# DISTANCE BASED OBJECT COUNTING IN INDUSTRIAL AUTOMATION

A project report submitted in partial fulfilment of

the requirements for the award of the

**BACHELOR DEGREE**

*in*

**Computer Applications**

*from*

**University of Calicut**

***Submitted***

***By***

**RAHUL PT (GVAWBCA042)**

**MUHAMMED SAFVAN EK (GVAWBCA039)**

**MUHAMMED FARIS TK (GVAWBCA037)**

**MOHAMMED SALMANUL FARIS P (GVAWBCA034)**

**MUHAMMAD SHAMEEMUDHEEN (GVAWBCA036)**

**MOHAMMED SHUHAIL MON (GVAWBCA035)**

***Carried out at***

**Department of Computer Applications**

Grace Valley College of Arts & Science,
**Maravattam, MARCH 2024-25**

**Department of Computer Applications**
**Grace Valley College of Arts & Science**
**Maravattam**

## Certificate

*This is to certify that the project report entitled* **“IoT-Powered Industrial Automation: Distance-Based Object Counting** *“is a bonafide record of the work done by* **RAHUL PT (GVAWBCA042), MUHAMMED SAFVAN EK (GVAWBCA039), MUHAMMED FARIS TK (GVAWBCA037), MOHAMMED SALMANUL FARIS P (GVAWBCA034), MUHAMMAD SHAMEEMUDHEEN (GVAWBCA036), MOHAMMED SHUHAIL MON (GVAWBCA035)** ***under our supervision and guidance. The report has been submitted in partial fulfilment of the requirement for award of the Bachelor Degree in Computer Applications from the University of Calicut for the year 2024- 25.***

| | |
|---|---|
| **Seetha Lakshmi R** | **Seetha Lakshmi R** |
| **Head of Department** | **project guide** |

Date:

Place:

Certified that the candidate was examined by us in the Project Viva Voice Examination held on…………………………….and his/her Register Number is…………………………..

**External Examiner:……………………………..**

# Declaration

Declaration We hereby declare that the project report entitled **"IoT-Powered Industrial Automation: Distance-Based Object Counting"** was carried out by us as part of our **Bachelor Degree Project in Computer Applications** under the guidance and supervision of **Ms. Seetha Lakshmi R, Head of the Department of Computer Applications at Grace Valley College of Arts & Science, Maravattam.** To the best of our knowledge and belief, this project contains no material previously published or written by another person, nor does it contain any material that has been accepted for the award of any other degree or diploma at any university or institute of higher learning, except where due acknowledgment has been made in the text. We also confirm that all the information provided in this project is based on our own research and work, and any references or sources used have been duly cited.

**DATE:**

**PLACE:**

**NAME:** Rahul.pt
Muhammed Safvan.EK
Muhammed Faris.TK
Mohammed Salmanul Faris.P
Muhammad Shameemudheen
Mohammed Shuhail Mon

# Acknowledgement

We are obediently thankful to **God Almighty,** praise and glory be to Him, for all His uncountable bounties and guidance, without which this work would have never been a reality. An endeavor over a long period can only be successful with the advice, encouragement, and guidance of many well-wishers, and we take this opportunity to express our sincere gratitude to all those who supported us in completing this project. We extend our deepest appreciation to our respected **Principal, Dr. M. Usman**, for his inspiration and for fostering an academic environment that encouraged us to carry out this project effectively. Our heartfelt gratitude goes to **Ms. Seetha Lakshmi R, Head of the Department of Computer Applications**, for granting us the necessary permissions and providing the facilities to conduct our project in a well-structured manner. We are also immensely grateful to **Ms. Varna, Assistant Professor, Ms. Murshida, Assistant Professor, and Mr. Ubaid, Lab Assistant in Computer Applications**, for their valuable guidance, timely advice, and unwavering moral support throughout the project. Their insights and constructive feedback have played a crucial role in refining our approach and ensuring the successful completion of our work. Additionally, we extend our sincere thanks to all the **teaching and non-teaching staff** of the Computer Application Department for their constant encouragement and assistance in various stages of this project. We also acknowledge the immense support and camaraderie of our **friends and classmates**, who provided valuable suggestions, motivation, and assistance whenever needed. Their collaboration has been instrumental in making this journey a fulfilling and rewarding experience. Finally, we express our heartfelt gratitude to our **families**, whose unconditional support, patience, and encouragement have given us the strength to complete this project successfully. This project is a testament to the collective efforts of everyone who has contributed in any way, and we are truly grateful for their support.

# Abstract

In industrial automation, accurate object counting is crucial for efficient inventory management, quality control, and operational optimization. Traditional methods for object counting often rely on manual inspection or fixed sensors, which can be labor-intensive and prone to inaccuracies. This paper proposes a distance-based object counting system utilizing Internet of Things (IoT) technology, which leverages sensors, such as ultrasonic or LiDAR, to detect and measure the distance to objects in real-time. The IoT framework allows for continuous monitoring and remote data processing, enabling precise counting without direct physical interaction. By integrating distance sensors with cloud-based analytics, the system enhances scalability, reliability, and efficiency. The proposed solution is tested in industrial environments, showing improved accuracy and reduced operational costs compared to conventional methods. This approach represents a significant advancement in automated object tracking, paving the way for more intelligent and responsive industrial systems.

# Contents

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

1. **IOT** – Internet of things
2. **DFD** – data flow diagram
3. **DB** – Database
4. **HTML** – Hypertext Mark-up Language
5. **CSS** – Cascading Style Sheets
6. **VS code** – Visual studio code
7. **IDE** – Integrated development environment
8. **UI** – user interface

# LIST OF TABLES

# INTRODUCTION

## 1 INTRODUCTION

In industrial automation, accurately tracking and counting objects is a fundamental requirement for various applications, including inventory management, production line monitoring, and logistics optimization. Traditional object counting methods, such as manual tallying or image-based systems, have several limitations, including human error, high labor costs, dependency on lighting conditions, and complex maintenance. These inefficiencies can lead to inaccurate data collection, operational delays, and increased costs.

To overcome these challenges, IoT-powered distance-based object counters offer a real-time, automated, and highly accurate solution. These systems utilize distance sensors such as ultrasonic, LiDAR, or infrared sensors to detect objects based on their proximity. The collected data is processed using a microcontroller (Arduino, ESP32, or Raspberry Pi), which then transmits the information to an IoT platform for remote monitoring and analysis.

How It Works:

The system functions by continuously measuring the distance between the sensor and objects within a predefined detection area. When an object enters and exits this area, the system identifies the transition and updates the count accordingly. The real-time data is then sent to a cloud-based IoT platform, where it can be analyzed and displayed on a dashboard or mobile app for monitoring and decision-making.

Key Advantages:

1. Automation & Accuracy – Reduces manual errors and ensures precise counting.
2. Real-time Monitoring – Enables live tracking of objects moving through an industrial process.
3. Cloud-Based Analytics – Allows remote access to object count data, enhancing decision-making.
4. Scalability – Easily integrates with existing industrial automation systems.
5. Cost-Effectiveness – Reduces labor costs and enhances operational efficiency.

Applications in Industry:

Manufacturing: Tracking the number of products on a conveyor belt. Warehousing & Logistics: Monitoring packages in storage or transit.

Retail & Inventory Management: Counting items in stock automatically. Smart Traffic Management: Counting vehicles for traffic control and analysis. By integrating IoT with distance-based object counting, industries can improve operational efficiency, reduce errors, and enable smart automation, making their processes more reliable and cost-effective.

## 2.METHODOLOGY

The IoT-based distance-based object counter follows a structured approach involving sensor integration, data processing, cloud connectivity, and real-time monitoring. The key steps are as follows:

1. Hardware Setup

Select a Distance Sensor: Ultrasonic Sensor (HC-SR04) – Measures distance using soundwaves.

LiDAR Sensor (VL53L0X) – Provides high-accuracy object detection.

Infrared Sensor – Detects objects using light reflection.

Microcontroller/Processor: Use ESP32, Arduino, or Raspberry Pi for data processing.

Communication Module: Implement Wi-Fi, LoRa, or Bluetooth for IoT connectivity.

Power Supply: Ensure a stable power source for continuous operation.

2. Software Development

Sensor Data Processing: Continuously measure the distance from the sensor. Use filtering techniques (e.g., Kalman filter, moving average) to remove noise. Define a threshold distance to detect objects. Count objects when they enter and leave the detection zone.

Microcontroller Programming: Write code using C++ (Arduino/ESP32) or Python (Raspberry Pi).

Implement event-based counting for precise detection.

Cloud Integration: Send data to an IoT platform (e.g., Things Board, Firebase, AWS IoT).Enable data visualization on dashboards.

3. Data Transmission & Cloud Processing

Use MQTT, HTTP, or Web Sockets to transmit sensor data in real time. Store and analyze object count trends to optimize operations. Implement alerts and notifications for anomalies.

4. Real-Time Monitoring & Analytics

Display live object count on a dashboard or mobile app. Use AI/ML analytics to detect patterns and optimize efficiency. Generate reports for process improvement and inventory tracking.

5. Testing & Optimization

Calibrate the sensor for accurate distance measurements. Test with different object sizes, speeds, and distances to refine accuracy. Optimize for low power consumption and high data reliability.

# 3.LITERATURE SURVEY

## LITERATURE SURVEY

### 3.1 INTRODUCTION

The literature survey of a project is a crucial aspect that involves a comprehensive review of existing literature, research papers, and projects relevant to the topic under study. In the context of your project on student database management system developed as a web app using the python, the literature survey serves several important purposes. It aims to explore and analyze previous works that have addressed similar challenges in managing student data, providing valuable insights into the approaches, methodologies, and technologies used in existing projects. By conducting a thorough literature survey, you can gain a deeper understanding of the current state of the art in student database management systems. This includes examining the various features, functionalities, and architectures employed in existing systems. It also involves studying the challenges faced by educational institutions in managing student data and the solutions proposed to address these challenges. Through this review, you can identify the strengths and weaknesses of existing systems, as well as any gaps or limitations that need to be addressed. Additionally, the literature survey provides a foundation for the theoretical framework of your project. By reviewing relevant literature, you can ground your work in established theories and concepts in the field of database management and student information systems. This theoretical grounding not only enhances the academic rigor of your project but also helps in shaping the design and implementation

of your system. Moreover, the literature survey serves as a basis for evaluating the significance and innovation of your project. By comparing your project with existing systems, you can highlight the novel features and contributions of your system. This comparative analysis can help you demonstrate the value proposition of your project and its potential impact in the field of student database management.

Furthermore, the literature survey enables you to position your project within the broader academic and practical landscape of student data management systems. By examining the evolution of student database management systems over time, you can contextualize the relevance and timeliness of your project. This historical perspective can also help you identify emerging trends and technologies that may influence the future development of student database management systems. In conclusion, the literature survey section of your project plays a crucial role in establishing the academic and practical foundation of your

work. By conducting a thorough review of existing literature and projects, you can demonstrate your knowledge of the field, highlight the novelty and contributions of your project, and position it within the broader context of student data management systems.

## 3.2 SURVEY ON SMART OBJECT DETECTION USING ESP32

### ABSTRACT

Smart object detection using ESP32 is an innovative approach that leverages IoT and embedded AI for real-time object identification and monitoring. The ESP32, particularly the ESP32-CAM variant, offers a cost-effective and efficient solution for applications in surveillance, home automation, and industrial automation. By integrating sensors such as ultrasonic modules and utilizing machine learning frameworks like Edge Impulse, the system enhances detection accuracy and decision-making. This study explores the implementation of smart object detection with ESP32, discussing its advantages, challenges, and practical applications. Key limitations such as processing constraints and environmental factors are also analyzed to optimize performance in real-world scenarios.

### Methodology

1. Hardware Selection & Setup:

- ESP32-CAM module for capturing images and processing data.
- Ultrasonic sensors for distance measurement to enhance object detection accuracy.
- OLED display for real-time feedback.
- Buzzer for alerts upon object detection.

2. Data Collection & Preprocessing:

- Capturing images using the ESP32-CAM and storing them locally or on a cloud platform.

- Labeling datasets to train machine learning models for specific object recognition tasks.

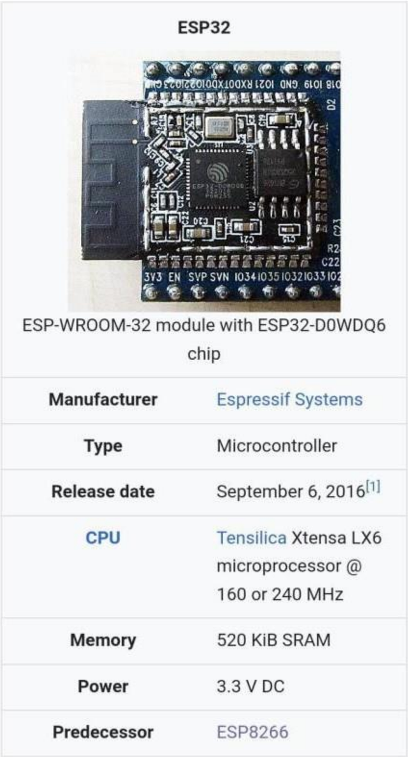
- Enhancing data quality by considering various lighting and environmental conditions.

3. Model Training & Deployment:

- Using platforms like Edge Impulse or TensorFlow Lite to train AI models.

- Optimizing models for ESP32's limited processing power.

- Converting trained models into a compatible format and deploying them on the ESP32 for real-time inference.



ESP32

ESP-WROOM-32 module with ESP32-D0WDQ6 chip

| | |
|---|---|
| **Manufacturer** | Espressif Systems |
| **Type** | Microcontroller |
| **Release date** | September 6, 2016[1] |
| **CPU** | Tensilica Xtensa LX6 microprocessor @ 160 or 240 MHz |
| **Memory** | 520 KiB SRAM |
| **Power** | 3.3 V DC |
| **Predecessor** | ESP8266 |

4. Integration & Testing:

- Programming the ESP32 to process sensor data and detect objects based on distance and vision.

