HazardScout: Multi-Purpose Emergency Response Robot

A.Y.2024-2025

CERTIFICATE

This is to certify that the project entitled **HazardScout: MultiPurpose Emergency Detection Robot** is a bonafide work of **Shashwat Shah, Kruti Shah, Khushi Jobanputra and Manasvi Gupta** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.Tech. in Computer Engineering

Prof. Ruhina Karani
Internal Guide

(Name and sign)
External Guide

Dr Meera Narvekar
Head of Department

Dr. Hari Vasudevan
Principal

Project Report Approval for B.Tech.

This project report entitled **HazardScout: MultiPurpose Emergency Detection Robot** by **Shashwat Shah, Kruti Shah, Khushi Jobanputra and Manasvi Gupta** is approved for the degree of ***B.Tech. in Computer Engineering*.**

Examiners

1.----------

2.----------

Date:

Place:

Declaration

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

(Signature)

Shashwat Shah 60004220126

(Signature)

Kruti Shah 60004210122

(Signature)

Khushi Jobanputra 60004210147

(Signature)

Manasvi Gupta 60004210235

Date:

Abstract

This project develops an IoT-enabled robotic system designed for autonomous, real-time gas leakage detection in complex industrial pipeline networks. The robot uses a unique locomotion mechanism that combines electric motors, a fidget spinner-inspired wheel configuration, and magnetic attachments, allowing it to traverse both vertical and horizontal pipes. This design ensures stability and adaptability to varying pipe diameters, bends, and joints, maintaining continuous contact for effective inspection.

The robot is equipped with a range of sensors, including a gas sensor for detecting combustible gases, a DHT11 sensor for monitoring ambient temperature and humidity, and an ultrasonic sensor for obstacle detection. Sensor data is processed by an onboard machine learning algorithm, which predicts the likelihood of gas leaks, enhancing detection accuracy and minimizing false alarms. Additionally, the robot features an onboard camera and LED torch to facilitate visual inspections in low-light or hard-to-reach areas.

All sensor data and visual information are transmitted to a cloud platform via Wi-Fi, enabling remote monitoring, data logging, and control through a centralized dashboard. In case of critical events, Twilio-based SMS notifications are triggered for immediate response. Designed with scalability and safety in mind, this system offers a reliable solution for continuous monitoring in industries such as oil, gas, and chemicals, improving worker safety, operational efficiency, and environmental protection through the integration of robotics, IoT, and machine learning.

Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1.	IOT	Internet of things
2.	LPG	Liquified petroleum gas
3.	MQ5	Metal Oxide Semiconductor Gas Sensor
4.	ML	Machine Learning
5.	PWM	Pulse with modulation
6.	DHT	Digital humidity and temperature sensor
7.	GUI	Graphical user interface

1. INTRODUCTION

1.1. Description

This work envisions an integrative IoT-powered robotic inspection system tailored for intelligent, real-time industrial pipe network gas leakage detection. Industrial pipelines tend to extend through intricate and risky terrain where manual inspection is inefficient as well as risky. In an attempt to overcome such limitations, envisaged is an autonomous robotic system serving to scan both the vertical and horizontal pipe networks to provide ongoing monitoring and immediate detection of leaks without the intervention of human beings.

The system uses a sturdy mechanical construction integrating traction wheels, strategically positioned magnets, and motors to have a strong hold on metallic pipe surfaces. One of its distinctive features is its fidget spinner-like configuration of three wheels, which provides 360° flexibility and balance. This makes it possible to move smoothly along pipe bends, elbow joins, changes in elevation, and different diameters—parameters which challenge conventional inspection tools.

The robot platform is equipped with an integrated set of environment sensors:

I. MQ-6 combustible gas sensor to sense combustible fuels like LPG, methane, and propane.

II. A temperature and humidity sensor to identify unusual thermal or moisture levels that act as indicators of pipe degradation or leaks.

III. An obstacle avoidance and proximity sensor using ultrasound, facilitating safe, collision-free travel through dense industrial environments.

In order to increase the trustworthiness of leak detection, the system uses a machine learning algorithm based on multi-sensor data to predict the risk and impact of a gas leak by processing patterns and irregularities. This is far better than detection using threshold-based methods. This model is adjustable in real time and has the capability to keep learning to react to changes in the environment dynamically.

A high-resolution onboard camera in conjunction with an LED flashlight ensures good visual feedback even under low-light or within enclosed pipeline interiors. Visual inspection and

documentation of faults or leaks is made possible in real-time. All sensor information and visual feeds are sent to a centralized cloud system, where approved users are able to view pipelines in real-time through a specific dashboard portal. In case of a leak, automated alerts and emergency procedures can instantaneously be initiated. By integrating the components of robotics, IoT, and AI, the system offers an intelligent and scalable solution to improve industrial safety, reduce human risk, and ensure structural integrity of gas distribution networks.

1.2. Problem Formulation

Industrial pipe leakages pose serious risks to safety and infrastructure, resulting in potentially disastrous consequences like fires, explosions, toxic exposures, environmental pollution, and loss of life. Risks are even greater in massive industrial facilities, where intricate pipe networks cover wide geographies, including above ground and enclosed areas, and thus are not easily accessible for monitoring using conventional methods. Current manual inspection methods are time-consuming, slow, and typically not effective in detecting minor or early leaks, which, if not detected, will propagate to devastating events. In addition, they typically have no predictive function and real-time response, resulting in slow detection, low situational awareness, and poor emergency readiness.
The structural intricacy of pipelines—complete with vertical sections, tight radii, joint clamps, rusty surfaces, and small clearances—adds to inspection and maintenance difficulty. Most of these areas are inaccessible to human inspectors or are unsafe to manually inspect. Low light within covered or underground portions further disrupts visual inspection and decreases opportunities for early intervention.

To meet these key limitations, the current project presents an IoT-enabled autonomous robotic system designed to conduct intelligent, real-time inspection and detection of gas leaks in industrial pipe environments. The robot has the ability to move independently along horizontal, vertical, and sloped segments of pipe due to a fidget spinner-inspired tri-wheeled mechanism augmented with magnetic grabbing capabilities. This mechanical innovation allows the robot to stabilize, fit into different pipe diameters, and navigate around physical obstructions such as pipe joints and bends. With a set of environment sensors—gas, temperature, humidity, and ultrasonic sensors—the robot gathers data regularly to detect abnormalities in the form of leaks or unsafe conditions. This data is analyzed via integrated machine learning, which increases detection accuracy by looking for patterns and correlations characteristic of initial leak development. Predictive analytics provides assurance that threats are identified before they happen, enabling proactive maintenance and early intervention.

For improved situational awareness, the system uses both a high-resolution imaging camera and an LED flash, enabling effective visual inspection under low-visibility situations. All sensor information and imagery is streamed in real time to a centralized cloud-hosted command system where remote operators can monitor the progress of the robot, view reports of anomalies, and execute emergency procedures if needed.

In short, the goal of the project is to revolutionize conventional gas pipe inspection processes using robotics, IoT, and machine learning to establish a smart, adaptive, and scalable inspection system. In addition to less reliance on manual labor, it both increases overall operational safety, accuracy in inspection, and response efficiency, guaranteeing proactive industrial management of gas leaks.

1.3. Motivation

Traditionally, industrial gas detection has depended on stationary sensors and manual inspection. Basic detection has been provided by fixed sensor grids and handheld detectors, but at the cost of disadvantages:

I. They are time-consuming, prone to making mistakes, and unsafe under hazardous conditions.
II. Fixed sensor networks only cover predefined sections and cannot detect spills along the entire length of the pipe, particularly in high places or inaccessible sections.
III. They have mobility but cannot move well within enclosed or constricted industrial environments and have difficulty maintaining proximity to pipes.

To transcend such limitations, our solution envisions an IoT-driven, autonomous robot to move along pipelines both horizontally and vertically by utilizing magnets to provide traction and fidget-spinner-inspired wheel setup for mobility. It incorporates sensors for monitoring different components of the gas, temperature, and humidity, along with ultrasonic obstacle detection and camera support with the feature of a light source to ensure maximum visibility. It further relies on machine learning to interpret data from sensors to identify leaks correctly and sends all information in real time to a monitoring center, facilitating faster and intelligent action.

It is motivated by the continuous, touchpoint monitoring requirement, which will leave no part of the pipe unscanned. Conventional solutions cannot achieve coverage across pipelines installed at heights, along ceilings, or in tight enclosures. By enabling the robot to actually move along such surfaces with firm traction, the system ensures inspection accuracy in areas that, heretofore, have been inaccessible or pose too much risk with human intervention.

Additionally, the addition of machine learning provides predictive intelligence to the system. Instead of simply responding to hazardous levels, the robot is able to identify patterns in temperature, gas concentrations, and changes in humidity to predict leaks before they reach critical levels. This predictive function allows the maintenance crews to take proactive action, cutting downtime, preventing accidents, and averting expensive damage to infrastructure.

Finally, the necessity of real-time situational awareness within industrial environments highlights the significance of the above project. Visual verification using onboard imagery, in addition to real-time sensor data, provides monitoring teams with actionable information at any given instant. From minor leaks to identifying sections of corroded pipes to navigating past obstructions, the robot ensures each inch of the pipe is scanned at high fidelity, ultimately facilitating safer, smarter, and more efficient industrial processes

1.4. Proposed Solution

To overcome the drawbacks of conventional methods of detecting leaks, we suggest an Internet of Things-enabled pipe-climbing robot with real-time pipe inspection and detection capabilities. It is provided with a gas sensor, temperature sensor, and humidity sensor, along with an ultrasonic sensor to detect the leak and obstacles with accuracy. A fidget spinner-inspired wheel with integrated magnets ensures stable gripping and movement along metallic pipes both horizontally and vertically, even along curvatures and junctions.

The device employs a machine learning algorithm to examine multi-sensor data to identify leaks correctly, eliminating false positive alarms. An imaging module provided with a torch supports low-light visibility inspection. All data is sent to a cloud-enabled monitoring and control center, where there is the capability to access remotely, control, and initiate alarms.

Due to its modular software and hardware design, the system is versatile for deployment in many different types of industrial environments. In oil refineries, chemical processing facilities, and urban gas supply pipelines, the robot is easily configured to run on pipelines of various diameters under any type of environment. This adaptability guarantees the solution is not merely efficient but scalable for mass industrial deployment.

Furthermore, the addition of cloud connectivity provides data centralization, making it possible for operators to monitor inspection history, observe environment trends, and base decisions accordingly. This data is also usable to update the machine learning model and refine its intelligence and responsiveness. Integration into industrial control systems is achievable, facilitating automatic shutdown or mitigation strategies in case of detection of a leak.

To support complex and harsh terrain along pipelines, the mechanical system of the robot consists of spring-loaded grippers to accommodate irregular terrain and deliver stable traction. Fidget spinner-inspired wheel geometry assists in balance while traveling along undulated sections, joints, and stair-like bumps normally in the way of inspection robots. It is so light in weight to not harm old or fragile pipeline infrastructure.

From operations and maintenance perspectives, ease of use has been considered in designing the robot. All its components are modular and simple to replace or update. An intuitive monitoring and control interface allows even nonspecialist employees to utilize the system with minimal training. Visualizations of alerts and analytics for enhanced situational awareness are possible using dashboards. In summary, the envisaged robotic system is groundbreaking in the detection of gas leaks with its cost-efficient, intelligent, and autonomous approach, which considerably shortens inspection downtime and lessens human risk. It adds efficiency, safety, and reliability to pipe inspection, benefiting proactive infrastructure management and more efficient industrial processes.

1.5. Scope of the project

This research is to develop an IoT-powered, autonomous robotic system for real-time detection of gas leaks and smart pipeline inspection in industrial setups. Its field is extensive and cross-disciplinary, spanning mechanical design, sensor integration, embedded computing, machine learning, cloud computing, and real-time monitoring at remote locations. It is proposed to generate a scalable, smart, and autonomous system to remedy the limitations of conventional gas leak detection systems, which normally are manual, discontinuous, and inefficient in industrial environments.

Hardware Scope

The hardware design consists of a lightweight, portable robot specifically intended to move along both horizontal and vertical metallic pipes. The robot's casing has permanent magnets to firmly attach to metallic surfaces and fidget-spinner-inspired wheels for maneuverability along bends, pipe clamps, and irregular terrain. Spring-loaded wheels enable the robot to fit any pipe diameter, and the chassis has a friction drive to ensure continuous contact even while turning and negotiating changes.

Embedded within the hardware are various sensors:

I. MQ-series Gas Sensor: It senses the availability of gases such as methane, propane, butane, and other flammable gases.
II. DHT11 or DHT22 Sensor: Sensing ambient temperature and humidity, which are determinative for correctly interpreting the behavior of the gas sensor.
III. Ultrasonic Sensor (e.g., HC-SR04): Identifies obstacles within reach to avoid collisions or falls.
IV. Imaging Module (Camera): Takes live video and images of pipe interiors for visual inspection.
V. Strong Torch/LEDs: Guarantees good visibility in dimly lit or cramped locations.

The robot is powered by rechargeable batteries and a microcontroller or single-board computer like Arduino, ESP32, or Raspberry Pi, facilitating edge computing, low-latency data processing, and direct integration with cloud services.

Software and Algorithmic Scope

The system is backed by an ML pipeline analyzing multivariate sensor data to predict the probability of a gas leak. With historical data and live sensor recordings, the ML model is able to identify patterns as well as anomalies within sensor data—namely changes in concentration of the gas, temperature, and humidity. This predictive model is much better at eliminating false positives and detecting leaks accurately than threshold-only methods.

The software stack comprises:

I. Sensor data preprocessing and normalization
II. Model training with supervised learning methods (such as Random Forest, SVM, or light CNNs)
III. Leak prediction and classification logic
IV. Edge computing functionality to execute simple logic onboard

V. Cloud integration, such as Firebase and AWS IoT Core, for real-time data upload, alert generation, and data visualization

All data is streamed to a centralized cloud monitoring platform from where operators view real-time metrics like gas levels, humidity, temperature, robot position, camera streaming, and obstacle detection alerts. The dashboard will have intuitive graphs, alerting mechanisms, inspection records, and the ability to remotely control.

Operational Scope

This robot is designed to work in various and risky industrial environments, including:

I. Gas refineries and petroleum processing plants
II. Urban natural gas distribution pipelines
III. Chemical processing facilities
IV. Underground utility tunnels
V. Industrial HVAC systems

It will work independently but has provisions for manual override using remote control in critical situations. It will conduct routine inspection, emergency inspection, and preventive maintenance work. Its design will also allow it to tolerate extreme conditions like high humidity, dust environments, variable temperature, and low light. It will support pipes ranging from 2 to 8 inches in diameter, depending on the adaptability of grippers.

Functional Scope

The main functions addressed in the current project are:

I. Autonomous navigation along intricate pipe geometries (overhead, vertical, and horizontal)
II. Real-time gas leak detection with multiple sensors
III. Ultrasonic sensing for obstacle detection and avoidance
IV. Visual inspection using image capture and real-time video feed
V. Data collection, pre-processing, and cloud server transmission
VI. Intelligent leak detection using a machine learning-trained model
VII. Operator user interface/dashboard with live metrics, alerts, and historical data
VIII. Alert generation through email/SMS/notification upon detection of leak
IX. Low-power and fail-safe operation with minimal diagnostics

2. SURVEY:

2.1 Literature Survey:

Bi, et al. [1] have proposed the design of magnetic-wheeled robots to climb metallic surfaces for the inspection of large industrial tanks and pipelines. The design of the robot involves the integration of strong magnetic adhesion and passive adaptive mechanisms in order to allow the device to sustain traction along vertical and curved surfaces. The technology is particularly useful in environments where human inspection will pose danger, including gas refineries and chemical plants under the risk of leaks. The study mentions possible uses in detecting leaks of gas, where sensors of concentrations of the gas are employed to detect levels of hazardous gases in real time.

Lee, et al. [2] propose a pipe-inspecting robot with articulated joints, enabling it to move within narrow and complex pipes. The robot is positioned to bend around obstacles such as sharp bends, clamps, and irregular ground, all typical of industrial pipes. Emphasis is placed on mobility through modularity and autonomous control, where the robot is positioned to access narrow areas to detect structural health and gas leaks. This mobility aspect is critical for a robot within challenging environments where access is critical.

Rahman, et al. [3] share the integration of various environmental sensors such as gas, temperature, and humidity sensors to build a more integrated robotic system for detecting leaks. One of the most significant contributions of the study is fusing the data from different sensors to enhance the accuracy of gas concentration detection. Through real sensor data, the robots are able to identify leaks at the earliest detection and issue alerts remotely, and preventive actions may be made prior to escalation of the situation. This real-time detection is most suitable for risky industrial environments.

