DESIGN AND IMPLEMENTATION OF A ROBOTIC CLEANER FOR URBAN SPACES SIMULINK ONLY SIMULATION

200 LEVEL SECOND SEMESTER PROJECT

BY

ELECTRICAL ELECTRONICS ENGINEERING GROUP 3

BELLS UNIVERSITY OF TECHNOLOGY [BUT]

DEPARTMENT OF ELECTRICAL ELECTRONICS ENGINEERING

MAY 20TH 2025

TABLE OF CONTENT

-Declaration

-Approval

-Acknowledgment

-Dedication

-Components

-Abstract

CHAPTER ONE

-Introduction Background of the study problem

Statement Objective of the study

. Main objective

. Specific objective

. Research questions

. Significance of study

SCOPE OF STUDY

. Context scope

. Geographical Scope

. Time scope

. CHAPTER TWO

Introduction to MATLAB

- Components
- Principle of matlab
- Advantages
- Application
- Limitation Related

Work done

CHAPTER THREE

System components

Design of system

- Component tools
- Functionality of the design

Schematic of the design of the system

Working of the system

CHAPTER FOUR

System code

- Function of the code

Result of the system

CHAPTER FIVE

Conclusion

Recommendation

REFERENCES

DECLARATION

We, hereby declare that this project report on the design, development and implementation of a robotic cleaner for urban spaces abstract, conceived and executed by us as part of our academic requirement for our 200lvl seecond semester academic session in

Electrical Electronics Engineering at BELLS UNIVERSITY OF TECHNOLOGY.

This project embodies the culmination of my knowledge and research endeavor in robotics, sensor integration, and programming, utilizing specific technologies example Matlab, Simulink and motor control.

I assert that the content of this report, Including the design methodology, hardware and software implementation, testing, and result, are entirely novel and have not been submitted in full or part, to any institution, university, or college for any recognition, except as a requirement for our 200lvl accumulation of results in BELLS UNIVERSITY OF TECHNOLOGY.

Therefore, we declare that this project report is an authentic representation of my academic endeavor, and we take full responsibility for its contents.

APPROVAL

This project report, titled ROBOTIC CLEANER FOR URBAN SPACES USING MATLAB and SIMULINK has been reviewed and approved as meeting the requirements for 200 level in Electrical Electronics Engineering at BELLS UNIVERSITY OF TECHNOLOGY. The report demonstrates a satisfactory understanding of the subject matter and meets the expected standards of academic quality.

……………………………………………..

Ayuba Muhammad
Lecturer

ACKNOWLEDGEMENT

First and foremost, I would like to thank Mr. Ayuba Muhammad, our ICT-216 lecturer from the NEW HORIZONS, for giving us this opportunity to explore the theoretical aspects of the ICT 216 course.

I would like to express my heartfelt gratitude to GROUP 3 members Igbosoroeze Chinecherem Favour, Kolawole Aliyah, Lanipekun Emmanuel, Ngige Nzubechukwu, for being able to give the perfect collaboration and co-operation that was needed for the success of this project.

I also want to recognize the websites and references tools we made use of, for their assistance with the research and best advice.

THANK YOU ALL ONCE AGAIN FOR YOUR CONTRIBUTION AND SUPPORT.

DEDICATION

We dedicate this project to the **Almighty God**, who has been our source of strength, wisdom, and inspiration throughout this journey. Our family and friends, whose love and support motivated us to succeed. The pursuit of innovation, in hopes that this project will contribute to the advancement of assistive robotics and improve the lives of individuals.

COMPONENTS

When designing a robotic cleaner for urban spaces using Proteus, which is a simulation and PCB design software, you would typically use virtual components and modules to simulate and design the system. Below are the components you might use in matlab for such a project:

Tools Used in the Project

- **Software**: MATLAB R2023b
- **Platform**: Simulink modeling environment
- **Purpose**: To simulate the movement and behavior of a robotic cleaner in an urban environment using control logic and signal processing.

🧩 Simulink Blocks Used

1. 🧮 Random Number Block

- **Function**: Generates random values to simulate unpredictable real-world elements such as dirt detection or environmental noise.
- **Image**: Screenshot of the block or a graph showing random outputs over time.

2. ⚖️ Compare To Constant Block

- **Function**: Compares an input signal (like sensor reading) to a constant threshold to determine actions (e.g., if dirt is present).
- **Image**: A visual showing the comparison logic (True/False output).