- Setting up a Wi-Fi connection for remote monitoring and data transmission to a cloud platform.

- Testing under different conditions to evaluate accuracy, response time, and reliability.

5. Analysis & Optimization:

- Identifying performance bottlenecks and refining the model.

- Implementing techniques like deep sleep mode for power efficiency.

- Fine-tuning parameters to improve real-time object detection accuracy.

## 3.3 Deep Learning-based Object Recognition for Counting Car Components to Support Handling and Packing Processes in Automotive Supply Chains

### Abstract

Complex supply chains in the automotive industry face challenges from high product variety. Digital image processing, leveraging specific geometric and optical properties of parts, eliminates the need for external markers, making it ideal for handling diverse components directly. This paper presents a deep learning-based digital image processing method to classify and count automotive components in standardized transport bins using time-of-flight (TOF) depth sensors. Traditional watershed methods were adapted to depth data for training a CNN-based classification system. The proposed approach demonstrated reliable counting of automotive parts during packing, showcasing its potential to enhance supply chain transparency by supplementing auto-identification and sensor technologies.

### Methodology

The methodology proposed in the study integrates deep learning techniques with time-of-flight (TOF) depth sensors and conventional segmentation algorithms to achieve automated classification, counting, and alignment verification of automotive components within transport bins. The steps involved are as follows:

1 Data Acquisition:

T0F and intensity sensor (Microsoft Kinect v2) captures 3D geometric data and depth images from a fixed height (minimum 0.7 m above the bin). The components in the bin are arranged in a single layer to ensure visibility of all parts.

2 Pre-processing:

Depth and intensity data are captured across multiple frames (30 frames per sequence). Consolidated depth images are generated by averaging the depth data, calculating mean intensity, and determining the standard deviation to improve object recognition. A binary mask is applied to focus only on the bin's relevant area, excluding background noise.

3 Segmentation and Training Data Generation:

A watershed algorithm is adapted to segment components based on the depth data. Thresholding and distance transformations help identify individual objects even in densely packed arrangements. The segmented data is used to generate labelled training datasets, where labels indicate the position and type of objects. Errors in labels are manually corrected.

4 Deep Learning Model Selection and Training:

The study employs Convolutional Neural Networks (CNNs) for detection and classification, using two architectures:

Faster R-CNN (with ResNet101 and Inception v2 as base networks). Single Shot Multi Box Detector (SSD) (with Inception v2 and Mobile Net as base networks). to optimize performance. Classes defined for training include "large pump," "small pump," and "wrong" (misaligned or foreign objects).

5 Evaluation Metrics:

The performance of CNNs is evaluated on the remaining 20% of the dataset using mean Average Precision (MAP), which measures detection reliability. Specific detection metrics are also recorded for large pumps, small pumps, and misaligned objects.

6 Implementation Workflow:

Sensor data is pre-processed and segmented to create labelled training datasets. CNNs are trained to detect, classify, and count components in bins. The trained model is evaluated for its ability to detect and verify the number and alignment of objects. This method demonstrates high reliability in detecting and counting components, emphasizing its applicability for dense arrangements and supply chain operations.

## 3.4 industrial Automated conveyor belts for object counting

### Abstract:

The distance-based object counter is an automated system designed to track and count objects in a given space based on their distance from a sensor. This technology is used in various industries, including manufacturing, logistics, and retail, to enhance operational efficiency and accuracy. By leveraging sensors such as ultrasonic, infrared, or LIDAR, the system measures the distance to objects within its range, differentiating them from background noise and clutter. This paper explores the application of distance-based object counting techniques, discusses the integration of sensors with software algorithms, and evaluates the effectiveness of such systems in real-time object tracking.

## Methodology:

Sensor Selection and Setup: Various distance sensors are evaluated for their effectiveness in different environments. Ultrasonic sensors, LIDAR, and infrared sensors are selected based on the nature of the objects and the operational setting. The sensors are placed strategically in the operational area, ensuring that their range and coverage align with the size and positioning of the objects to be counted.

## 3.5 Analysis of obstacle detection with distance measuring using Arduino uno and ultrasonic sensor

## Abstract

This study presents the development and analysis of a distance measurement system using an Arduino Uno and an ultrasonic sensor. Traditional tools for distance measurement, such as tape measures, are often prone to human error, making automated systems more desirable. The proposed system uses ultrasonic waves at a frequency of 40 kHz to measure distances from 0.5 to 4 meters with a 1 cm accuracy. The device calculates the distance based on the time it takes for ultrasonic waves to reflect back from an object. Results are displayed on an LCD module, showcasing high accuracy and cost-effectiveness. The system has significant potential applications in obstacle detection, robotics, construction, parking assistance, and industrial automation, with future improvements targeting extended range and multi-dimensional measurements.

## Methodology

The methodology encompasses the following stages:

1. System Design and Hardware Setup

The system employs an Arduino Uno microcontroller, an ultrasonic sensor, and an LCD module for distance measurement and display. The ultrasonic sensor emits high-frequency sound waves, and the Arduino calculates the time taken for the echo to return, determining the distance. The components are connected using jumper wires, with detailed circuit diagrams guiding the setup.

2. Software Development

The Arduino IDE is used to program the microcontroller. The code processes the input signals from the sensor, calculates distances using time-of-flight data, and outputs results to the LCD display.

3. Experimental Setup

Objects were placed at varying distances (5 cm to 50 cm), and measurements were recorded.The setup's accuracy was evaluated by comparing the measured distances with actual physical measurements.

4. Data Collection and Analysis

Results were compiled into tables, noting the actual distance, measured distance, and percentage error. Graphical representations and error analysis were conducted to assess the sensor's performance.

5. Validation and Improvements

The system’s accuracy was validated through repeated trials, and its efficiency was tested across different ranges. Potential enhancements, such as environmental adjustments and extended range capabilities, were identified for future work. This systematic approach ensures the device's reliability and feasibility for real-world applications.

## 3.6 Object Detection & Count in Image

## Abstract

The project focuses on implementing object detection and counting using machine learning algorithms. Different methods such as OpenCV, TensorFlow, and YOLO (You Only Look Once) were explored and applied to detect and count objects, particularly fruits on trees. The study achieved high accuracy, approximately 97%, in detecting and counting objects within images. By leveraging YOLO, the framework demonstrated efficiency in both speed and accuracy, proving suitable for applications like fruit yield estimation. The project highlighted the advantages of neural networks, especially CNN and YOLO, while also addressing challenges like computational demands and limitations in detecting smaller or overlapping objects.

## Methodology

1. Initial Setup:

Installed tools like Anaconda, TensorFlow, and Jupyter Notebook. Used Python libraries such as TensorFlow, SciPy, and Keras for object detection.

2. Dataset Preparation:

Downloaded datasets such as Cars and Cats & Dogs. Trained models using approximately 8,000 images for cars and 6,000 images for cats and dogs.

3. Implementation Approaches:

CNN Approach:

Convolutional Neural Networks were used for initial object detection and classification. Training and testing were performed to classify objects like cars, cats, and dogs.

YOLO Framework:

Applied YOLO for real-time detection. The image was divided into multiple regions to predict bounding boxes and associated probabilities.

4. Advanced Techniques:

Blob detection was employed for identifying objects by grouping pixels with shared properties like grayscale values. Implemented steps such as thresholding, grouping, merging, and calculating centers and radii for objects.

5. Fruit Detection Example:

Applied YOLO to detect apples on a tree from an image dataset. The model identified and counted apples, providing precise results showcased in visual outputs.

6. Analysis:

YOLO outperformed CNN in terms of speed and real-time capability. Challenges with YOLO included difficulty in detecting smaller or overlapping objects and dependence on sufficient lighting.

## 3.7 Distance Measurement with the Help of Ultrasonic Sensor

## Abstract

This study explores the principles and applications of ultrasonic sensors for distance measurement. Ultrasonic sensors utilize high-frequency sound waves to detect objects and measure distances with precision, making them valuable in industrial automation, vehicle electronics, and water level monitoring. These sensors operate by transmitting ultrasonic waves and analyzing the time delay of reflected echoes to calculate distance. Their performance relies on piezoelectric ceramics and advanced signal processing technologies, ensuring high accuracy and adaptability in varied environments. This paper focuses on the design, operation, and practical applications of ultrasonic sensors, highlighting their role in enhancing automation and detection systems.

## Methodology

1. Principle of Operation:

Ultrasonic sensors transmit high-frequency sound waves through a piezoelectric transducer. Reflected echoes from obstacles are captured, and the time delay is analyzed to compute distance using the speed of sound in the medium.

2. Sensor Configuration:

The study involves the use of open-structure and enclosed ultrasonic sensors to suit diverse environmental conditions. High-frequency ultrasonic sensors, optimized for industrial use, were tested for precision within a 2 cm to 400 cm range.

3. Experimental Setup:

A sensitivity measuring circuit was implemented to evaluate the response of sensors under controlled conditions. Sensors were calibrated using known distances to establish accuracy, with outputs recorded as voltage pulses proportional to the distance.

4. Data Analysis:

The performance of the sensors was assessed based on parameters like sound pressure, sensitivity, and accuracy (±0.3 cm). The results were analyzed to determine the sensors' efficacy across varied applications such as automatic doors, intrusion alarms, and industrial robots.

5. Applications:

Testing was extended to real-world scenarios like water level monitoring, obstacle detection in vehicles, and automated systems in factories. This methodology ensures a comprehensive evaluation of ultrasonic sensors, emphasizing their versatility and reliability in modern engineering applications.

## 3.8 HOME AUTOMATION USING ARDUINO AND IoT

## Abstract

This paper presents a cost-effective, flexible, and reliable home automation system using Arduino and IoT. The system leverages an Arduino microcontroller, specifically the ATmega328 IC, and a Wi-Fi module (ESP8266) to enable authorized users to remotely control appliances such as lights, fans, and TVs through a smartphone application or web browser. The system operates independently of servers and offers added security. Experimental testing demonstrated its effectiveness in various environments, showcasing its potential to automate modern homes by integrating IoT technology for seamless control and monitoring.

## Methodology

1. System Components:

Hardware:
Arduino UNO microcontroller with ATmega328 IC.
ESP8266 Wi-Fi module for internet connectivity.
Relays for controlling electrical loads.
16x2 LCD display to show system status.
Software:
Arduino IDE for programming the microcontroller.
Ubidots IoT platform for creating a dashboard to control devices.

2. Implementation Steps:

The Arduino UNO board is connected to the ESP8266 Wi-Fi module for receiving commands over the internet. Relays are used to control the state (ON/OFF) of appliances based on user inputs.
The 16x2 LCD displays the status of the connected loads.

A smartphone application or web browser is used to send commands via the Ubidots platform.

3. System Workflow:

The user logs in to the application and accesses the IoT dashboard.
The Arduino reads the switch statuses from the Ubidots platform.
Based on the switch inputs, the Arduino toggles the relays to control the appliances.
Feedback on the current state of the appliances is displayed on the LCD in real time.

4. Testing and Evaluation:

The system was tested under different load conditions and in multiple environments. Results showed reliable operation with a delay of 2-3 seconds for switching appliances on and off.

5. Advantages and Future Enhancements:

Reduced wiring costs and power consumption.
Platform independence, allowing access from any web browser.
Proposed future improvements include reducing switching delays, integrating speech recognition, and expanding Wi-Fi range.

# 4. OBJECTIVE

## 4.1 EXISTING SYSTEM

In traditional industrial automation setups, object counting is a fundamental task often implemented using basic sensor-based systems. The most commonly used technologies include Infrared (IR) sensors, photoelectric sensors, and ultrasonic sensors. These sensors are typically placed along conveyor belts, where they detect the presence of objects as they pass through a specific point. When an object is detected, the sensor sends a signal to a microcontroller or PLC (Programmable Logic Controller), which then increments a counter. This method has been widely used in manufacturing and packaging industries due to its simplicity and low cost. Infrared and photoelectric sensors work based on the interruption of a light beam. When an object crosses the beam, it triggers a signal. While these sensors are effective for basic counting tasks, they often suffer from limitations such as short detection range, sensitivity to environmental factors (e.g., dust, ambient light, and temperature), and difficulty in detecting transparent or reflective objects. Ultrasonic sensors, which measure the distance between the sensor and the object using sound waves, offer slightly better accuracy and non-contact measurement. However, they too face challenges such as signal interference, limited resolution, and difficulty in distinguishing closely spaced objects. Moreover, most of these traditional systems are standalone, meaning they lack real-time monitoring and remote access capabilities. Data is usually stored locally and often requires manual intervention for analysis and reporting. These systems are not integrated with cloud platforms or smart factory networks, making them less suitable for modern Industry 4.0 environments. Additionally, traditional systems do not support advanced features such as predictive maintenance, anomaly detection, or data-driven decision-making. As industries aim to improve productivity and efficiency, there is a growing need to upgrade these conventional systems with IoT-enabled solutions that offer better accuracy, connectivity, and real-time analytics.

## 4.2 PROBLEM DEFINITION

We have taken seven papers for the review.

## 4.3 DISADVANTAGE OF EXISTING SYSTEM

1. Inaccuracy in Counting – Errors due to sensor misalignment, overlapping objects, or fast-moving items.

2. Environmental Interference – Dust, vibrations, temperature changes, and lighting conditions can affect sensor performance.

3. Limited Object Detection – Difficulty in detecting transparent, reflective, or very small objects accurately.

4. Connectivity Issues – IoT-based systems may face network failures, leading to data loss or delays.

5. High Power Consumption – Some sensors and communication modules consume excessive power, reducing efficiency.

6. Data Latency – Delays in transmitting or processing data can impact real-time monitoring.

7. Complex Maintenance – Regular calibration and sensor cleaning are needed to maintain accuracy.

8. Integration Challenges – Compatibility issues with existing industrial systems like PLCs and SCADA.

9. Security Risks – IoT devices can be vulnerable to cyber threats, risking data manipulation or breaches.

10. High Initial Cost – Advanced sensors and IoT infrastructure can be expensive to implement.

# 5. PROPOSED SYSTEM

## 5.1 INTRODUCTION

The proposed system will automate the process of counting objects moving through a designated area in an industrial setting, such as a warehouse, assembly line, or manufacturing plant. The system uses an **ultrasonic sensor** to measure the distance of objects, an **ESP32** for data processing, a **cloud platform** for remote monitoring, and **real-time alerts** through an **OLED display** and a **buzzer** for operational feedback.