Zhao, et al. [4] target robots applying thermal and infrared sensing to identify fire risks in industrial setups. It has autonomous fire-extinguishing capabilities, which allow for the extinguishment of fires using chemical extinguishers or jets of water. This capability of the robot to fight fires actively and support operations at high temperatures is the perfect fit for industrial fire management and protection. All these attributes fit your design's intention of integrating fire suppression under its capabilities.

Hirose, et al. [5] report the deployment of snake robots due to the success they have seen in their deployment in the 2017 Mexico City earthquake. Serpentine motion-enabled robots have access to collapsed and enclosed regions inaccessible to conventional robots and are therefore appropriate for inspection after disaster events, detection of leaks from gases, and the detection of hazards in hard-

to-reach industrial environments.

Hossin, et al. [6] discuss integrating different sensors (such as thermal, gas, and smoke sensors) onboard autonomous robots to identify fires and leaks of flammable gases. With fusion of data from the sensors, the robot will correctly identify the origin of a fire or gas leak and take action to suppress it in real time. Robots like these are deployable within risky industrial environments where it would be unsafe for humans. Multi-sensor data fusion is stressed in the paper as mandatory to achieve accuracy in detection of hazards.

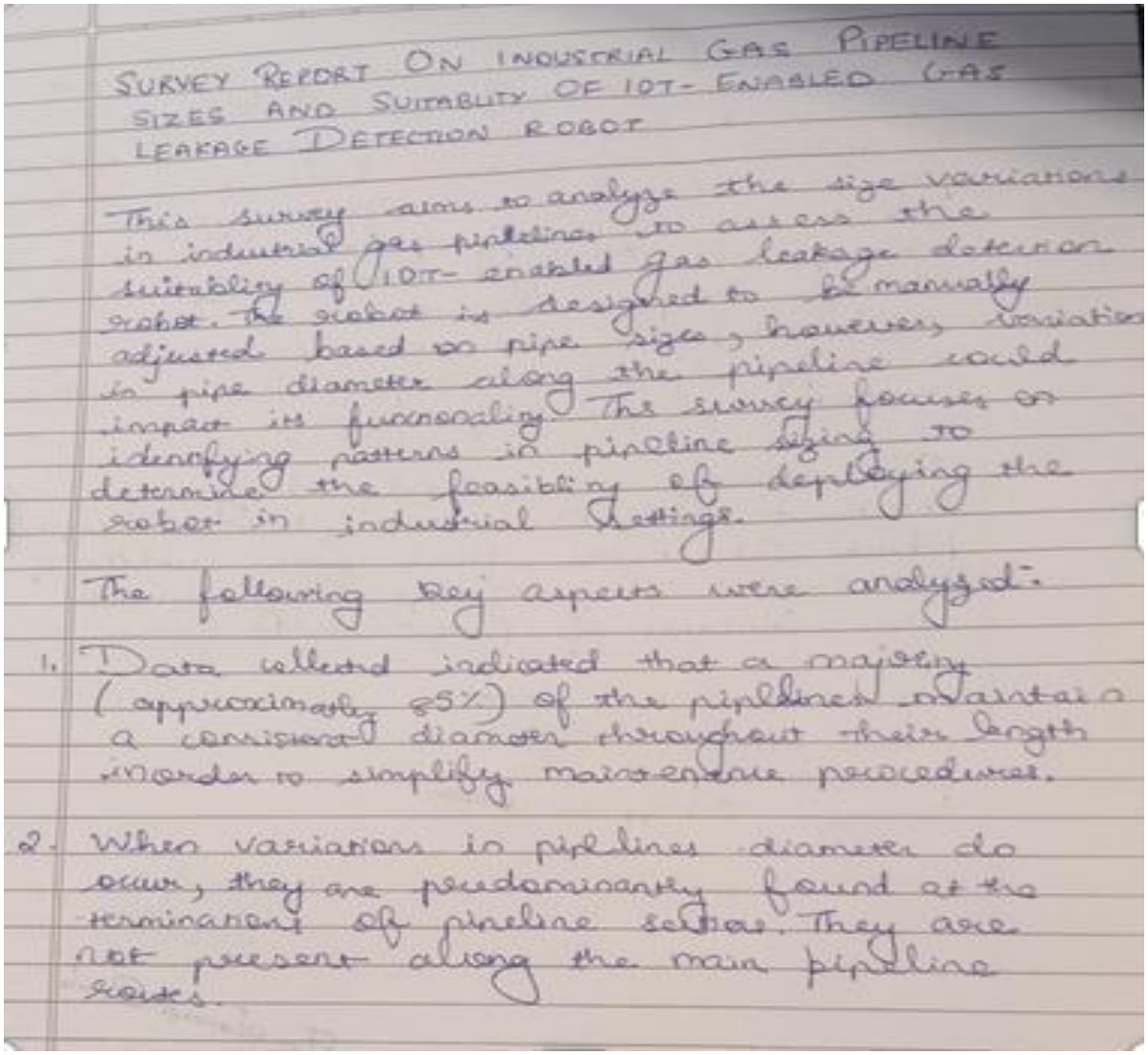
Zhang, et al. [7] discuss magnetic wall-climbing robots intended for petrochemical environments. They are constructed for monitoring potentially leaking gas from corroded tanks and pipelines. Continuous and secure monitoring within extremely risky locations is facilitated by their magnetic adhesion and mobility along metallic vertical planes.

Dhakad, et al. [8] highlight the importance of real-time data transmission from robots working in risky locations like refineries and chemical factories. With high-resolution cameras and sensor arrays to monitor the environment, the robots have the capability to transfer data to far-end operators in real time, giving critical information about industrial apparatus conditions and the level of hazardous gases. This is distinctly reducing reliance on human inspection and protecting employees from harm by restricting human access to harmful environments.

Taher, et al. [9] examine the implementation of wireless sensor networks (WSNs) in mobile robots to monitor industrial environments for toxic gas levels. The WSN-enabled robots scan the environment round the clock for hazardous levels of gases and trigger alarms autonomously should levels exceed set limits. It is autonomous in nature and sends alerts along with data to far-away operators to support better decisions and safety in real-time.

Ehsan, et al. [10] introduce a singular robotic system utilizing water jet mobility as well as fire suppression. When integrated with thermal imaging and real-time control, the robot is able to move through obstacles and target fire hotspots with great specificity in industrial settings where conventional solutions are inadequate.

2.2 Field Survey:



SURVEY REPORT ON INDUSTRIAL GAS PIPELINE SIZES AND SUITABILITY OF IOT-ENABLED GAS LEAKAGE DETECTION ROBOT

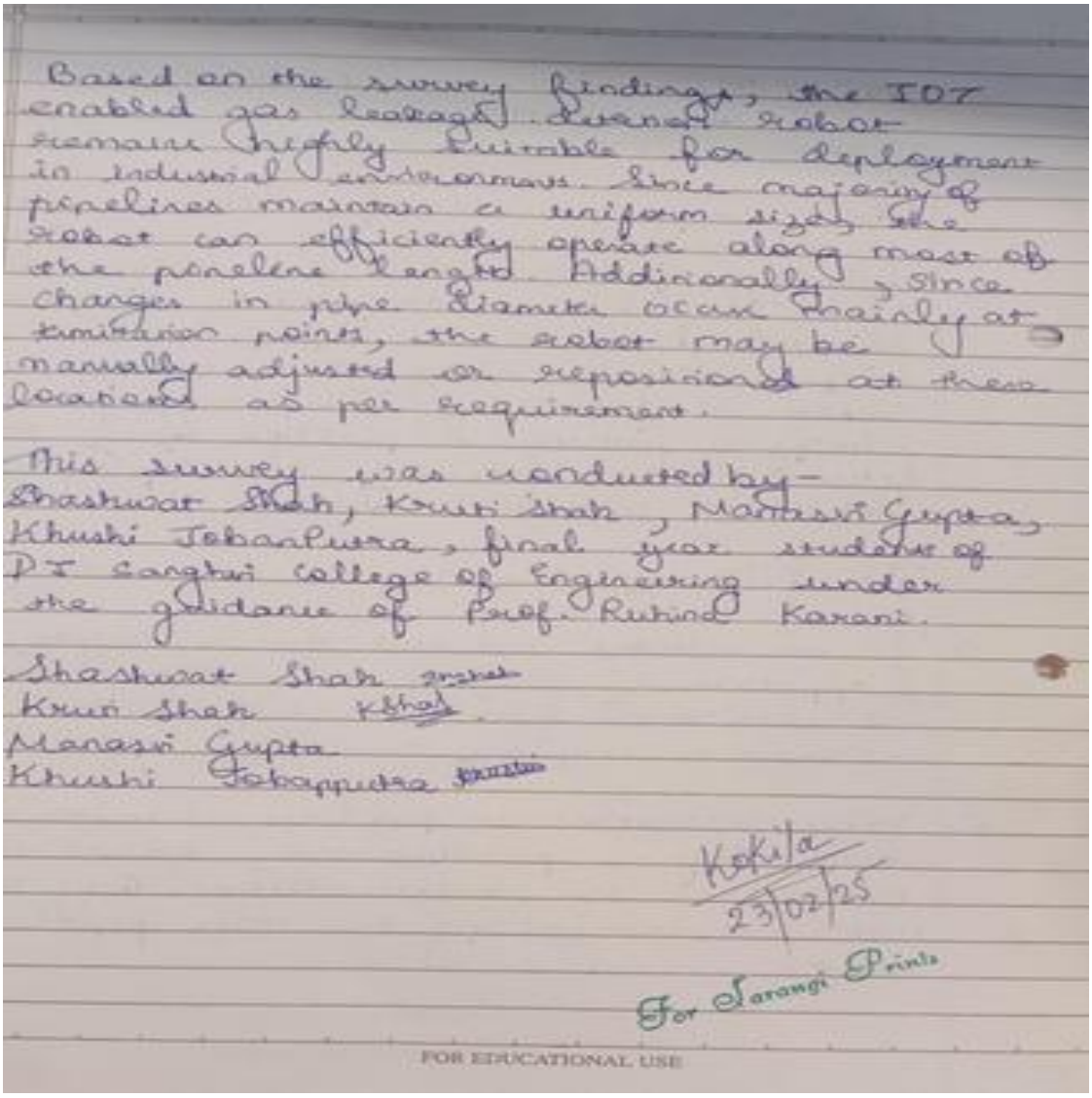
This survey aims to analyze the size variations in industrial gas pipelines to assess the suitability of IOT-enabled gas leakage detection robot. The robot is designed to be manually adjusted based on pipe sizes, however, variation in pipe diameter along the pipeline could impact its functionality. The survey focuses on identifying patterns in pipeline sizing to determine the feasibility of deploying the robot in industrial settings.

The following key aspects were analysed:

1. Data collected indicated that a majority (approximately 85%) of the pipelines maintain a consistent diameter throughout their length in order to simplify maintenance procedures.

2. When variations in pipelines diameter do occur, they are predominantly found at the terminations of pipeline sections. They are not present along the main pipeline routes.

Fig No 2.2.1 Field survey conducted on industrial gas pipeline sizes



Based on the survey findings, the IOT enabled gas leakage detection robot remains highly suitable for deployment in industrial environments. Since majority of pipelines maintain a uniform size, the robot can efficiently operate along most of the pipeline length. Additionally, since changes in pipe diameter occur mainly at limitation points, the robot may be manually adjusted or repositioned at these locations as per requirement.

This survey was conducted by-
Shashwat Shah, Kruti Shah, Manasvi Gupta, Khushi Jobanputra, final year students of D.J. Sanghvi College of Engineering under the guidance of Prof. Rutvi Karani.

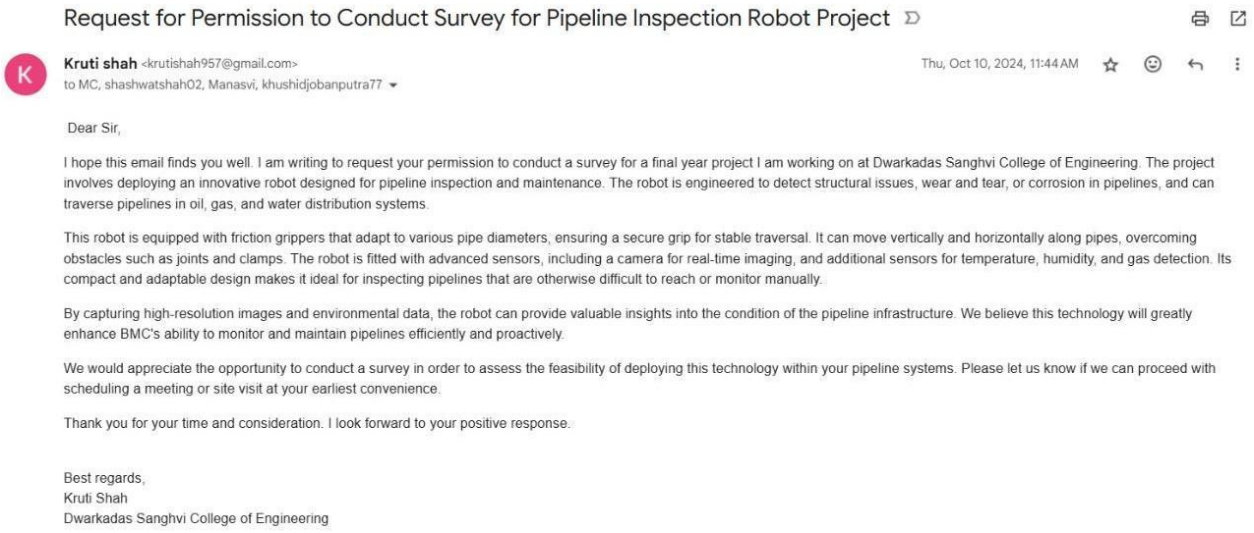
Shashwat Shah
Kruti Shah
Manasvi Gupta
Khushi Jobanputra

Kokila
23/02/25

For Sarangi Prints

FOR EDUCATIONAL USE

Fig No 2.2.1 Field survey conducted on industrial gas pipeline sizes



Request for Permission to Conduct Survey for Pipeline Inspection Robot Project

Kruti shah <krutishah957@gmail.com> Thu, Oct 10, 2024, 11:44 AM
to MC, shashwatshah02, Manasvi, khushidjobanputra77

Dear Sir,

I hope this email finds you well. I am writing to request your permission to conduct a survey for a final year project I am working on at Dwarkadas Sanghvi College of Engineering. The project involves deploying an innovative robot designed for pipeline inspection and maintenance. The robot is engineered to detect structural issues, wear and tear, or corrosion in pipelines, and can traverse pipelines in oil, gas, and water distribution systems.

This robot is equipped with friction grippers that adapt to various pipe diameters, ensuring a secure grip for stable traversal. It can move vertically and horizontally along pipes, overcoming obstacles such as joints and clamps. The robot is fitted with advanced sensors, including a camera for real-time imaging, and additional sensors for temperature, humidity, and gas detection. Its compact and adaptable design makes it ideal for inspecting pipelines that are otherwise difficult to reach or monitor manually.

By capturing high-resolution images and environmental data, the robot can provide valuable insights into the condition of the pipeline infrastructure. We believe this technology will greatly enhance BMC's ability to monitor and maintain pipelines efficiently and proactively.

We would appreciate the opportunity to conduct a survey in order to assess the feasibility of deploying this technology within your pipeline systems. Please let us know if we can proceed with scheduling a meeting or site visit at your earliest convenience.

Thank you for your time and consideration. I look forward to your positive response.

Best regards,
Kruti Shah
Dwarkadas Sanghvi College of Engineering

Fig No 2.2.2: Request to BMC to conduct survey

3. SYSTEM ANALYSIS

3.1 Functional Requirements

3.1.1 Robot Mobility Control

The system should enable the robot to move in forward, backward, left, and right directions through remote commands. The robot should be able to climb vertically and move horizontally on metallic gas pipelines. The robot should automatically stop when an obstacle is sensed using the ultrasonic sensor.

3.1.2 Gas Leak Detection

The gas sensor will always monitor the surroundings for any gas leak. On sensing a gas leak, the system will provide a warning to the Monitoring and Control Center.

3.1.3 Environmental Monitoring

The robot will read temperature and humidity values using the temperature and humidity sensor. The environmental data read will be stored in the cloud-based storage.

3.1.4 Data Processing and Analysis

The processing unit will send sensor data to a machine learning model running on the data store. The machine learning model will process temperature, humidity, and gas data to forecast the probability of a leak.

3.1.5 Visual and Imaging Inspection

The robot shall capture images of its surroundings using the imaging device. Captured images shall be transmitted and stored in the data store for further inspection.

