End-Term Project report on

ESP32-CAM and OpenCV-Based Low-Cost Object Counting System for Real-Time Inventory and Resource Management

Submitted by

Abhishek Sahoo (2241019026)

Debadutta Barik (2241002036)

Jaydev Sahoo (2241019078)

Archit Kumar Sahoo (2241019040)

B. Tech. (CSE(IOT) 5th Semester (Section–2241037)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Institute of Technical Education and Research

SIKSHA 'O' ANUSANDHAN
DEEMED TO BE UNIVERSITY

Bhubaneswar, Odisha, India.

(December, 2024)

Abstract

This paper proposes a low-cost, real-time object counting system for inventory and resource management, utilizing the ESP32-CAM module and OpenCV, a powerful open-source computer vision library. The system uses the ESP32-CAM's built-in camera to capture images or video streams of objects in a monitored area. These images are then processed using OpenCV, employing computer vision techniques such as object detection, background subtraction, contour analysis, and tracking to count the objects within the frame. The ESP32-CAM is a compact and affordable solution, featuring both a camera and Wi-Fi capabilities, which makes it highly suitable for various resource management and inventory applications.

The proposed system offers a scalable, real-time approach for industries such as retail, warehousing, and manufacturing, where efficient inventory control and resource tracking are essential. By providing an automated method to count and monitor objects, the system reduces human errors and enhances operational efficiency. Its wireless capabilities enable remote monitoring and data transmission, making it adaptable to a wide range of environments.

In addition, the system's integration with OpenCV ensures accurate and reliable object detection, even under varying lighting conditions. The simplicity and low cost of the ESP32-CAM module make this solution an attractive alternative to expensive industrial-grade object tracking systems. The paper demonstrates the practicality of the system, highlighting its effectiveness for real-time inventory and resource management applications. Ultimately, the proposed solution represents an affordable, efficient, and scalable method for enhancing resource management and monitoring in dynamic environments.

Contents

Chapter 01: Introduction

1.1. Introduction

Effective inventory and resource management is essential for businesses in industries such as retail, warehousing, and manufacturing. Traditional methods of counting and tracking inventory are often manual, time-consuming, prone to human error, and lack real-time capabilities. With the increasing demand for automation and accuracy, there is a growing need for low-cost, scalable solutions to streamline inventory and resource management. This paper introduces a real-time object counting system based on the ESP32-CAM module and OpenCV, which addresses these challenges in a cost-effective manner.

The ESP32-CAM is a compact, low-cost microcontroller with integrated camera and Wi-Fi capabilities, making it ideal for real-time monitoring and object detection tasks. By combining the ESP32-CAM with OpenCV, a powerful open-source computer vision library, the system processes captured images or video streams to identify and count objects accurately. Image processing techniques such as background subtraction, contour detection, and object tracking are employed to ensure precise object counting under varying conditions.

This system offers several advantages, including reduced reliance on manual labor, improved accuracy, and the ability to monitor inventory remotely through wireless communication. The proposed solution is not only scalable but also adaptable to various environments, providing an affordable alternative to expensive industrial-grade object tracking systems. Ultimately, it provides an efficient method for real-time inventory and resource management, benefiting businesses looking to improve operational efficiency.

1.2. Background

Inventory and resource management is a critical aspect of many industries, including retail, manufacturing, and logistics. Traditional inventory control methods, such as manual counting or barcode scanning, often lead to inaccuracies, inefficiencies, and delays, especially in environments with high volumes of items. These methods also lack real-time tracking capabilities, making it difficult to maintain accurate stock levels and optimize resource allocation. As businesses continue to seek more efficient and automated solutions, the need for affordable and scalable systems has become more apparent.

Recent advancements in computer vision and low-cost hardware have paved the way for more effective object detection and tracking systems. The ESP32-CAM, a low-cost microcontroller with an integrated camera and Wi-Fi connectivity, has gained popularity due to its affordability, compactness, and versatility. OpenCV, a widely-used open-source computer vision library, offers powerful image processing tools that can be leveraged to develop object counting systems capable of real-time inventory tracking.

1.3. Project Objectives

The primary objective of this project is to develop a low-cost, real-time object counting system for inventory and resource management using the ESP32-CAM module and OpenCV. The system aims to provide an affordable and scalable solution that can accurately detect and count objects in environments such as warehouses, retail stores, and manufacturing facilities. The project focuses on designing an efficient object counting mechanism by integrating the ESP32-CAM module's camera and Wi-Fi capabilities for seamless data capture and remote monitoring. It also involves implementing image processing techniques, including background subtraction, contour detection,

and object tracking, using OpenCV to ensure precise counting and minimize errors. Additionally, the system will enable wireless communication to transmit the counted data for real-time updates and integration with existing inventory management systems, enhancing operational efficiency. A key goal is to ensure the system's scalability and flexibility, making it adaptable for various applications and accessible to businesses of different sizes. Ultimately, this project aims to provide an affordable, automated alternative to manual inventory control, improving accuracy and resource management while reducing human error.

1.4. Scope

The scope of this project is to develop a low-cost, real-time object counting system for inventory and resource management using the ESP32-CAM module and OpenCV. The system will capture images or video streams of objects in various environments, such as warehouses, retail stores, and manufacturing facilities, using the ESP32-CAM's built-in camera. These images will be processed with OpenCV, employing techniques like background subtraction and contour detection to accurately count the objects. The project aims to create an affordable and scalable solution for small to medium-sized businesses, offering an automated alternative to manual counting methods.

