AN IOT BASED AUTONOMOUS VEHICLE FOR REMOTLY MONITORING AND PREDICTING ENVIRONMENTAL PARAMETERS USIGN ML TECHNIQUES

PARVATEESAM KUNDA [1], PASUMARTHI RAMAVENKATA DURGASATYA SAI RAVINDRA[2], CHAPPIDI BABU[3], DEGALA SRINIVAS[4], DASARI TEJA SAI RAM[5]

[1] Assistant Professor, ECE, Godavari Institute of Engineering and Technology, Rajahmundry, AP

[2,3,4,5] UG Student, ECE, Godavari Institute of Engineering and Technology, Rajahmundry, AP

kundaparvateesam@ggu.edu.in, pasumarthiravindra@gmail.com, babichappidi13@gmail.com, degalasrinivas00@gmail.com, saidasari191210@gmail.com

Abstract- ***This paper presents the design and implementation of an autonomous IoT-enabled robotic vehicle developed for continuous environmental monitoring and predictive analysis. The system is built around an ESP32 microcontroller and integrates multiple sensors to measure temperature, humidity, air quality, gas leakage, smoke, alcohol vapours, vibration, motion, and obstacle distance. The vehicle is capable of autonomous navigation using ultrasonic-based obstacle avoidance while simultaneously collecting environmental data from diverse locations. The acquired sensor data is transmitted to a cloud platform in real time for visualization, storage, and analysis.***

To enhance system intelligence, machine learning techniques are employed to analyze historical and real-time sensor data for anomaly detection and future trend prediction. A live video streaming module based on ESP32-CAM is incorporated to provide real-time visual feedback, thereby improving situational awareness and operational safety in remote or hazardous environments. The proposed system offers a low-cost, scalable, and flexible solution suitable for applications such as smart agriculture, industrial safety monitoring, pollution surveillance, and disaster-prone area assessment.

Keywords: IoT, ML, Autonomous vehicle, Cloud, Scaling

1. INTRODUCTION

Environmental monitoring has become a crucial requirement due to increasing industrial activity, urban expansion, and the growing impact of pollution and climate variability. Conventional environmental monitoring approaches are largely dependent on fixed sensor stations and manual data collection, which restricts spatial coverage and delay real-time response. These limitations reduce their effectiveness in dynamic or hazardous environments. The emergence of the Internet of Things (IoT) has enabled the deployment of interconnected sensing devices capable of continuous data acquisition and wireless communication. IoT-based monitoring systems allow real-time access to environmental data through cloud platforms, significantly improving efficiency, scalability, and accessibility. However, most existing IoT monitoring solutions remain stationary and lack mobility, which limits their ability to cover large or inaccessible regions. Autonomous robotic platforms address this limitation by providing mobility and flexibility in data collection. By mounting sensors on mobile vehicles, environmental parameters can be measured across wide geographical areas, including locations that may be unsafe for human operators.

Autonomous navigation further reduces the need for manual intervention and enhances operational efficiency. In addition to mobility, the integration of machine learning techniques enables intelligent interpretation of sensor data. Machine learning models can identify abnormal patterns, detect hazardous conditions, and predict future environmental trends based on historical observations. This predictive capability supports proactive decision-making rather than reactive responses. In this work, an autonomous environmental monitoring vehicle based on the ESP32 microcontroller is developed. The system integrates multiple sensors for comprehensive environmental sensing, wireless cloud connectivity for remote monitoring, live video streaming for visual feedback, and machine learning-based analytics. The proposed approach aims to provide an intelligent, reliable, and scalable solution for modern environmental monitoring applications. The primary objective of this project is to design and develop an IoT-based autonomous vehicle capable of continuously monitoring key environmental parameters, including temperature, humidity, air quality, gas leakage, smoke, alcohol presence, vibration, and motion. The system aims focuses on integrating live video streaming to provide real-time visual feedback for improved situational awareness. Additionally, machine learning techniques are incorporated to analyze collected data, detect anomalies, and predict future environmental trends. The overall goal is to create a scalable, cost-effective, and intelligent monitoring solution suitable for smart cities, agriculture, industrial safety, and disaster management application.