3.1.6 Lighting and Visibility

The robot will have a remotely operated light source to enable illumination in environments with poor lighting.

3.1.7 Real-Time Communication

The system will provide real-time communication between the robot and the Monitoring and Control Center. The robot will transmit periodic status updates such as location, sensor readings, and visual feed.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

The system shall provide real-time data processing with a maximum delay of 2 seconds between the detection of a gas leak and the alert being sent to the Monitoring and Control Center. The robot shall be capable of continuously operating for at least 6 hours on a single charge under normal operating conditions. The ultrasonic sensor shall have a minimum detection range of 1 meter and a maximum range of 3 meters to detect obstacles.

3.2.2 Reliability Requirements

The system shall have an uptime of at least 99.9% during its operation in a controlled environment. The robot shall be able to function effectively in both high and low temperatures ranging from -10°C to 50°C. The gas sensor shall maintain an accuracy of 95% or higher for gas leak detection.

3.2.3 Scalability Requirements

The system shall be scalable to accommodate additional robots or monitoring points without significant changes to the existing infrastructure. The cloud-based data store shall be capable of handling large volumes of sensor data and images generated by multiple robots in real-time.

3.2.4 Usability Requirements

The system shall include an easy-to-use user interface (UI) for remote monitoring and control. The system shall allow operators to interact with the robot and control its movements, adjust settings, and view sensor data in real-time

3.2.5 Security Requirements

The system shall employ encrypted communication between the robot, the Monitoring and Control Center, and the cloud-based data store to ensure the confidentiality and integrity of data. Access to the monitoring and control system shall be authenticated and authorized to prevent unauthorized users from gaining access.

3.2.6 Maintainability Requirements

The system shall be designed to allow easy maintenance, including robot hardware repairs, software updates, and sensor replacements. The robot's sensors and components shall have a lifespan of at least 2 years under normal operating conditions.

3.2.7 Environmental Requirements

The robot shall be designed to operate effectively in industrial environments, including exposure to dust, moisture, and vibration without performance degradation.
The system shall be designed to be energy-efficient, reducing environmental impact during operation.

3.3 Specific Requirements

3.3.1 Performance Requirements

The system will ensure real-time processing of data with a maximum time lag of 2 seconds between the identification of a gas leak and the alarm being sent to the Monitoring and Control Center. The robot will be able to run continuously for a minimum of 6 hours on a single battery under normal operating conditions. The

minimum and maxim detection ranges of an ultrasonic sensor will be 1 meter and 3 meters respectively. have a minimum detection range of 1 meter and a maximum detection range of 3 meters to identify obstacles.

3.3.2 Reliability Requirements

The system should have a minimum uptime of 99.9% when operating while a controlled environment. The robot should be able to operate efficiently under high and low temperature conditions between -10°C and 50°C. The gas sensor should have an accuracy rate of 95% or more for gas leak detection.

3.3.3 Scalability Requirements

The system should be extensible to be able to take more robots or monitoring points in without a radical overhaul of existing infrastructure. The cloud-based storage facility should support high levels of sensor data and images produced by many robots at the same time.

3.3.4 Usability Requirements

The system will have a simple-to-use user interface (UI) for remote monitoring and control. The system will enable operators to exchange information with the robot and command its movement, configure settings, and check sensor data in real-time.

3.3.5 Security Requirements

The system will utilize encrypted communication among the robot, the Monitoring and Control Center, and the cloud-based data repository to provide confidentiality and integrity for data. Access to the monitoring and control system will be authenticated and approved to block unauthorized users from entering.

3.4 Use-Case Diagrams and description

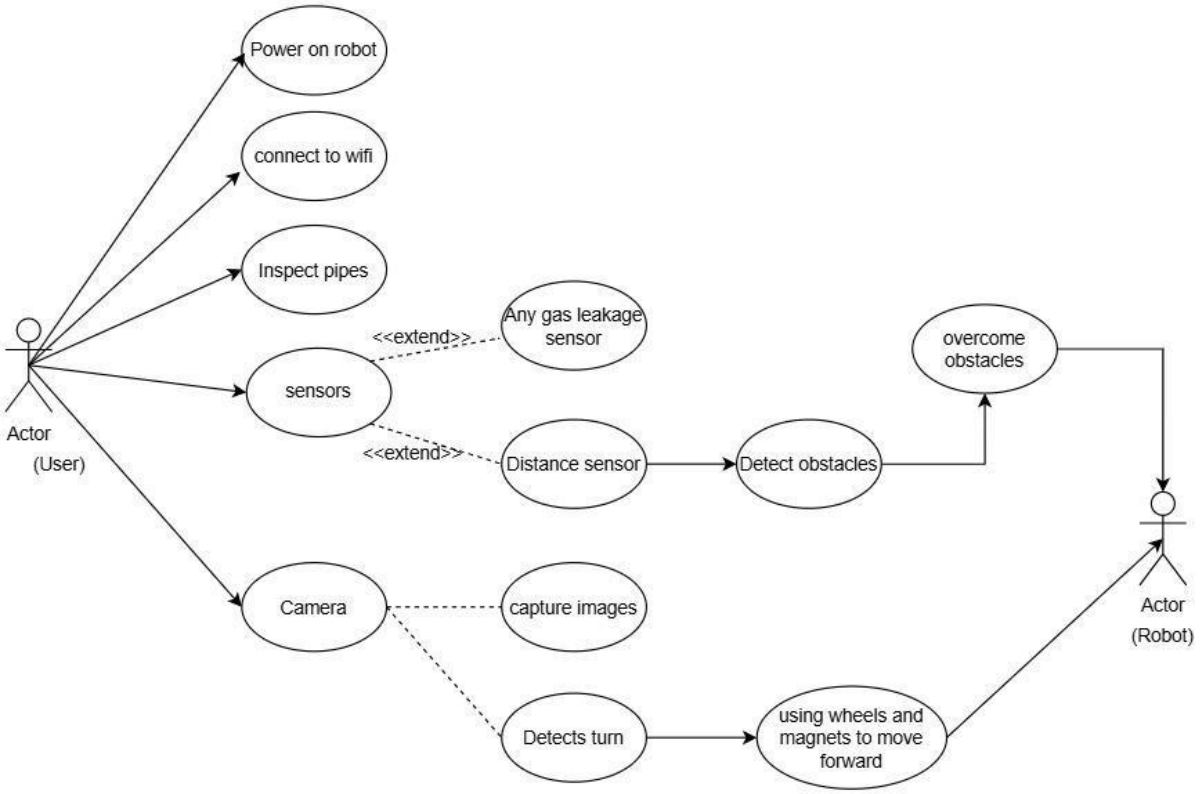


Fig 3.4.1: Use Case Diagram of the IOT enabled robot

Actors

1. User (Human Operator) – Initiates and controls the robot functions.

2. Robot – Performs the actual pipe inspection, including navigation, sensing, and image capturing.

Use Cases and Relationships

User-Initiated Use Cases

Power on robot: The user starts the robot system.

Connect to Wi-Fi: Enables the robot to transmit data or be controlled remotely.

Inspect pipes: The main task initiated by the user, involving various sub-functions.

Sensors (Extended Use Case):

- <<extend>> Any gas leakage sensor: Detects gas leaks inside the pipe.
- <<extend>> Distance sensor: Helps in obstacle detection and navigation.

Camera (Extended Use Case):

- Capture images: Takes real-time images for inspection.
- Detects turn: Identifies changes in direction or pipe turns using visual input.

Robot-Initiated Use Cases

- Detect obstacles: Uses sensors to identify barriers within the pipe.
- Overcome obstacles: Robot maneuvers around obstacles autonomously.
- Using wheels and magnets to move forward: Core locomotion mechanism inside pipes.

4 ANALYSIS MODELING

4.1 Sequence Diagrams

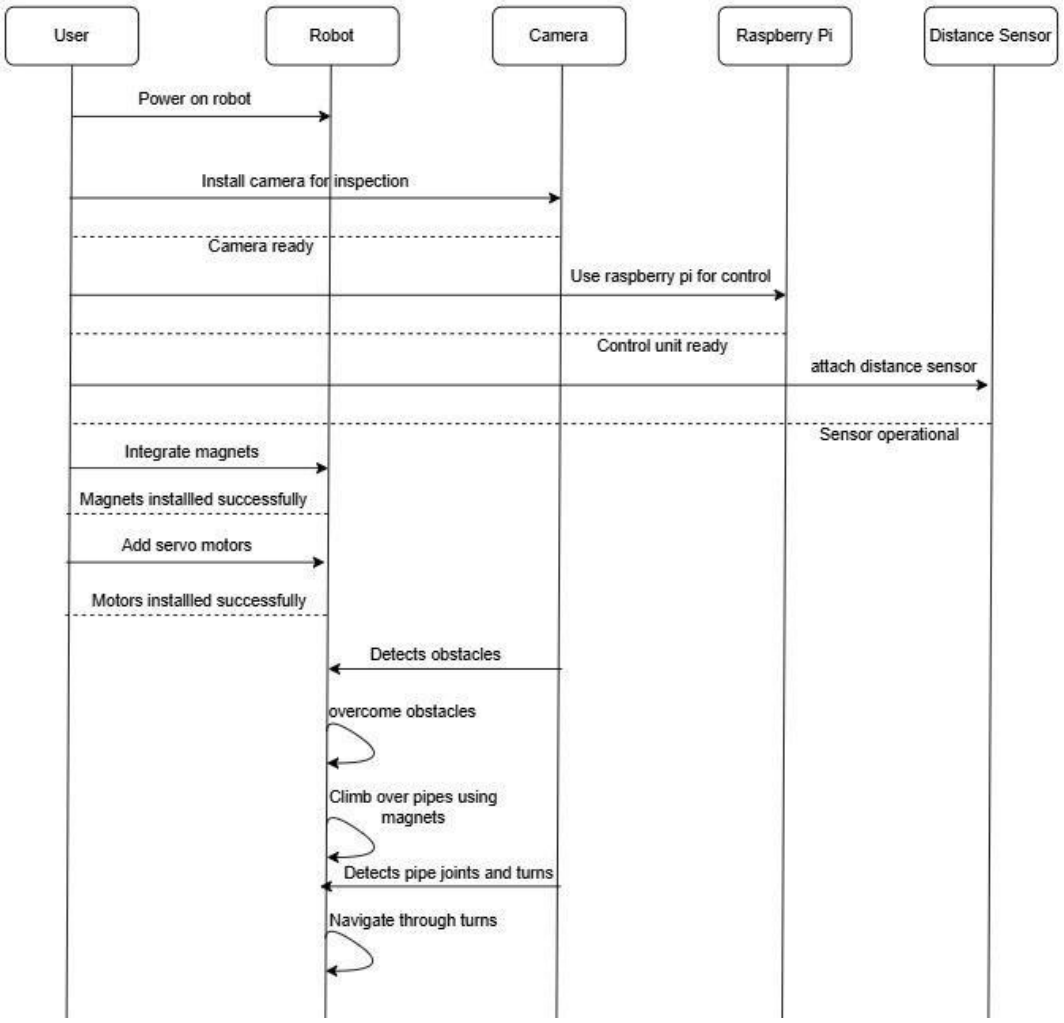


Fig 4.1.1: Sequence diagram depicting the flow of the robot.

Participants (Lifelines)

1. User – Human operator who initiates the process.

2. Robot – Core mechanical system that performs the inspection.

3. Camera – Vision module for capturing and analyzing pipe interiors.

4. Raspberry Pi – Acts as the control unit/microcontroller.

5. Distance Sensor – Senses distance to detect obstacles.

Flow of Actions

System Initialization

- User → Robot: Power on robot

  The inspection process begins when the user powers on the robot.

- Robot → Camera: Install camera for inspection

  A camera module is attached for visual inspection inside pipes.

- Camera → Robot: Camera ready

  The camera confirms successful installation.

- User → Raspberry Pi: Use Raspberry Pi for control

  The user assigns the control functionality to the Raspberry Pi.

- Raspberry Pi → User: Control unit ready

  Raspberry Pi is operational and ready for communication and control

- User → Raspberry Pi: Attach distance sensor

  The user connects the distance sensor to the Raspberry Pi.

- Distance Sensor → Raspberry Pi: Sensor operational Confirmation that the sensor is functioning correctly.

The operation begins with the user powering on the robot and then installing a camera for visual inspection to commence. The camera then reports itself ready to the robot.
Subsequently, the user commands the Raspberry Pi, the robot's control system, to take over, which it acknowledges by showing its readiness. The user proceeds to mount a distance sensor to increase the robot's awareness of the environment, and the sensor acknowledges its functionality. To facilitate the robot's movement on pipe surfaces, the user incorporates magnets, an operation that the robot acknowledges as successful. Likewise, servo motors are installed for movement or manipulation under control, which the robot verifies as successfully installed. Equipped, the robot automatically performs its inspection activities, first sensing and surmounting any obstacles in its way. With its embedded magnets, the robot is capable of climbing over pipes.

In addition, it can identify pipe turns and joints, allowing it to move along the intricate geometry of pipeline systems.

4.2 Functional Modeling

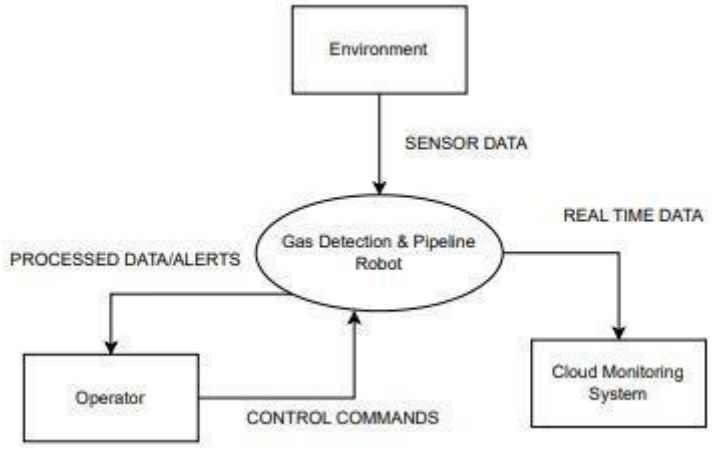


Fig 4.2.1: DFD Level 0 diagram depicting the various entities and processes.

The system described provides an inclusive solution for gas detection and pipeline monitoring via a robotic system. The robot, within a designated environment, is fitted with multiple sensors that constantly collect important information. Real-time raw sensor information is passed to a cloud-based monitoring system to facilitate constant observation and detailed evaluation of the pipeline condition. The robot itself or, more likely, the cloud system interprets this data to detect potential gas leaks or pipeline abnormalities and produces processed data and alarms that are relayed to a human operator. This operator can then send control commands back to the robot, controlling movement, starting inspections, changing sensor calibrations, or performing required actions based on the received data. This interactive flow of data and commands sets a closed-loop system that makes continuous monitoring, leak detection, and timely intervention in pipeline maintenance and safety possible.

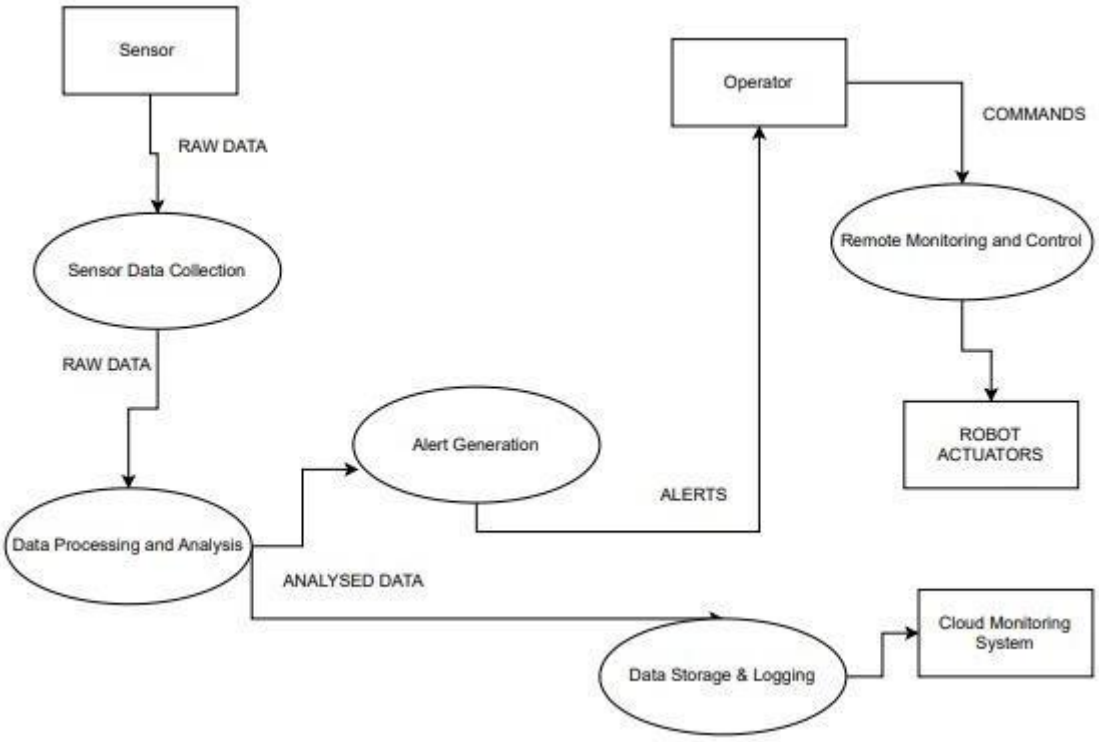


Fig 4.2.2: DFD Level 1 depicting processes and entities in detail.