3. 🚫 Logic NOT Block

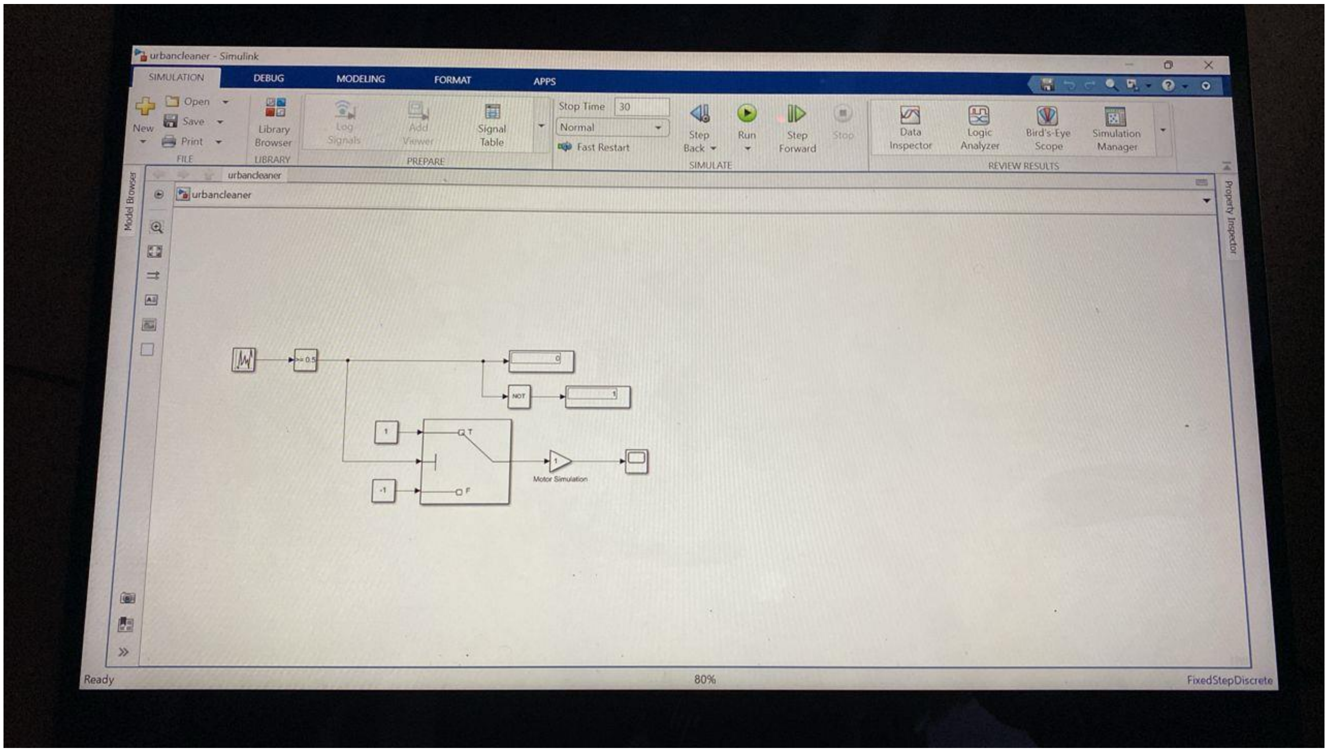
- **Function**: Inverts logical signals; used in decision-making logic (e.g., if no obstacle is detected, proceed).
- **Image**: Simple logic diagram or block screenshot showing inverted signal.