PROBLEM STATEMENT

Despite advancements in environmental monitoring technologies, existing systems face several challenges. Traditional monitoring setups are mostly static and provide limited spatial coverage, making them unsuitable for monitoring large, remote, or dynamically changing environments. Manual data collection methods increase operational costs and expose personnel to potential safety risks. Although IoT-based monitoring systems improve real- time data acquisition, many current implementations rely solely on threshold-based alerts and lack intelligent data analysis. Without predictive capabilities, such systems are unable to forecast environmental changes or detect anomalies at an early stage. Furthermore, the absence of mobility restricts their effectiveness in hazardous zones such as industrial sites, disaster-affected areas, and regions with toxic gas exposure.

Another significant limitation is the lack of real-time visual monitoring. Sensor readings alone may not always provide sufficient context for decision-making, especially in complex environments. The absence of live visual feedback reduces situational awareness and limits the reliability of remote monitoring operations. Therefore, there is a need for an integrated system that combines autonomous mobility, multi-sensor environmental monitoring, cloud-based data access, live video streaming, and machine learning-based prediction. Such a system can improve safety, reduce human involvement, and enable proactive environmental management.

2. HARDWARE IMPLEMENTATION

Proposed Block Diagram

To sum up, the proposed block diagram shows a multipurpose IoT based robotic along with solar tracker system which is comprised of two Arduino UNO controllers, node MCU Wi-Fi module, sensors, actuators and motor driving block. Arduino UNO 1 is the main control Arduino used here, which reads the LDR sensor values (light dependant resistance), GPS, voltage, and dual servo motor which takes care of solar tracking. Four LDRs sense light in all four directions, and Arduino UNO 1 checks their analog values to move the servo motors for panel alignment. GPS sends up to the minute to collect sensor data in real time and transmit it wirelessly to a cloud platform for remote monitoring and storage. Another objective is to implement autonomous navigation using obstacle detection to enable safe movement in unknown environments. The project also location information, and the voltage sensor checks battery status. Arduino UNO 1 interacts with NodeMCU that makes provision for Wi-Fi connection and supports the system data- LDR readings, GPS location, servo angles, battery voltage to be viewed remotely through Blynk IoT app.

Robot’s movement is controlled by Arduino UNO 2. It listens on RF for commands from an RF receiver which has an external, RF remote connected to it and then controls the DC motors using the L298 motor driver. This separation between controllers allows for the servo track to continue uninterrupted and robot motion smooth. The system is placed under a 12V power supply and controlled by a power supply circuit. The robot is remote controlled using Blynk and RF control for local operations, enabling it to be used for solar tracking, automation and mobile robotics.

Figure 1 Proposed Block Diagram

Hardware Components

ESP 32 Microcontroller Board

The ESP32 microcontroller serves as the central control unit of the system. It provides sufficient processing capability, multiple GPIO interfaces, and built-in Wi-Fi connectivity, making it suitable for handling sensor data acquisition, motor control, and cloud communication simultaneously. The ESP32 processes data from all connected sensors and transmits the information to the cloud platform.

Figure 2 ESP 32 Dev Module

ESP 32-CAM

The ESP32-CAM module is used to capture and stream live video from the vehicle. It provides real-time visual feedback of the surrounding environment, enabling remote users to monitor conditions, verify sensor readings, and ensure safe navigation in hazardous areas.

MOTOR DRIVER

A motor driver is an essential electronic interface used to control the direction, speed, and operation of DC motors by providing sufficient current and voltage that a microcontroller cannot supply directly. Since microcontrollers like the ESP32 operate at low power levels, a motor driver acts as an intermediary between the control unit and the motors, enabling safe and efficient motor operation.