The diagram above shows a system of remote monitoring and control in which sensors collect raw data, which is compiled and processed and analyzed in order to derive meaningful information. From this analysis, the system issues alerts that are conveyed to a human operator. The operator sends commands via a remote monitoring and control interface, which are

converted into actions by actuators of the robot. Most importantly, the data that has been examined, and perhaps even the raw data, is retained and recorded for easy access and examination later. Furthermore, retained data can be sent to a cloud monitoring system, which enables remote access and possibly more advanced analytical capacity. This architecture provides for unceasing data collection and smart analysis, allowing human operators to remotely monitor the system and apply control when required, backed up by thorough data logging and possible cloud-based augmentation.

4.3 TimeLine Chart

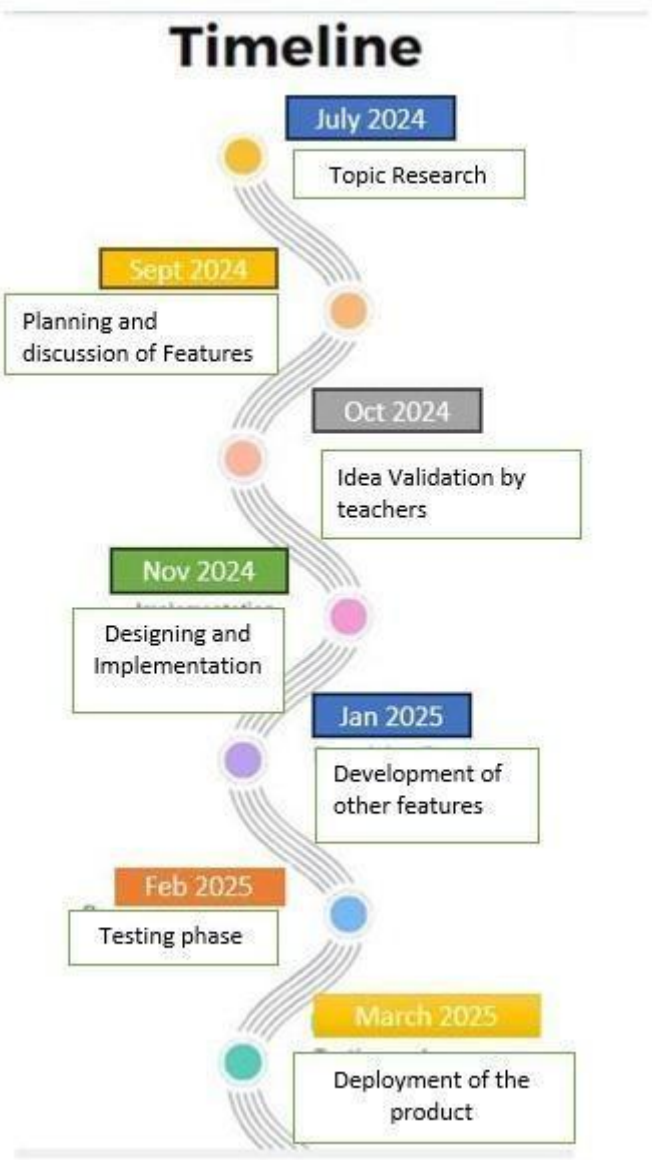


Fig 4.3.1: Depicting timeline of the project completion

5. DESIGN

5.1 Architectural Design

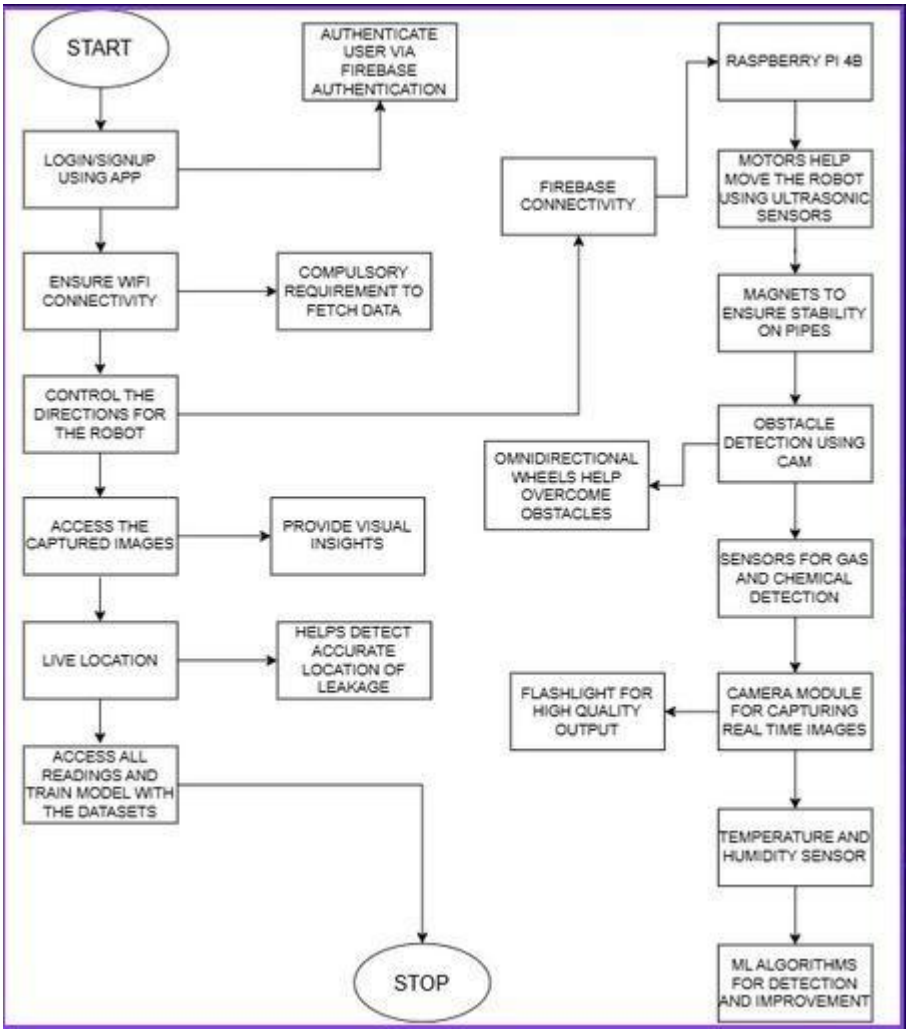


Fig 5.1.1: Depicting system architecture of the project.

The workflow of the robotic pipe inspection and leak detection system is depicted in the flowchart. The user logs in or registers using a mobile application, which then validates the user using Firebase. A Wi-Fi connection is of utmost importance for the application to retrieve data and function smoothly. At the same time, the robot unit gets connected to Firebase, probably for real-time data transfer and control. The movement of the robot is then commanded using the app by users, while the robot runs using a Raspberry Pi 4B, guided using motors and ultrasonic sensors. Magnets stabilize the robot during pipe traversal. Obstructions are detected using a camera, and omnidirectional wheels help maneuver around obstacles. A camera takes

photographs, which gives visual information to the user, and the

user is able to view the photographs through the application. For detection of leaks, the robot has sensors to detect chemicals and gases, and there is a flashlight to acquire high-quality visual data under different light conditions. The robot's live current location is also tracked, which, along with sensor inputs, helps to identify the correct site of any leakage. In addition to this, temperature and humidity sensors are integrated to collect environmental information. All the reading taken and data sets are made available to the user and are usable for training machine learning algorithms to increase the detection accuracy and overall efficiency of the system. With the conclusion of the inspection or user input, the procedure ends

5.2 User Interface Design

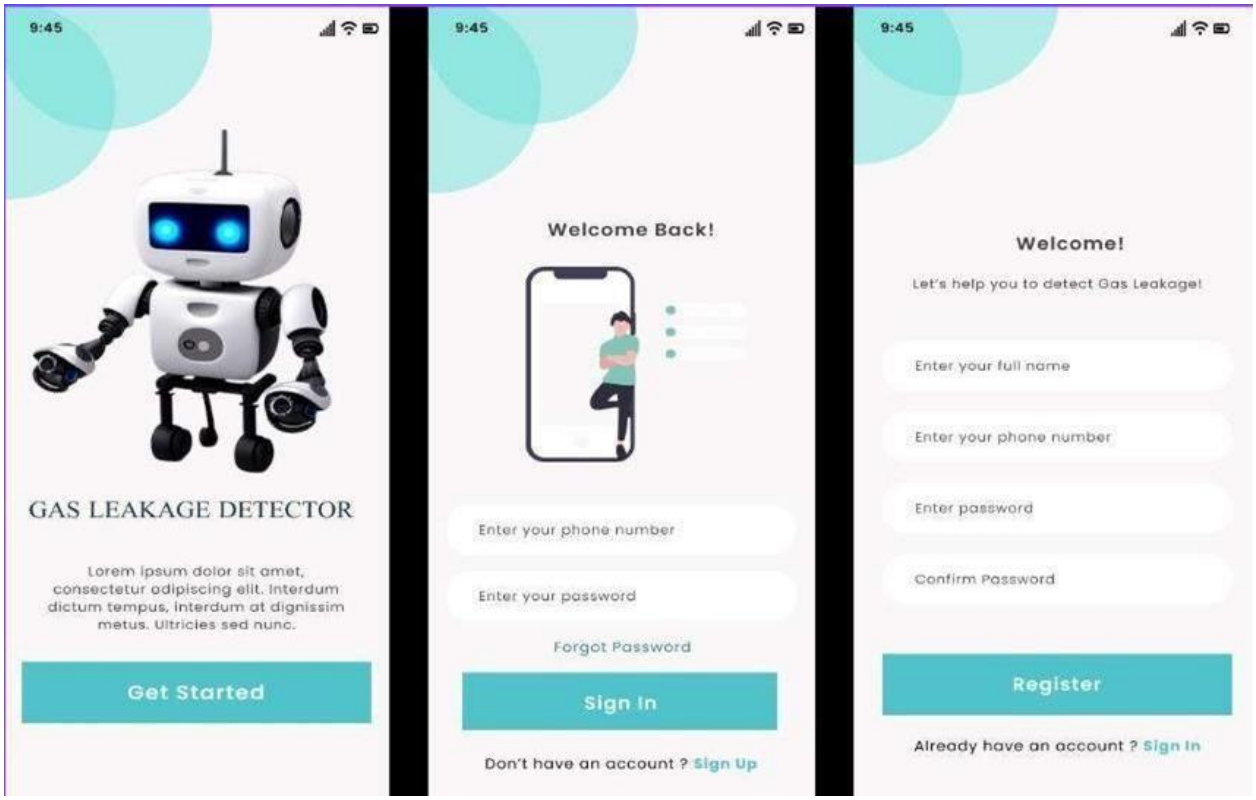


Fig 5.2.1: Authentication pages of the flutter app

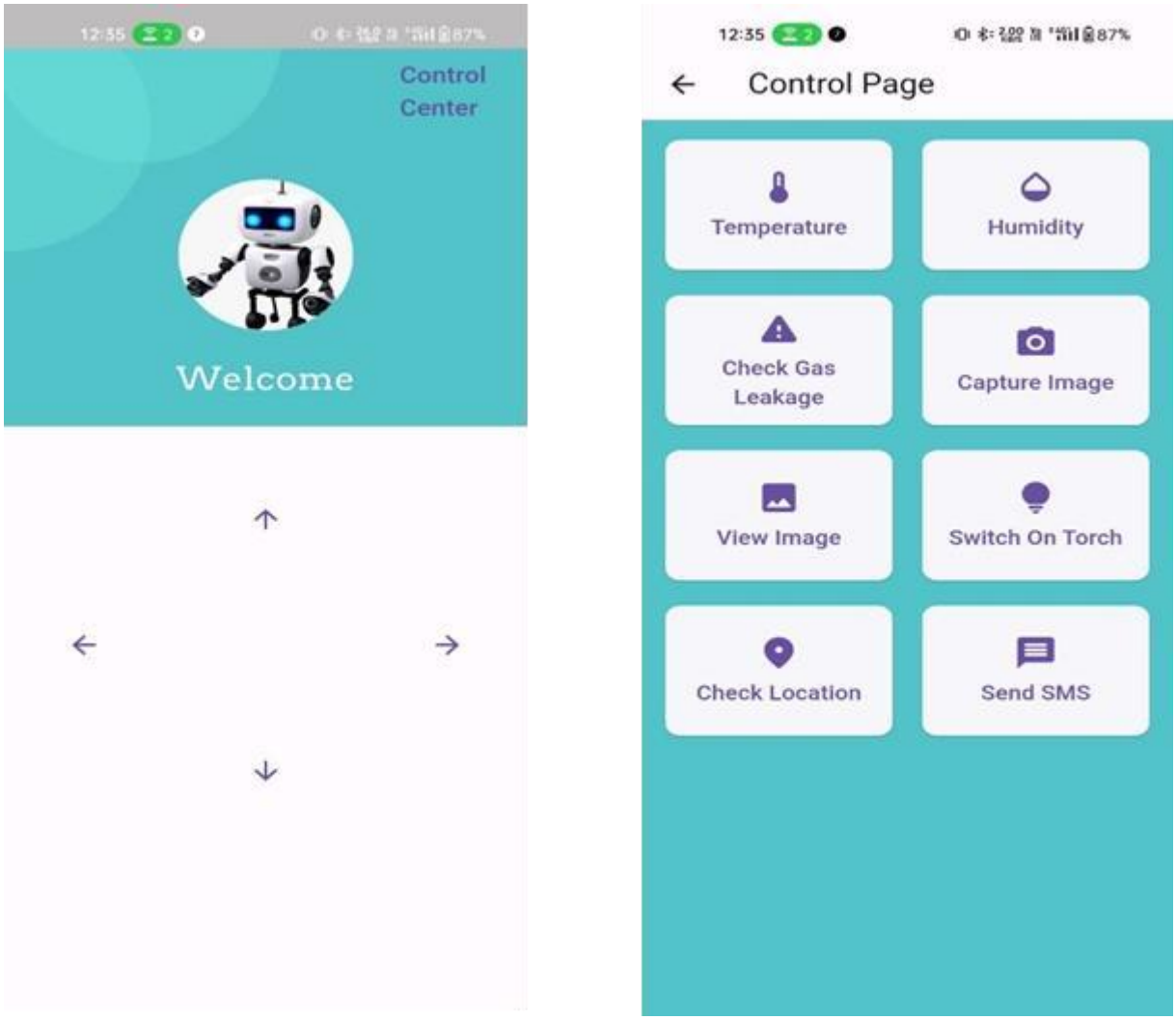


Fig 5.2.2: Control page of the flutter application.

5.3 Autocad Designs

Figure 5.3.1 is the perspective view of the intended IoT-enabled robot for detecting gas pipe leaks within industrial and residential environments. It is installed with different sensors, such as a temperature-humidity sensor and a gas sensor, which run in the background to observe the environment and identify possible toxic leaks. Processing of the sensor inputs is real-time by using an onboard processor and is communicated to a remote data storage system. This information is then presented at a centralized monitoring and control center to allow operators to take instantaneous decisions.

An ultrasonic sensor located at the front of the robot is instrumental in detecting and evading obstacles. It gauges the distance to surrounding objects using ultrasonic waves, assisting the robot in safely traveling along pipe networks with complex geometry. For stability and adherence during climbs up pipes, particularly cylindrical or metallic ones, the robot uses high-strength magnets attached to its chassis.

For inspection using the camera in low light or at night, the robot has incorporated a high-powered torch complemented by an imaging component, which records real-time images of the environment. For mobility, four wheels have been incorporated, while castor wheels with 360-degree rotation are utilized for easy turning and navigation in narrow areas. A motor driver circuit controls the rotation of the wheels using a set of motors, following commands from the processing unit depending on sensor input from the environment and navigation algorithms.