The system will also feature wireless communication capabilities, enabling real-time data transmission for remote monitoring and integration with existing inventory management systems. The scope of the project excludes the use of advanced industrial-grade hardware, focusing on providing a practical and cost-effective solution that improves operational efficiency, reduces errors, and enhances resource management.

1.5. Project Management

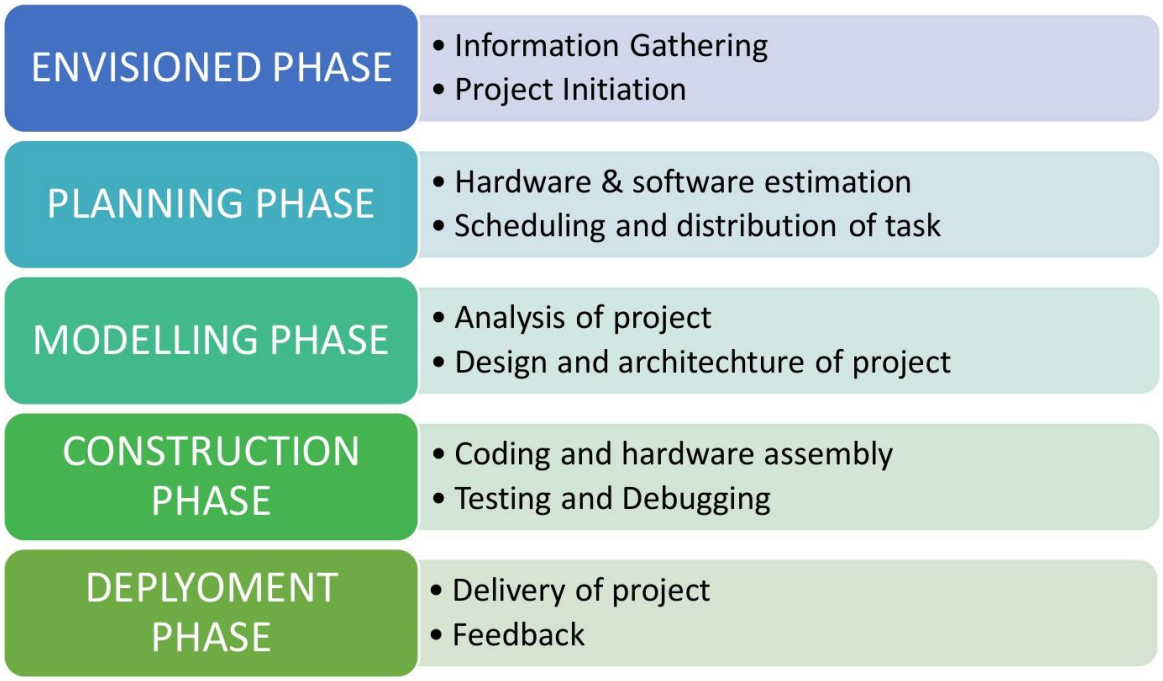


Figure 1. Model of phases in project management.

1.6. Overview and Benefits

The ESP32-CAM and OpenCV-based object counting system is a cost-effective solution for real-time inventory and resource management. By integrating the ESP32-CAM's camera with OpenCV's image processing capabilities, the system enables accurate object detection and counting in various environments, such as warehouses, retail stores, or manufacturing facilities. This system automates the traditionally manual and error-prone process of inventory counting, improving efficiency and reducing human error. With its low-cost hardware, scalability, and flexibility, it provides a practical alternative to more expensive sensor-based tracking systems. Benefits include increased accuracy in tracking inventory, real-time monitoring of resources, reduced operational costs, and the ability to make faster, data-driven decisions.

1.7. Organization of the Report

The report is organised into the following chapters. Each chapter is unique on its own and is described with necessary theory to comprehend it.

Chapter 2 deals with background survey and review, Chapter 3 has the description of the theoretical aspects that has been acquired to commence the project work.

Chapter 02: Background Review & Survey

2.1. Related Works

The need for efficient inventory and resource management has led to the development of various automated systems, with object counting being a critical aspect. Traditional methods of inventory management often rely on manual counting, which can be time-consuming, prone to errors, and costly. Advances in computer vision and embedded systems have paved the way for more accurate and automated solutions. One such approach is using the ESP32-CAM, a low-cost microcontroller with integrated camera capabilities, combined with OpenCV, an open-source computer vision library.

Recent research and applications have focused on using embedded systems for object detection and counting. Systems leveraging Raspberry Pi and other low-cost cameras have demonstrated real-time object counting in warehouses and retail environments. However, these solutions often involve higher hardware costs and complex setups. The ESP32-CAM, with its affordability, ease of integration, and Wi-Fi capabilities, offers a more accessible solution for small to medium-sized businesses.

Previous works have utilized OpenCV for object recognition tasks in various domains such as agriculture, security, and logistics. However, most studies focus on expensive hardware or specific applications, limiting scalability.

Chapter 03: Theoretical Aspects

3.1. Internet of Things (IoT)

The integration of ESP32-CAM and OpenCV in an object counting system leverages the Internet of Things (IoT) for real-time inventory and resource management. The IoT connects physical devices to the internet, enabling remote monitoring, data collection, and management through sensors and embedded systems. In this system, the ESP32-CAM serves as an IoT device, capturing images of objects in an inventory or resource area. The camera is connected to a microcontroller that processes the captured images using OpenCV, an open-source computer vision library, to identify and count objects.