Figure 3 Motor driver

In the proposed IoT-based autonomous environmental monitoring rover, the motor driver is used to control the movement of DC geared motors mounted on the robotic chassis. Control signals from the ESP32 microcontroller are sent to the motor driver, which then regulates motor rotation for forward, backward, left, and right movements. This allows the rover to navigate autonomously using ultrasonic sensor feedback or be remotely controlled through an IoT dashboard. The motor driver thus plays a critical role in enabling autonomous navigation, obstacle avoidance, and precise mobility, ensuring reliable movement while environmental data and live video are continuously collected.

DC Motor

A gear motor is a type of electrical motors commonly used in low-speed, but high-torque applications. RPM – The motor's RPM specified speed. By providing gear, it is possible to reduce the output speed into any desired value, at the same time that torque is markedly increased. This is called gear reduction. The DC motor is definitely a range of operation itself with higher voltage yielding faster RPM. This is represented as RPM = $K_1 \times V$, wherein K 1 where k ais the motor torque constant, and V is the voltage.

Figure 4 DC Motor

The gears operate in accordance with the law of conservation of angular momentum. A smaller gear will turn faster but will have less torque; a larger gear turns slower and produces more torque. If more than one gear is attached, the devices determine the velocity and torque of the final shaft. In DC motors RPM/Torque is inversely proportional that mean if more torque required to motor speed will get slow.

Sensors and Actuators

The system integrates multiple sensors, including DHT22 for temperature and humidity measurement, MQ2, MQ3, MQ6, and MQ135 for gas and air quality detection, a vibration sensor for shock detection, MPU6050 for motion and orientation monitoring, and an ultrasonic sensor for obstacle detection. DC motors controlled through a motor driver enable vehicle movement.

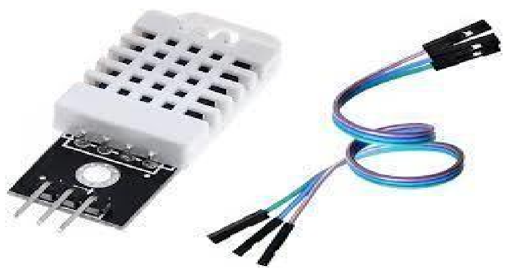


Figure 5 DHT 22 Temperature & Humidity Sensor

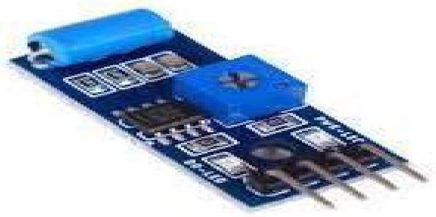


Figure 6 Vibration Sensor

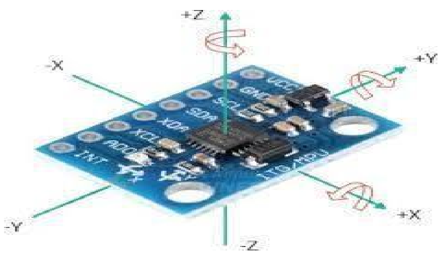


Figure 7 MPU 6050

Figure 8 MQ2 (Smoke Sensor)

Figure 9 MQ3(Alcohol Sensor)

Figure 10 MQ6 (Gas Leakage Sensor)

Figure 11 MQ135 (Air Quality Sensor)

Figure 12 Ultrasonic Sensor

Battery & Power Management

A rechargeable battery supplies power to the entire system. A battery management module ensures safe charging, voltage regulation, and protection against overcurrent and deep discharge, supporting stable and long-duration operation.

3. SOFTWARE IMPLEMENTATION

Embedded C Programming

Embedded C is an extension to the traditional C, and is used in programming for embedded systems. It provides several features not normally available in standard C, such as fixed-point arithmetic, named address spaces and I/O hardware access, making it ideal for devices with limited memory or custom hardware peripherals. These extensions aside, Embedded C retains the standard C syntax such as functions may use local variables and can have loops, conditional statements, arrays, structures etc. and various other distinctions that make it suitable for run or processing efficient programs to be deployed on hardware devices.