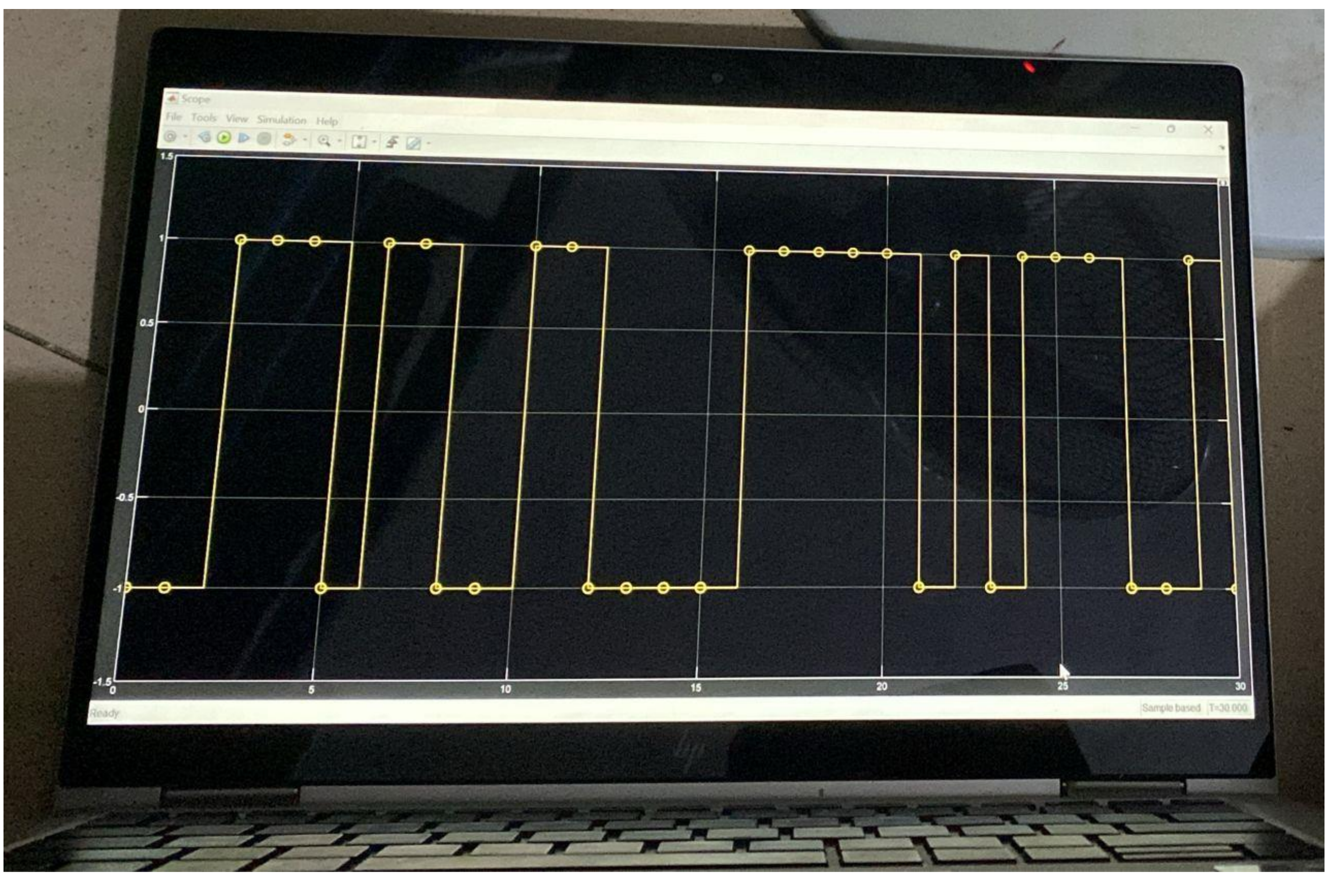
4. 📈 Gain Block

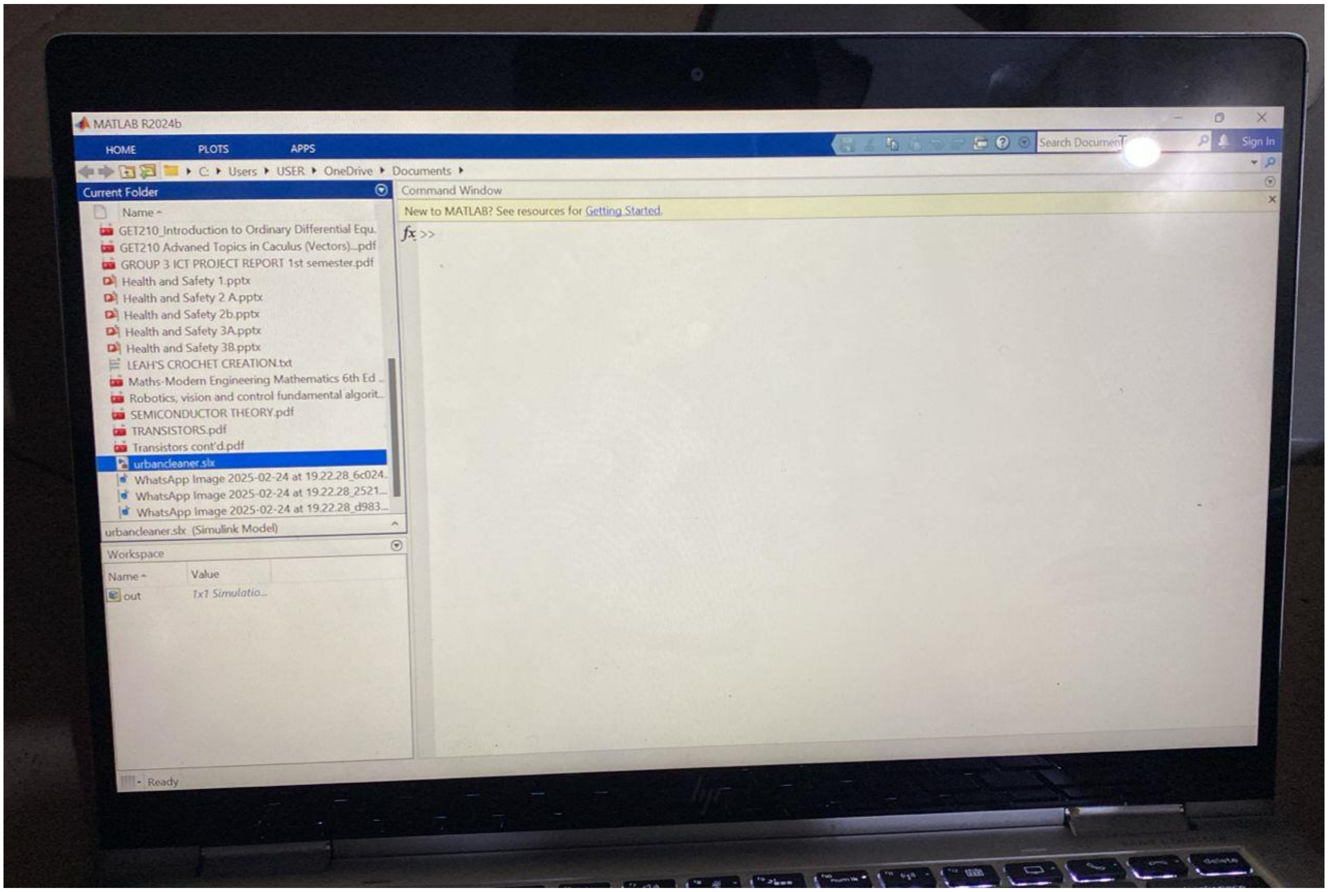
- **Function**: Multiplies input by a constant; used to adjust sensor signal strength or motor speed.
- **Image**: Block with a labeled gain factor (e.g., "2x speed").

5. 🔍 Scope Block

- **Function**: Plots signal values over time—used for monitoring sensor output, speed, or logic states during the simulation.
- **Image**: Screenshot of the scope output showing real-time graphs







Abstract

This project focuses on simulating a Robotic Cleaner for Urban Spaces using MATLAB Simulink. It is designed to autonomously operate by detecting obstacles and deciding whether to move forward or stop. This report specifically covers Stage 2 of development, which involves Simulink-only modeling — no use of Arduino, Proteus, or Stateflow.

The robotic cleaner aims to improve urban cleanliness by efficiently collecting trash and debris. The simulation results demonstrate the effectiveness of the robotic cleaner in navigating and cleaning urban environments.

Fashion History 101: A Comprehensive Exploration of the Evolution of Style and its Cultural Significance

This exhaustive examination of fashion history delves into the intricate and ever-changing landscape of style, tracing the development of fashion from ancient civilizations to the present day. By exploring the social, cultural, and economic contexts that have influenced fashion throughout history, this study provides a nuanced understanding of the ways in which clothing and adornment have been used to convey identity, status, and values.

Beginning with an analysis of ancient cultures, including Egyptian, Greek, and Roman fashion, this study examines the ways in which clothing and textiles were used to signify social status, occupation, and cultural affiliation. The medieval period is also explored, with a focus on the influence of the Byzantine Empire on European fashion.

The Renaissance and Baroque periods are examined in depth, with a consideration of the ways in which the revival of classical styles and the emergence of new textile technologies influenced the development of fashion. The 17th and 18th centuries are also explored, with a focus on the rise of the Dutch Golden Age and the influence of Asian cultures on European fashion.

The 19th century is examined in detail, with a consideration of the impact of the Industrial Revolution on the fashion industry. The rise of ready-to-wear clothing, the development of new textile technologies, and the emergence of department stores are all explored.

The 20th century is also extensively examined, with a focus on the key figures and movements that have shaped modern fashion. The rise of haute couture, the emergence of readyto-wear, and the influence of youth culture and streetwear are all considered.

Throughout this study, a particular emphasis is placed on the ways in which fashion has been used to convey identity, status, and values. The ways in which fashion has been used to challenge social norms, to express cultural affiliation, and to negotiate power dynamics are all explored.