The IoT aspect comes into play as the system communicates data over Wi-Fi to central servers or cloud platforms for real-time monitoring and analysis. This remote accessibility allows businesses to track inventory and resources from anywhere. Additionally, IoT enables seamless integration with other systems, improving operational efficiency, data-driven decision-making, and automation, while reducing human error and resource wastage in inventory management processes.

3.2. Features of IoT

a) Intelligence

Intelligence in IoT refers to the ability of devices to process and analyze data autonomously, making decisions or triggering actions based on real-time inputs. In the context of an ESP32-CAM and OpenCV-based object counting system, intelligence is embedded in the system's ability to detect and count objects using computer vision. The ESP32-CAM captures images, and OpenCV processes them to

identify and count objects, reducing the need for manual intervention. Additionally, the system can analyze patterns, recognize anomalies, and make automated decisions, such as alerting for low stock levels or miscounts, enhancing operational efficiency in real-time inventory management.

b) Connectivity

Connectivity in IoT refers to the ability of devices to communicate and share data over networks. In the ESP32-CAM and OpenCV-based object counting system, connectivity enables seamless data transmission between the camera module, the processing unit, and cloud platforms or centralized servers. The ESP32-CAM, with built-in Wi-Fi capabilities, allows real-time uploading of object count data for monitoring and analysis from anywhere. This connectivity ensures continuous data flow, enabling remote access, system integration, and automation of inventory management tasks. Additionally, it facilitates real-time updates and alerts, improving decision-making and operational efficiency.

c) Dynamic Nature

The dynamic nature of IoT refers to the system's ability to adapt and respond to changing conditions in real time. In the ESP32-CAM and OpenCV-based object counting system, this feature allows the system to continuously monitor and process inventory or resource data as it changes. As objects are added or removed, the system dynamically updates object counts, adapting to new input without manual intervention. The dynamic nature also enables the system to learn from data patterns, detect anomalies, and adjust operations accordingly, ensuring real-time accuracy in

inventory management and resource tracking in ever-changing environments.

d) Enormous Scale

The enormous scale of IoT refers to the ability to deploy and manage a vast number of interconnected devices across diverse locations. In the context of the ESP32-CAM and OpenCV-based object counting system, this scalability allows the solution to be expanded easily from a single device to a network of cameras monitoring inventory across multiple warehouses or stores. The system can handle large volumes of data from numerous devices, providing real-time updates and insights at a global scale. This scalability ensures that businesses of any size can implement the system, improving resource management and inventory tracking across vast networks.

e) Sensing

Sensing in IoT refers to the ability of devices to capture and measure data from the physical environment. In the ESP32-CAM and OpenCV-based object counting system, sensing is achieved through the camera module of the ESP32-CAM, which captures images of objects in an inventory or resource area. The system uses computer vision algorithms via OpenCV to process these images, detecting and counting objects in real time. This sensing capability enables the system to autonomously monitor inventory levels, track resource usage, and provide accurate data without human intervention, ensuring efficient and accurate management of resources.

f) Heterogeneity

Heterogeneity in IoT refers to the diversity of devices, technologies, and protocols that can work together within an IoT ecosystem. In the case of the ESP32-CAM and OpenCV-based object counting system, heterogeneity allows the integration of various devices, such as different types of cameras, sensors, and computing platforms. The ESP32-CAM, with its low-cost hardware and Wi-Fi connectivity, can seamlessly interact with other IoT devices and cloud platforms, regardless of differences in specifications. This flexibility enables the system to scale and adapt to various environments, ensuring compatibility across diverse inventory and resource management setups.

g) Security

Security in IoT ensures the protection of data, devices, and networks from unauthorized access and potential threats. In the ESP32-CAM and OpenCV-based object counting system, security features are crucial to safeguard inventory data and ensure privacy. The ESP32-CAM, with its Wi-Fi capabilities, can be secured using encryption protocols like WPA2 to protect the communication between devices and cloud platforms. Additionally, data transmitted for real-time object counting should be encrypted to prevent interception. Secure authentication mechanisms and access controls help prevent unauthorized access to the system, ensuring the integrity and confidentiality of sensitive inventory and resource data.

3.3. Advantages of IoT

a) Communication

IoT communication offers several advantages, especially in real-time inventory and resource management. With the ESP32-CAM and OpenCV system, communication enables seamless data exchange between devices, cloud platforms, and central servers, allowing for instant updates on object counts and inventory status. This communication ensures remote monitoring, enabling businesses to track resources from any location. Additionally, real-time data transfer reduces delays, improving decision-making and operational efficiency. IoT communication also supports scalability, enabling the system to grow with the business by adding more devices or expanding to multiple locations without losing efficiency or accuracy.

b) Automation and Control

IoT automation and control significantly enhance efficiency in real-time inventory and resource management. In the ESP32-CAM and OpenCV-based system, automation allows for continuous monitoring and object counting without manual intervention. The system automatically processes images, updates inventory data, and triggers actions based on real-time inputs, such as sending alerts for low stock levels. This reduces human error, speeds up inventory management, and minimizes operational costs. Additionally, IoT-based control allows businesses to remotely manage and adjust the system, improving responsiveness and ensuring resources are efficiently tracked and allocated, optimizing overall operational workflows.

c) Information

The advantage of IoT information in real-time inventory and resource management lies in its ability to provide accurate, up-to-date data for informed decision-making. In the ESP32-CAM and OpenCV-based system, IoT enables continuous collection and processing of object count data, which is transmitted to central platforms for analysis. This real-time information allows businesses to monitor inventory levels, track resources, and detect discrepancies instantly, improving operational efficiency. Additionally, the system's integration with cloud platforms enables easy access to historical and live data from anywhere, fostering better resource planning, reducing waste, and enhancing overall business operations.