It is essentially developed in embedded C, since it has basic functionality and implemented using less number of lines and has ease to understand, increased reliability, Portability for any processor & Scalability for any application. The coordinator, mobile phones, washing machines, digital cameras — most of the electronic devices that we use on a daily basis embed microcontrollers programmed using such Embedded C; it's all code-implement-compile-as-hex and program-to-microcontroller.

Microcontroller programming process varies across operating system such as Windows, Linux or RTOS, but

the concept is same. The aim is to write, compile and upload structured Embedded C code in order for the micro controller to control hardware in a way that will perform some task or full fill functionalities of an embedded system.

Programming of Arduino-based microcontrollers (such as the Nano used in Duino Kits) requires installation of the Arduino IDE. The software is freely available at the Arduino site. cc by using the Windows, Mac or Linux version as appropriate. Prerequisites You need the computer and an Arduino-compatible board, as well as a USB cable to plug the board in for program loading (upload). The Installer (. exe) version is preferable since it allows USB driver installation, and zip based installation need to manually install drivers.

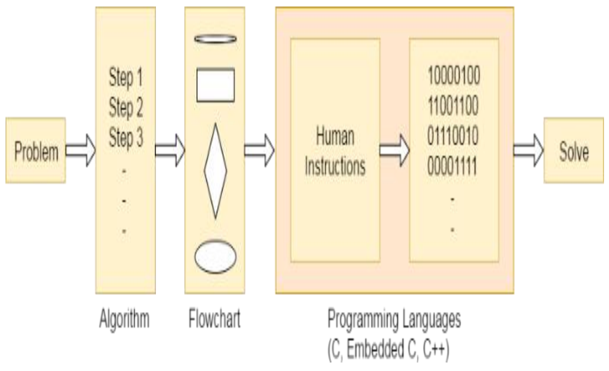


Figure 13 Flow of Embedded C

When you install it will ask were you want to installed, you can just choose C:\Program Files\Arduino. This is the default place, and it is OK for most users, who will have IDE files, libraries, drivers and examples in a single convenient places. But advanced users may prefer custom locations, particularly when working with limited storage or multiple projects

Once the directory is selected, the installation process begins. The installer copies program files, examples, and libraries, installs essential USB drivers, and registers Arduino settings with the operating system. A progress bar indicates installation status. After completing the process, shortcuts are created, file associations are set, and the IDE becomes ready for uploading sketches to the Arduino board.

Arduino IDE

You can use the Arduino IDE for writing, editing and uploading programs to your Arduino. Clean design, easy to understand and use for beginners in embedded systems. The interface has important areas like Menu Bar, Toolbar, Sketch Editor, Message Area and Serial Monitor. Functions useful for file management, compiling and uploading are available on the menus of the Menu Bar: File, Sketch, Tools. Quick-access buttons such as Verify, Upload, New, Open and Save are located in the toolbar.

But, it's really the Sketch Editor that is your primary coding workspace (where you will write code with syntax highlighting and error underlining). The area Message below shows the compilation result, when there is warning or if upload fails, it would help you to easy debug the errors. By opening up the Serial Monitor, you can communicate with the board in real time and view sensor data as well as debug messages.

For IDE users can find these options in \ [Tools> Board\] > and their corresponding port. Beginners are able to grasp how the sensors are used through reference on Examples menu for sensor use, communication examples, and sample coding structure. IDE output area: The Message Area and Serial Monitor provide vital program feedback and are both part of the IDE. Together, these elements make the Arduino IDE an efficient tool for learning, testing and developing microcontroller-based applications.

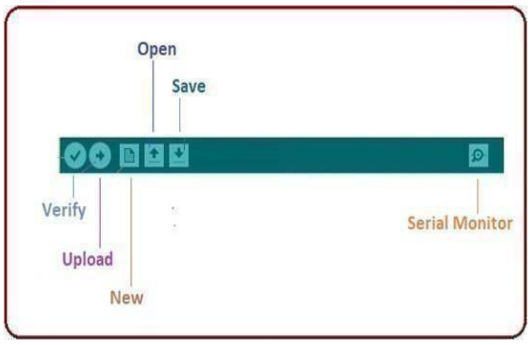
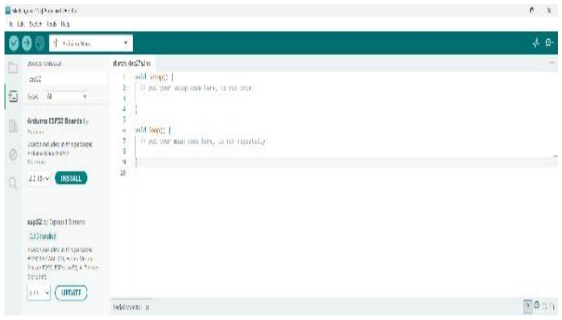


