual modules such as sensor data acquisition, communication protocols, and result display logic to ensure each works as intended. Integration testing then evaluates the data flow between components—including sensor readings, machine learning prediction, and frontend display—to verify system coherence. Functional testing ensures core features such as data transmission, ML-based predictions (pH, EC, soil type, and crop), and real-time display are working properly. Performance testing assesses system responsiveness, accuracy under varied conditions, and scalability for additional sensors or extended datasets. Accuracy testing verifies the precision of predictions against known soil profiles. Usability testing checks the clarity, accessibility, and user-friendliness of the display interfaces. Security testing ensures that data is transmitted and stored safely without unauthorized access. Finally, regression

testing validates that any updates do not break existing functionality, while acceptance testing confirms that the system meets user expectations and real-world performance goals.

6.4 Test Cases

Test case for Sensor Initialization and Data Collection:

Test case ID	TC001
Description	Ensure that all sensors initialize properly and send valid readings.
Pre-requisites	ESP32 powered and connected to sensors (Soil Moisture, DS18B20, DHT11)
Steps	1. Power on the ESP32. 2. Wait for sensor data to be read.
Expected Result	Sensor values (moisture, soil temp, humidity, air temp) are displayed without errors.

Table 6.1: **Sensor Initialization and Data Collection**

Test case for Model Prediction via Flask API:

Test case ID	TC002
Description	Verify that the ML model hosted on Flask predicts pH, EC, soil type, and crop correctly.
Pre-requisites	Flask server running, ML model loaded
Steps	1. Send a POST request to /predict endpoint with valid sensor data. 2. Check the response.

Expected Result	Response includes pH, EC, soil type, and crop predictions.

Table 6.2: Model Prediction via Flask API

Test case for Display on 16×2 LCD:

Test case ID	TC003
Description	Verify that the LCD displays prediction results correctly.
Pre-requisites	ESP32 connected to LCD, model predictions ready
Steps	1. Run the prediction code. 2. Observe the LCD screen.
Expected Result	LCD should display readable values like "pH: 6.7", "Crop: Rice" etc.

Table 6.3: Display on 16×2 LCD

Test case for Web Dashboard Display:

Test case ID	TC004
Description	Ensure latest predictions are fetched and shown in the HTML dashboard.
Pre-requisites	Frontend connected to /latest endpoint
Steps	1. Open the dashboard. 2. Wait for 5 seconds for auto-refresh.
Expected Result	Web UI shows latest sensor-based predictions.

Table 6.4: Web Dashboard Display

Test case for Flask Endpoint /latest Response:

Test case ID	TC005
Description	Ensure the /latest endpoint returns the most recent prediction results.
Pre-requisites	At least one prediction made previously
Steps	1. Send GET request to /latest. 2. Observe returned values.
Expected Result	JSON response should include recent prediction results.

Table 6.5: Flask Endpoint /latest Response

Test case for Model Training and Export:

Test case ID	TC006
Description	Validate model training and .pkl export process.
Pre-requisites	Dataset available, Python script for training ready
Steps	1. Run the training code for Linear Regression and Random Forest 2. Check for model.pkl files
Expected Result	pH_model.pkl, EC_model.pkl, soil_type_model.pkl, crop_model.pkl should be generated

Table 6.6: Model Training and Export

SCREENS

7. SCREENS

Figure 7.1: Display of Ambient Temperature

Description:

This screen shows the ambient temperature recorded by the system using an environmental sensor. The value displayed is 28.5°C, which represents the surrounding air temperature during the soil testing process. This parameter is crucial in determining environmental conditions affecting soil behavior and crop growth.

Figure 7.2: Display of Predicted pH Value

Description:

This screen displays the predicted soil pH value as 6.79, calculated using the Linear Regression model. The pH value is an important indicator of soil acidity or alkalinity and significantly affects nutrient availability for crops.

Figure 7.3: Display of Predicted Soil Type

Description:

This screen shows the predicted soil type as "clay" based on the input parameters. The prediction is made using a Random Forest Classifier model trained on historical soil datasets. The soil type information helps in selecting suitable crops and understanding soil behavior.