d) Monitoring

The advantage of IoT monitoring in the ESP32-CAM and OpenCV-based object counting system lies in its ability to provide continuous, real-time oversight of inventory and resources. IoT monitoring allows businesses to track object counts remotely, ensuring accurate, up-to-date information without manual checks. This constant surveillance reduces human error and allows for immediate detection of discrepancies or low stock levels. With IoT-enabled monitoring, businesses can efficiently manage resources, identify trends, and make data-driven decisions. Furthermore, remote monitoring improves operational efficiency by enabling proactive actions, such as automated restocking alerts or adjustments, ensuring smoother operations and reduced downtime.

e) Efficiency

The advantage of IoT efficiency in the ESP32-CAM and OpenCV-based object counting system is its ability to streamline inventory management by automating data collection and analysis. The system continuously captures images, processes them in real time, and updates object counts without human intervention. This reduces the time and effort spent on manual inventory checks, accelerating the overall process. IoT connectivity allows for seamless data transfer to cloud platforms, enabling remote monitoring and quick decision-making. With accurate and up-to-date information, businesses can optimize resources, reduce waste, and improve operational workflows, resulting in significant time and cost savings.

3.4. Disadvantages of IoT

a) Compatibility

A key disadvantage of IoT in the ESP32-CAM and OpenCV-based object counting system is compatibility. Integrating various IoT devices with differing communication protocols, hardware, and software can be challenging. The ESP32-CAM may face compatibility issues with certain sensors or cloud platforms, limiting the system's scalability or flexibility. Additionally, different IoT devices might require custom configurations or adaptations, increasing setup time and complexity. Compatibility issues could also hinder seamless system integration.

b) Complexity

A disadvantage of IoT in the ESP32-CAM and OpenCV-based object counting system is the complexity of implementation and maintenance. The integration of hardware, software, and cloud platforms requires technical expertise, making setup challenging for non-technical users. Additionally, system updates, troubleshooting, and scaling can be complicated, requiring specialized knowledge. Managing data security, device compatibility, and connectivity issues also adds to the overall complexity of maintaining an efficient IoT-based solution.

c) Privacy/Security

A significant disadvantage of IoT in the ESP32-CAM and OpenCV-based object counting system is privacy and security concerns. As data is transmitted over networks, there is a risk of unauthorized access or data breaches. Sensitive inventory information can be intercepted or tampered with, leading to potential financial losses or data misuse. Ensuring robust encryption, secure authentication, and regular updates is essential to protect data privacy and secure the IoT system from vulnerabilities.

d) Safety

A disadvantage of IoT in the ESP32-CAM and OpenCV-based object counting system is safety risks. IoT devices, if not properly secured, can be vulnerable to hacking or unauthorized access, potentially leading to disruptions in inventory management or even physical safety hazards. For instance, unauthorized manipulation of the system could result in incorrect data, leading to overstocking or understocking. Ensuring proper system monitoring, security

protocols, and regular updates is essential for maintaining safety.

3.5. Application areas of IoT

The ESP32-CAM and OpenCV-based object counting system can be applied in various IoT-driven sectors, such as retail, warehouse management, manufacturing, and logistics. In these areas, IoT facilitates real-time inventory tracking, resource optimization, and automated monitoring. It can be used for stock management, supply chain monitoring, quality control, and asset tracking, providing accurate data, reducing human error, and enhancing operational efficiency across diverse industries.

3.6. IOT Technologies and Protocols

a) Bluetooth

Bluetooth is a wireless communication technology commonly used in IoT for short-range data transmission. In the ESP32-CAM and OpenCV-based object counting system, Bluetooth can enable device communication in environments where Wi-Fi is unavailable or impractical. It allows for the transfer of object count data between devices or to a local gateway, ensuring real-time monitoring and control. Bluetooth's low power consumption and ease of integration make it suitable for smaller-scale, localized IoT applications.

b) Zigbee

Zigbee is a low-power, wireless communication protocol designed for IoT applications, ideal for creating mesh networks. In the ESP32-CAM and OpenCV-based object

counting system, Zigbee can be used for reliable, long-range communication between multiple devices, such as cameras and sensors, in large environments like warehouses. Its low energy consumption and scalability make it well-suited for real-time inventory management, ensuring efficient data transfer over extended distances without compromising power efficiency.

c) Z-Wave

Z-Wave is a wireless communication protocol designed for home automation and IoT applications, focusing on low-power, secure communication over short to medium distances. In the ESP32-CAM and OpenCV-based object counting system, Z-Wave can facilitate reliable communication between devices, such as sensors and cameras, in environments like retail stores or warehouses. Its mesh network capabilities ensure stable data transmission, while its energy efficiency and security features are well-suited for real-time inventory management.

d) Wi-Fi

Wi-Fi is a widely used wireless communication protocol in IoT applications, offering high-speed data transfer and reliable connectivity. In the ESP32-CAM and OpenCV-based object counting system, Wi-Fi enables seamless real-time communication between the camera module and cloud platforms or central servers. It allows the system to transmit object count data over long distances, making it ideal for large-scale inventory and resource management. Wi-Fi’s widespread availability and scalability make it a suitable choice for IoT-based solutions.

e) Cellular

Cellular communication, such as 4G or 5G, offers wide-area connectivity for IoT devices, making it ideal for remote or mobile applications. In the ESP32-CAM and OpenCV-based object counting system, cellular networks can transmit real-time inventory data from locations without stable Wi-Fi access. This ensures that businesses can monitor resources in remote or large-scale environments, providing flexibility and scalability for real-time inventory management, even in areas with limited connectivity options.