## 5.2 SYSTEM WORKFLOW:

1. Object Detection:

The **ultrasonic sensor** continuously scans the area to detect objects. It emits sound waves and measures the time it takes for the sound to reflect off an object. Based on this measurement, it calculates the distance to the object. When an object enters the detection range, the sensor detects a change in distance, indicating the presence of an object.

2. Data Processing:

The **ESP32 microcontroller** receives the data from the ultrasonic sensor and processes it to determine if an object has passed through the detection zone. The ESP32 maintains a real-time count of objects detected. This count is updated whenever an object is detected and passes the sensor's threshold.

3. Local Feedback (Display & Alerts):

The **OLED display** shows the current count of objects detected by the sensor. This provides immediate feedback to operators working in the area. If the object count exceeds a predefined threshold (set by the operator or system configuration), the **buzzer** is activated to alert the operator of this event.

4. Cloud Data Sync:

The **ESP32** is connected to a Wi-Fi network, allowing it to send data to a **cloud platform** for remote monitoring and analysis. This could be done via MQTT or HTTP protocols, depending on your cloud setup. The cloud platform stores the object count data in real time and provides analytics to identify patterns (e.g., peak object flow periods) and track long-term performance. Alerts, reports, and historical data can be accessed from anywhere through the cloud interface.

## 5.3 SYSTEM FEATURES:

1. Real-time Object Counting:

The system continuously tracks and updates the object count based on sensor data. Operators can monitor the count in real time via the OLED display.

2. Automated Alerts:

When the object count surpasses a certain threshold, the buzzer provides an audible alert. This can prevent bottlenecks or issues on the production line or warehouse.

3. Remote Monitoring:

Cloud integration allows for **remote access** to the object count data, trends, and analytics. Supervisors or managers can monitor the system from anywhere using a computer or mobile device.

4. Scalability:

The system can be expanded to multiple locations or sections within a factory, allowing for easy integration of additional sensors or ESP32 units.

5. Data Analytics:

Cloud-based analytics provide insights into the flow of objects, allowing operators and managers to make informed decisions about production scheduling, inventory management, or maintenance.

## 5.4 SYSTEM BENEFITS:

1. Increased Accuracy:

The use of the ultrasonic sensor allows for highly accurate object counting, reducing human error and manual counting efforts.

2. Improved Operational Efficiency:

The automated system streamlines object counting, eliminating the need for manual labor and improving overall workflow in industrial processes.

3. Real-Time Monitoring:

Operators are kept informed of object counts and any potential issues (via the display and buzzer), enabling faster response times and better decision-making.

4. Data-Driven Insights:

The integration with the cloud platform allows for long-term data storage, performance tracking, and trend analysis, which helps optimize operations over time.

5. Cost Efficiency:

By minimizing errors, automating processes, and allowing for proactive issue resolution (via alerts), the system can reduce downtime and improve overall operational cost efficiency.

## 5.5 FUTURE ENHANCEMENTS:

AI Integration: Incorporating machine learning algorithms to detect object types, sizes, or even detect anomalies.

Multi-Sensor Integration: Adding more ultrasonic sensors or combining different sensor types (e.g., cameras or infrared sensors) to improve object detection accuracy or cover larger areas.

Energy Optimization: Implementing sleep modes or energy-saving features in the ESP32 to reduce power consumption in industrial settings.

Advanced Analytics: Using cloud AI/ML to predict trends, maintenance needs, and optimize operations further.

# 6. SYSTEM DESIGN

## 6.1 BLOCK DIAGRAM

A block diagram is a specialized, high-level flowchart used in engineering. It is used to design new systems or to describe and improve existing ones, It’s structure provides a high-level overview of major system components, key process participants, and important working relationships. Block diagrams are made similar to flowcharts. You will want to create blocks, often represented by rectangular shapes that represent important points of interest in the system from input to output. Lines connecting the blocks will show the relationship between these components.

### SYMBOLS USED IN BLOCK DIAGRAMS

Block diagrams use very basic geometric shapes: boxes and circles, The principal parts and functions are represented by blocks connected by straight and segmented lines illustrating relationships. When block diagrams are used in electrical engineering, the arrows connecting components represent the direction of signal flow through the system. Whatever any specific block represents should be written on the inside of that block. A block diagram can also be drawn in increasing detail if analysis requires it. Feel free to add as little or as much detail as you want using more specific electrical schematic symbols.

- Identify the system. Determine the system to be illustrated. Define components, inputs, and outputs.
- Create and label the diagram. Add a symbol for each component of the system, connecting them with arrows to indicate flow. Also, label each block so that it is easily identified.

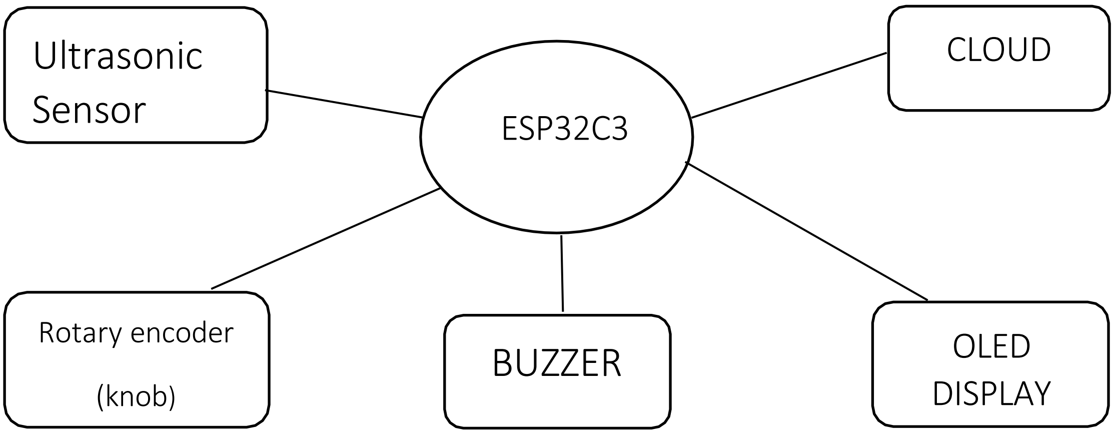


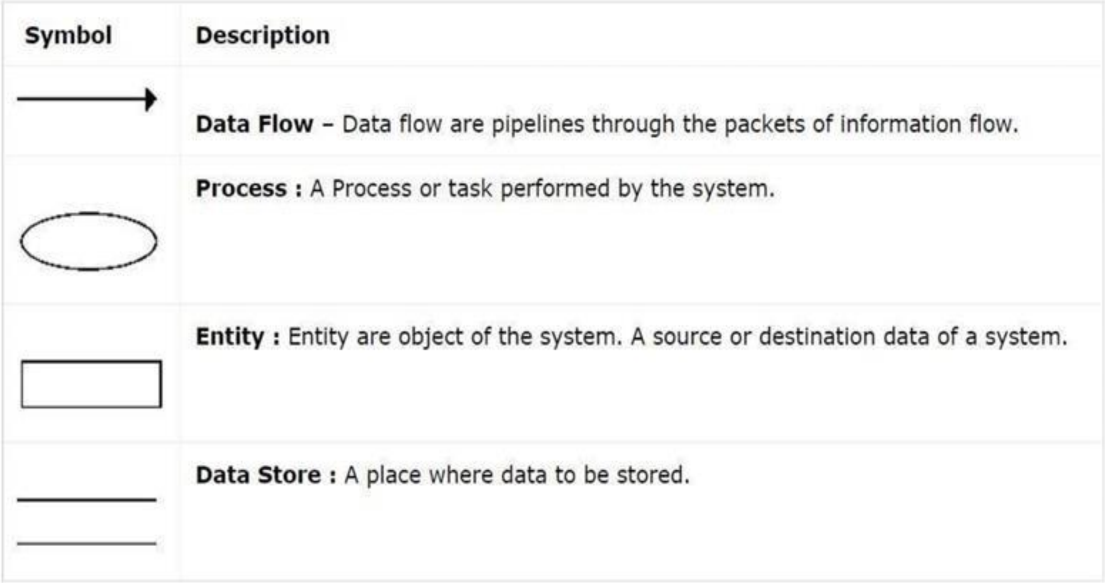
**Fig 1: Block Diagram**

## 6.2 DATAFLOW DIAGRAM

A Data Flow Diagram (DFD) is a specialized diagram used in system analysis and design to visualize how data flows through a system. It provides a high-level representation of processes, data inputs, outputs, and storage within the system. DFDs help in designing new systems or analyzing and improving existing ones by illustrating how data moves between components. It helps us understand how data is captured, processed, and utilized within the system.

### SYMBOLS USED IN DATAFLOW DIAGRAMS

DFDs use standardized symbols to represent different elements of a system: Processes: Represented by circles or rectangles with rounded edges, indicating actions perform on data. Data Stores: Represented by open-ended rectangles, showing where data is stored. External Entities: Represented by squares, indicating sources or destinations of data. Data Flows: Represented by arrows, showing the movement of data between components.



| Symbol | Description |
|---|---|
| ⟶ | **Data Flow –** Data flow are pipelines through the packets of information flow. |
| ⬭ | **Process :** A Process or task performed by the system. |
| ▭ | **Entity :** Entity are object of the system. A source or destination data of a system. |
| ═ | **Data Store :** A place where data to be stored. |

### Steps to Create a Data Flow Diagram

1. Identify the System: Determine the scope of the system, including key processes, inputs, outputs, and data storage points.

2. Create and Label the Diagram: Define processes, data stores, and external entities, and use arrows to indicate data flow between them. Label each component clearly for easy identification.

3. Detail as Needed: A DFD can be drawn at different levels of complexity (e.g., Level 0, Level 1, Level 2) to provide varying degrees of system detail.