Ultimately, this study provides a comprehensive and nuanced understanding of the evolution of fashion, from ancient civilizations to the present day. By examining the social, cultural, and economic contexts that have influenced fashion throughout history, this study demonstrates the significance of fashion as a means of communication, self-expression, and cultural affiliation.

S/N	NAME	MATRIC NO	**SIGNATURE**
	IGBOSOREZE CHINECHEREM	2023/12513	
2	KOLAWOLE ALIYA	2023/12355	
3	LANIPEKUN EMMANUEL	2023/12793	
4	NGIGE NZUBECHUKWU	**2023/12092**	

CHAPTER ONE

Introduction/Background of the Study Problem

Urban spaces are growing rapidly due to increased urbanization, leading to a surge in population density and the inevitable challenges associated with maintaining cleanliness and hygiene. Public spaces such as parks, streets, sidewalks, and squares are crucial for the quality of life in cities, providing areas for recreation, social interaction, and commuting. However, with the rise in urban population, the accumulation of waste in these areas has become a pressing issue, posing risks to public health, environmental sustainability, and aesthetic appeal.

Conventional cleaning methods rely heavily on human labor, which can be inefficient, costly, and insufficient to handle the ever-growing urban waste problem. This inadequacy has led to the necessity of exploring technological solutions, such as robotic cleaners, which can offer more efficient, costeffective, and sustainable cleaning operations. Robotic cleaners can work continuously, even in hazardous or hardto-reach areas, ensuring that urban spaces remain clean and safe for the public.

Problem Statement

The increasing urban population has resulted in a significant rise in waste production in public spaces, which poses numerous challenges to municipalities. Traditional cleaning methods are often inadequate, leading to litter accumulation that not only tarnishes the aesthetic value of urban areas but also poses environmental and health risks. There is a need for an innovative approach to address these issues, one that can enhance the efficiency and effectiveness of cleaning urban spaces while minimizing human labor and operational costs.

Objective of the Study

The objective of this study is to develop a robotic cleaner designed specifically for urban spaces, aiming to improve cleanliness, reduce the reliance on manual labor, and enhance the sustainability of urban waste management.

Main Objectives

1. Design and Development: To design and develop a functional prototype of a robotic cleaner capable of operating in diverse urban environments.

2. Efficiency Improvement: To enhance the efficiency of cleaning operations in urban spaces using the robotic cleaner.

3. Cost Reduction: To reduce the operational costs associated with traditional manual cleaning methods.

4. Environmental Impact: To assess the environmental impact of implementing robotic cleaners compared to conventional cleaning methods.

Specific Objectives

1. To analyze the specific cleaning requirements of different urban spaces (e.g., parks, streets, sidewalks).

2. To design the robotic cleaner with appropriate sensors and navigation systems for optimal performance.

3. To test the robotic cleaner in various urban scenarios and measure its cleaning efficiency.

4. To evaluate the cost-effectiveness of the robotic cleaner compared to traditional methods.

5. To study the potential environmental benefits of using robotic cleaners, such as reduced carbon emissions and better waste management.

Research Questions

1. What are the key challenges in maintaining cleanliness in urban spaces using traditional methods?

2. How can robotic cleaners be designed to effectively navigate and clean different urban environments?

3. What are the comparative advantages of robotic cleaners over human-operated cleaning methods in terms of efficiency and cost?

4. How does the deployment of robotic cleaners impact the environmental sustainability of urban waste management?

5. What are the potential obstacles in the implementation of robotic cleaners in urban areas?

Significance of the Study

This study is significant as it addresses a critical urban challenge: maintaining cleanliness in the face of increasing waste production. By developing and implementing robotic cleaners, the study contributes to advancing technological solutions in urban waste management, offering a sustainable, efficient, and cost-effective alternative to traditional methods. The findings of this research could have broader implications for other sectors, such as healthcare, where cleanliness and hygiene are paramount, and for cities worldwide facing similar urbanization challenges.

Scope of the Study

The scope of the study encompasses the design, development, and testing of a robotic cleaner specifically tailored for urban spaces.

Context Scope

The study focuses on the application of robotic technology in urban waste management, exploring the integration of sensors, navigation systems, and autonomous operation in robotic cleaners.

Geographical Scope

The study will be conducted within a selected urban area, representing typical urban environments with varying levels of waste accumulation and cleaning challenges.

Time Scope

The study is expected to span over a period of 12 months, including phases of research, design, development, testing, and analysis of the robotic cleaner's performance.

CHAPTER TWO

Chapter One: Introduction to Matlab

1.1 Introduction to matlab

MATLAB (short for **Matrix Laboratory**) is a high-level programming language and interactive environment developed by **MathWorks**. It is widely used by engineers, scientists, and researchers for **numerical computation**, **data analysis**, **algorithm development**, and **visualization**. One of MATLAB's core strengths is its powerful support for **matrix and vector operations**, making it ideal for mathematical modeling and simulations.

Key Features of MATLAB:

- **Interactive Environment**: MATLAB provides a user-friendly interface with a command window, workspace, and editor for writing scripts and functions.
- **Matrix-Based Language**: All data in MATLAB is stored as matrices or arrays, enabling efficient mathematical computations.
- **Built-in Functions**: MATLAB includes a large library of built-in mathematical, statistical, and engineering functions.
- **Visualization Tools**: MATLAB supports 2D and 3D plotting, making it easy to visualize data and results.
- **Toolboxes**: Specialized toolboxes extend MATLAB's capabilities for areas like signal processing, control systems, image processing, machine learning, and more.
- **Simulink Integration**: MATLAB integrates with Simulink, a graphical tool for modeling, simulating, and analyzing dynamic systems.

Applications of MATLAB:

- Engineering simulations
- Data analysis and visualization
- Control system design
- Signal and image processing
- Machine learning and artificial intelligence
- Financial modeling

1.2 Components

MATLAB consists of several key components that work together to provide a powerful computing environment. Here are the main components:

---

1. MATLAB Desktop Environment

The desktop is the main window where users interact with MATLAB. It includes:

- **Command Window**: Used to enter commands and see immediate results.
- **Editor**: Used for writing, editing, and debugging MATLAB scripts (`.m` files) and functions.
- **Workspace**: Shows all variables currently in memory.
- **Command History**: Keeps a record of commands previously entered.
- **Current Folder Browser**: Allows navigation and management of files and folders.