Figure 5.3.2 shows the top view of the robot with the relative positioning of the key components such as the processing unit, the gas sensor, temperature-humidity sensor, ultrasonic sensor, imaging device, torch, motor drive, as well as the set of drive wheels. This view is useful to conceptualize the efficient and tight configuration aimed at reducing space while maximizing sensor coverage and efficiency of movement.

Figure 5.3.3 is the bottom view of the robot, looking at the placement of the magnetic gripper mechanism and the castor wheels. This configuration allows the robot to stick firmly to metallic pipes and move smoothly even when going from horizontal to vertical orientations.

Figure 5.3.4 presents the front view of the robot with the front-facing elements like the processing unit, the gas sensor, the ultrasonic sensor, imaging device, and the temperature-humidity sensor. Visible and seen besides the front wheels is one of the castor wheels, all of which are involved in the balanced movement and overcoming obstacles.

Figure 5.3.5 illustrates the back of the robot, highlighting the placement of primary back components such as the processing unit, gas sensor, ultrasonic sensor, imaging device, temperature-humidity sensor, magnets, motors, rear wheels, and a castor wheel. This drawing is meant to illustrate the symmetrical configuration of the components so the robot is properly balanced and has balanced weight distribution.

Figure 5.5.6 is the left side view of the robot, which better showcases the lateral positioning of the processing unit, gas sensor, ultrasonic sensor, imaging device, motor driver, magnets, motors, wheels, and castor wheel. It shows the robot's low-profile, high-utility configuration to enable it to navigate narrow corridors and intricate pipe network.

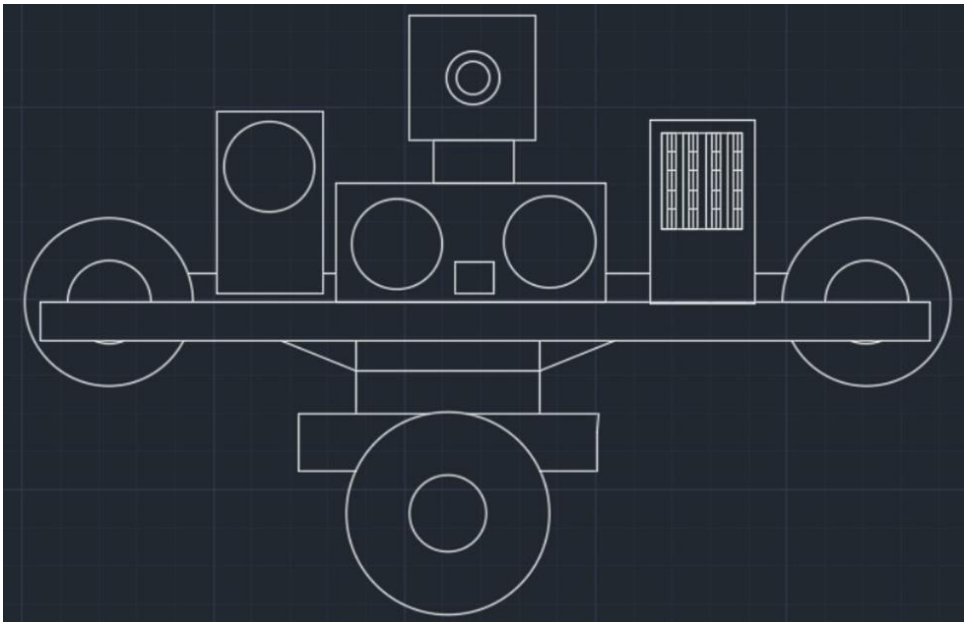


Fig 5.3.1: Front View of Robot

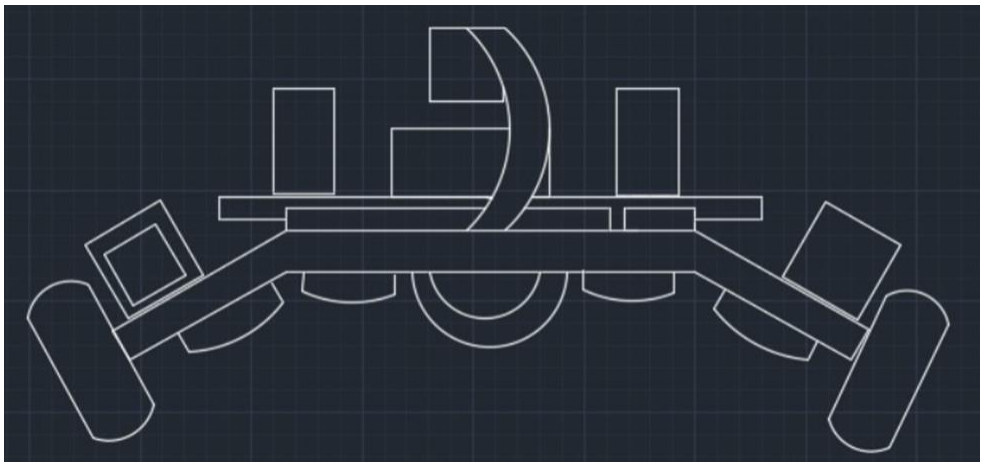


Fig 5.3.2: Rear View of Robot

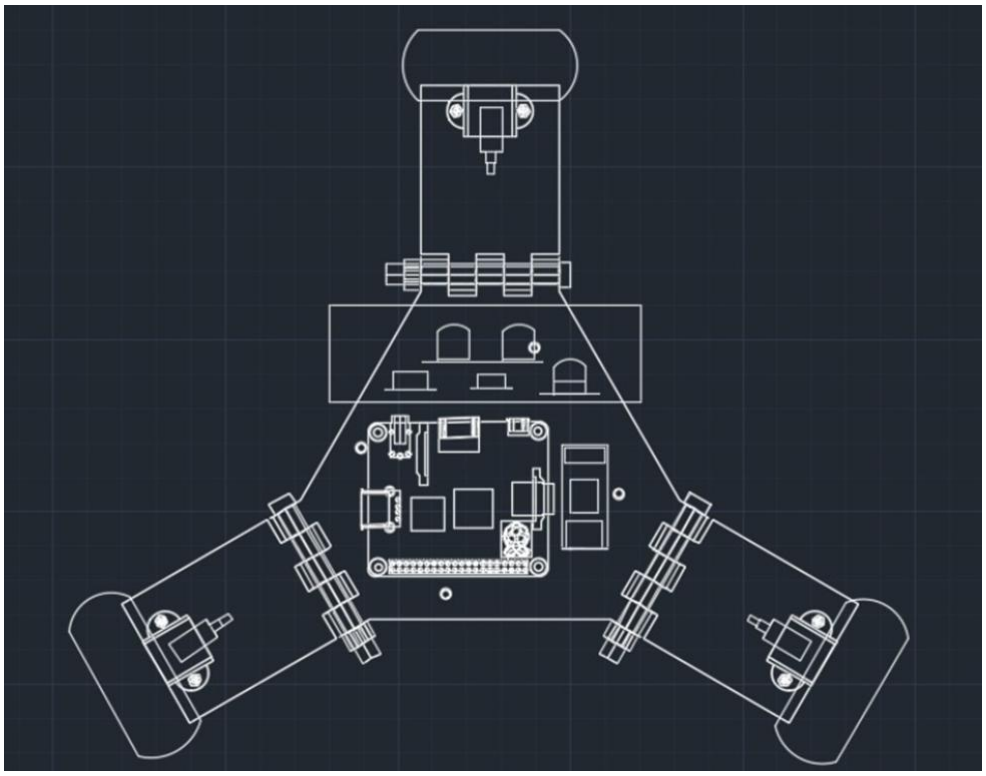


Fig 5.3.3: Top View of Robot

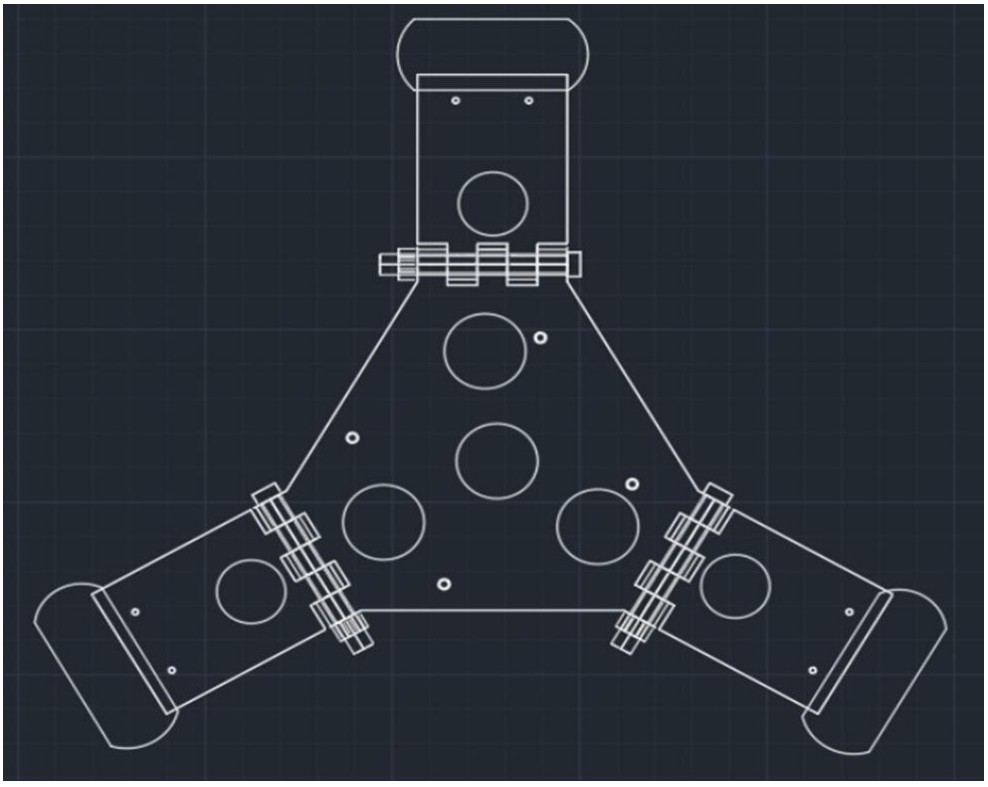


Fig 5.3.4: Bottom View of Robot

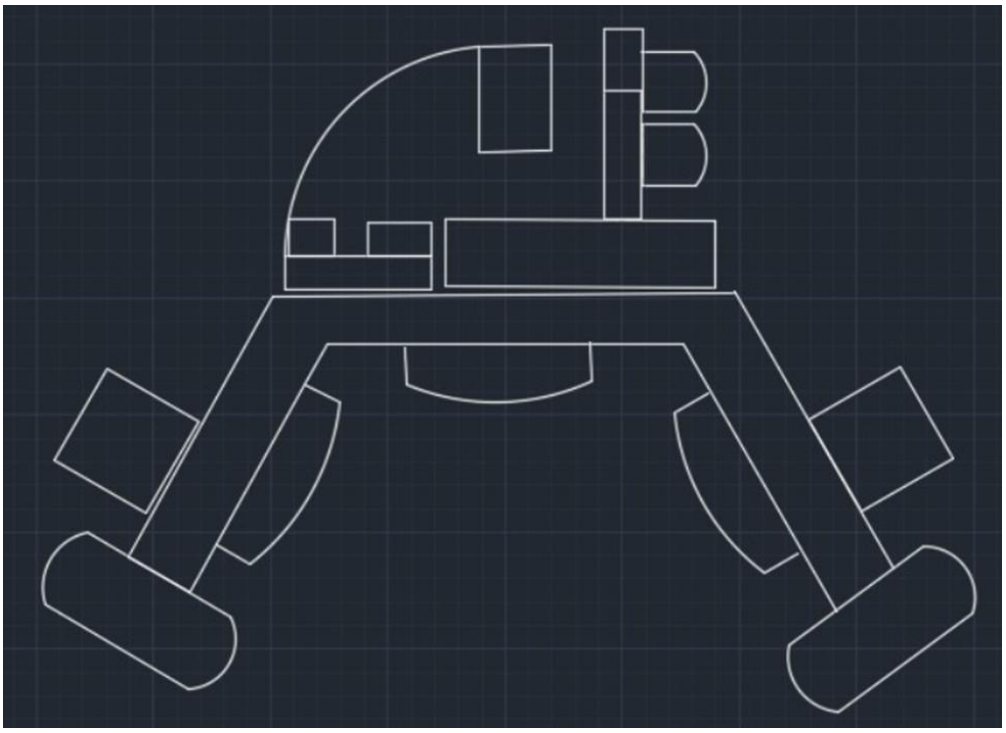


Fig 5.3.5: Left View of Robot

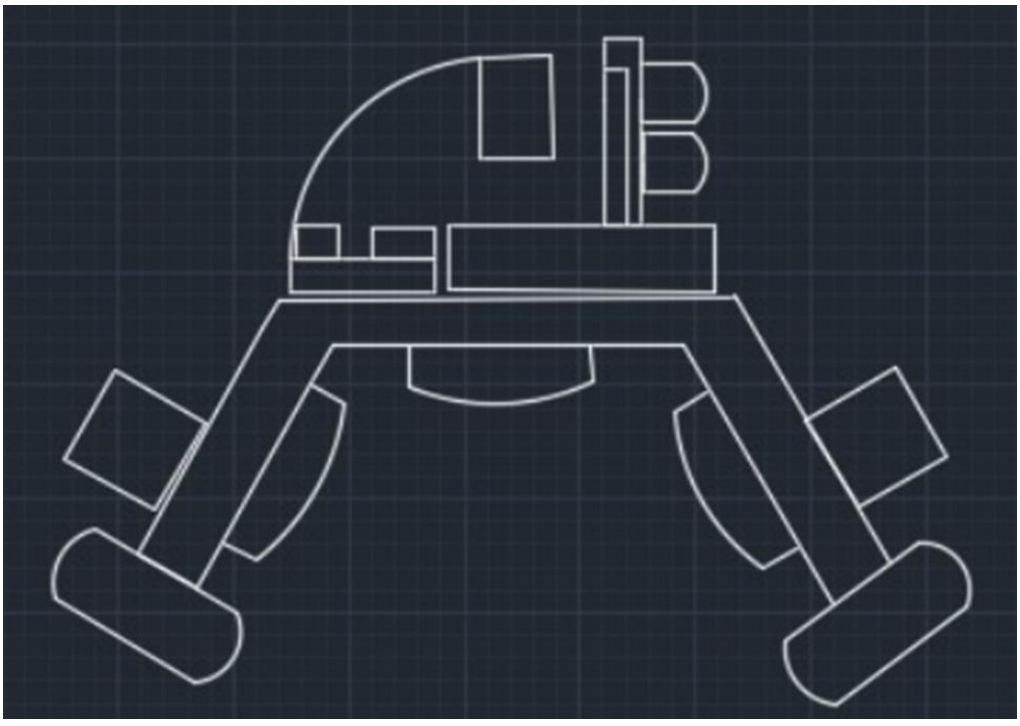


Fig 5.3.6: Right View of Robot

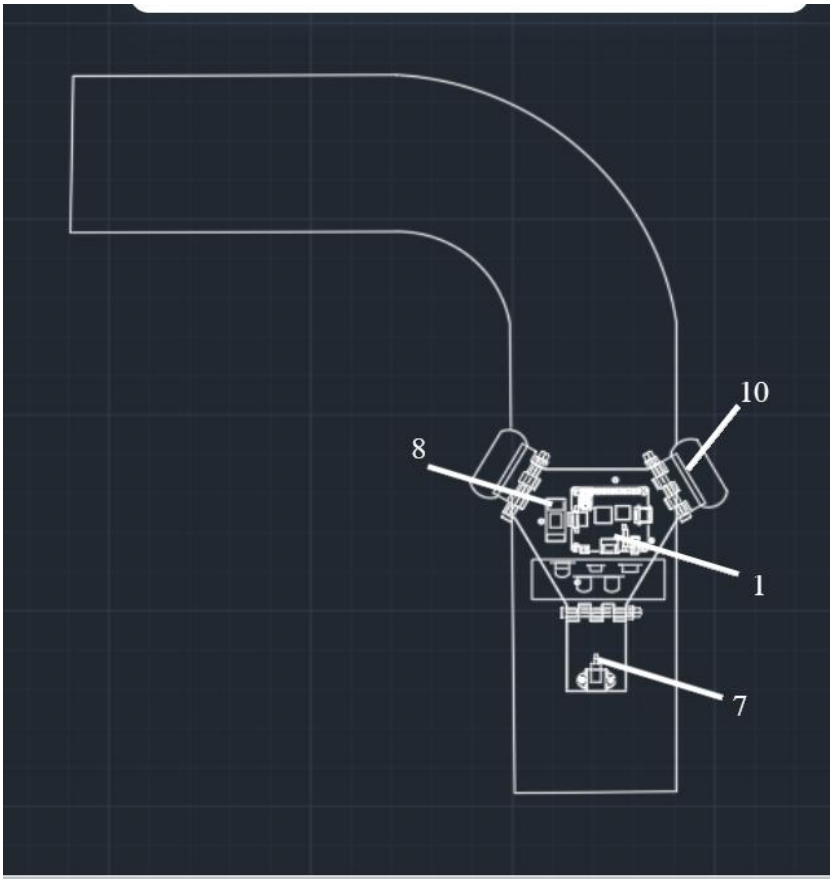


Fig 5.3.7 : Perspective View of Robot

6. IMPLEMENTATION

6.1 Algorithms / Methods Used

The IoT-powered robotic device covered by the code leverages sensor-informed decision-making, real-time cloud communication, and automated actuation to provide efficient detection of gas leaks and industrial monitoring. At its core, the system employs threshold-based algorithms to parse environment sensor data—specifically, from an MQ5 sensor for detecting gas, a DHT11 sensor for temperature and humidity detection, and an ultrasonic sensor for detecting obstacles. All the sensors run under predefined threshold conditions to enable the robot to identify anomalies like gas leaks, overheating, and nearness to obstacles. If such anomalies are detected, the system is pre-programmed to react accordingly, such as ordering halting motion or emergency alerts. Navigation of the robot is done through FSM-style logic where input flags (such as forward, left, right, backward) received from a Firebase Firestore document are translated into corresponding GPIO pin configurations. These govern the rotation of the motor, enabling the robot to drive autonomously in any direction or stop safely as needed. In addition, the ultrasonic sensor enables simple obstacle avoidance to evade collisions by causing the robot to stop if there is any object within the critical distance.