Figure 14 Arduino Uno Interface

Interface of ESP32 in Arduino IDE

ESP32 support in the Arduino IDE must be installed for programming WiFi- enabled boards like NodeMCU and Wemos D1 Mini. The reason is: because the IDE does not have ESP32 libraries in its library manager, you need to add it yourself. Begin by opening File → Preferences, and pasting the following URL in to the Additional Boards Manager URLs piece of code. Now, save this and then visit Tools > Board > Boards Manager, and look for ESP32 and install package called "ESP32 by ESP32 Community."



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 15 Install ESP32 Board

After the installation of the tool and rebooting the IDE, ESP32 boards shows up in Tools → Board again beside the board under ESP32 Boards that I can choose models such as NodeMCU 1.0 among others. Once the right board and COM port has been selected, Arduino IDE is entirely prepared to compile and flash IoT applications into ESP32 modules.

Web Page Interface

The web page interface acts as the primary user interaction layer for the IoT-based autonomous environmental monitoring vehicle. It provides a real-time dashboard that displays sensor readings such as temperature, humidity, air quality, gas concentration, smoke, alcohol levels, vibration status, and distance measurements from the ultrasonic sensor. The interface retrieves live data from the cloud IoT platform using HTTP or MQTT protocols, ensuring continuous and remote monitoring of environmental parameters.

In addition to sensor visualization, the web interface integrates **live video streaming** from the ESP32-CAM module, allowing users to observe the rover's surroundings in real time. This visual feedback enhances navigation, safety, and situational awareness, especially in hazardous or remote environments. Control options for rover movement (forward, backward, left, right, stop) can also be included, enabling remote manual control when required.

The dashboard is designed with a simple and responsive layout using web technologies such as HTML, CSS, and JavaScript, making it accessible from desktops, tablets, and mobile devices. Graphical representations like line charts and gauges help users analyze historical trends and real-time variations of environmental parameters.

Overall, the web page interface enhances usability, supports informed decision-making, and serves as a vital link between the autonomous rover, cloud analytics, and end users.

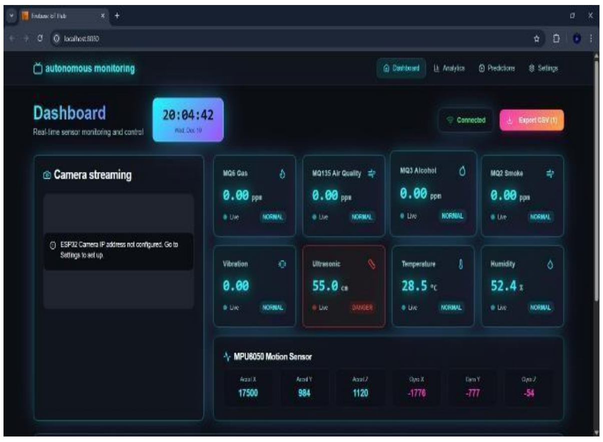


Figure 16 Web Dashboard

4. RESULTS

The environmental monitoring vehicle was successfully designed, implemented, and tested under real-time operating conditions. The system effectively collected environmental parameters such as temperature, humidity, gas concentration, smoke levels, air quality, vibration, and motion data using multiple onboard sensors. All sensor readings were accurately acquired by the ESP32 microcontroller and transmitted wirelessly to the cloud platform, demonstrating reliable IoT connectivity and real-time data monitoring.