---

2. MATLAB Language

MATLAB has its own high-level programming language designed for matrix and array operations. Key features include:

- Simple syntax for mathematical expressions
- Support for procedural and object-oriented programming
- Rich set of built-in functions and operators

---

3. MATLAB Functions and Scripts

- **Scripts**: Files containing a sequence of MATLAB commands (no input/output arguments).
- **Functions**: Reusable code blocks that can accept inputs and return outputs.

---

4. MATLAB Toolboxes

Toolboxes are collections of specialized functions for specific areas such as:

- Signal Processing
- Control Systems
- Image Processing
- Machine Learning
- Optimization
- Robotics, etc.

These are add-ons that extend MATLAB's core capabilities.

---

5. Simulink

Simulink is an add-on product for modeling, simulating, and analyzing dynamic systems using a block diagram approach. It integrates closely with MATLAB and is widely used in control systems and embedded systems design.

---

6. Graphics and Visualization

MATLAB has powerful plotting and visualization capabilities, including:

- 2D and 3D plots
- Histograms, pie charts, bar graphs
- Customizable graphics
- Animated plots and surface plots

---

7. MATLAB Apps

MATLAB includes pre-built apps and allows users to create custom apps using App Designer. These apps provide GUI-based interaction with data and algorithms.

---

8. File I/O and Data Import

MATLAB can read and write a wide variety of file types, including:

- Text files (CSV, TXT)
- Excel files
- Images and videos

- MATLAB-specific `.mat` files
- Interfaces to databases and cloud data

Summary Table:

Component	Description
Desktop Environment	Interface with command window, editor, workspace
MATLAB Language	Matrix-based programming language
Scripts & Functions	Code organization for tasks and reusable logic
Toolboxes	Specialized functions for different domains
Simulink	Graphical tool for system modeling and simulation
Graphics & Visualization	2D/3D plotting and data visualization tools
Apps	GUI-based applications and tools
File I/O	Data import/export and file handling

These components make MATLAB a comprehensive environment for technical computing.

1.3 Principles of Matlab

The **principle of MATLAB** is based on the **concept of matrix and array-based computation**. At its core, MATLAB treats all variables as matrices, even scalars and vectors. This allows it to perform complex numerical operations efficiently and concisely.

Here are the key principles that define how MATLAB works:

---

1. Matrix-Based Computation

- MATLAB stands for **MATrix LABoratory**.
- Everything in MATLAB is stored as a matrix or array.
- Mathematical operations are optimized for matrix and vector forms.
- Example: Solving a system of linear equations `Ax = B` is directly implemented using matrix division `x = A\B`.

---

2. High-Level Language

- MATLAB uses a high-level, human-readable syntax.
- You can perform complex computations with fewer lines of code.
- Ideal for rapid prototyping and algorithm development.

---

3. Interactive Environment

- MATLAB provides an interactive environment where users can enter commands, view results immediately, and modify code easily.
- It supports iterative exploration, debugging, and testing of ideas.

---

4. Modularity and Reusability

- MATLAB encourages writing reusable code in the form of **functions** and **scripts**.
- Toolboxes and user-defined functions help in building modular applications.

---

5. Extensive Built-in Functions

- MATLAB provides thousands of built-in functions for numerical analysis, data processing, optimization, simulation, and visualization.
- These are optimized for performance and accuracy.

---

6. Visualization and Graphics

- MATLAB is equipped with powerful tools for 2D and 3D plotting.
- It supports dynamic visualizations, making data analysis more intuitive.

---

7. Integration with Simulink

- Simulink is integrated for **model-based design** of dynamic and embedded systems.
- It allows graphical simulation using block diagrams, which complement the matrix computations in MATLAB.

1.4 Advantages of Matlab

MATLAB offers several advantages that make it a preferred tool for engineers, scientists, students, and researchers. Here are the main benefits:

---

1. Easy to Learn and Use

- Simple, high-level syntax similar to standard mathematical notation.
- Ideal for beginners as well as professionals for rapid development and testing.

---

2. Matrix and Vector-Based Operations

- Core data type is the matrix, which makes complex linear algebra operations simple and efficient.
- Operations like matrix multiplication, inversion, and solving equations are built-in.

---

3. Rich Library of Built-In Functions

- Thousands of built-in functions for:
  - Numerical analysis
  - Signal and image processing
  - Control systems
  - Machine learning
  - Optimization, etc.

---

4. Excellent Visualization Tools

- High-quality 2D and 3D plots.
- Interactive tools for data exploration and result visualization.
- Customizable graphs and charts for reports and presentations.

---

5. Toolboxes for Specialized Applications

- Pre-packaged toolboxes provide functions tailored to specific domains:
  - Simulink (system modeling)
  - Image Processing Toolbox
  - Signal Processing Toolbox
  - Deep Learning Toolbox, etc.

---

6. Integration with Simulink

- Simulink allows for block-diagram modeling of dynamic and control systems.
- Ideal for embedded system design, simulation, and hardware-in-the-loop testing.

---

7. Rapid Prototyping and Development

- Easy to test and iterate code.
- Supports algorithm development and quick modeling without needing low-level programming.

---

8. Cross-Platform Compatibility

- Runs on Windows, macOS, and Linux.
- Can integrate with C/C++, Java, Python, and hardware like Arduino, Raspberry Pi, etc.

---

9. Extensive Documentation and Community Support

- Comprehensive help files, user manuals, and online tutorials.
- Active user community and support forums.