**LEVEL 0:**

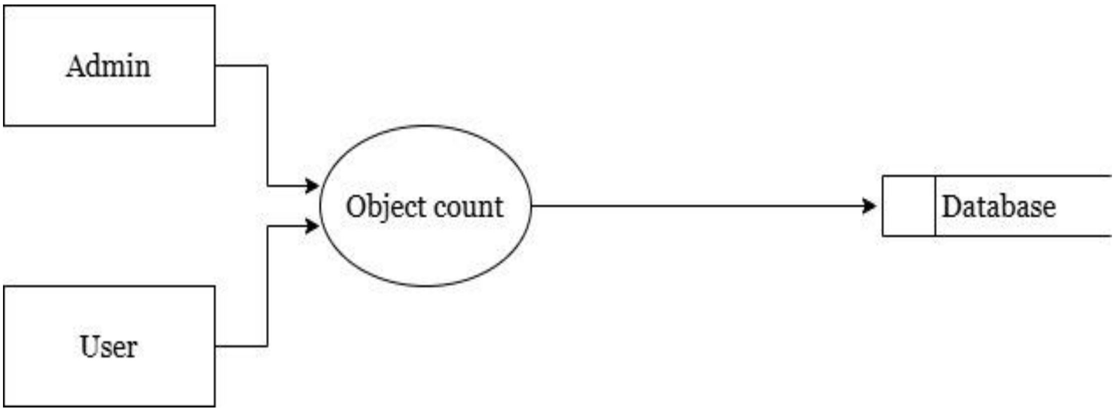


**Fig 2: LEVEL 0 DFD**

**LEVEL 1:**

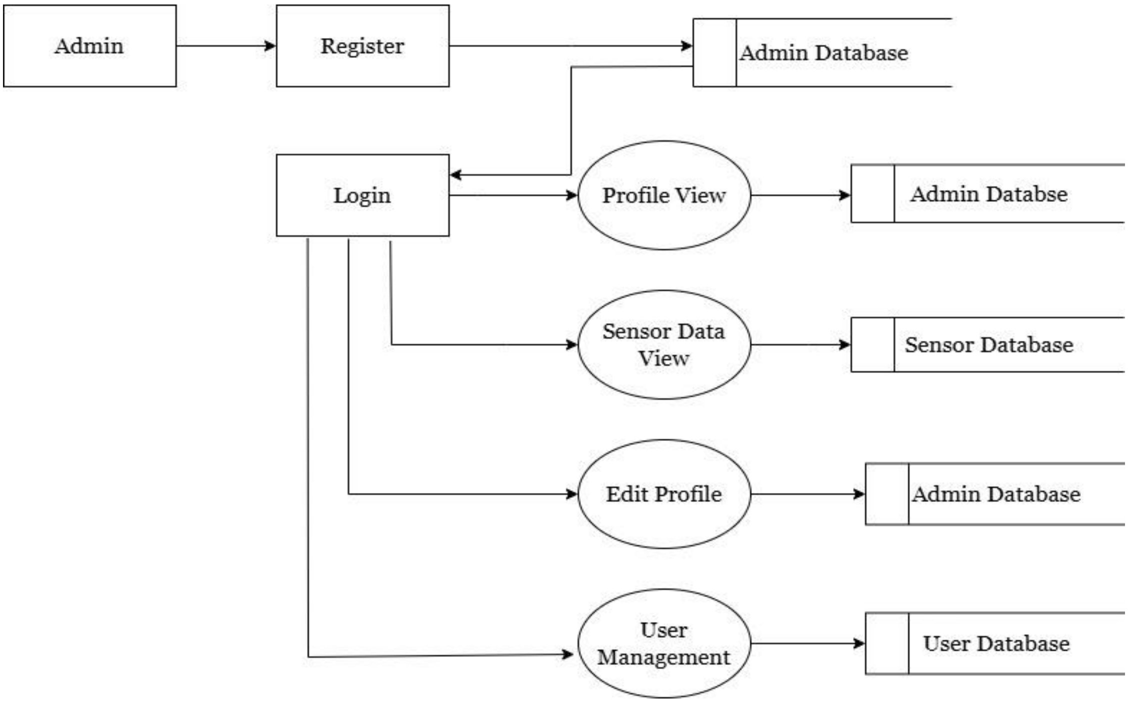


**Fig 3: LEVEL 1 DFD**

**LEVEL 2:**

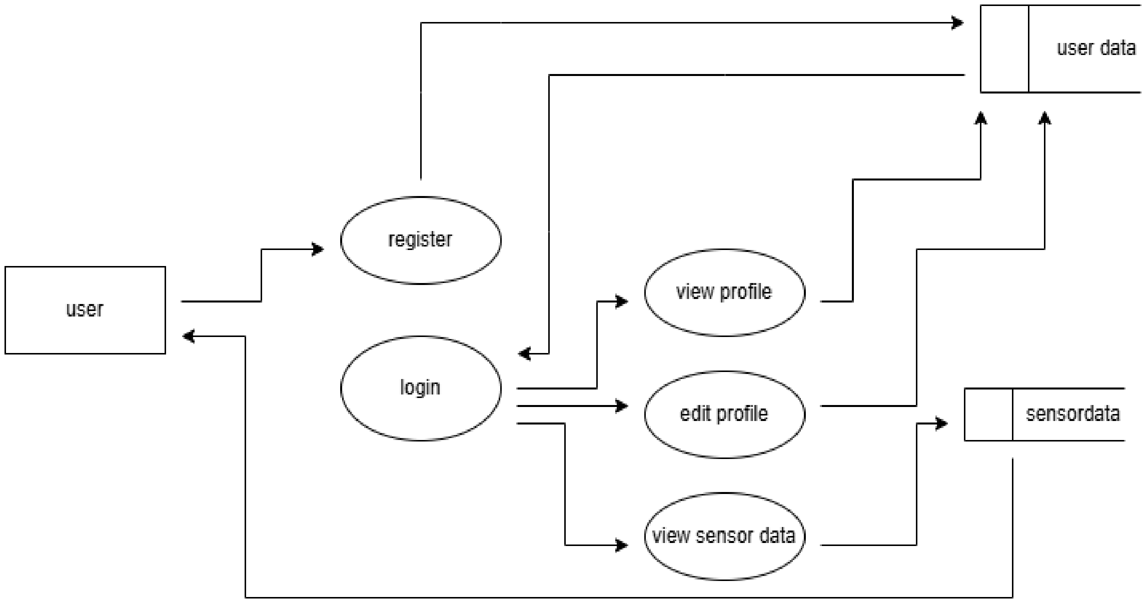


**Fig 4: LEVEL 2 DFD**

# 7. HARDWARE REQUIREMENT SPECIFICATION

Hardware is one of the most crucial requirements for building a Distance Based Object Counting Industrial Automation. This section summarizes the hardware requirements for the IoT components involved. Specifically, it outlines the necessary hardware specifications for building a Distance Based Object Counting in Industrial Automation.

## 7.1 ultrasonic sensor

- Detects objects based on their distance

## 7.2 ESP32C3 microcontroller

- Handles data processing and communication bet ween the sensor, display, and cloud.

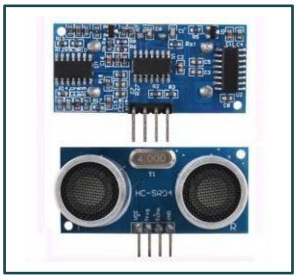


**Fig 5: Ultrasonic Sensor**

- Acts as the main controller and communication device.
- Connects to the local server over Wi-Fi.
- Processes sensor input and counts objects.
- Interfaces with the rotary encoder for adjusting system settings.
- Controls the OLED display, LED, and buzzer for local feedback.

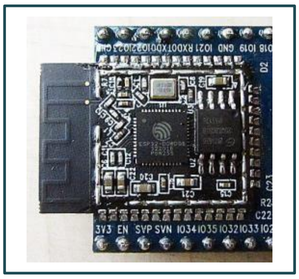


**Fig6:ESP32C3**

- Sends data to the cloud server/website for remote monitoring.

## 7.3 OLED DISPLAY

- Provides real-time feedback by displaying the count.

## 7.4 BUZZER&LED

- Alerts operators of specific conditions. (e.g., crossing a threshold)

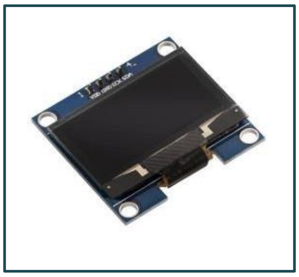


**Fig 7: OLED Display**

### 7.5 CLOUD

**Fig 8: Buzzer**

- The system integrates with a cloud platform to store and analyze data for monitoring and reporting, ensuring scalability and reliability for industrial environments.

### 7.6 Rotary encoder

**Fig 9: Rotary encoder**

- Used for adjusting the object detection threshold. (e.g., setting the minimum distance for detection).
- Can be used to manually reset the object counter.
- Helps fine-tune system parameters based on factory conditions.

**Tools Required:**

The selection of hardware configuration is a very important task related to the Software development. The processor should be powerful to handle entire operations.

❖Processor: Intel® Core™ i3

❖RAM: 4 GB RAM minimum, 8GB recommended

❖ Hard Disk: 150 GB

❖Keyboard: Standard

❖Mouse: Normal

❖Monitor: Plug and play monitor

# 8. SOFTWARE REQUIREMENT SPECIFICATION

One of the most difficult tasks is selecting software, once the system requirement is find out then we have to determine whether a particular software package fits for those system requirements. This section summarizes the application requirement.

- Python Web Application
- ESP32C3 Firmware

The major element in building a system is the selection of compactable software. software requirement of the system on which the project was developed is as follows:

- Operating System: Microsoft windows
- Front End: HTML, CSS, JavaScript
- Back End: python, SQLite3
- Frame work: Django
- IDE: VS code, Arduino IDE
- Web Browser: Google Chrome

## 8.1 PYTHON WEB APPLICATION

The Python web application serves as the central hub of the system, orchestrating the flow of real-time object count data. Leveraging Python's robust web frameworks, such as Flask or Django, it efficiently receives distance-based object detection data transmitted from the ESP32-C3 microcontroller via Wi-Fi. The application then processes this raw data and presents it through an intuitive and visually appealing web interface, allowing users to monitor industrial automation processes in real time. Python's advantages, including its rapid development cycle, extensive library ecosystem, and cross-platform compatibility, make it an ideal choice for handling real-time data processing and web-based visualization. The simplicity and readability of Python enhance the system's efficiency, enabling seamless monitoring and control of the object counting process. Additionally, Python's scalability ensures that the application can accommodate future enhancements and increasing data loads, guaranteeing long-term reliability for industrial automation.

## 8.2 ESP32C3 FIREWARE

The ESP32-C3 firmware serves as the embedded intelligence within the IoT-powered industrial automation system, executing critical real-time functions. It manages the acquisition of distance data from sensors such as ultrasonic or LiDAR, ensuring accurate object detection. Based on these readings and commands received from the Python web application, the firmware processes object counts and transmits the data to the server for real-time monitoring.

Additionally, it establishes and maintains a stable Wi-Fi connection, enabling seamless communication with the web server. The firmware is responsible for regularly sending object count data, ensuring continuous tracking, logging, and analysis. As the core of the system, this embedded software ensures reliable and autonomous object counting, contributing to efficient industrial automation.

## 8.3 FRONT-END

### HTML:

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

### CSS:

CSS, or Cascading Style Sheets, is a language used for describing the presentation of a document written in HTML or XML, including colors, layout, and fonts. It enables web developers to control the appearance and layout of multiple web pages simultaneously, ensuring consistency and ease of maintenance. CSS works by selecting HTML elements and applying styling rules to them, allowing for precise control over design elements such as spacing, alignment, and responsiveness. It plays a crucial role in creating visually appealing and user-friendly websites and is constantly evolving to meet the demands of modern web design.

### JAVASCRIPT:

JavaScript is a versatile and dynamic programming language primarily used for building interactive and dynamic web applications. As a client-side scripting language, JavaScript runs directly in web browsers, enabling developers to enhance the functionality and interactivity of web pages. Its lightweight and interpreted nature allows for quick development iterations and immediate feedback during the coding process. JavaScript's flexibility enables developers to manipulate HTML and CSS, handle user events, and dynamically update content on web pages, creating a more engaging and responsive user experience.

## 8.4 BACK-END

### PYTHON:

Python serves as the cornerstone of our IoT-powered industrial automation system, providing a versatile and robust platform for its core functionalities. Its role extends across multiple critical areas, starting with the development of the web application's central logic. Python's clear syntax and extensive libraries facilitate efficient data processing, implementation of object counting algorithms, and secure user authentication.

Furthermore, Python enables seamless interaction with the PostgreSQL/MySQL database, ensuring reliable data storage and retrieval of object count records. Crucially, it acts as the communication bridge between the web application and the ESP32-C3 microcontroller, enabling the exchange of real-time distance sensor data and control commands. This comprehensive utilization of Python highlights its critical importance in creating a responsive, efficient, and reliable industrial automation system for distance-based object counting.

### SQLite3:

SQLite3 plays a pivotal role in the IoT-powered industrial automation system as the embedded relational database, providing a lightweight and efficient solution for persistent data storage. It securely houses historical object count records, enabling users to track trends and analyze production patterns over time. Additionally, SQLite3 stores critical system settings, such as sensor calibration data and threshold configurations, ensuring that the system operates optimally based on industrial requirements. It also manages user authentication and access control, allowing for secure login and personalized user experiences. This seamless integration of SQLite3 ensures that the system maintains a reliable, structured, and easily accessible repository of essential data, contributing to its efficiency and usability in industrial automation.

### PHP:

The back-end of the IoT-powered industrial automation system for distance-based object counting is developed using PHP, providing a robust and efficient platform for managing data processing, system logic, and communication with the frontend interface. PHP handles the collection and storage of real-time sensor data transmitted from the ESP32-C3 microcontroller, processes the object count values, and updates the SQLite3 database accordingly. It also manages critical operations such as user authentication, threshold configuration, alert generation, and data retrieval for analytics and visualization. The PHP-based backend ensures smooth interaction between hardware and software components, offering a scalable, secure, and maintainable environment for industrial automation tasks. Its simplicity and wide compatibility make PHP an ideal choice for developing a lightweight yet powerful server-side system in real-time object counting applications.

## 8.5 FRAMEWORK

### Django:

Django, a powerful high-level Python web framework, forms the structural backbone of our IoT-powered industrial automation system for distance-based object counting. Its comprehensive features streamline development, ensuring a robust, scalable, and efficient platform for real-time data processing and monitoring. Django’s Object-Relational Mapper (ORM) simplifies database interactions, abstracting complex SQL queries and enabling seamless storage and retrieval of object count data. Its URL routing and view handling efficiently manage web requests and responses, directing traffic to the appropriate application logic for real-time object monitoring. Additionally, Django’s built-in authentication and authorization mechanisms provide essential security, safeguarding user data and system access. The templating engine facilitates dynamic rendering of HTML pages, ensuring a user-friendly and interactive web interface for monitoring industrial processes. By leveraging these features, Django empowers us to build a well-structured, secure, and responsive industrial automation system for accurate object counting and data-driven decision-making.

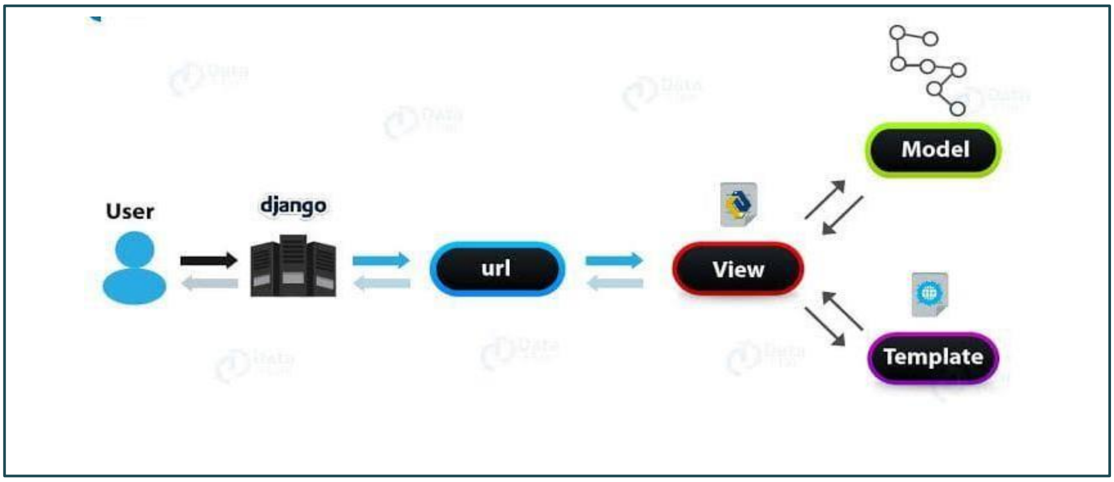


Fig 10: Django MVC Architecture

## 8.6 IDE

### VS Code:

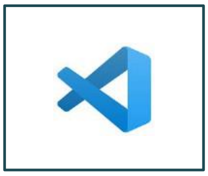


Fig11: VS Code

VS Code, a highly versatile and widely adopted code editor, served as our primary development environment throughout the IoT-powered industrial automation system for distance-based object counting. Its exceptional support for Python, HTML, CSS, JavaScript, and other essential web development languages significantly streamlined the coding process. Features such as intelligent code completion and syntax highlighting enhanced code readability and minimized errors, while the robust debugging tools facilitated efficient troubleshooting of both the ESP32-C3 firmware and the Python web application. Seamless Git integration ensured effective version control, enabling collaborative development and easy management of code changes. Furthermore, the vast library of extensions allowed us to tailor VS Code to our specific needs, enhancing functionality and productivity. Its user-friendly interface and comprehensive features made VS Code an indispensable tool for developing and maintaining our industrial automation system with precision and efficiency.