The live video streaming feature using the ESP32-CAM module functioned efficiently, providing continuous visual feedback of the rover's surroundings. This visual data significantly enhanced situational awareness and assisted in safe navigation, especially in obstacle-prone and hazardous environments. The ultrasonic sensor successfully detected obstacles, enabling the rover to avoid collisions during autonomous movement.

Machine Learning models applied to the collected sensor data demonstrated effective anomaly detection and trend prediction capabilities. The system was able to identify abnormal conditions such as elevated gas levels, poor air quality, and sudden environmental changes, allowing timely alerts and proactive responses. The prediction accuracy improved with continuous data collection, highlighting the benefit of integrating ML techniques with IoT systems.

The motor driver and DC geared motors ensured smooth and controlled rover movement, while the power management system using a rechargeable battery and charger module provided stable operation for extended durations. Overall, the experimental results confirm that the proposed system is reliable, scalable, and efficient for real-time environmental monitoring and prediction.

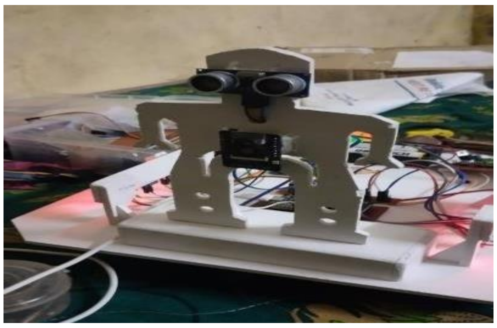


Figure 17 Front View

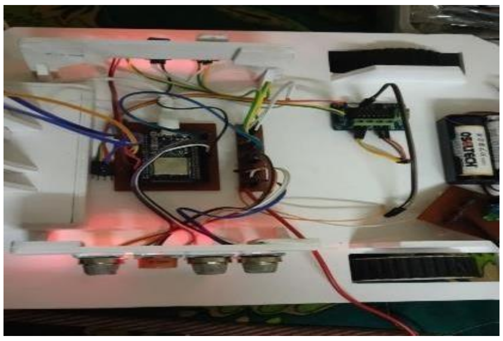


Figure 18 Back View

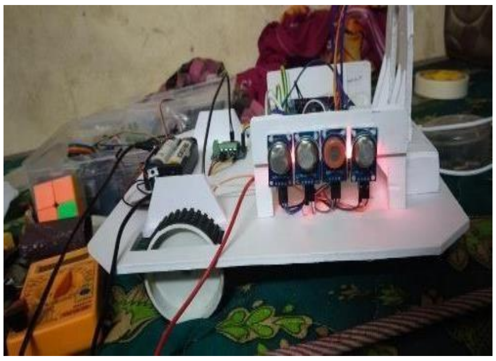


Figure 19 Side View

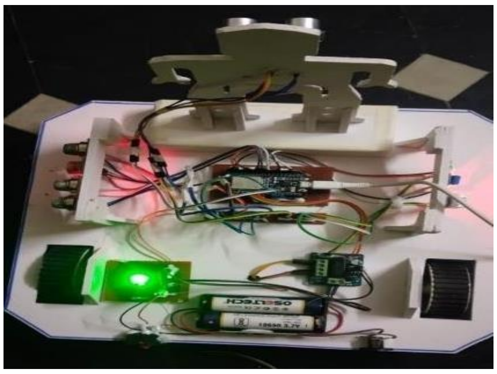
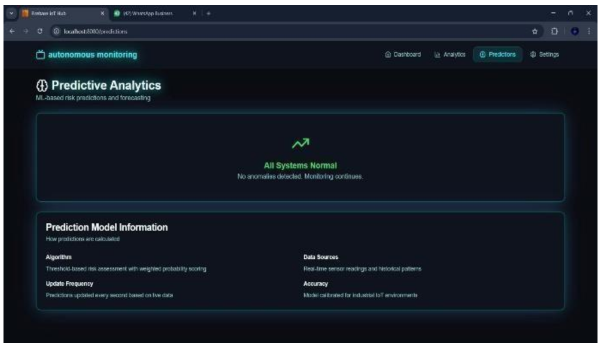