f) NFC

NFC (Near Field Communication) is a short-range wireless technology that enables secure communication between devices when they are in close proximity. In the ESP32-CAM and OpenCV-based object counting system, NFC can be used for quick, on-site identification and data exchange, such as tagging objects or verifying inventory. While its range is limited, NFC provides an efficient, secure method for asset tracking and management in environments requiring fast, localized communication.

g) LoRaWAN

LoRaWAN (Long Range Wide Area Network) is a low-power, wide-area network protocol designed for long-range communication in IoT applications. In the ESP32-CAM and OpenCV-based object counting system, LoRaWAN enables reliable data transmission over large distances, ideal for monitoring remote or expansive inventory systems, such as in agriculture or large warehouses. Its low energy consumption and extended range make it suitable for real-time resource tracking in challenging environments with limited connectivity.

3.7. Project Layout

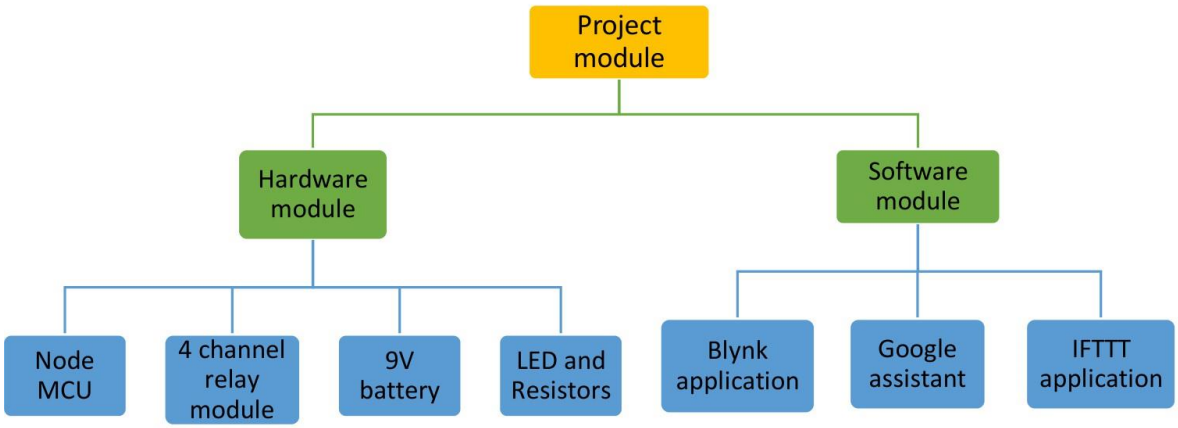


Figure 2. Layout of project module

3.7.1. Brief Description

The **ESP32-CAM and OpenCV-based Object Counting System** is a low-cost, real-time solution designed for inventory and resource management. This system leverages the capabilities of the ESP32-CAM, a compact microcontroller with an integrated camera, to capture images or video in an inventory area, while OpenCV, an open-source computer vision library, is used to process and analyze these images for object detection and counting.

3.7.2. Key Components:

3.7.2.1. **ESP32-CAM**: A microcontroller with Wi-Fi and Bluetooth support, ideal for remote monitoring and wireless communication. It is equipped with a camera that captures real-time images of objects in the inventory area.

3.7.2.2. **OpenCV**: A software library that performs real-time image processing, object detection, and counting. It

detects objects based on predefined features, such as shape or color, and can adapt to various environments.

3.7.3. Functionality:

3.7.3.1. The system captures images of inventory items, processes them through OpenCV algorithms, and counts the objects in real-time.

3.7.3.2. It can distinguish between different objects or categorize them, making it suitable for a variety of inventory environments such as warehouses, retail stores, or factories.

3.7.3.3. The ESP32-CAM enables remote monitoring by connecting the system to a network, allowing data to be accessed and managed from a distance.

3.7.4. Applications:

3.7.4.1. **Inventory Management**: Helps track the quantity of products or resources, reducing human error.

3.7.4.2. **Resource Tracking**: Ideal for managing equipment, tools, or other assets in a dynamic environment.

3.7.4.3. **Cost-Effective**: The combination of low-cost hardware and software makes this system accessible to small and medium-sized enterprises, improving automation and operational efficiency.

Chapter 04: Hardware Requirements

4.1. Arduino Uno

The hardware requirements for the ESP32-CAM and OpenCV-based object counting system include an **ESP32-CAM** module, a **camera** for image capture, **Arduino Uno** for additional control, and a **power supply**.

4.1.1. Features

The hardware requirements for the ESP32-CAM and OpenCV-based object counting system include:

- **ESP32-CAM Module**: Integrated microcontroller with Wi-Fi/Bluetooth, camera module for capturing real-time images.
- **Camera**: Captures high-resolution images for object detection and counting.
- **Arduino Uno**: Optional, used for additional control or interfacing with other peripherals.
- **Power Supply**: Provides necessary power to the ESP32-CAM and associated components.
- **Wi-Fi/Network Connectivity**: For real-time data transmission and remote monitoring.

4.1.2. Pin Configuration

The ESP32-CAM pin configuration for the object counting system involves several key pins for proper functioning. The **GPIO 0** is used for boot mode selection, while **GPIO 1 (TX)** and **GPIO 3 (RX)** are utilized for serial communication (transmit and receive). General-purpose I/O pins, such as **GPIO 4 to GPIO 15**, can be used for connecting additional external devices or sensors. The camera module connects to specific pins, including **D0, D1, D2, D3, D4, D5, D6, D7, XCLK, PCLK, VSYNC, and HREF**.