Figure 7.4: Display of Server Error Message

Description:

This screen appears when the device fails to connect with the backend server due to issues like no internet, server downtime, or API errors. During this time, predictions are paused until the connection is restored, ensuring user awareness of the issue.

Figure 7.5: Display of Reading Next Message

Description:

This screen indicates that the device is processing the current sensor reading and preparing to fetch the next parameter. It ensures smooth sequential display of all sensor values for better readability.

Figure 7.6: Display of Suitable Crop

Description:

This screen shows the final prediction result the most suitable crop for the given land conditions based on sensor readings and the trained machine learning model. It helps the user make informed agricultural decisions.

The system also displays vital real-time readings such as **soil moisture**, **soil temperature**, and humidity, which are crucial for understanding the current land and weather conditions. The **soil moisture** value helps determine how dry or wet the soil is, guiding irrigation decisions. The **soil temperature** reading reflects the warmth of the soil, which directly affects seed germination and microbial activity. Meanwhile, humidity represents the moisture content in the surrounding air, influencing both soil evaporation rates and plant transpiration. These sensor values, combined with machine learning predictions, help provide accurate insights into soil health and optimal crop planning.

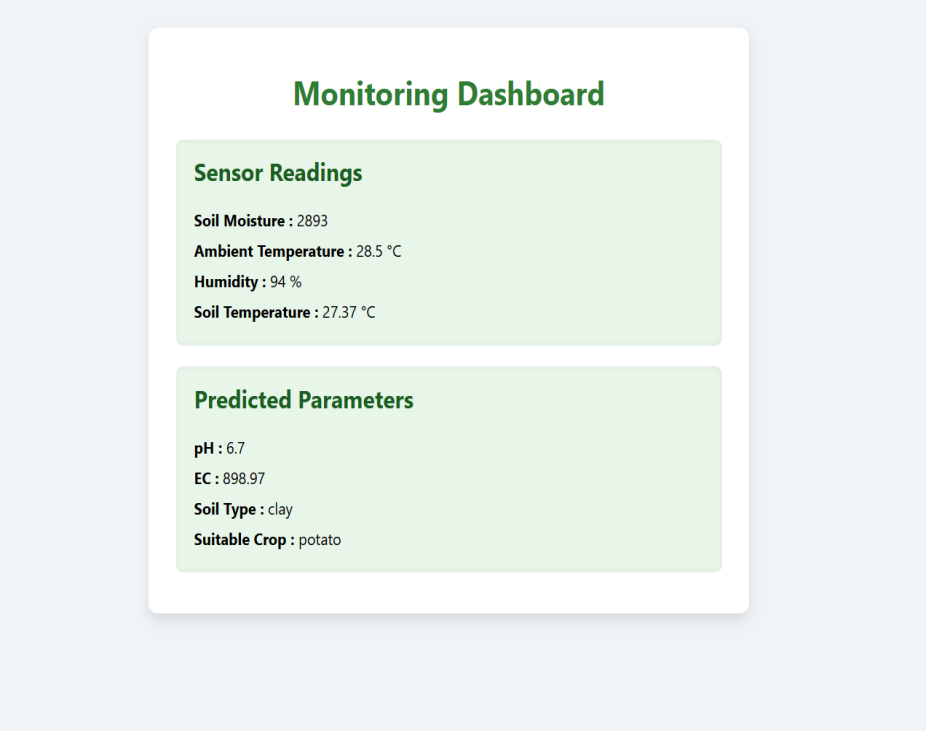

Figure 7.7: frontend web dashboard screen

Description:

This screen represents the Monitoring Dashboard of the Land Parameter Measuring Device, built using HTML, CSS, and JavaScript. It is divided into two main sections: Sensor Readings and Predicted Parameters. The Sensor Readings section displays live values for soil moisture, ambient temperature, humidity, and soil temperature, all fetched from the IoT-connected sensors. The Predicted Parameters section shows the machine learning model's predictions, including pH level, electrical conductivity (EC), soil type, and the most suitable crop for the current land condition. The dashboard updates in real-time using JavaScript and ensures a clear, organized, and user-friendly presentation for monitoring and decision-making.