### FUTURES OF VS Code:

Cross-platform support for Windows, macOS, and Linux. Integrated terminal for running shell commands within the editor. Built-in version control with Git integration. Extensive language support with syntax highlighting and code formatting. Live Share feature for collaborative editing and debugging. Task running and build automation with tasks. json and launch. json configuration. Accessibility features such as screen reader support and high contrast themes. Regular updates and active community support. Highly customizable with themes and settings. Fast and responsive performance, even with large codebases.

## Arduino IDE:

**Fig 12: Arduino**

The Arduino IDE played a crucial role in programming the ESP32-C3 microcontroller, the hardware core of our IoT-powered industrial automation system for distance-based object counting. Utilizing the Arduino language (a variant of C/C++), this IDE provided a straightforward and accessible platform for developing the microcontroller's firmware. Its user-friendly interface simplified the process of writing, compiling, and uploading code, enabling rapid prototyping and iteration. Additionally, the Arduino IDE provided access to a rich collection of pre-built libraries, facilitating seamless interaction with ultrasonic, LiDAR, or infrared distance sensors used for object detection. This combination of ease of use and hardware integration made the Arduino IDE an essential tool for developing the embedded control logic of our industrial automation system, ensuring efficient and accurate object counting in real-time production environments.

## 8.7 TABLES USED

### TABLES:

In the IoT-powered industrial automation system for distance-based object counting, tables serve as fundamental components of the database, responsible for storing, organizing, and managing all essential data. A table is a structured set of fields (columns), where each field represents a specific attribute of the system. Understanding how to create, modify, and maintain these tables is crucial for ensuring data consistency and reliability. This includes defining primary keys, adding or removing columns, and establishing relationships between data entities.

**Database diagrams** are also used to represent the system’s architecture in a graphical format, helping developers to visualize data flow, define relationships between tables, and manage database objects more effectively. These diagrams play a key role in planning and maintaining a scalable and efficient data model.

### Sensor data reading:

This table is designed to store real-time sensor readings received from the ESP32-C3 microcontroller, including distance values, object count, and system status. The fields allow tracking of sensor activity based on timestamp and location, enabling efficient monitoring and analysis of industrial workflows. Additional fields such as status and alert triggered help in identifying abnormal conditions and initiating timely responses.

| Field | Datatype | Unique | Default value |
|---|---|---|---|
| Id | INT | YES | - |
| Date & time | TIME_STAMP | - | CURRENT TIME |
| Sensor | STRING | - | - |
| Location | VARCHAR | - | - |
| Set count | VARCHAR | - | NULL |
| Set time | VARCHAR | - | NULL |
| Object count | VARCHAR | - | NULL |
| States | STRING | - | |

**Table 1: sensor data**

# 9. TESTING

The testing phase was essential to validate the reliability, accuracy, performance, security, and usability of the IoT-powered Industrial Automation system for Distance-Based Object Counting. A comprehensive approach involving unit testing, integration testing, system testing, acceptance testing, and regression testing was implemented. The system underwent thorough security assessments to identify and mitigate vulnerabilities, while usability testing ensured a smooth user experience across different platforms and devices. Post-deployment updates were followed by regression testing to ensure continued system stability and performance. This structured testing strategy helped ensure an efficient, robust, and user-friendly object counting solution in an industrial automation context.

## 9.1 System Testing:

System testing involved evaluating the end-to-end functionality of the object counting system under real-world conditions. All modules—including sensor data acquisition, object detection logic, counting algorithms, and the web-based monitoring interface—were tested to ensure smooth operation. Key functionalities such as real-time distance measurement, count updates, threshold alerts, and data logging were validated across user roles and system components. Integration between the ESP32-C3 firmware and the Django-based web application was also tested to ensure reliable data flow and operational coherence.

## 9.2 Unit Testing:

Individual components were subjected to unit testing to ensure each function performed as expected. Critical modules, such as distance calculation algorithms, object detection triggers, ESP32-C3 sensor readings, and web interface data rendering, were rigorously tested. Communication between hardware and software was verified at the code level to ensure correct data transmission, edge-case handling, and proper error reporting.

## 9.3 Acceptance Testing:

Acceptance testing was conducted to verify that the system met predefined functional and performance criteria. Real-time object counting, threshold-based notifications, system accuracy, and response time were tested to ensure compliance with industrial automation requirements. The integration of ESP32-C3 and the Django web platform was validated for seamless user interaction, ensuring the system delivered expected outcomes under practical usage scenarios.

## 9.4 Integration Testing:

Integration testing focused on verifying the interaction between hardware and software modules. Real-time sensor data from the ESP32-C3 was monitored to confirm successful integration with the web-based dashboard. Features like data visualization, count logs, threshold alerts, and system control commands were tested for consistent communication and synchronization between components.

## 9.5 Security Testing:

Security testing aimed to safeguard system data and ensure secure communication. JWT-based user authentication and access control mechanisms were tested against unauthorized access. Penetration testing simulated intrusion scenarios to uncover vulnerabilities. Communication between the ESP32-C3 and the Django backend was also validated for data integrity and encryption compliance.

## 9.6 Usability Testing:

The usability of the web interface was tested to ensure that operators and administrators could easily interact with the system. User feedback was gathered regarding interface clarity, responsiveness, navigation flow, and dashboard readability. Adjustments were made to improve user experience, visual hierarchy, and control accessibility, ensuring an intuitive and efficient system for industrial environments.

## 9.7 Regression Testing:

Regression testing ensured that newly implemented features or updates did not compromise existing functionality. After each code modification, tests were run to verify continued performance of object detection, data logging, dashboard updates, and alert mechanisms. This maintained overall system reliability and avoided disruptions in critical automation workflows.

## 9.8 Error Handling Testing:

Robust error handling was tested by simulating invalid input scenarios and sensor anomalies. The system's ability to display relevant error messages, validate inputs, and maintain functional stability under unexpected conditions was assessed. Edge cases, such as extreme distances, sensor failures, or incorrect data formats, were tested to ensure resilience and consistent system behavior.

## 9.9 Compatibility Testing:

The web interface was tested across various browsers (Chrome, Firefox, Safari, Edge) and devices (PCs, tablets, smartphones) to ensure cross-platform compatibility. The responsiveness and consistency of the UI were evaluated to deliver a smooth user experience, regardless of screen size or operating system, making the system accessible and reliable for diverse user environments.

# 10. CONCLUSION

In conclusion, our team has successfully designed and implemented an IoT-powered Industrial Automation system for Distance-Based Object Counting, focusing on enhancing operational efficiency and automation in industrial environments. This system integrates ESP32-C3 microcontroller-based sensors with a Django-powered web application to provide accurate object detection, real-time distance measurement, and automated object counting capabilities. The primary objective was to streamline the object counting process in industrial settings, reducing manual errors and optimizing productivity. The system ensures reliable real-time data acquisition, efficient data transmission, and intuitive user interface features, allowing operators to monitor object flow, set distance thresholds, and receive automated alerts when predefined conditions are met. The use of JWT-based secure authentication further ensures that only authorized personnel can access the monitoring dashboard and perform administrative operations. Throughout the development, a modular and scalable architecture was followed. The web interface supports features such as live count display, sensor status visualization, threshold configuration, and automated logging of count data, making it suitable for a wide range of industrial applications. The integration of ESP32-C3 firmware with the Django backend ensures seamless communication between hardware and software components, promoting a robust and real-time automation ecosystem. Our development process leveraged tools like VS Code as the primary development environment, supporting efficient code collaboration and testing. By combining IoT technology with web-based automation, this project demonstrates a significant step forward in achieving intelligent, automated object counting within industrial setups. Looking ahead, there are multiple opportunities to further enhance the system. Improving mobile responsiveness will expand accessibility across devices for remote monitoring. Extending the system's capabilities to include production analytics, conveyor control integration, and predictive maintenance features can offer deeper insights and increased automation efficiency. Future improvements may also include batch configuration options, dashboard analytics visualization, and AI-driven anomaly detection to refine operational workflows. Incorporating interactive and dynamic web elements can also improve user experience and engagement. By continuously evolving the system through user feedback and technological advancements, this IoT-powered solution can serve as a scalable, adaptable, and indispensable tool in modern industrial automation.

# 11. FUTURE WORKS

In the forthcoming development phases, the project will focus on enhancing both the functionality and user experience of the IoT-powered Distance-Based Object Counting system. A primary objective is to introduce mobile responsiveness, enabling users to seamlessly access the system across various devices such as smartphones, tablets, and desktops. This will significantly improve accessibility for on-site personnel and remote operators. Further enhancements include the integration of advanced analytics and reporting modules, allowing administrators and supervisors to gain deeper insights into object flow patterns, operational efficiency, and production trends. This data-driven approach will support informed decision-making and performance optimization within industrial environments. The system will also aim to provide greater administrative flexibility, allowing bulk configuration options for sensors, thresholds, and user management. Features such as batch addition of sensors, automated system calibration, and enhanced user role management will be introduced to streamline system scalability and ease of operation. To improve communication and responsiveness, a real-time notification and alert system will be integrated. This feature will inform users of critical system events such as sensor faults, threshold breaches, or count anomalies, ensuring timely intervention and reducing operational downtime. In terms of user experience, visually appealing dashboards and cinematic UI enhancements will be implemented to make the interface more engaging and intuitive. Interactive graphs, dynamic visual elements, and real-time status indicators will contribute to an enriched user interaction experience. Future iterations will also explore third-party integrations, such as cloud storage, industrial automation protocols (like MQTT/Modbus), calendar synchronization, and communication tools for improved functionality and interoperability with other systems in a smart factory environment. Finally, a strong emphasis will be placed on ensuring the scalability and maintainability of the system architecture. The project will be designed to support future upgrades, additional sensor nodes, and higher data loads without compromising performance, thus ensuring long-term sustainability and adaptability in evolving industrial automation landscapes.

# 12. REFERNCES

**1. W3Schools** – A widely used online platform for learning web development technologies such as HTML, CSS, JavaScript, and other frontend tools. It provided valuable support during the development of the web interface for the object counting system.

LINK: https://www.w3schools.com

**2. MDN Web Docs** – A comprehensive resource offering detailed documentation and tutorials for modern web technologies like HTML, CSS, JavaScript, and DOM manipulation, which contributed to building a responsive and user-friendly frontend interface.

LINK: https://developer.mozilla.org

**3. ChatGPT** – Leveraged to generate technical insights, troubleshoot development challenges, and brainstorm effective system architecture, communication flow, and backend logic throughout the development of the IoT-based object counting system.

**4. Bard (ai)** – Utilized for guidance and inspiration related to UI/UX design best practices, contributing to a clean, intuitive, and efficient web dashboard layout for monitoring and control purposes.

**5. Draw.io** – Employed to design flowcharts, system architecture diagrams, and data flow schematics, aiding in visual planning and presentation of the IoT-based automation system.

LINK: https://app.diagrams.net

**6. Django Documentation** – The official Django framework documentation was referenced extensively to develop the backend web application, manage data flow, authentication, and ensure scalability and maintainability of the platform.

LINK: https://docs.djangoproject.com

**7. ESP32-C3 Technical Reference Manual** – Consulted to understand the hardware capabilities, GPIO interfacing, distance sensor integration, and communication protocols used for real-time data transmission between the microcontroller and the web application.

LINK: https://en.m.wikipedia.org/wiki/Main_Page

**9. Sensor Manufacturer Documentation** (e.g., HC-SR04, VL53L0X, etc.) – Product datasheets and manuals were referred to for configuring distance sensors, understanding their operational characteristics, and ensuring accurate object detection and counting.

# 13. APPENDICES

## 13.1 SCREENSHOTS

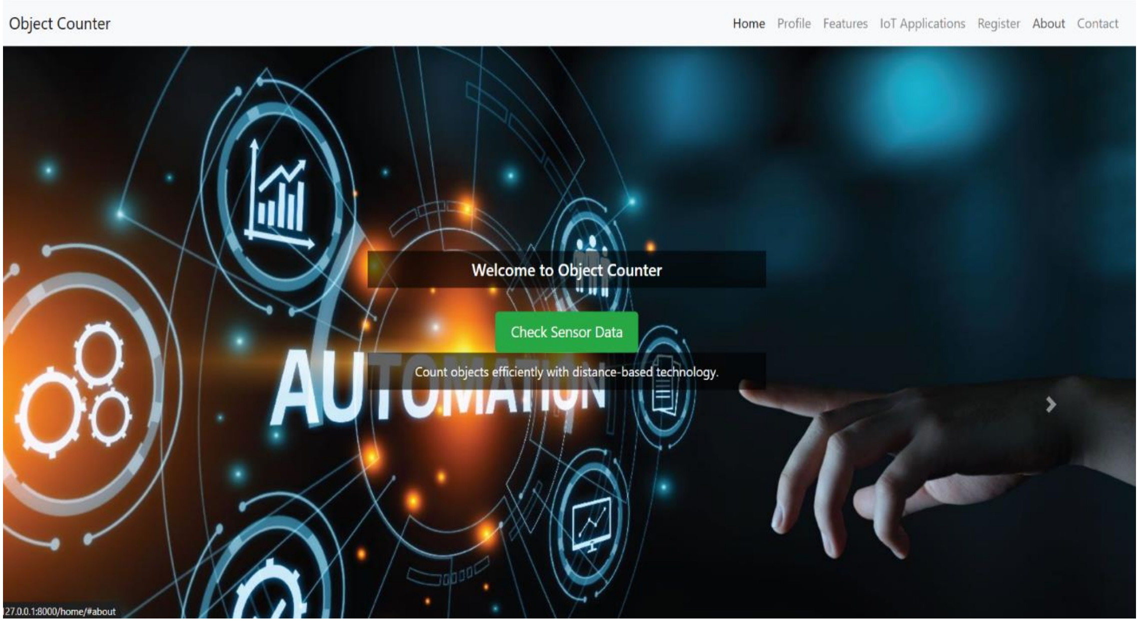

Object Counter
Home
Profile
Features
IoT Applications
Register
About
Contact
Welcome to Object Counter
Check Sensor Data
Count objects efficiently with distance-based technology.
AUTOMATION
127.0.0.1:8000/home/#about