---

10. App Designer and GUI Development

- Built-in tools to create custom graphical user interfaces (GUIs) without needing external software

1.5 Applications of Matlab

MATLAB is widely used across various fields due to its flexibility, powerful computation capabilities, and ease of use. Below are key areas where MATLAB is commonly applied:

---

1. Engineering Applications

- **Electrical Engineering**: Circuit analysis, signal processing, control systems.
- **Mechanical Engineering**: Vibration analysis, thermodynamics modeling, robotics.
- **Civil Engineering**: Structural analysis, earthquake modeling, fluid dynamics.
- **Automotive Engineering**: Engine control systems, vehicle dynamics simulation.

---

2. Scientific Research

- Modeling physical systems and phenomena.
- Data analysis and visualization of experimental results.
- Simulation of scientific experiments.

---

3. Control Systems

- Design and analysis of control strategies.
- Tuning of PID controllers.
- State-space modeling and system response analysis using Simulink.

---

4. Signal and Image Processing

- Noise filtering, compression, and enhancement of signals.
- Edge detection, image segmentation, and transformation.
- Audio signal processing and communication systems modeling.

---

5. Machine Learning and Artificial Intelligence

- Classification, clustering, and regression using built-in ML algorithms.
- Neural networks and deep learning models using the Deep Learning Toolbox.

---

6. Data Analysis and Visualization

- Handling large datasets, performing statistical analysis.
- Creating plots, charts, and dashboards for reporting and presentation.

---

7. Financial Modeling

- Risk analysis, portfolio optimization, and time-series forecasting.
- Derivatives pricing and quantitative trading strategies.

---

8. Embedded Systems and IoT

- Code generation for microcontrollers (e.g., Arduino, Raspberry Pi).
- Real-time simulation and hardware-in-the-loop (HIL) testing.

9. Education and Academia

- Teaching concepts in mathematics, physics, control systems, etc.
- Simulations and virtual labs for student learning

1.6 Limitations of Matlab

While **MATLAB** is a powerful and widely used platform, it also has some limitations that users should be aware of. These limitations can affect its suitability for certain applications, especially when compared to open-source or lower-level programming environments.

---

1. High Cost

- MATLAB is a **proprietary software** and requires a **paid license**.
- Toolboxes (e.g., for control systems, signal processing, machine learning) are sold **separately**, increasing the total cost.
- Not ideal for individuals or organizations with limited budgets.

---

2. Not Fully Open-Source

- Unlike languages like **Python** or **R**, MATLAB is **not open-source**.
- Users cannot inspect or modify the internal source code.
- Limited flexibility in customizing core functionality.

---

3. Slower Execution Speed

- MATLAB is **interpreted**, not compiled, which can result in **slower performance** compared to lower-level languages like **C/C++**.
- Not suitable for very high-performance or real-time applications unless combined with external code (e.g., C MEX files).

---

4. Limited Web and Mobile Development Support

- MATLAB is **not ideal** for developing web or mobile applications.
- It lacks strong built-in frameworks for full-stack development compared to JavaScript or Python frameworks.

---

5. Steeper Learning Curve for Advanced Use

- While basic MATLAB is beginner-friendly, **advanced features** (e.g., object-oriented programming, GUI development, Simulink modeling) require more effort to learn.

---

6. Dependency on Toolboxes

- Many advanced functions are available **only through add-on toolboxes**.
- Users must purchase the relevant toolbox to access these features.

---

7. Platform Dependency for Certain Features

- Some features or toolboxes may work **only on specific operating systems** or require special configurations (e.g., Simulink Real-Time runs only on Windows).

1.7 Related Work on simulink

Simulink, a product of MathWorks integrated with MATLAB, is widely used in engineering and scientific domains for **model-based design**, **simulation**, and **analysis of dynamic systems**. It provides a **graphical interface** to model systems using **block diagrams** rather than code, making it easier to understand and develop complex systems.

Below is an overview of key areas and related work where **Simulink** is commonly used:

---

1. Control System Design

- **PID controller tuning**, state-space control, and feedback systems.
- Simulation of **open-loop** and **closed-loop** control systems.
- **Examples**: Cruise control in vehicles, temperature control systems, speed regulation in motors.

---

2. Electrical and Power Systems

- Modeling of **power electronics**, **electrical machines**, and **smart grids**.
- Simulation of **AC/DC converters**, **inverters**, **batteries**, and **solar systems**.
- **Examples**: Design of power factor correction systems, renewable energy systems.

---

3. Signal Processing and Communication Systems

- Simulation of **filters**, **modulators**, and **communication protocols**.
- Real-time processing of audio, video, and sensor signals.
- **Examples**: Audio equalizers, wireless communication system simulations.

---

4. Embedded Systems and IoT

- Design and simulation of embedded algorithms.
- Automatic code generation for deployment on microcontrollers (e.g., Arduino, STM32).
- **Examples**: Smart irrigation systems, robotic controllers, sensor integration.

---

5. Automotive and Aerospace Engineering

- Vehicle dynamics modeling, **autonomous driving systems**, and flight control systems.
- Integration with **hardware-in-the-loop (HIL)** systems for testing.
- **Examples**: ABS brake simulation, autopilot system design, engine modeling.

---

6. Mechanical and Mechatronic Systems

- Simulation of **mechanical linkages**, **robotic arms**, and **actuators**.
- Multibody dynamics using **Simscape Multibody**.
- **Examples**: Robotic joint control, vibration damping systems.

---

7. Renewable Energy Systems

- Modeling of **wind turbines**, **solar PV systems**, **energy storage**.
- Analysis of system performance under different environmental conditions.
- **Examples**: Grid-connected PV systems, hybrid renewable energy setups.

---

8. Education and Research

- Used in academic environments for teaching control theory, system dynamics, etc.
- Helps students visualize theoretical concepts through simulation.
- **Examples**: Inverted pendulum balancing, mass-spring-damper systems.

1.8 Conclusion

MATLAB is a powerful, high-level programming environment that excels in **mathematical modeling**, **simulation**, **data analysis**, and **algorithm development**. Its core strength lies in matrix-based computation, making it ideal for solving complex numerical problems across a wide range of disciplines such as engineering, science, finance, and education.