CONCLUSION

8. CONCLUSION

The Land Parameter Measuring Device project demonstrates the effective integration of IoT and Machine Learning to deliver a smart, real-time solution for soil monitoring and agricultural decision-making. By collecting data from various environmental sensors such as soil moisture, ambient temperature, humidity, and soil temperature the system provides accurate predictions for essential soil parameters including pH, Electrical Conductivity (EC), soil type, and the most suitable crop.

The ESP32-based IoT hardware ensures portability and low power consumption, while the ML models deployed through a Flask server enable seamless prediction and result processing, both locally and via the cloud. The system includes a LCD display for field readability and a web dashboard built using HTML, CSS, and JavaScript for remote monitoring.

This device supports farmers in making timely and data-driven agricultural decisions, improves soil management, and potentially increases crop yield. The modular design allows for future scalability, such as the integration of additional sensors. Overall, the project showcases how modern technologies can be harnessed to empower sustainable and smart farming practices.

FUTURE SCOPE

9. FUTURE SCOPE

The Land Parameter Measuring Device holds significant potential for future enhancements and broader applications. Advanced analytics powered by AI can be integrated to study long-term soil and environmental patterns, enabling more accurate seasonal crop recommendations and land health forecasting. The system can be expanded to support predictive irrigation suggestions by analyzing historical trends alongside real-time data. Integration with satellite or drone-based remote sensing can offer large-scale monitoring capabilities, enhancing agricultural planning for larger farmlands. A mobile application could be introduced for seamless user interaction, enabling farmers to receive notifications, reports, and expert suggestions directly on their smartphones. Furthermore, the system could be linked with government agricultural databases or smart farming platforms to contribute to regional soil mapping and sustainable farming practices. With support for more sensor types, such as NPK or light sensors, and the inclusion of multilingual voice outputs, the device can be made even more user-centric, scalable, and impactful across diverse agricultural regions.

REFERENCE

10. REFERENCES

10.1 Academic Papers and Journals

- Verma, A., Singh, D., & Thakur, R. (2020). *Smart Agriculture Using IoT, Machine Learning and Cloud Computing*. International Journal of Advanced Science and Technology, 29(4), 13499–13507.
- Ameen, W., Liu, X., & Ullah, I. (2020). *IoT-based smart soil monitoring system using machine learning*. IEEE Access, 8, 134963–134972.
- Rahman, A., & Haider, S. (2022). *Predictive analysis of soil parameters using ML algorithms*. International Journal of Agricultural Technology, 18(2), 145–153.
- M. Mehta, K. Jain, and M. Singh, *Soil Parameter Monitoring System Using IoT and ML*, International Journal of Engineering Research & Technology (IJERT), Vol. 10 Issue 6, June 2021.
- Jha, A. & Kumari, R. (2019). *Crop recommendation system based on soil parameters using ML*. International Journal of Scientific & Engineering Research, 10(4), 240–245.
- Kumar, D., & Choudhary, A. (2018). *A Survey on ML Techniques in Agriculture*. International Research Journal of Engineering and Technology (IRJET), 5(4), 1634–1638.

10.2 Web References

- Soil Parameter Monitoring using ESP32 and Sensors – YouTube
- IoT Based Smart Agriculture using ESP32 & Blynk – YouTube
- ESP32 Sensor Data to Flask Server (Python) – YouTube
- Machine Learning Regression for pH Prediction – YouTube
- How to Host Flask App on PythonAnywhere – YouTube
- Soil pH and EC Sensing and Prediction Model – YouTube
- Crop Recommendation System using ML – YouTube
- ESP32 + LCD + Sensors Setup Reference – YouTube

APPENDIX

11. APPENDIX

11.1 List of Figures

11.2 List of Tables