These pins enable communication between the ESP32 and the camera for image capturing. Additionally, **5V** and **GND** pins are used for powering the ESP32-CAM and associated components.

4.2. ESP–32

The ESP32-CAM and OpenCV-based object counting system utilizes the **ESP32 microcontroller**, which provides Wi-Fi and Bluetooth connectivity for real-time data transmission and remote monitoring. The ESP32-CAM is equipped with an integrated camera for capturing images of objects, which are processed by OpenCV for object detection and counting. This low-cost system efficiently manages inventory and resources by using the ESP32's powerful processing capabilities and wireless features for seamless operation.

4.3. Sensors

In the ESP32-CAM and OpenCV-based object counting system, additional sensors may include **motion sensors** (e.g., PIR) for detecting object movement, **proximity sensors** for identifying item presence, and **temperature or humidity sensors** for environment monitoring. These sensors enhance real-time inventory management by detecting changes in the environment or object status, improving accuracy and efficiency in resource tracking and management.

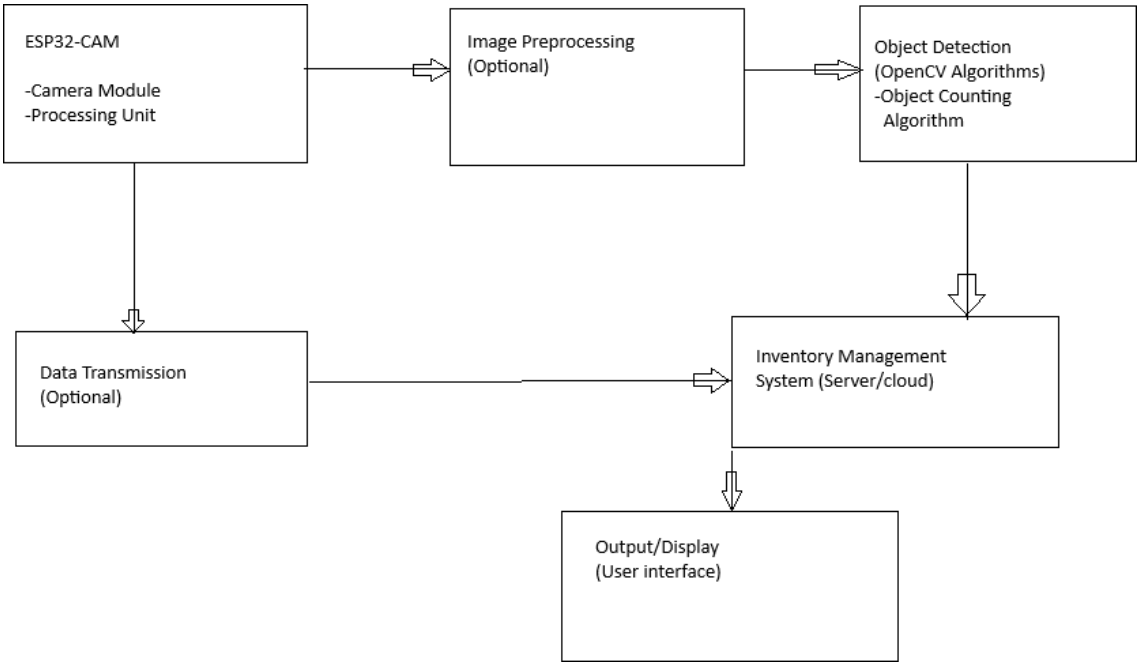
4.3.1. Pressure Sensor

In the ESP32-CAM and OpenCV-based object counting system, a **pressure sensor** can be integrated to detect weight or pressure changes when objects are placed or removed from a surface, such as a shelf or container. This sensor enhances inventory management by providing additional data, ensuring more accurate counting and monitoring of resources, and helping to track stock levels or detect item movement in real-time.

4.3.2. Temperature Sensor

In the ESP32-CAM and OpenCV-based object counting system, a **temperature sensor** (such as DHT11 or DHT22) can be used to monitor environmental conditions in real-time. This sensor ensures the system operates optimally by tracking temperature fluctuations, which could affect inventory items sensitive to heat. Integrating this sensor helps maintain better control over inventory, especially for temperature-sensitive products in warehouses or storage areas.