**Fig 13: Home Page**

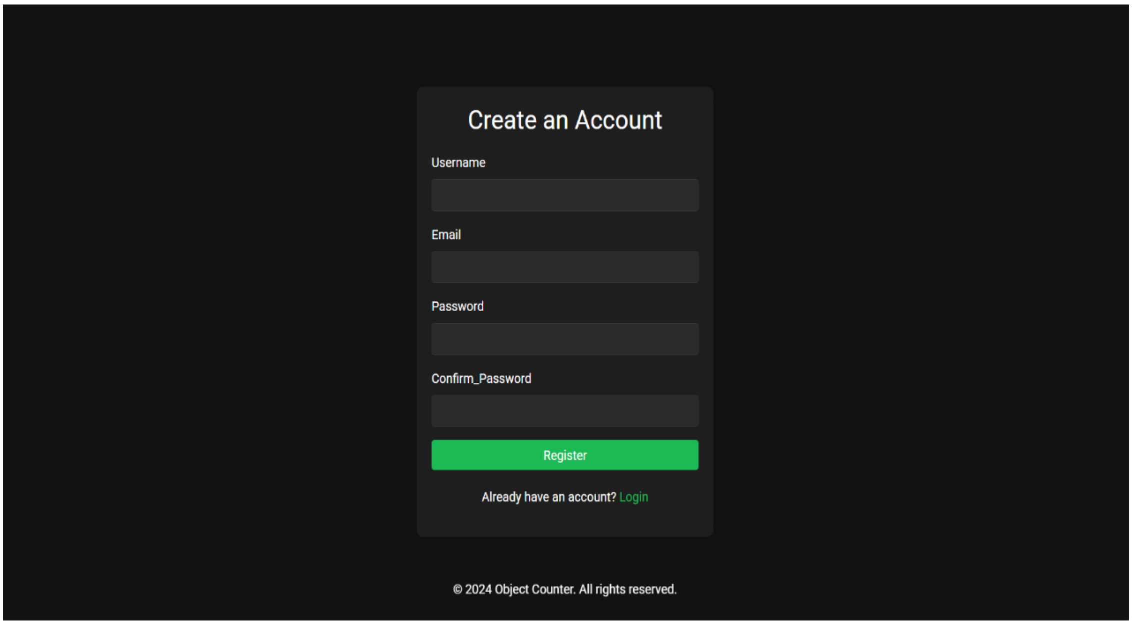

Create an Account
Username
Email
Password
Confirm_Password
Register
Already have an account? Login
© 2024 Object Counter. All rights reserved.

**Fig 14: Signup Page**

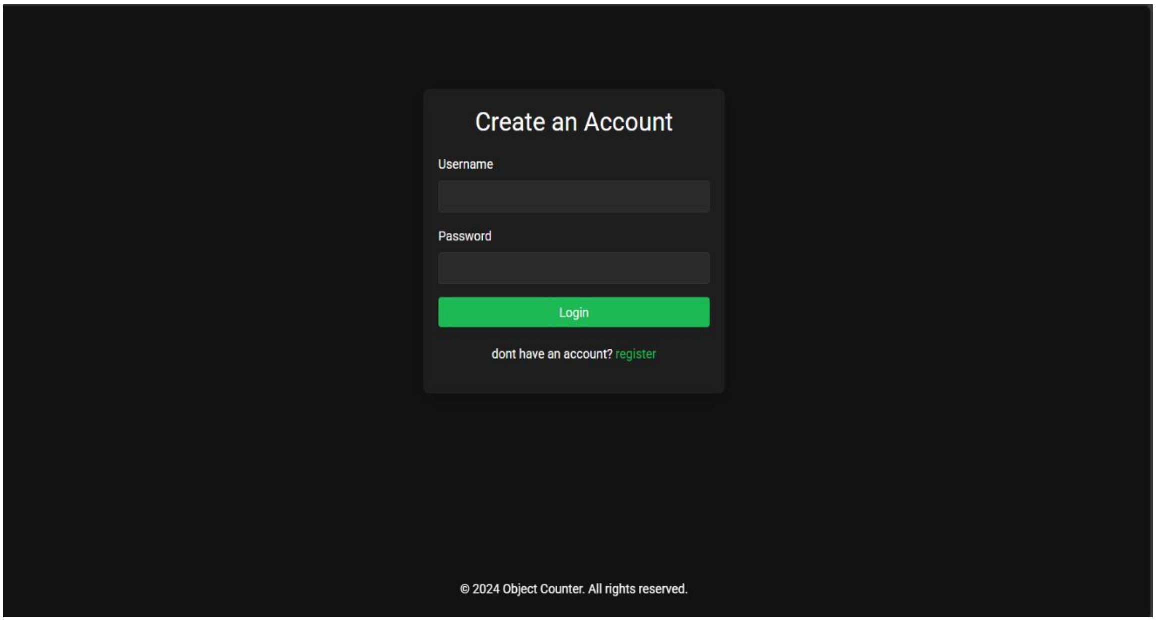
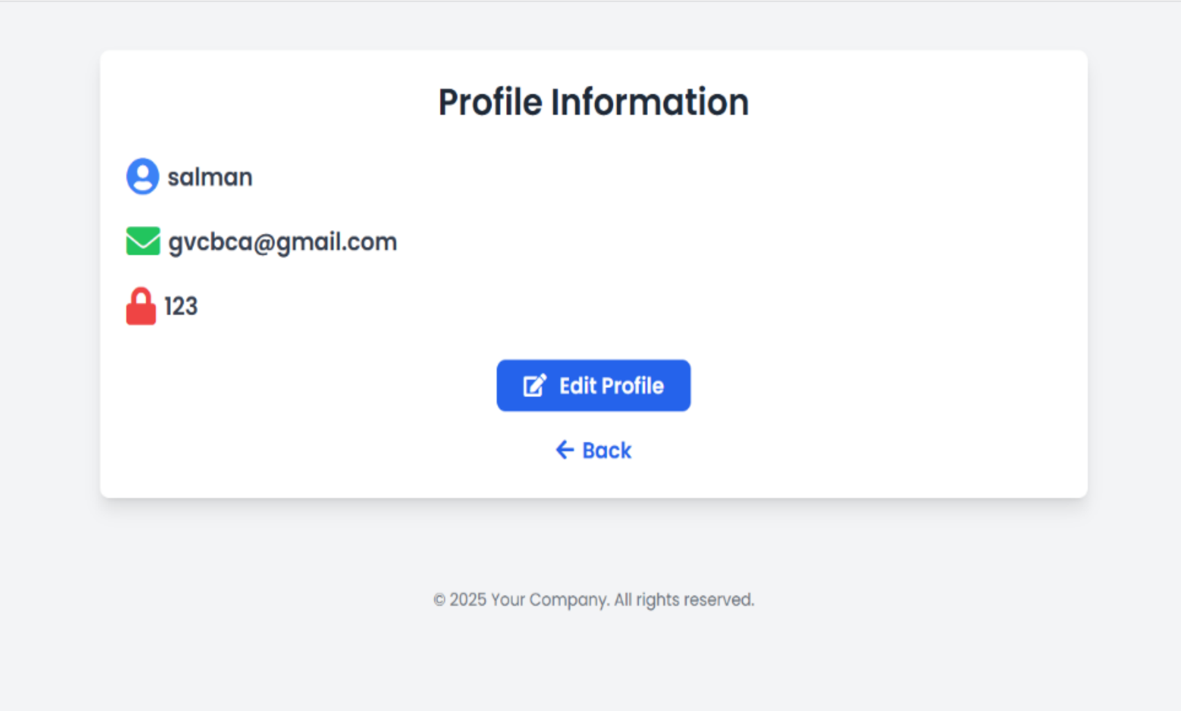

Create an Account
Username
Password
Login
dont have an account? register
© 2024 Object Counter. All rights reserved.

**Fig 15: Login Page**



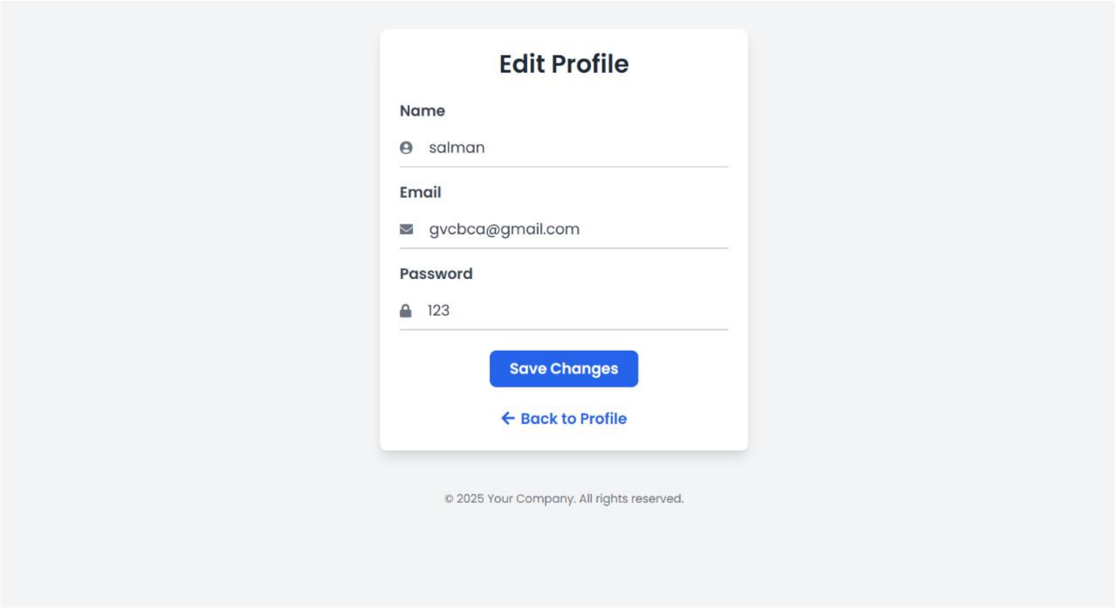
Profile Information

salman

gvcbca@gmail.com

123

**Fig 16: Profile View**



## Edit Profile

Name

salman

Email

gvcbca@gmail.com

Password

123

© 2025 Your Company. All rights reserved.