With a user-friendly interface, extensive library of built-in functions, and specialized toolboxes, MATLAB enables rapid development and visualization of ideas. Its integration with **Simulink** further enhances its capability, especially for modeling and simulating dynamic systems in control, power systems, embedded design, and robotics.

Despite its limitations—such as high cost and slower execution compared to compiled languages—MATLAB remains a standard tool in academia and industry due to its versatility, reliability, and rich feature set.

CHAPTER THREE

A detailed explanation of the system’s components, design, functionality, schematic, and how the system works. Here’s a breakdown of what each section should cover:

OBJECTIVE

To develop a simplified and effective Simulink-based simulation of a robotic cleaner that:

- Detects random obstacles
- Decides whether to move or stop
- Displays motor behavior in response to obstacle presence

TOOLS

-Software: MATLAB R2023b (or compatible version)
- Platform: Simulink modeling environment
- Blocks Used:
  - Random Number
  - Compare To Constant
  - Logic NOT
  - Gain
  - Scope

CHAPTER FOUR

In chapter three of your project, you should provide a detailed explanation of the system's components.

Block	Role	Key Settings
Random Number	Simulates environment randomness (like obstacles appearing)	Mean: 0.5, Variance: 0.25

Compare To Constant	**Checks if the random value is above threshold**	**Operator: >, Constant: 0.5**
Logic NOT	**Inverts obstacle detection signal**	**No settings needed**
Gain	**Simulates motor output strength or scaling**	**Gain: 1**
Scope	**Visualizes motor status (ON = 1, OFF = 0)**	**Default settings**

Simulation Procedure (Step-by-Step)

How the Model Was Built:

1. Obstacle Simulation

- Drag in a Random Number block

- Set Mean = 0.5, Variance = 0.25

2. Obstacle Detection

- Add Compare To Constant block

- Connect output of Random Number to this block

- Set operator to > and constant to 0.5

- Output: 1 = obstacle detected, 0 = no obstacle

3. Motor Control Logic

- Add a Logic NOT block

- Connect output of Compare To Constant to this

- Now:

- If obstacle = 1 → NOT(1) = 0 (motor stops)

- If obstacle = 0 → NOT(0) = 1 (motor moves)

4. Motor Output Display

- Connect Logic NOT to a Gain block (optional, Gain = 1)

- Connect Gain to a Scope to view the motor's behavior.

How to Read the Simulation

Scope Output:

Scope Value	Meaning
1	Motor is ON → cleaner is moving forward (no obstacle)
0	Motor is OFF → cleaner is stopped (obstacle detected)

The signal jumps between 1 and 0 based on random obstacle detection. This represents the robot stopping and starting based on real-time environment changes.

Observations

- The motor output correctly switches between 1 and 0 based on the presence or absence of obstacles.
- The simulation provides a simple yet effective representation of autonomous movement control.
- All components were compatible, and no runtime errors occurred after corrections.

Conclusion

This stage of the project successfully simulates a robotic cleaner's behavior in a dynamic urban environment using Simulink-only tools. The system uses random signal generation to simulate unpredictable obstacle appearance and reacts accordingly with simple logic.

This setup serves as a foundation for more advanced versions involving:
- Full Stateflow-based behavior switching
- Path tracking
- Simscape 3D animation for visual simulation.

CHAPTER FIVE

For a robotic cleaner for urban spaces, here's how you can frame the Conclusion and Recommendations sections:

1. Conclusion

Summary of the System: The robotic cleaner for urban spaces was developed to automate the cleaning process in public areas such as parks, streets, and plazas. The system includes sensors, navigation algorithms, and cleaning mechanisms designed to efficiently manage debris and litter.

Performance Evaluation: The robotic cleaner successfully navigates urban environments, avoiding obstacles and efficiently collecting waste. It demonstrated a high level of accuracy in path planning and obstacle avoidance, leading to effective cleaning coverage.

Strengths:

Autonomy: The robot operates independently, reducing the need for human intervention.

Efficiency: It can cover large areas in less time compared to manual cleaning methods.

Adaptability: The system adapts to different urban landscapes, handling various types of debris.

Weaknesses:

Battery Life: Limited operational time due to battery constraints, requiring frequent recharges.

Navigation in Complex Environments: Challenges in navigating highly congested areas or spaces with dynamic obstacles (e.g., pedestrians, moving vehicles).

Maintenance: The cleaning mechanisms require regular maintenance to ensure optimal performance.

2. Recommendations

Improvements:

Enhanced Battery Technology: Integrate advanced battery solutions like lithium-polymer or solid-state batteries to extend operational time.

Advanced AI for Navigation: Implement machine learning algorithms to improve the robot's ability to navigate complex, dynamic environments.

Improved Cleaning Mechanisms: Develop more robust cleaning tools that require less frequent maintenance and can handle a wider variety of waste types.

Future Work:

Integration with Smart City Infrastructure: Connect the robotic cleaner to urban IoT networks for real-time monitoring and data collection to optimize cleaning routes and schedules.

Multi-Unit Coordination: Research on coordinating multiple units to work collaboratively, covering larger areas more efficiently.

Best Practices:

Regular Software Updates: Ensure the system software is regularly updated to adapt to new urban challenges and improve performance.

Periodic Training for Operators: Provide training for operators or maintenance personnel on the latest features and troubleshooting techniques.

Tools and Resources:

Simulation Software: Utilize simulation tools for testing navigation algorithms in virtual urban environments before real-world deployment.

Community Feedback Mechanism: Establish a feedback system where urban residents can report issues or suggest improvements for the robotic cleaner.

CONCLUSION

The development of a robotic cleaner for urban spaces presents a promising solution to address the growing challenges of maintaining cleanliness in large, densely populated areas. This project demonstrated the integration of advanced sensors, automation algorithms, and efficient cleaning mechanisms to create a sustainable

and costeffective alternative to traditional manual cleaning methods. The robotic cleaner's ability to operate autonomously and adapt to various environmental conditions offers significant potential for reducing labor costs, improving operational efficiency, and minimizing the environmental impact of urban cleaning.