Figure 20 Top View

The results validate that the integration of IoT, autonomous mobility, live streaming, and machine learning offers a powerful solution for smart environmental monitoring applications such as agriculture, industrial safety, disaster management, and smart cities.


autonomous monitoring

Dashboard · Analytics · Predictions · Settings

Predictive Analytics

ML-based risk predictions and forecasting

All Systems Normal

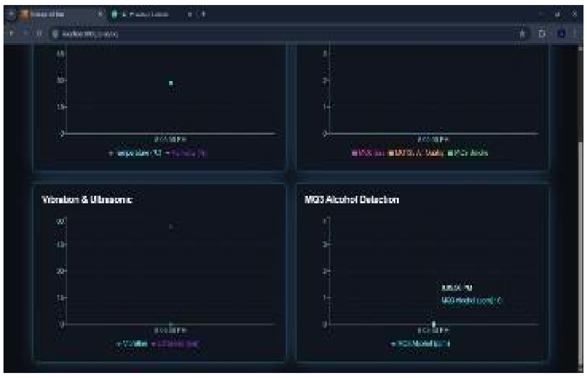
No anomalies detected. Monitoring continues.

Prediction Model Information

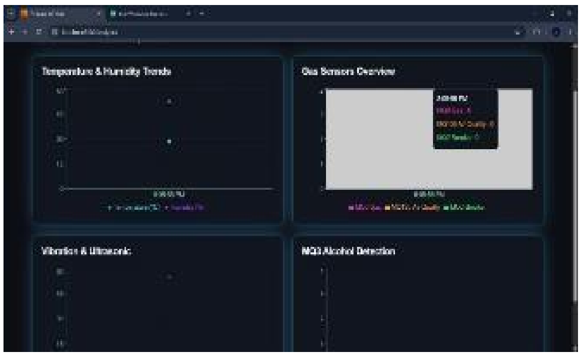
How predictions are calculated

Algorithm	**Data Sources**
Threshold-based risk assessment with weighted probability scoring	Real-time sensor readings and historical patterns
Update Frequency	**Accuracy**
Predictions updated every second based on live data	Model calibrated for industrial IoT environments

(a) Dashboard Predictive Analytics



(b) Sensors Outputs In Graphical Representation



5. CONCLUSION

This paper presented an IoT-based autonomous environmental monitoring vehicle that integrates multi- sensor data acquisition, cloud connectivity, live video streaming, and machine learning-based analysis. The system provides a reliable and scalable solution for real- time environmental monitoring in diverse applications.

FUTURE SCOPE

Future enhancements may include the integration of advanced deep learning models, GPS-based location tracking, solar-powered energy management, and coordinated multi-vehicle deployment. These improvements can further enhance system.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010

[2] S. R. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of Things (IoT): A literature review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164– 173, 2015.

[3] Espressif Systems, "ESP32 Series Datasheet," Espressif Systems Inc., 2023. [Online]. Available: https://www.espressif.com

[4] Espressif Systems, "ESP32-CAM Technical Reference Manual," Espressif Systems Inc., 2022.

[5] A. Al-Fuqaha et al., "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[6] M. A. Hassan et al., "IoT-based smart environmental monitoring using machine learning," *IEEE Access*, vol. 8, pp. 152422–152436, 2020.

[7] Hanwei Electronics, "MQ Series Gas Sensor Technical Datasheets (MQ2, MQ3, MQ6, MQ135)," 2021.

[8] Aosong Electronics, "DHT22 Temperature and Humidity Sensor Datasheet," 2020.

[9] InvenSense, "MPU6050 Six-Axis Motion Tracking Device Datasheet," 2019.

[10] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[11] S. Thrun et al., "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[12] K. Ashton, "That 'Internet of Things' thing," *RFID Journal*, 2009.

[13] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, Springer, 2016.

[14] IEEE Standards Association, "IEEE Standard for IoT Architecture Framework," IEEE Std 2413-2019.

[15] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.