In addition, the robot incorporates smart multimedia and communication capabilities. The image acquisition system utilizes a PiCamera module driven by a cloud flag (click_image) to take periodic snapshots of the inspection zone, helpful during low light conditions due to the LED flashlight powered by GPIO. Those images are stored in Firebase Storage in real time, and their URLs are appended to the Firestore document to allow for visualized inspection at a distance. It even incorporates Twilio's SMS API to alert in emergency conditions where the concentration of the gas crosses a voltage level threshold (e.g., >1.5V), under control of a one-time switch (sms_enabled) to prevent repeated messages. IP geolocation using the geocoder library is utilized by the robot to periodically update its latitude and longitude in the cloud, facilitating operators to monitor its trajectory and inspection route. All data is centralized within a structured dictionary (data) and written to Firestore, making it a single interface for control as well as monitoring. This fusion approach—using event-driven programming, sensor fusion, and cloud synchronization—allows scalable deployment for real-time, remote, and intelligent industrial pipe monitoring systems.

The one-second iteration loop, which runs at one-second frequencies, is the time-driven polling

mechanism coordinating all units in harmony. To minimize latency and save resources, operations such as image acquisition and current locations are intervalled by using iteration counters. This tightly integrated yet componentized control setup helps the robot to operate autonomously in hostile environments, and as such, is suitable for applications in detecting gas leaks, checking industrial machinery, and disaster recovery surveys. With its integration of cloud computing, embedded systems, and communication protocols, the robot presents one efficient embodiment of edge-to-cloud Internet of Things infrastructure for intelligent industry 4.0 applications

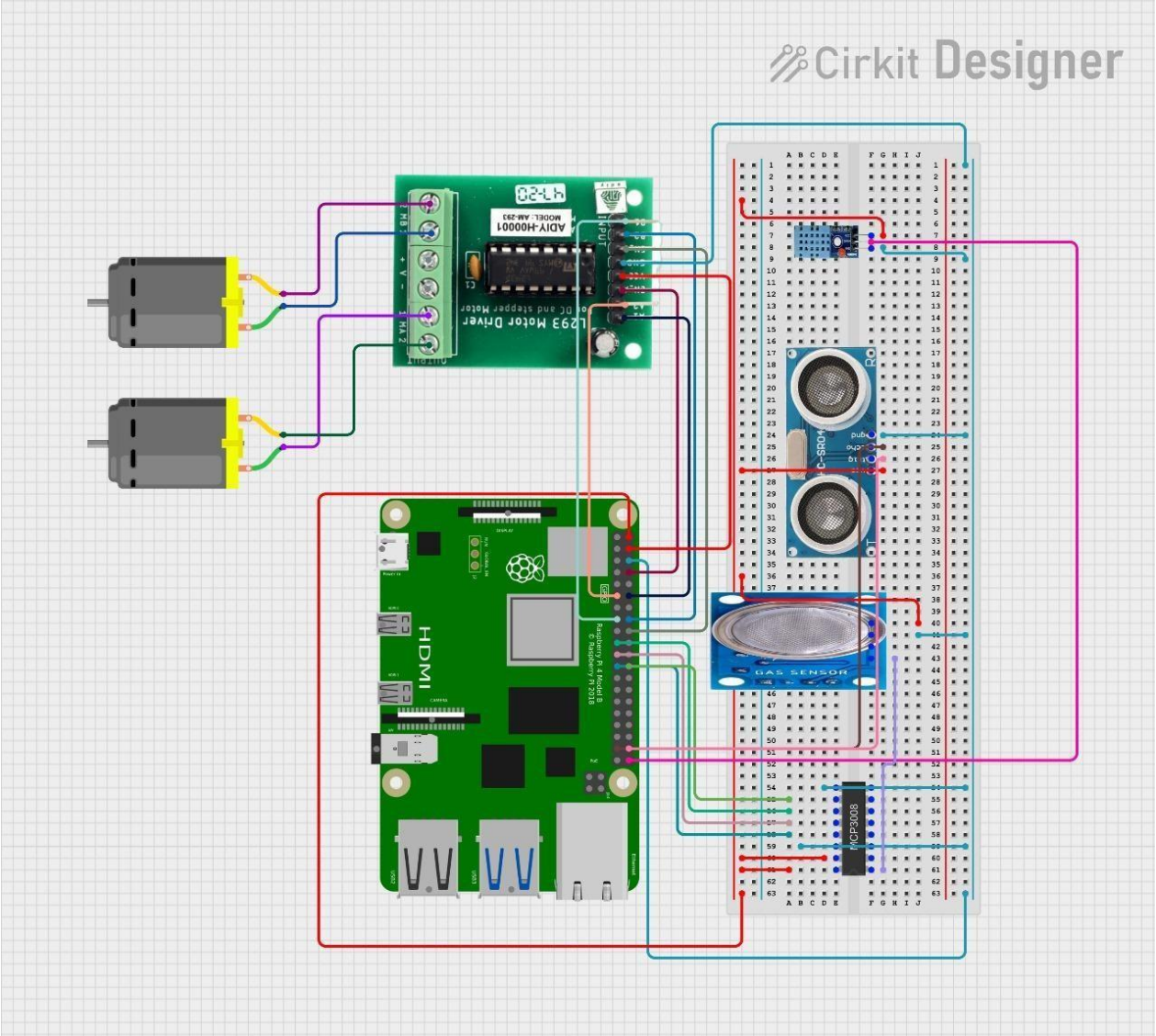


Fig 6.1.1: Circuit Diagram depicting connections of the robot

The circuit illustration shows a basic robot circuit constructed on a breadboard around a microcontroller unit of Raspberry Pi controlling the behavior of the system. An L293D motor driver is powered and driven by control signals from the Raspberry Pi to drive two DC motors

for movement. For sensing the environment, an ultrasonic sensor is interfaced to the Raspberry Pi to detect distances to help in obstacle detection. A gas sensor is incorporated to detect certain gases to add another form of input to the Raspberry Pi from the environment, probably through analog values. A voltage regulator stabilizes power for the sensors. There is power and ground distribution along the breadboard connecting all the components, and there are different colored wires bringing control and data signals to and from the Raspberry Pi to the motor driver, the ultrasonic sensor, and the gas sensor to form a basic yet functional experiment setup for robotics concerning motion control and sensing the environment.

6.2 Working of the project

1. Main Functionality code

```
import firebase_admin
from firebase_admin import credentials, storage from firebase_admin import firestore
import dht11

import RPi.GPIO as GPIO from time import sleep
from gpiozero import DistanceSensor import busio
import digitalio import board
import adafruit_mcp3xxx.mcp3008 as MCP

from adafruit_mcp3xxx.analog_in import AnalogIn import geocoder
from picamera import PiCamera import datetime
from twilio.rest import Client



############## Setup Starts ############## # GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)


GPIO.setwarnings(False)
```

```
# Set GPIO Pins PWMA = 4 #7
AIN1 = 18 #12

AIN2 = 17 #11

PWMB = 24 #18

BIN1 = 22 #15

BIN2 = 23 #16

LED_PIN = 27 GPIO.setup(PWMA, GPIO.OUT) GPIO.setup(AIN1, GPIO.OUT)
GPIO.setup(AIN2, GPIO.OUT) GPIO.setup(PWMB, GPIO.OUT) GPIO.setup(BIN1, GPIO.OUT)
GPIO.setup(BIN2, GPIO.OUT) GPIO.setup(LED_PIN, GPIO.OUT)


# Setup the DHT11 Sensor DHT11PIN = 21
instance = dht11.DHT11(pin=DHT11PIN) # temperature & humidity


# Setup the Distance Sensor

sensor = DistanceSensor(26,20) # Distance sensor


# Setup the MQ5 Sensor

spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI) # create the spi bus
cs = digitalio.DigitalInOut(board.D5) # create the cs (chip select) mcp = MCP.MCP3008(spi, cs) #
create the mcp object
chan = AnalogIn(mcp, MCP.P0) # create an analog input channel on pin 0

# Connect to the Firebase


bucket_name = "gas-leakage-f87ff.appspot.com"

cred = credentials.Certificate('serviceAccountKey.json') firebase_admin.initialize_app(cred,
{'storageBucket': bucket_name})


# Create a Cloud Firestore client database = firestore.client()
new_collection = database.collection("users") document = new_collection.document('pi')
```

```
# Twilio Setup account_sid = '' auth_token = ''
client = Client(account_sid, auth_token) iterations = 0
############## Setup Complete ##############


# Turn off all the motors def turnOffMotors():
GPIO.output(AIN1, GPIO.LOW) GPIO.output(AIN2, GPIO.LOW) GPIO.output(PWMA,
GPIO.LOW) GPIO.output(BIN2, GPIO.LOW) GPIO.output(BIN1, GPIO.LOW)
GPIO.output(PWMB, GPIO.LOW)
print("Motors are off") return


# Forward motion def forward():
GPIO.output(AIN1, GPIO.LOW) GPIO.output(AIN2, GPIO.HIGH) GPIO.output(PWMA,
GPIO.HIGH)


GPIO.output(BIN2, GPIO.LOW) GPIO.output(BIN1, GPIO.HIGH) GPIO.output(PWMB,
GPIO.HIGH)
print("Moving forward") return


# Left motion def left():
GPIO.output(AIN1, GPIO.LOW) GPIO.output(AIN2, GPIO.HIGH) GPIO.output(PWMA,
GPIO.HIGH) GPIO.output(BIN2, GPIO.HIGH) GPIO.output(BIN1, GPIO.LOW)
GPIO.output(PWMB, GPIO.HIGH)
print("Moving left") return


# Right motion def right():
GPIO.output(AIN1, GPIO.HIGH) GPIO.output(AIN2, GPIO.LOW) GPIO.output(PWMA,
GPIO.HIGH) GPIO.output(BIN2, GPIO.LOW) GPIO.output(BIN1, GPIO.HIGH)
GPIO.output(PWMB, GPIO.HIGH)
print("Moving right") return


# Backward motion def backward():
GPIO.output(AIN1, GPIO.HIGH) GPIO.output(AIN2, GPIO.LOW) GPIO.output(PWMA,
```

```
GPIO.HIGH) GPIO.output(BIN2, GPIO.HIGH)

GPIO.output(BIN1, GPIO.LOW) GPIO.output(PWMB, GPIO.HIGH)
print("Moving backward") return

# Stop motion def stop():
GPIO.output(AIN1, GPIO.LOW) GPIO.output(AIN2, GPIO.LOW) GPIO.output(PWMA,
GPIO.LOW) GPIO.output(BIN2, GPIO.LOW) GPIO.output(BIN1, GPIO.LOW)
GPIO.output(PWMB, GPIO.LOW)
print("Stop") return



# Read the temperature and humidity def readHumidityTemperature():
result = instance.read() if result.is_valid():
      print("Temp: %d C" % result.temperature +' '+"Humid: %d %%" % result.humidity)
# setHumidityTemperature(result) return result

# Read the distance def readDistance():
distance = sensor.distance
print("Distance: %.1f cm" % (distance * 100)) if distance < 0.06:
   stop() return distance

# Read the Gas Quality def readGasQuality():
gas = str(chan.voltage)
print("ADC Voltage: " + gas + "V") return gas

# Send SMS def sendSMS():
message = client.messages.create( from_='+12184026613', to='+919820077642',
body= "GAS Leakage Detected",
)
```

```
print(message.sid)

print("Sending SMS using Twilio")



# Click Image def clickImage():
print("Clicking Image") # Initialize camera camera = PiCamera()
camera.resolution = (1024, 768)



# Generate image path

image_path = f'images/image_{datetime.datetime.now().isoformat()}.jpg'



# Capture image camera.start_preview()
sleep(2) # Camera warm-up time camera.capture(image_path) camera.stop_preview()
camera.close()



# Upload to Firebase Storage bucket = storage.bucket()
blob = bucket.blob(image_path) blob.upload_from_filename(image_path) print(f'Image uploaded
to {image_path}')



# # Generate a URL to access the image

   #                    url     =     blob.generate_signed_url(datetime.timedelta(seconds=300),
method='GET') # URL valid for 5 minutes
# print(f'Access URL: {url}\n')



public_url                                                                                    =

f'https://firebasestorage.googleapis.com/v0/b/{bucket_name}/o/{image_path.re place("/",
"%2F")}?alt=media'



return public_url
```

```
    # Get all values from pi def getAllData():
    print("getAlldata Called")
    data = document.get().to_dict() print("Completed | getAllData") print(data)
    return data


def readAndSetAllData(data): try:
    dht = readHumidityTemperature()
        data["hum"], data["temp"] = str(dht.humidity), str(dht.temperature) except Exception as e:
    print(f"Error reading humidity and temperature: {e}")


    try:
    data["dist"] = str(readDistance())


    except Exception as e:
    print(f"Error reading distance: {e}")


    try:
        data["leakage"] = str(readGasQuality()) except Exception as e:
    print(f"Error reading gas quality: {e}")



if iterations % 5 == 0: try:
    geoLocation = geocoder.ip('me') data["lat"] = str(geoLocation.latlng[0]) data["long"] =
    str(geoLocation.latlng[1])
    except Exception as e:
    print(f"Error obtaining geolocation: {e}")


    try:
```

```
        document.set(data) except Exception as e:
        print(f"Error setting document data: {e}") return data
    # Main Loop while True:
    print("\nMain Loop Started") data = getAllData()
if data['forward'] == True: forward()
elif data['left'] == True: left()
    elif data['right'] == True:


    right()
elif data['backward'] == True: backward()
    else:
    stop()



if data["torch"] == True: GPIO.output(LED_PIN, GPIO.HIGH)
    else:
    GPIO.output(LED_PIN, GPIO.LOW)



    if iterations % 4 == 0 :
if data["sms_enabled"] == True and float(data["leakage"]) > 1.5: try:
    sendSMS() data["sms_enabled"] = False
except Exception as e: print("Error sending SMS: ", e)


if data["click_image"] == True: try:
    imageUrl = clickImage() data["view_images"].append(imageUrl) data["click_image"] = False
    except Exception as e:
            print("Error clicking image: ", e) print(readAndSetAllData(data))



    iterations += 1
    print("Main Loop Ended:",iterations, end="\n") sleep(1)
```

2. firebase_options.dart

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars, avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions; import 'package:flutter/foundation.dart'
show defaultTargetPlatform, kIsWeb, TargetPlatform;


/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///  options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
static FirebaseOptions get currentPlatform { if (kIsWeb) {
 return web;
 }

switch (defaultTargetPlatform) { case TargetPlatform.android:
 return android;
 case TargetPlatform.iOS:
 return ios;
 case TargetPlatform.macOS:
```