Fig 17: Edit profile

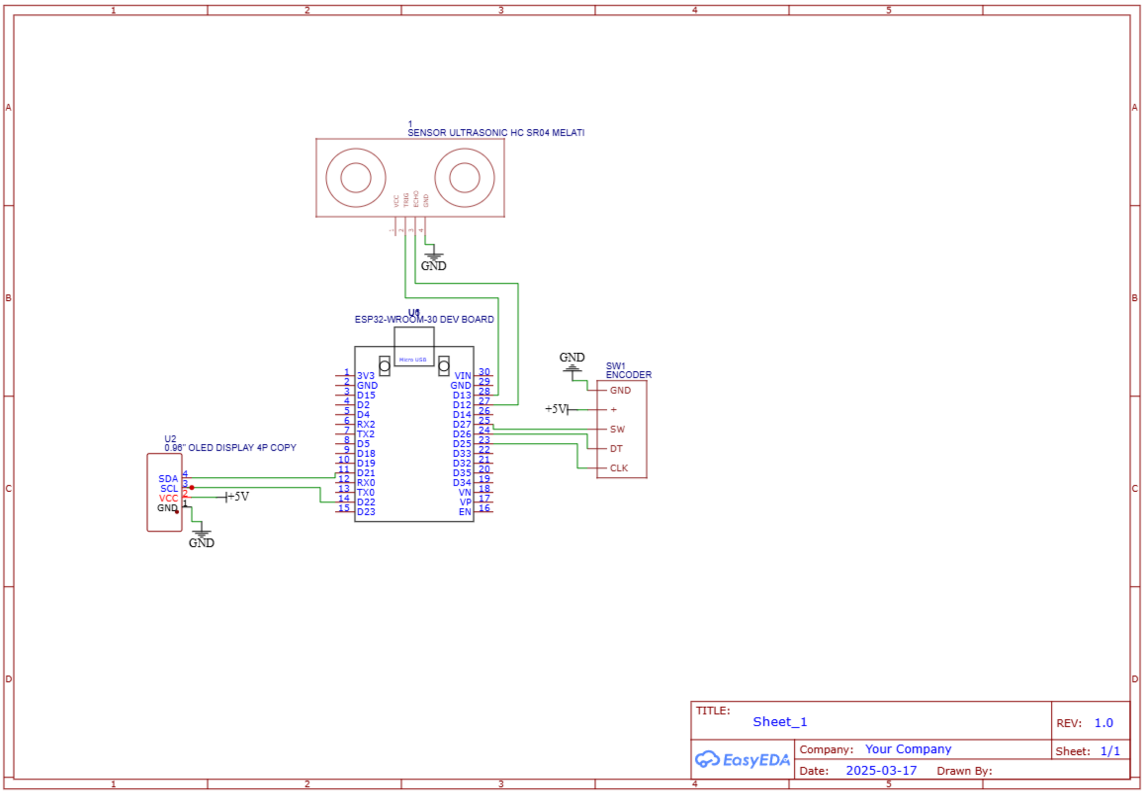


Fig 18: Schematic Diagram

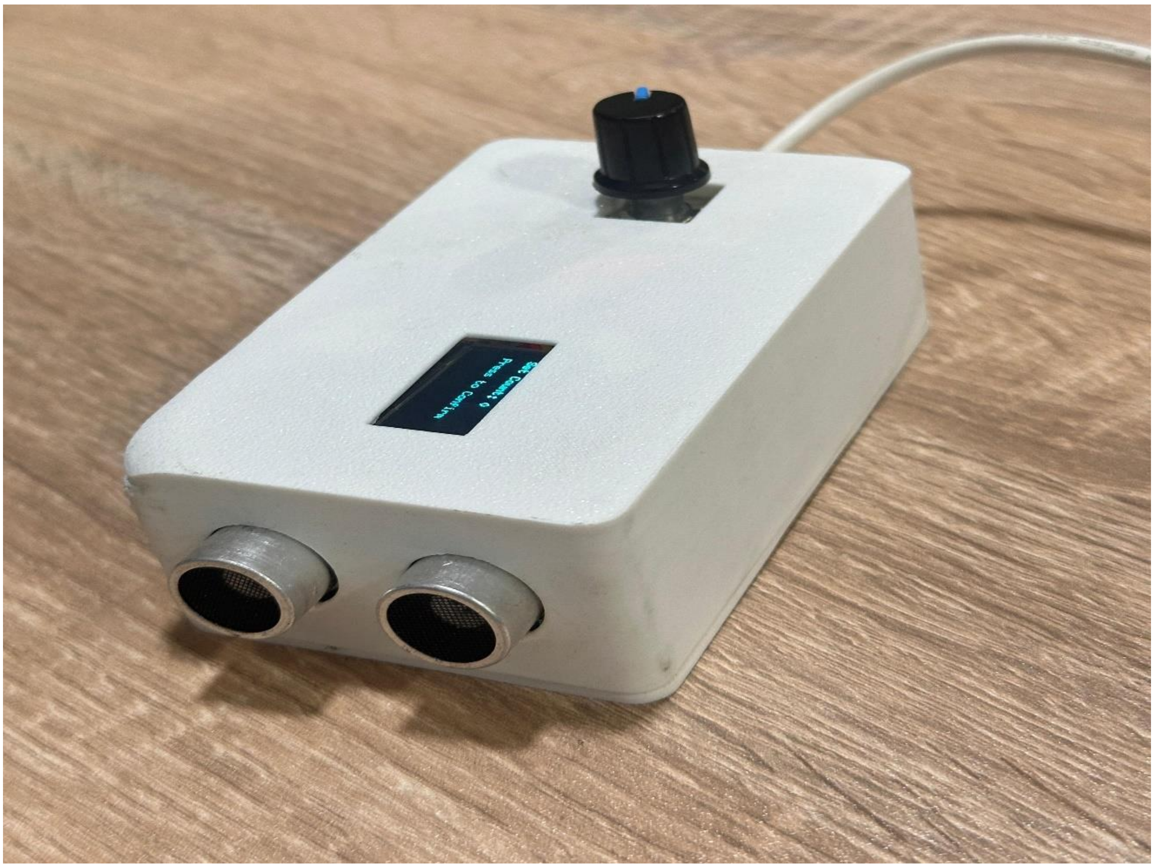
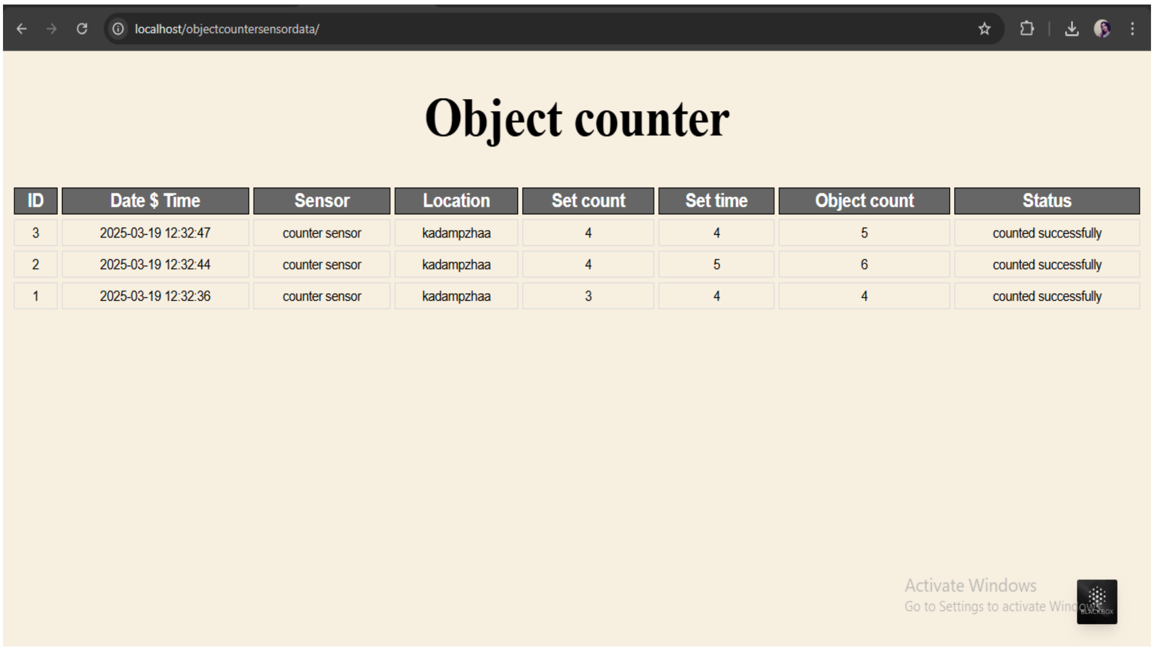


Fig 19:  Image of Hardware


localhost/objectcountersensordata/
Activate Windows
Go to Settings to activate Windows.

# Object counter

| ID | Date $ Time | Sensor | Location | Set count | Set time | Object count | Status |
|---|---|---|---|---|---|---|---|
| 3 | 2025-03-19 12:32:47 | counter sensor | kadampzhaa | 4 | 4 | 5 | counted successfully |
| 2 | 2025-03-19 12:32:44 | counter sensor | kadampzhaa | 4 | 5 | 6 | counted successfully |
| 1 | 2025-03-19 12:32:36 | counter sensor | kadampzhaa | 3 | 4 | 4 | counted successfully |

Fig 20: Object count Reading

# 14. SOURCE CODES

## 14.1 Source Code of Home Page

```
{% comment %} {% load static %}

<!DOCTYPE html>

<html lang="en">


<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Distance-Based Object Counter</title>

    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">

    <link rel="stylesheet" href="{% static 'style.css' %}">

    <style>

        /* Ensure the button is centered over the first carousel image */

        .carousel-item.active .carousel-caption {

            position: relative;

        }

        .carousel-item.active .btn-check-data {

            position: absolute;

            bottom: 20px;

            left: 50%;

            transform: translateX(-50%);

            z-index: 10;

        }

    </style>

</head>


<body>
```

```
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">Object Counter</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="#">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'profile' %}">Profile</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#features">Features</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#iot-applications">IoT Applications</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="http://localhost/objectcountersensordata">Sensor data</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#about">About</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#contact">Contact</a>
                </li>
            </ul>
        </div>
    </nav>

    <div id="carouselExample" class="carousel slide" data-ride="carousel">
        <div class="carousel-inner">
            <div class="carousel-item active">
                <img src="{% static 'images/Automation.jpg'%}" class="d-block w-100" alt="Traffic Detection">
                <div class="carousel-caption d-none d-md-block">
                    <h5>Welcome to Object Counter</h5>
                    <a href="http://localhost/objectcountersensordata" class="btn btn-success btn-lg btn-check-data">Check Sensor Data</a>
                    <p>Count objects efficiently with distance-based technology.</p>
                    <!-- Button to trigger the modal inside the first carousel item -->

                </div>
            </div>
            <div class="carousel-item">
                <img src="{% static 'images/tech.jpeg'%}" class="d-block w-100" alt="YOLO v3 Traffic Detection">
                <div class="carousel-caption d-none d-md-block">
                    <h5>Advanced Object Detection</h5>
                    <p>Utilizing YOLO v3 for real-time traffic detection and analysis.</p>
                </div>
            </div>
            <div class="carousel-item">
                <img src="{% static 'images/conveyorbelt.jpeg'%}" class="d-block w-100" alt="Ultrasonic Distance Circuit">
                <div class="carousel-caption d-none d-md-block">
                    <h5>Easy to Use</h5>
                    <p>Intuitive interface that simplifies object counting.</p>
                </div>
            </div>
        </div>
        <a class="carousel-control-prev" href="#carouselExample" role="button" data-slide="prev">
            <span class="carousel-control-prev-icon" aria-hidden="true"></span>
            <span class="sr-only">Previous</span>
        </a>
        <a class="carousel-control-next" href="#carouselExample" role="button" data-slide="next">
            <span class="carousel-control-next-icon" aria-hidden="true"></span>
            <span class="sr-only">Next</span>
        </a>
    </div>


    <section id="features" class="py-5">
        <div class="container">
            <h2 class="text-center">Features</h2>
            <div class="row">
                <div class="col-md-4">
                    <div class="card">
                        <img src="{% static 'images/conveyorbelt.jpeg'%}" class="card-img-top" alt="Real-Time Tracking">
                        <div class="card-body">
                            <h5 class="card-title">Real-Time Tracking</h5>
                            <p class="card-text">Track objects as they move in real-time using advanced detection technologies like YOLO v3.</p>
                        </div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="card">
                        <img src="{% static 'images/belt.jpeg'%}" class="card-img-top" alt="Distance Measurement">
                        <div class="card-body">
                            <h5 class="card-title">Distance Measurement</h5>
                            <p class="card-text">Measure distances with high accuracy using advanced algorithms and IoT connectivity.</p>
                        </div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="card">
                        <img src="{% static 'images/tech.jpeg'%}" class="card-img-top" alt="User-Friendly Interface">
                        <div class="card-body">
                            <h5 class="card-title">User-Friendly Interface</h5>
                            <p class="card-text">Easy navigation and operation for all users, enhancing accessibility.</p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>


    <section id="iot-applications" class="py-5">
        <div class="container">
            <h2 class="text-center">IoT Applications</h2>
            <p>Our distance-based object counting technology is revolutionizing various industries through IoT integration. Here are some key applications:</p>
            <ul>
                <li><strong>Smart Retail:</strong> Automatically track inventory levels in real-time, reducing out-of-stock situations.</li>
                <li><strong>Warehouse Management:</strong> Monitor the movement of goods and optimize space utilization.</li>
                <li><strong>Traffic Monitoring:</strong> Count vehicles in real-time to manage traffic flow and enhance safety.</li>
```

```
                <li><strong>Event Management:</strong> Track attendee numbers at events to ensure safety and compliance.</li>

                <li><strong>Wildlife Conservation:</strong> Monitor animal populations in natural habitats using remote sensors.</li>

            </ul>

        </div>

    </section>



    <section id="about" class="bg-light py-5">

        <div class="container">

            <h2 class="text-center">About Us</h2>

            <p>At Object Counter, we leverage cutting-edge IoT technology to provide efficient and accurate distance-based object counting solutions. Our mission is to enhance operational efficiency across various sectors, from retail to conservation. With a focus on innovation, we aim to transform how industries manage and monitor their assets, ensuring smarter and more sustainable practices.</p>

        </div>

    </section>



    <section id="contact" class="py-5">

        <div class="container">

            <h2 class="text-center">Contact Us</h2>

            <form>

                <div class="form-group">

                    <label for="name">Name</label>

                    <input type="text" class="form-control" id="name" placeholder="Your Name" required>

                </div>

                <div class="form-group">

                    <label for="email">Email</label>

                    <input type="email" class="form-control" id="email" placeholder="Your Email" required>

                </div>

                <div class="form-group">
```

```
                    <label for="message">Message</label>
                    <textarea class="form-control" id="message" rows="3" placeholder="Your Message" required></textarea>
                </div>
                <button type="submit" class="btn btn-primary">Send Message</button>
            </form>
        </div>
    </section>


    <footer class="text-center py-4">
        <p>&copy; 2024 Object Counter. All rights reserved.</p>
    </footer>


    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core"></script>
</body>


</html> {% endcomment %}
{% load static %}
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Distance-Based Object Counter</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="{% static 'style.css' %}">
    <style>
        /* Ensure the carousel image covers the entire space */
        .carousel-item img {
```

```
    width: 100%;
    height: 100%;
    object-fit: cover;
}

/* Style the carousel-caption */
.carousel-caption {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    color: white;
    text-align: center;
    z-index: 10;
}

/* Add background to carousel caption for readability */
.carousel-caption h5, .carousel-caption p {
    background-color: rgba(0, 0, 0, 0.6); /* Semi-transparent background */
    padding: 10px;
}

/* Style the button in the carousel caption */
.carousel-caption .btn-check-data {
    margin-top: 20px;
    font-size: 18px;
    padding: 10px 20px;
}

/* General section styling */
section {
```

```
            padding: 5rem 0;
        }
    </style>
</head>

<body>

    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">Object Counter</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="#">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'profile' %}">Profile</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#features">Features</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#iot-applications">IoT Applications</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/register/">Register</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#about">About</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#contact">Contact</a>
                </li>
            </ul>
        </div>
    </nav>


    <!-- Carousel -->
    <div id="carouselExample" class="carousel slide" data-ride="carousel">
        <div class="carousel-inner">
            <div class="carousel-item active">
                <img src="{% static 'images/Automation.jpg'%}" class="d-block w-100"
alt="Traffic Detection">
                <div class="carousel-caption d-none d-md-block">
                    <h5>Welcome to Object Counter</h5>
                    <a href="http://localhost/objectcountersensordata" class="btn btn-success btn-lg
btn-check-data">Check Sensor Data</a>
                    <p>Count objects efficiently with distance-based technology.</p>
                </div>
            </div>
            <div class="carousel-item">
                <img src="{% static 'images/tech.jpeg'%}" class="d-block w-100" alt="YOLO v3
Traffic Detection">
                <div class="carousel-caption d-none d-md-block">
                    <h5>Advanced Object Detection</h5>
                    <p>Utilizing YOLO v3 for real-time traffic detection and analysis.</p>
                </div>
            </div>
            <div class="carousel-item">
```

```
                <img src="{% static 'images/conveyorbelt.jpeg'%}" class="d-block w-100" alt="Ultrasonic Distance Circuit">
                <div class="carousel-caption d-none d-md-block">
                    <h5>Easy to Use</h5>
                    <p>Intuitive interface that simplifies object counting.</p>
                </div>
            </div>
        </div>
        <a class="carousel-control-prev" href="#carouselExample" role="button" data-slide="prev">
            <span class="carousel-control-prev-icon" aria-hidden="true"></span>
            <span class="sr-only">Previous</span>
        </a>
        <a class="carousel-control-next" href="#carouselExample" role="button" data-slide="next">
            <span class="carousel-control-next-icon" aria-hidden="true"></span>
            <span class="sr-only">Next</span>
        </a>
    </div>

    <!-- Features Section -->
    <section id="features" class="bg-light">
        <div class="container">
            <h2 class="text-center">Features</h2>
            <div class="row">
                <div class="col-md-4">
                    <div class="card">
                        <img src="{% static 'images/conveyorbelt.jpeg'%}" class="card-img-top" alt="Real-Time Tracking">
                        <div class="card-body">
                            <h5 class="card-title">Real-Time Tracking</h5>
                            <p class="card-text">Track objects as they move in real-time using advanced detection technologies like YOLO v3.</p>
                        </div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="card">
                        <img src="{% static 'images/belt.jpeg'%}" class="card-img-top" alt="Distance Measurement">
                        <div class="card-body">
                            <h5 class="card-title">Distance Measurement</h5>
                            <p class="card-text">Measure distances with high accuracy using advanced algorithms and IoT connectivity.</p>
                        </div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="card">
                        <img src="{% static 'images/tech.jpeg'%}" class="card-img-top" alt="User-Friendly Interface">
                        <div class="card-body">
                            <h5 class="card-title">User-Friendly Interface</h5>
                            <p class="card-text">Easy navigation and operation for all users, enhancing accessibility.</p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>


    <!-- IoT Applications Section -->
    <section id="iot-applications" class="bg-light">
        <div class="container">
```

```html
        <h2 class="text-center">IoT Applications</h2>
        <p>Our distance-based object counting technology is revolutionizing various industries through IoT integration. Here are some key applications:</p>
        <ul>
          <li><strong>Smart Retail:</strong> Automatically track inventory levels in real-time, reducing out-of-stock situations.</li>
          <li><strong>Warehouse Management:</strong> Monitor the movement of goods and optimize space utilization.</li>
          <li><strong>Traffic Monitoring:</strong> Count vehicles in real-time to manage traffic flow and enhance safety.</li>
          <li><strong>Event Management:</strong> Track attendee numbers at events to ensure safety and compliance.</li>
          <li><strong>Wildlife Conservation:</strong> Monitor animal populations in natural habitats using remote sensors.</li>
        </ul>
      </div>
    </section>

    <!-- About Us Section -->
    <section id="about" class="bg-light py-5">
      <div class="container">
        <h2 class="text-center">About Us</h2>
        <p>At Object Counter, we leverage cutting-edge IoT technology to provide efficient and accurate distance-based object counting solutions. Our mission is to enhance operational efficiency across various sectors, from retail to conservation. With a focus on innovation, we aim to transform how industries manage and monitor their assets, ensuring smarter and more sustainable practices.</p>
      </div>
    </section>

    <!-- Contact Section -->
    <section id="contact" class="py-5">
      <div class="container">
        <h2 class="text-center">Contact Us</h2>
        <form>
          <div class="form-group">
            <label for="name">Name</label>
            <input type="text" class="form-control" id="name" placeholder="Your Name" required>
          </div>
          <div class="form-group">
            <label for="email">Email</label>
            <input type="email" class="form-control" id="email" placeholder="Your Email" required>
          </div>
          <div class="form-group">
            <label for="message">Message</label>
            <textarea class="form-control" id="message" rows="3" placeholder="Your Message" required></textarea>
          </div>
          <button type="submit" class="btn btn-primary">Send Message</button>
        </form>
      </div>
    </section>

    <!-- Footer -->
    <footer class="text-center py-4">
      <p>&copy; 2024 Object Counter. All rights reserved.</p>
    </footer>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core"></script>
  </body>
  </html>
```