4.5. Block diagram of the proposed system

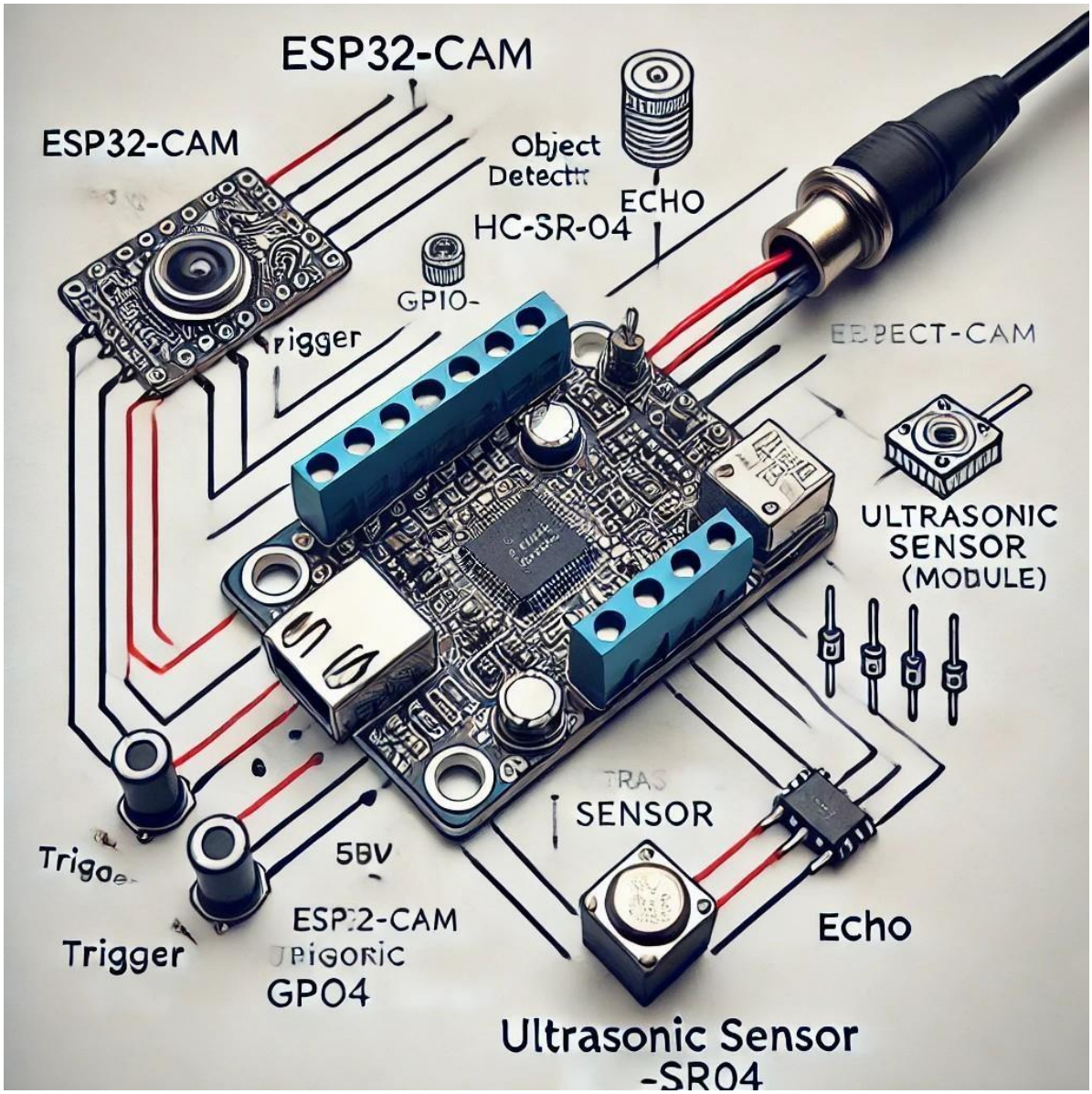


4.5.1. Working of the system

The **ESP32-CAM and OpenCV-based object counting system** works by capturing images using the ESP32-CAM module. The ESP32 is connected to a Wi-Fi network and acts as a server, transmitting images to a connected computer or Raspberry Pi. On

the server side, **OpenCV** processes the images to detect and count objects. The images are first preprocessed (grayscale conversion, thresholding), and then **contour detection** is used to identify individual objects. The number of objects is counted by detecting contours and drawing bounding boxes. This real-time data helps track inventory or resources for efficient management.

4.5.2. Circuit Diagram



4.5.3. Components Required

Table 1. Component listing.

Sl. No.	Component and Specification	Quantity
1.	ESP32DEVBOARD38PIN	1
2.	ESP CAMERA	1
3.	USB TO TTL CONVERTOR	1
4.	ESP32 CAMMPUSBBOARDBLACK	1
5.	MICRO SDCARD8GB	1
6.	JUMPERWIREF-F	20
7.	USB TO USB CONNECTOR	1

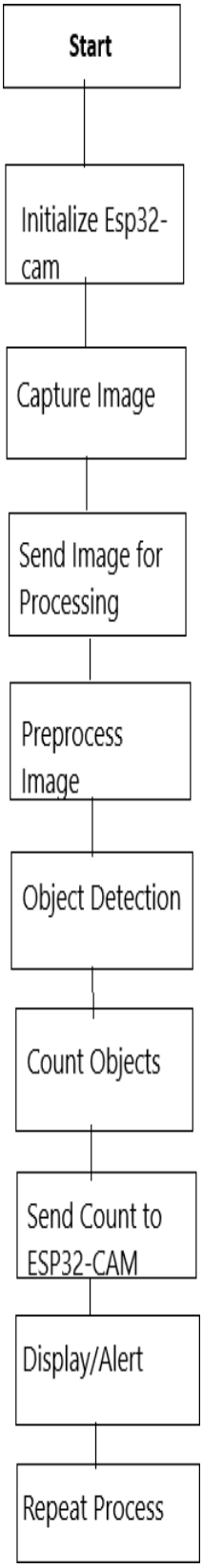
Chapter 05: Software Requirements

5.1. Arduino IDE (Embedded C / C++)

For the **ESP32-CAM and OpenCV-based object counting system**, the primary software requirement is the **Arduino IDE**, which is used for programming the ESP32-CAM module in **Embedded C/C++**. Key steps include:

1. **Arduino IDE**: Install the latest version to program the ESP32-CAM. This allows users to write, compile, and upload code to the ESP32.
2. **ESP32 Board Support**: Add ESP32 board definitions through the **Arduino Board Manager** to enable compatibility with the ESP32-CAM module.
3. **Required Libraries**:
    - **WiFi.h**: For Wi-Fi connectivity, allowing the ESP32-CAM to connect to a local network.
    - **esp_camera.h**: Provides functions to interface with the camera module, enabling image capture.
    - **WebServer.h**: Creates a web server on the ESP32-CAM for transmitting captured images to the server.
4. **OpenCV (Server-Side)**: Install **opencv-python** and **numpy** on the server to process images, detect objects, and count them in real-time.
5. **Serial Monitor**: Use the monitor for debugging and tracking the system's performance and image transmission.