```
return macos;
case TargetPlatform.windows:
return windows;
case TargetPlatform.linux:
throw UnsupportedError(
'DefaultFirebaseOptions have not been configured for linux - ' 'you can reconfigure this by running the FlutterFire CLI again.',
);
default:
throw UnsupportedError(
'DefaultFirebaseOptions are not supported for this platform.',
);
}
}

static const FirebaseOptions web = FirebaseOptions(
apiKey: 'AIzaSyDF1HvOKGFcSYhIy1VvzC0X2Z3DBiS8KLk', appId: '1:680846974193:web:8ea586bbcd9fd9dfb65d50', messagingSenderId: '680846974193', projectId: 'gas-leakage-f87ff',
authDomain: 'gas-leakage-f87ff.firebaseapp.com', storageBucket: 'gas-leakage-f87ff.appspot.com',
);

static const FirebaseOptions android = FirebaseOptions( apiKey: 'AIzaSyAN0ud17qCu7pMA4QfiMo_p385uzXwPcyc', appId: '1:680846974193:android:365ee7563b3344d1b65d50', messagingSenderId: '680846974193',

projectId: 'gas-leakage-f87ff',
storageBucket: 'gas-leakage-f87ff.appspot.com',
```

```
);


static const FirebaseOptions ios = FirebaseOptions(
apiKey: 'AIzaSyC_luE5xqNFa_KKBVUfXkUB2wRjmuHmkVs', appId:
'1:680846974193:ios:876001b84d1e21acb65d50',
messagingSenderId: '680846974193', projectId: 'gas-leakage-f87ff',
storageBucket: 'gas-leakage-f87ff.appspot.com', iosBundleId: 'com.example.gasLeakageDetector',
);


static const FirebaseOptions macos = FirebaseOptions(
apiKey: 'AIzaSyC_luE5xqNFa_KKBVUfXkUB2wRjmuHmkVs', appId:
'1:680846974193:ios:876001b84d1e21acb65d50',
messagingSenderId: '680846974193', projectId: 'gas-leakage-f87ff',
storageBucket: 'gas-leakage-f87ff.appspot.com', iosBundleId: 'com.example.gasLeakageDetector',
);


static const FirebaseOptions windows = FirebaseOptions(
apiKey: 'AIzaSyDF1HvOKGFcSYhIy1VvzC0X2Z3DBiS8KLk', appId:
'1:680846974193:web:fe0a0dac452772f2b65d50', messagingSenderId: '680846974193',
projectId: 'gas-leakage-f87ff',
authDomain: 'gas-leakage-f87ff.firebaseapp.com', storageBucket: 'gas-leakage-f87ff.appspot.com',
);


}
```

3. main.dart

```
import 'package:firebase_core/firebase_core.dart'; import 'package:flutter/material.dart';
import 'package:gas_leakage_detector/Intro.dart'; import
```

```
'package:gas_leakage_detector/Login.dart';
import 'package:gas_leakage_detector/firebase_options.dart';


void main() async { WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform); runApp(const IPD());
}


class IPD extends StatelessWidget { const IPD({super.key});


@override
Widget build(BuildContext context) { return const MaterialApp( debugShowCheckedModeBanner:
false, home: Scaffold(
backgroundColor: Colors.white, body: IntroductionPage(),
),
);
}
}
```

4. trump

```
import firebase_admin
from firebase_admin import credentials from firebase_admin import firestore import dht11
import RPi.GPIO as GPIO


from time import sleep
from gpiozero import DistanceSensor
```

```
############# Setup Stats #############

# GPIO Mode (BOARD / BCM) GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

GPIO.setup(7, GPIO.OUT) # Connected to PWMA GPIO.setup(11, GPIO.OUT) # Connected to
AIN2 GPIO.setup(12, GPIO.OUT) # Connected to AIN1

GPIO.setup(15, GPIO.OUT) # Connected to BIN1 GPIO.setup(16, GPIO.OUT) # Connected to
BIN2 GPIO.setup(18, GPIO.OUT) # Connected to PWMB

# # Setup the DHT11 Sensor

# instance = dht11.DHT11(pin=21) # temperature & humidity

# # Setup the Distance Sensor

# sensor = DistanceSensor(26,20) # Distance sensor

# Connect to the Firebase

cred = credentials.Certificate('serviceAccountKey.json') firebase_admin.initialize_app(cred)
# Create a Cloud Firestore client database = firestore.client()
collection = database.collection("wheels") print(collection)
############# Setup Complete #############

# Get the controller def getController():
try:
    controller = collection.document('controller').get().to_dict() except Exception as e:
    print(f"An error occurred: {e}") print(controller)
```

```
    return controller


    # Set the controller
def setController(left, forward, behind, right): collection.document('controller').set({'left': left,
    'forward': forward, 'behind':
behind, 'right': right}) return


    # Turn off all the motors def turnOffMotors():
    GPIO.output(12, GPIO.LOW) GPIO.output(11, GPIO.LOW) GPIO.output(7, GPIO.LOW)
    GPIO.output(16, GPIO.LOW) GPIO.output(15, GPIO.LOW) GPIO.output(18, GPIO.LOW)
    print("Motors are off") return


    # Forward motion def forward():
    GPIO.output(12, GPIO.LOW) GPIO.output(11, GPIO.HIGH) GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.LOW) GPIO.output(15, GPIO.HIGH)


    GPIO.output(18, GPIO.HIGH)
    print("Moving forward") return


    # Left motion def left():
    GPIO.output(12, GPIO.LOW) GPIO.output(11, GPIO.HIGH) GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH) GPIO.output(15, GPIO.LOW) GPIO.output(18, GPIO.HIGH)
    print("Moving left") return


    # Right motion def right():
    GPIO.output(12, GPIO.HIGH) GPIO.output(11, GPIO.LOW) GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.LOW)


    GPIO.output(15, GPIO.HIGH) GPIO.output(18, GPIO.HIGH)
    print("Moving right") return


    # Backward motion def backward():
```

```
GPIO.output(12, GPIO.HIGH) GPIO.output(11, GPIO.LOW) GPIO.output(7, GPIO.HIGH)
GPIO.output(16, GPIO.HIGH) GPIO.output(15, GPIO.LOW) GPIO.output(18, GPIO.HIGH)



print("Moving backward") return


# Stop motion def stop():
GPIO.output(12, GPIO.LOW) GPIO.output(11, GPIO.LOW) GPIO.output(7, GPIO.LOW)
GPIO.output(16, GPIO.LOW) GPIO.output(15, GPIO.LOW) GPIO.output(18, GPIO.LOW)
print("Stop") return


# # Read the temperature # def readTemperature():
#                       result = instance.read() # if result.is_valid():

#                           print("Temperature: %d C" % result.temperature) #   return
result.temperature
#                       else:
#                           print("Error: %d" % result.error_code) #        return result.error_code


# # Set the temperature
# def setTemperature(temp):
#                       collection.document('temperature').set({'temp': temp}) #          return


# # Read the humidity # def readHumidity():
#                       result = instance.read() # if result.is_valid():



#                           print("Humidity: %d %%" % result.humidity) #        return result.humidity
#                       else:
#                           print("Error: %d" % result.error_code) #        return result.error_code


# # Set the humidity
```

```
# def setHumidity(hum):
#                     collection.document('humidity').set({'hum': hum}) #     return


# # Read the distance # def readDistance():



#                     distance = sensor.distance
#                     print("Distance: %.1f m" % distance) #  return distance


# # Set the distance
# def setDistance(dist):
#                     collection.document('distance').set({'dist': dist}) #         return


# # Read the Gas Quality # def readGasQuality():
#                     gas = 0 # Add your logic here #  print(gas)
#                     return gas



# # Set the Gas Quality
# def setGasQuality(gas):
#                     collection.document('gas').set({'gas': gas}) #      return




iterations = 0 # Main Loop while True:
print("Main Loop Started") controller = getController()
# if controller['forward'] == True: #   forward()
# elif controller['left'] == True: #       left()
# elif controller['right'] == True: #      right()
# elif controller['behind'] == True: #  backward()
# else:
```

```
#                       stop()