## 14.2 Source Code of Login Page

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register - Object Counter</title>
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #121212; /* Dark background */
            color: #ffffff; /* Light text color */
            font-family: 'Roboto', sans-serif;
        }
        .container {
            max-width: 400px;
            margin-top: 100px;
            padding: 20px;
            background-color: #1e1e1e; /* Slightly lighter dark for the form */
            border-radius: 8px;
            box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);

        }
        h2 {
            text-align: center;
            margin-bottom: 20px;
        }
        .form-control {
            background-color: #2a2a2a; /* Darker input background */
            border: 1px solid #3a3a3a;
            color: #ffffff; /* Input text color */
        }
        .form-control:focus {
            background-color: #2a2a2a;
            border-color: #1db954; /* Spotify green */
            box-shadow: 0 0 0 0.2rem rgba(29, 185, 84, 0.25);
        }
        .btn-primary {
            background-color: #1db954; /* Spotify green */
            border: none;
        }
        .btn-primary:hover {
            background-color: #1aa34a; /* Slightly darker green */
        }
        .text-center {
            margin-top: 20px;
        }
        .footer {
            position: fixed;
            bottom: 10px;
            left: 0;
        right: 0;
            text-align: center;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Create an Account</h2>
    <form method="POST" action=>
```

```
        {% csrf_token %}
        <div class="form-group">
            <label for="username">Username</label>
            <input type="text" class="form-control" id="username" name="username"
required>
        </div>

        <div class="form-group">
            <label for="password">Password</label>
            <input type="password" class="form-control" id="password"
name="password" required>
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
    <div class="text-center">
        <p>don’t have an account? <a href="/register/"
                    style="color: #1db954;">register</a></p>
    </div>
</div>

<div class="footer">
<p>&copy; 2024 Object Counter. All rights reserved.</p>
</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></s
cript>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script
>
</body>
</html>
```

## 14.3 Source Code of Signup Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register - Object Counter</title>
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=sw
ap" rel="stylesheet">
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #121212; /* Dark background */
            color: #ffffff; /* Light text color */
            font-family: 'Roboto', sans-serif;
        }
        .container {
        max-width: 400px; margin-top: 100px; padding: 20px;
            background-color: #1e1e1e; /* Slightly lighter dark for the form */
            border-radius: 8px;
            box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);
        }
        h2 {
            text-align: center;
            margin-bottom: 20px;
        }
        .form-control {
            background-color: #2a2a2a; /* Darker input background */
            border: 1px solid #3a3a3a;
            color: #ffffff; /* Input text color */
        }
        .form-control:focus {
            background-color: #2a2a2a;
            border-color: #1db954; /* Spotify green */
            box-shadow: 0 0 0 0.2rem rgba(29, 185, 84, 0.25);
        }
        .btn-primary {
            background-color: #1db954; /* Spotify green */
            border: none;
        }
        .btn-primary:hover {
            background-color: #1aa34a; /* Slightly darker green */
        }
        .text-center {
            margin-top: 20px;
        }
        .footer {
            position: fixed;
            bottom: 10px;
            left: 0;
            right: 0;
            text-align: center;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Create an Account</h2>
    <form method="POST" action="{% url 'register' %}">
    {% csrf_token %}
    <div class="form-group">
      <label for="username">Username</label>
      <input type="text" class="form-control" id="username" name="username"
required>
    </div>
    <div class="form-group">
      <label for="email">Email</label>
      <input type="email" class="form-control" id="email" name="email" required>
    </div>
    <div class="form-group">
      <label for="password">Password</label>
      <input type="password" class="form-control" id="password"
name="password" required>
    </div>
    <div class="form-group">
      <label for="password">Confirm_Password</label>
      <input type="password" class="form-control" id="Confirm_Password"
name="Confirm_Password" required>
    </div>
    <button type="submit" class="btn btn-primary btn-block">Register</button>
  </form>
  <div class="text-center">
    <p>Already have an account? <a href="/login/"
              style="color: #1db954;">Login</a></p>
  </div>
</div>


<div class="footer">
  <p>&copy; 2024 Object Counter. All rights reserved.</p>
</div>


<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>
```

## 14.4 Source Code of ESP32 Fireware

```
#include <Wire.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <U8g2lib.h>


#define ENCODER_CLK  32
#define ENCODER_DT   33

#define ENCODER_SW   25


#define TRIG_PIN  15

#define ECHO_PIN  18

#define BUZZER_PIN  26


const char* ssid = "OPPOA54";  // Change to your WiFi SSID

const char* password = "10101010";  // Change to your WiFi password

const char* serverURL = "http://192.168.95.214/objectcountersensordata/post-esp-data.php";


String sensorName = "objectcounter";

String sensorLocation = "Thrissur";


U8G2_SSD1306_128X64_NONAME_F_SW_I2C display(U8G2_R0, SCL, SDA, U8X8_PIN_NONE);
```

```
volatile int encoderValue = 0;
int lastStateCLK;
bool countSet = false;
bool timeSet = false;
int objectCount = 0;
int setCount = 0;
int setTime = 10;
unsigned long startTime = 0;
bool timeRunning = false;

void IRAM_ATTR readEncoder() {
  int currentStateCLK = digitalRead(ENCODER_CLK);
  if (currentStateCLK != lastStateCLK) {
    if (digitalRead(ENCODER_DT) != currentStateCLK) {
      encoderValue++;
    } else {
      encoderValue--;
    }
  }
  lastStateCLK = currentStateCLK;
}

float getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2;
}
```

```
void sendDataToWeb() {
  Serial.println("Sending data to server...");
  display.clearBuffer();
  display.setCursor(10, 30);
  display.print("Sending data...");
  display.sendBuffer();

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverURL);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    String postData = "sensor_name=" + sensorName +
             "&sensor_location=" + sensorLocation +
             "&value1=" + String(setCount) +
             "&value2=" + String(setTime) +
             "&value3=" + String(objectCount) +
            "&value4=" + (objectCount >= setCount ? "Successful" : "Timed Out");

    int httpResponseCode = http.POST(postData);
    Serial.print("Server Response Code: ");
    Serial.println(httpResponseCode);

    http.end();
  } else {
    Serial.println("WiFi not connected!");
  }

  delay(2000);
}
```

```
void resetSystem() {
    countSet = false;
    timeSet = false;
    objectCount = 0;
    encoderValue = 0;
    timeRunning = false;
}

void setup() {
    Serial.begin(9600);
    WiFi.begin(ssid, password);
    Serial.print("Connecting to WiFi...");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to WiFi!");

    pinMode(ENCODER_CLK, INPUT_PULLUP);
    pinMode(ENCODER_DT, INPUT_PULLUP);
    pinMode(ENCODER_SW, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(ENCODER_CLK), readEncoder, CHANGE);

    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(BUZZER_PIN, OUTPUT);

    lastStateCLK = digitalRead(ENCODER_CLK);
    display.begin();
    display.clearBuffer();
```

```
  display.setFont(u8g2_font_6x10_tf);
}

void loop() {
  if (!countSet) {
    display.clearBuffer();
    display.setCursor(10, 20);
    display.print("Set Count: ");
    display.print(encoderValue);
    display.setCursor(10, 40);
    display.print("Press to Confirm");
    display.sendBuffer();

    if (digitalRead(ENCODER_SW) == LOW) {
      delay(300);
      setCount = encoderValue;
      countSet = true;
      encoderValue = 10;
    }
    return;
  }

  if (!timeSet) {
    display.clearBuffer();
    display.setCursor(10, 20);
    display.print("Set Time (s): ");
    display.print(encoderValue);
    display.setCursor(10, 40);
    display.print("Press to Confirm");
    display.sendBuffer();
    if (digitalRead(ENCODER_SW) == LOW) {
        delay(300);
        setTime = encoderValue;
        if (setTime < 1) setTime = 1;
        if (setTime > 60) setTime = 60;
        timeSet = true;
        objectCount = 0;
        timeRunning = true;
        startTime = millis();
    }
    return;
}

unsigned long elapsedTime = millis() - startTime;
if (elapsedTime >= setTime * 1000) {
    display.clearBuffer();
    display.setCursor(10, 30);
    display.print("Time Reached!");
    display.sendBuffer();
    digitalWrite(BUZZER_PIN, HIGH);
    delay(1000);
    digitalWrite(BUZZER_PIN, LOW);
    sendDataToWeb();
    resetSystem();
    return;
}

float distance = getDistance();
if (distance < 10) {
    objectCount++;
    delay(500);
  }

  if (objectCount >= setCount) {
    display.clearBuffer();
    display.setCursor(10, 30);
    display.print("Count Reached!");
    display.sendBuffer();
    digitalWrite(BUZZER_PIN, HIGH);
    delay(1000);
    digitalWrite(BUZZER_PIN, LOW);
    sendDataToWeb();
    resetSystem();
    return;
  }

  int timeLeft = (setTime * 1000 - elapsedTime) / 1000;
  display.clearBuffer();
  display.setCursor(10, 10);
  display.print("Set: ");
  display.print(setCount);
  display.setCursor(10, 30);
  display.print("Count: ");
  display.print(objectCount);
  display.setCursor(10, 50);
  display.print("Time: ");
  display.print(timeLeft);
  display.print("s");
  display.sendBuffer();
  delay(500);
  }
```

## 14.5 Backend source code of website

```
from django.shortcuts import render,redirect,HttpResponse
from .models import *
# Create your views here.
def index(request):
    return render(request,'index.html')
def register(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        Confirm_Password = request.POST.get('Confirm_Password')


        if password != Confirm_Password:
            return HttpResponse("<script>alert('Password do not match!');
window.history.back();</script>")
        if userregister.objects.filter(username=username).exists():
            return HttpResponse("<script>alert('Username already exists!');
window.history.back();</script>")
        if userregister.objects.filter(email=email).exists():
            return HttpResponse("<script>alert('Email alreday registered!');
window.history.back();</script>")
        user=userregister(username=username,email=email,password=password)
        user.save()
        return HttpResponse("<script>alert('Registration Successful! Please login.');
window.location.href='/login/';</script>")
    return render(request,'register.html')


def login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            user = userregister.objects.get(username=username,password=password)
            request.session['username'] = user.username
            return redirect('index')
        except user.DoesNotExist:
             return render(request,'login.html',{'error':'Invalid email or password.'})
    return render(request,'login.html')


def ben(request):
    return render(request,'salmu.html')



def profile(request):
    username=request.session.get('username')
    form=userregister.objects.filter(username=username)
    return render(request,'profile.html',{'form':form})
```