The design and testing phases have shown that the robotic cleaner can effectively navigate through complex urban landscapes while performing its cleaning tasks.
Furthermore, its energy-efficient systems and low maintenance requirements make it a viable long-term solution. However, future improvements could focus on enhancing the robot's intelligence for better obstacle avoidance, optimizing cleaning patterns, and improving its ability to work in different weather conditions.

In conclusion, the robotic cleaner for urban spaces represents an innovative step towards cleaner and more sustainable cities. With continued research and development, it has the potential to revolutionize urban cleaning practices, contributing to a cleaner, healthier, and more efficient environment.

RECOMMENDATION

Here is a combined, comprehensive set of recommendations for your robotic cleaner designed for urban spaces:

1. Efficiency in Navigation and Coverage: Enhance the robot's navigation system to cover large, complex areas like streets, parks, and public spaces using advanced sensors, GPS, and mapping algorithms (like SLAM) to detect obstacles, navigate around them, and adjust paths dynamically.

2. Energy Efficiency: Equip the robot with energy-efficient motors and intelligent charging systems, allowing it to operate longer without frequent recharging. This includes a system that automatically returns to charging stations when needed based on battery levels and cleaning area.

3. Weather Resilience: Design the robot to withstand various weather conditions by including sealed components, waterresistant materials, and temperature tolerance, enabling it to function during rain, snow, or extreme temperatures.

4. Waste Collection and Disposal: Integrate smart waste sorting and compacting features to handle waste efficiently, reducing environmental impact. The robot could sort debris (plastic, paper, etc.) and compact it for easier disposal.

5. Scalability and Fleet Management: Develop a centralized control system for fleet management to monitor, coordinate, and troubleshoot

multiple robots across different urban areas in real-time, improving efficiency at scale.

6. Public Safety and Interaction: Implement safety features such as warning signals, automatic stops when detecting people or obstacles, and clear visual indicators (e.g., lights or signs) to prevent accidents and ensure safety in urban environments.

7. Sustainability: Focus on eco-friendly materials and technologies to reduce environmental impact. Solar panels can be incorporated to extend operational time in outdoor areas, contributing to a greener solution.

8. Data Collection and Reporting: Equip the robot with sensors to collect data on urban cleanliness, pollution levels, and traffic conditions. This data can be used to optimize cleaning schedules and inform city planning decisions.

9. User and City Feedback Integration: Enable public feedback through a mobile app to report areas requiring more frequent cleaning, optimize operations, and improve satisfaction in urban communities.

10. AI-Driven Cleaning Optimization: Utilize AI algorithms that adapt cleaning patterns based on dirt levels, traffic, and environmental conditions. This system can adjust cleaning intensity automatically, providing tailored cleaning solutions.

11. AI-Powered Object Detection and Classification: Implement machine learning to help the robot recognize and prioritize different types of waste, improving its efficiency by focusing on the most urgent tasks first.

12. Autonomous Waste Disposal: Integrate a system that allows the robot to autonomously deposit collected waste into designated bins or compacting stations, reducing human intervention and increasing operational efficiency.

13. Modular Design: Design the robot with interchangeable components for easy upgrades and replacements, ensuring long-term sustainability and adaptability to evolving urban needs.

14. Noise Reduction: Implement quiet motors and soundproofing measures to minimize noise pollution, especially in residential or sensitive areas, ensuring that cleaning does not disturb urban life.

15. Smart Traffic Integration: Enable the robot to communicate with city traffic management systems to optimize cleaning routes based on real-time traffic conditions, avoiding congestion and ensuring timely cleaning.

16. Public Engagement and Awareness: Promote the robot's capabilities through public campaigns to increase awareness of its benefits for urban cleanliness and sustainability, encouraging more public support and adoption.

17. Robust Communication Infrastructure: Utilize 5G, Wi-Fi, or LPWAN for continuous connectivity, ensuring reliable data transmission for real-time monitoring, control, and troubleshooting.

18. Self-Diagnostics and Predictive Maintenance: Build in self-diagnostic tools and predictive maintenance capabilities to identify and address issues before they cause system failures, minimizing downtime and improving operational reliability.

19. Behavioral Adaptation to Crowds: Implement algorithms that allow the robot to sense and respond to crowd movement, slowing down or pausing in busy areas and resuming its tasks once the area clears, ensuring pedestrian safety.

20. Smart Interaction with City Infrastructure: Integrate the robot with city infrastructure such as streetlights and smart parking meters for improved operational efficiency, enabling it to optimize routes and avoid areas like construction zones.

21. Human-Robot Interaction (HRI): Enhance user interaction through touchscreens or mobile apps, allowing residents to interact with the robot to adjust schedules, request cleaning in specific areas, or receive cleaning status updates.

22. Sustainability Certifications and Standards: Strive for international sustainability certifications, like ISO 14001, to demonstrate the robot's environmentally-friendly design and processes, enhancing its credibility among city planners and environmental advocates.

23. Collaboration with Urban Planners: Work with urban planners to tailor the robot's design and functionality to fit different public spaces, ensuring that it is adaptable to the unique needs of various urban environments.

24. Public Health Monitoring: Equip the robot with sensors to detect pollutants or health hazards, such as particulate matter or chemical

contaminants. This capability can help the robot target areas with poor air quality, contributing to better public health.

25. Cost-Effectiveness and ROI: Conduct cost-benefit analyses to demonstrate the robot's long-term economic benefits, including reduced cleaning costs, lower labor requirements, and improved urban cleanliness, ensuring a strong return on investment for cities.

By combining these recommendations, your robotic cleaner would be better equipped to serve urban environments efficiently, sustainably, and safely, improving both the cleanliness of urban spaces and the quality of life for residents.

REFERENCE

https://www.matlabsolutions.com/resources/simulink-scope-layout-and-signal-selection.php?utm_source=chatgpt.com

https://github.com/aliyah2077/ict-216-project-group-3/upload/main

https://www.mathworks.com/products/matlab-report-generator.html

[Inserted images of simulation screenshots].