# setDistance(readDistance())

# setTemperature(readTemperature()) # setHumidity(readHumidity())
# setGasQuality(readGasQuality())


iterations += 1

print("Main Loop Ended:",iterations)
```

7. TESTING

7.1 Test cases

To confirm the performance of the robot in real and critical operational environments, an extensive set of test cases was formulated. Every test case was intended to check a particular functionality and confirm the reliability and safety of the system in a deployment environment.

1. Gas Detection Test: Controlled conditions with different levels of combustible gases were established with a safe gas supply. The readings of the MG6 sensor were checked to ascertain that the system appropriately identifies gas levels above the established safety limit. On detection, it was confirmed that the alarm system—comprising LED lighting, buzzer, and SMS notification—was triggered. The test also verified the flag reset action once the gas level returned to normal.

2. Temperature and Humidity Response Test: The rapid temperature and humidity change was simulated using a heat source (e.g., hairdryer) and humidifier. Sensor response times were recorded and real-time values were verified against calibrated references to ensure reliability. Edge cases like sudden environmental changes were also tested to ensure sensor stability and regular Firebase updates.

3. Obstacle Detection Test: Different-sized, different-material physical objects were laid in front of the robot at various distances. The sensing ability of the ultrasonic sensor and the giving of a stop command was confirmed in detecting obstacles. The avoidance mechanism of the robot was checked not to run into any obstacle, even under a low lighting environment or a reflective surface.

4. Navigation Test: Commands were issued via the Firebase interface—forward, backward, left, and right—to test directional response of the robot's motors. Tests under sloping surfaces and thin paths were conducted to measure motor torque, grip, and control accuracy. Log data was examined to ensure that Firebase received and executed commands satisfactorily with near-zero latency.

5. Camera and Torch Function Test: The feature to capture an image was activated through a flag in Firebase across different light settings (dim, bright, and ambient normal lighting). The native torch was flipped to maximize visibility, and images captured were evaluated for their clarity, resolution, and orientation. The test ensured that the robot can properly record its environment when inspecting.

6. Cloud Data Sync Test: A real-time data stream from all the sensors, camera, and GPS module was synced to Firebase Firestore and Firebase Storage. Consistency of data was checked by comparing the synced data with live values. Other tests involved checking timestamps, ensuring that image URLs were stored properly, and checking if geolocation data was properly mapped.

7. SMS Alert Test: The gas detection level was manually driven above threshold to activate the SMS alert system through Twilio. The test confirmed the message payload, recipient phone number, and response time. Post-alert behavior was monitored to confirm the readiness of the system for future alerts by properly resetting the gas leak detection flag.

7.2 Type of Testing used

To guarantee that the robot performs accurately throughout its hardware-software environment, various levels of testing methodologies were utilized in development and integration stages. The different types of tests assured that the robot operates as specified under normal and abnormal circumstances.

7.2.1 Unit Testing: The individual modules and components were tested in isolation to ensure proper functionality. Separate testing for motor controls (directional movement, speed control), sensor reading (gas, temperature, humidity), camera interface, and Firebase operations (data read/write) was done. Mock data inputs were utilized to mimic sensor behavior and check component output.

7.2.2 Integration Testing: Once individual modules were verified, they were combined to test data flow and operational coherence among subsystems. For example, the reading of the gas sensor was linked with the SMS alert function and Firebase update function, and the entire process was tested holistically. It ensured end-to-end communication between hardware actions and cloud responses.

7.2.3 System Testing: The whole robotic system was subjected to realistic conditions simulating real pipeline inspection cases. Tests covered complete operational cycles including navigation, sensor monitoring, hazard detection, image taking, and remote command execution. The aim was to ensure that the system met all functional requirements with ease.

7.2.4 White-Box Testing: Internal logic like sensor thresholding algorithms, control flow between input and output modules, and decision trees for movement and alerts were reviewed. Code coverage was optimized to cover all branches, loops, and exception conditions.

7.2.5 Black-Box Testing: The system's response was tested against input conditions without examining internal code. For example, passing movement commands or exposing the robot to gas with no

knowledge of internal sensor calibration ensured the observable behavior met user expectations.

7.2.6 Real-Time Testing: Priority was given to testing time-critical features like live sensor updates to Firebase, camera activation delay, and SMS alert latency of transmission. Real-time feedback was essential for emergency response and remote monitoring reliability.

7.2.7 Smoke Testing: Performed after complete hardware-software assembly, this first test verified all major components (motors, sensors, camera, cloud communication) were working. Any major failures were identified and resolved immediately before further testing.

7.2.8 Regression Testing: Once code changes or new hardware modules were introduced, regression testing ensured features previously tested—like movement, detection, or cloud uploads—remained functional as intended. Automated and manual test scripts were used to ensure system integrity post-changes.

8. RESULTS AND DISCUSSIONS

Fig 8.1: Depicting temperature and humidity results obtained.

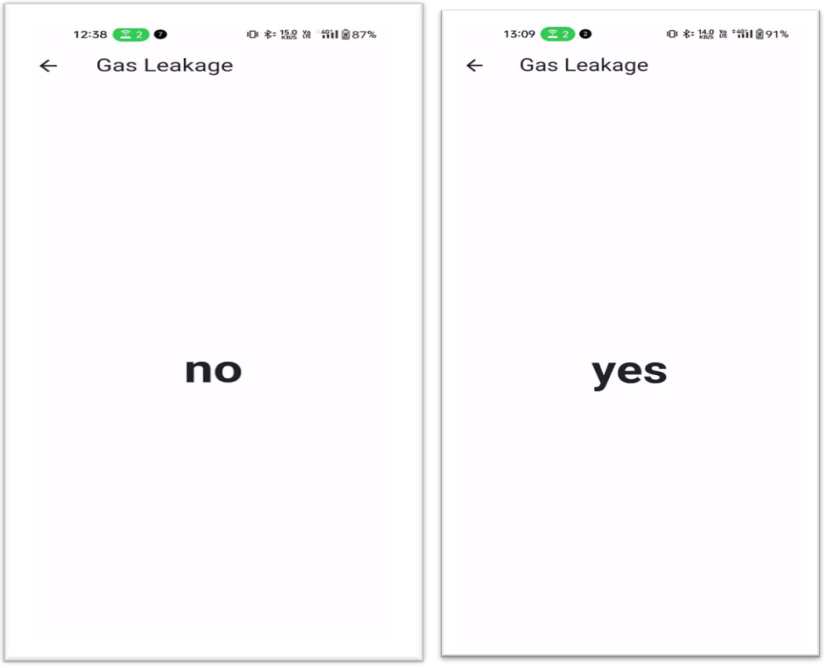


Fig 8.2: Depicting whether gas leakage has been detected.

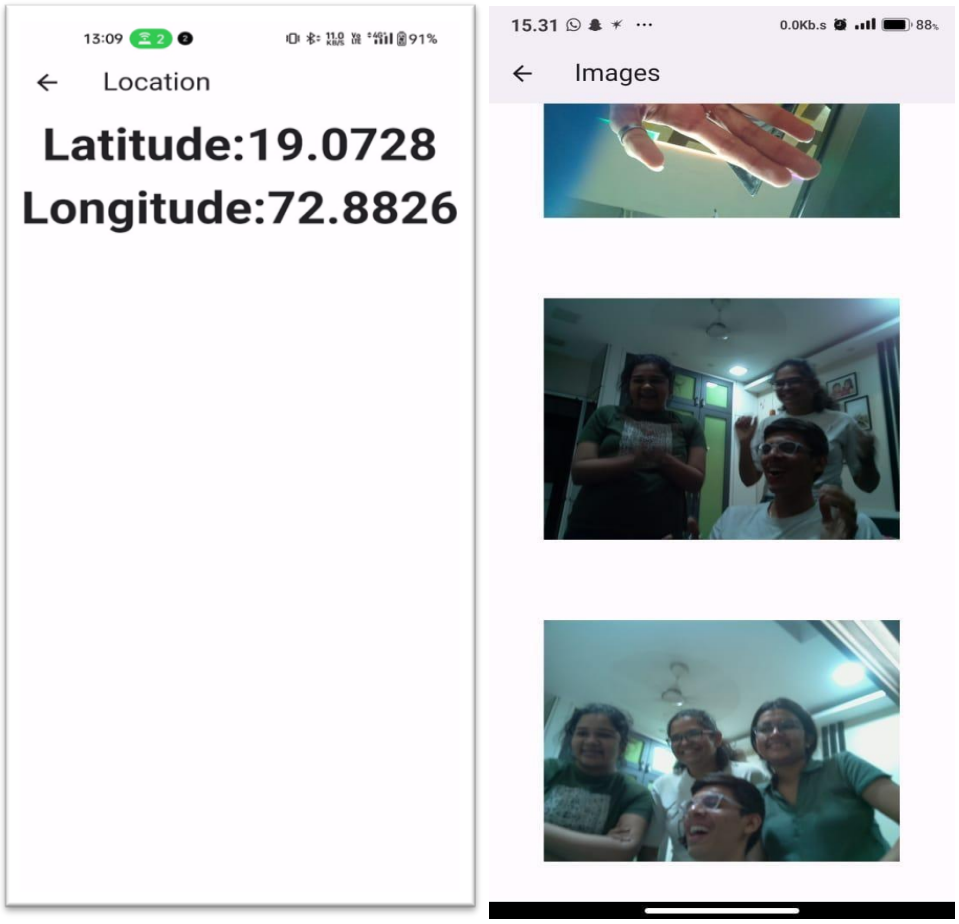


Fig 8.3: Location and images captured displayed.

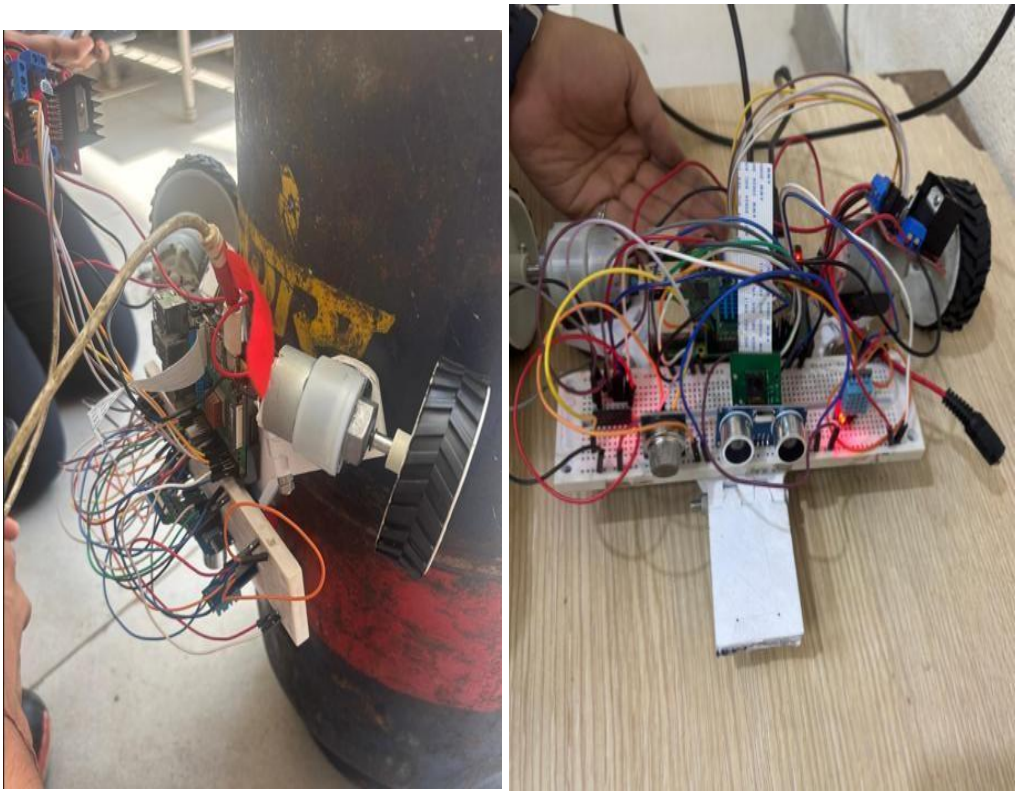


Fig 8.4: Working prototype images

The test period produced encouraging results confirming the functionality and performance of the proposed robotic system. Temperature and humidity sensor (DHT11) delivered appropriate responses to changes in atmospheric conditions, as provided in Fig 8.1 illustrating real-time data under simulated conditions. The system successfully sensed changes under the influence of hot air and differing humidity, proving its capability to detect atmospheric conditions important for early detection of hazards.

As shown in Fig 8.2, the gas leakage detection module worked according to expectations. Whenever the MG6 sensor detected gas levels above the set threshold, the system instantaneously instigated safety actions. These comprised setting the gas leak flag in Firebase, activating the audible and visible alert, and initiating an SMS alert through Twilio. This shows the system's ability to react fast in case of danger, with immediate alerts to the responsible personnel.

Figure 8.3 shows how the robot logs data effectively—capturing its location and images, then uploading them to Firebase Storage and Firestore. Even in dim lighting, the robot’s built-in torch allowed its camera to consistently take clear photos, making remote monitoring more reliable.The GPS module accurately recorded the robot's location, aiding in remote monitoring and documentation of inspection sites.

Figure 8.4 showcases images of the fully assembled working prototype. These images highlight the key components such as the sensor modules, camera system, magnetic grippers, friction wheels, and flashlight. During testing, the robot successfully navigated across metal surfaces, detected obstacles, gathered environmental data, and transmitted it to the cloud. These results confirmed that the system is operational and ready for real-world use. The smooth integration of both hardware and software demonstrated the reliability and practical potential of the robot for pipeline inspection tasks.

9. CONCLUSIONS & FUTURE SCOPE

This project is a significant step forward in industrial efficiency and safety by successfully demonstrating an intelligent and real-time gas leakage detection system for critical industrial pipelines. Leaning on the Internet of Things (IoT) capability, the robotic system offers a comprehensive inspection solution never achievable with traditional methods. The robot's sophisticated mechanical design allows it to move independently along vertical as well as horizontal pipe segments to ensure extensive pipeline network coverage. The robot is equipped with a set of advanced environmental sensors that can detect a range of significant parameters indicating gas leaks, transmit real-time and localized measurements. The onboard camera system also ensures visual confirmation and close-up views of potential areas of concern.

The integration of machine learning-based algorithms is the differentiating component, enabling the system to look beyond detection to achieve enhanced prediction accuracy for leaks. By understanding historical patterns and real-time sensor inputs, the system can recognize subtle anomalies and predict impending leak events before they become significant hazards. This forward-looking measure enables prompt intervention and preventive maintenance, minimizing downtime while ensuring continuity in operations. Cloud connectivity forms an integral component of system design, enabling secure and reliable remote monitoring. Plant managers and safety personnel are able to monitor real-time information, receive alerts, and review trends remotely from any place, enabling intelligent decision-making and quick response to emergency conditions. By automating the inspection process and reducing human exposure to potentially hazardous environments, this system contributes significantly to minimizing industrial pipeline risk and ensuring safer and wiser industrial pipeline management, protecting individuals and minimizing environmental footprint. This technology creates a strong foundation for future innovation in autonomous industrial inspection and predictive maintenance technologies, marking a new generation of operational excellence.

The present solid design provides many paths for future development and extension, with even more capability and wider use promised. Adding more sensor modalities is a natural extension. Thermal imaging sensors, for example, would present a non-contact means of measuring temperature changes associated with gas leaks, providing another level of verification and possibly locating leaks even before considerable gas concentration buildup. Furthermore, the inclusion of high-end infrared (IR) cameras that are able to detect the spectral signatures of specific invisible gases would significantly enhance the system's ability to detect a wider range of potential threats, even those outside the capabilities of conventional sensors.

To further improve the robot's operating performance and responsiveness, there are potential future developments involving AI-based path planning algorithms. These sophisticated algorithms would enable the robot to intelligently move through complex and dynamic pipe systems, including heavily branched or clogged regions, minimizing its inspection route while cutting down traversal time. This would be particularly valuable for large industrial plants with huge pipeline systems.

Examining blockchain integration presents a captivating potential for enhancing data integrity and security. With the implementation of a decentralized and tamper-evident ledger, inspection information, sensor information, and maintenance records could be logged securely and auditable, creating an unchangeable record of pipeline condition and regulatory compliance. This would be particularly beneficial for highly regulated sectors.

The technological and design basis of this system has robust potential for extension to other hostile environments apart from industrial pipes. With improved mechanized resilience of the robot and appropriate sensor shielding, the system can be used for extreme environments such as chemical plants, nuclear power houses, and even underwater pipes to fulfill crucial inspection needs in such high-risk segments. The use would significantly increase the market value and societal impact of the technology.

Looking into the future of collaborative robotics, swarm robotics application in collaborative inspection and leak tracing is a revolutionary innovation. With the application of a team of networked robots that can travel in unison with one another and share information in real-time, it would potentially reduce the time spent inspecting and enhance coverage across broad industrial installations. The collaborative strategy thus would help facilitate more efficient identification and location of leaks, and this would yield faster response rates and less chances of major events. Complex communication protocols and coordinating algorithms would form the critical inputs for releasing the complete power potential of swarm-based industrial inspections.

By further developing these channels for future development, this initial project can continue to evolve as a completely new solution for protecting the security, efficiency, and environmental appropriateness of major industrial facilities in nearly all fields of application.

REFERENCES / BIBLIOGRAPHY

1. Bi, S., Lu, Y., & Zhao, J. (2020). Design of magnetic-wheeled robots capable of climbing metallic surfaces for inspection of industrial tanks and pipelines.
2. Lee, C. H., & Choi, H. R. (2018). Development of a pipe-inspecting robot with articulated joints for navigating narrow and complex pipelines.
3. Rahman, M., & Mohammed, S. B. (2019). Integration of environmental sensors for robotic gas leak detection in industrial environments.
4. Zhao, X., Wang, H., & Yang, L. (2021). Thermal and infrared imaging-based robotic system for industrial fire hazard detection and suppression.
5. Hirose, S., & Morishima, A. (2017). Application of snake robots in post-disaster inspections and industrial hazard identification.
6. Hossin, M. A., Hossain, M. S., & Nahid, A.-A. (2020). Multi-sensor autonomous robots for real-time detection of fire and gas leaks in hazardous areas.
7. Zhang, D., & Wei, B. (2022). Magnetic wall-climbing robots for gas leak inspection in petrochemical environments.
8. Dhakad, A., Panwar, A., & Sharma, A. (2021). Real-time data transmission systems for robots in hazardous industrial environments.
9. Taher, A. A., Hassanein, H. S., & Abdel Hamid, S. (2019). Integration of wireless sensor networks in mobile robots for toxic gas monitoring.
10. Ehsan, S., Wahid, M., & Ghazali, M. F. (2022). Robotic systems using water jet propulsion and thermal imaging for industrial firefighting.
11. Bi, S., Lu, Y., & Zhao, J. (2020). Design of magnetic-wheeled robots capable of climbing metallic surfaces for inspection of industrial tanks and pipelines.
12. Lee, C. H., & Choi, H. R. (2018). Development of a pipe-inspecting robot with articulated joints for navigating narrow and complex pipelines.

13. Rahman, M., & Mohammed, S. B. (2019). Integration of environmental sensors for robotic gas leak detection in industrial environments.

14. Zhao, X., Wang, H., & Yang, L. (2021). Thermal and infrared imaging-based robotic system for industrial fire hazard detection and suppression.

15. Hirose, S., & Morishima, A. (2017). Application of snake robots in post-disaster inspections and industrial hazard identification.

16. Hossin, M. A., Hossain, M. S., & Nahid, A.-A. (2020). Multi-sensor autonomous robots for real-time detection of fire and gas leaks in hazardous areas.

17. Zhang, D., & Wei, B. (2022). Magnetic wall-climbing robots for gas leak inspection in petrochemical environments.

18. Dhakad, A., Panwar, A., & Sharma, A. (2021). Real-time data transmission systems for robots in hazardous industrial environments.

19. Taher, A. A., Hassanein, H. S., & Abdel Hamid, S. (2019). Integration of wireless sensor networks in mobile robots for toxic gas monitoring.

20. Ehsan, S., Wahid, M., & Ghazali, M. F. (2022). Robotic systems using water jet propulsion and thermal imaging for industrial firefighting.

ABSTRACT

HazardScout – Multi-Purpose Emergency Response Robot

An IOT enabled robot for gas leakage detection that can traverse pipes horizontally as well as vertically while efflciently overcoming obstacles. The processing unit (l)receives input from the gas sensor (2) and the temperature humidity sensor (7) to detect gas leaks . The imaging device (5) is used to provide visual inspection . The strong magnets (4) beneath enable the robot to adhere to the pipes. The castor wheels (9) assist the robot to climb pipes vertically and overcome obstacles. The ultrasonic sensor (6) detects the proximity of obstacles, ensuring the robot avoids collisions with them . The processing unit (l) tracks the location and sends alert to the monitoring and control center (l l) if gas leakages are detected.

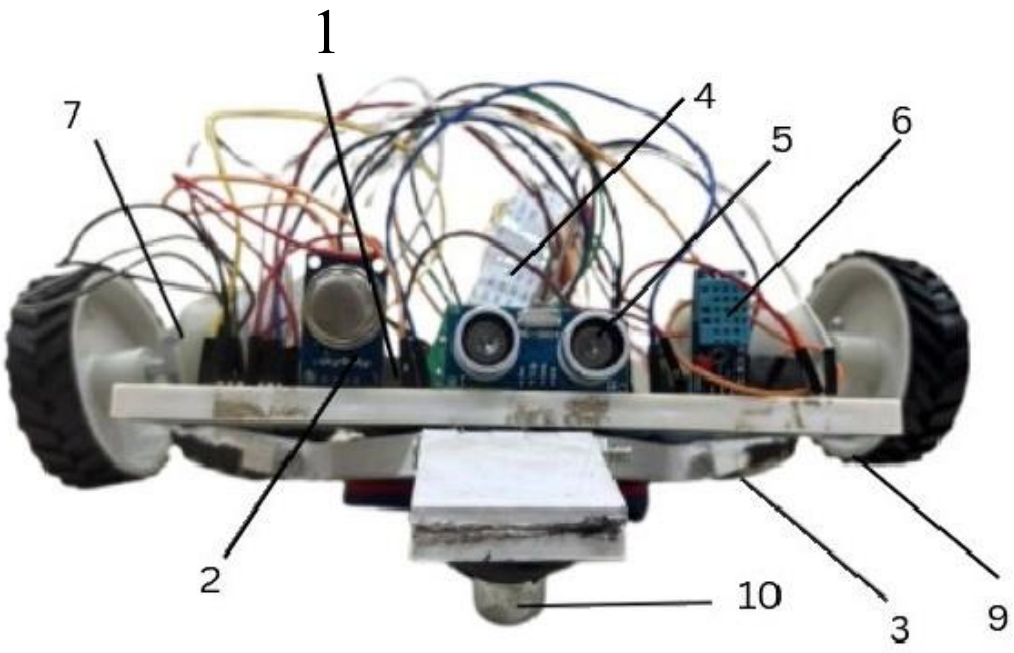


Figure 1

Plagiarism Report

Copyleaks

Analysis Report

Plagiarism Detection and AI Detection Report

HazardScout_BlackBook.pdf

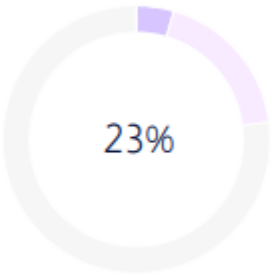
Scan details

Scan Time	Total Pages	Total Words
May 7th, 2025 at 09:04 UTC	47	11669

Plagiarism Detection

Plagiarism Types	Text Coverage	Words
Identical	1.6%	185
Minor Changes	1%	122
Paraphrased	0.9%	110
Excluded		
Omitted Words		0

AI Detection



	Text Coverage	Words
AI Text	23%	2,679
Low Frequency		530
Medium Frequency		0
High Frequency		4
Human Text	77%	8,990
Excluded		
Omitted Words		0

Plagiarism

Results (14)

*Results may not appear because the feature has been disabled.

Repository	Internal Database	Filtered / Excluded
0	0	0

Internet Sources	Current Batch
14	0

Plagiarism Types	Text Coverage	Words
Identical	1.6%	185
Minor Changes	1%	122
Paraphrased	0.9%	110
Excluded		
Omitted Words		0

About Plagiarism Detection

Our AI-powered plagiarism scans offer three layers of text similarity detection: Identical, Minor Changes, and Paraphrased. Based on your scan settings we also provide insight on how much of the text you are not scanning for plagiarism (Omitted words).

Identical

One to one exact word matches. Learn more

Minor Changes

Words that hold nearly the same meaning but have a change to their form (e.g. "large" becomes "largely"). Learn more

Paraphrased

Different words that hold the same meaning that replace the original content (e.g. "large" becomes "big"). Learn more

Omitted Words

The portion of text that is not being scanned for plagiarism based on the scan settings. (e.g. the 'Ignore quotations' setting is enabled and the document is 20% quotations making the omitted words percentage 20%) Learn more

Copyleaks Internal Database

Our Internal Database is a collection of millions of user-submitted documents that you can utilize as a scan resource and choose whether or not you would like to submit the file you are scanning into the Internal Database. Learn more

Filtered and Excluded Results

The report will generate a complete list of results. There is always the option to exclude specific results that are not relevant. Note, by unchecking certain results, the similarity percentage may change. Learn more

Current Batch Results

These are the results displayed from the collection, or batch, of files uploaded for a scan at the same time. Learn more