5.2. Logic and Flowchart



Chapter 06: Project development & Testing Aspects

The development and testing of the **ESP32-CAM and OpenCV-based object counting system** for real-time inventory management involves several key phases:

6.1. Development:

- **6.1.1 Hardware Setup**: Connect the **ESP32-CAM** module to a camera, ensure proper power supply, and establish Wi-Fi connectivity.
- **6.1.2 Software Programming**: Using **Arduino IDE**, write code to initialize the camera, capture images, and send them via Wi-Fi to a server. Use libraries like **WiFi.h**, **esp_camera.h**, and **WebServer.h**.
- **6.1.3 Image Processing**: On the server side, use **OpenCV** and **numpy** to process images, detect objects, and count them based on contours.

6.2.Testing:

- **6.2.1 Functionality Testing**: Test individual components like Wi-Fi connection, image capture, and transmission.
- **6.2.2 Image Processing Accuracy**: Validate the accuracy of object detection and counting under various conditions (e.g., different lighting and object types).
- **6.2.3 Real-Time Performance**: Assess the system's real-time performance by monitoring object counts in live settings.
- **6.2.4 Optimization**: Adjust image resolution, frame rate, and threshold values for optimal system performance.

Through continuous testing and debugging, the system is refined for reliable inventory and resource management.

Chapter 07: Conclusion & Future Scope

7.1. Result

The ESP32-CAM and OpenCV-based object counting system provides an affordable and effective solution for real-time inventory and resource management. It leverages computer vision technology to ensure accurate object detection and tracking, making it an ideal choice for small to medium-scale applications. Future developments could focus on integrating advanced machine learning models to improve accuracy in challenging conditions such as low light and object occlusion. Additionally, scalability improvements could allow the system to handle large-scale inventories in industries like retail, logistics, and manufacturing, further enhancing automation, reducing human error, and optimizing resource management in diverse environments.

7.2. Conclusion

The ESP32-CAM and OpenCV-based object counting system offers a low-cost, efficient solution for real-time inventory and resource management. By utilizing computer vision, it enables accurate object detection and counting, streamlining operations in various industries. This system's affordability makes it accessible for small and medium-sized businesses. Future improvements could involve integrating advanced machine learning algorithms to enhance detection accuracy in challenging environments, such as low lighting or high object density. Additionally, expanding its capabilities to handle large-scale inventories and incorporating automated reporting could further optimize resource management and increase efficiency in sectors like retail, manufacturing, and logistics.

7.3. Limitations

The ESP32-CAM and OpenCV-based object counting system, while cost-effective, faces several limitations. The processing power of the ESP32-CAM is limited, which may affect the system's performance when handling complex or high-volume counting tasks. Environmental factors such as poor lighting, reflections, or object occlusions can reduce the accuracy of object detection. Additionally, the system might struggle with real-time processing for large inventories or fast-moving objects. Scalability could also be a challenge for large-scale deployments, requiring hardware or software upgrades. Addressing these limitations is essential for improving the system’s reliability and efficiency in diverse operational environments.

.

7.4. Further Enhancement and Future Scope

Future enhancements for the ESP32-CAM and OpenCV-based object counting system could focus on integrating machine learning algorithms, such as deep learning-based object detection, to improve accuracy and robustness, especially in complex environments with varying lighting and occlusions. The system could also benefit from the use of more powerful processors or edge AI solutions to enhance real-time performance for larger inventories. Additionally, expanding the system’s scalability to handle high-volume environments and incorporating features like automated stock level reporting, cloud integration, and predictive analytics could further optimize resource management across industries such as retail, logistics, and manufacturing.

References

https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/

https://www.youtube.com/watch?v=9fER61M3yFo

Appendix 01

A01.1. Code Listing

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

const char* WIFI_SSID = "iot_11";
const char* WIFI_PASS = "rpw1@iot";

WebServer server(80);



static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
  Serial.println("CAPTURE FAIL");
  server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
```

```
}

void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}


void setup(){
  Serial.begin(115200);
  Serial.println();
  {
```

```
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");

  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);

  server.begin();
}

void loop()
{
```

```
  server.handleClient();
}
```

A01.2. Main Code

```
import cv2
import matplotlib.pyplot as plt
import cvlib as cv
import urllib.request
import numpy as np
from cvlib.object_detection import draw_bbox
import concurrent.futures

url='http://192.168.10.162/cam-hi.jpg'
im=None

def run1():
    cv2.namedWindow("live transmission", cv2.WINDOW_AUTOSIZE)
    while True:
        img_resp=urllib.request.urlopen(url)
        imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
        im = cv2.imdecode(imgnp,-1)

        cv2.imshow('live transmission',im)
        key=cv2.waitKey(5)
        if key==ord('q'):
            break

    cv2.destroyAllWindows()
```

```
def run2():
    cv2.namedWindow("detection", cv2.WINDOW_AUTOSIZE)
    while True:
        img_resp=urllib.request.urlopen(url)
        imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
        im = cv2.imdecode(imgnp,-1)

        bbox, label, conf = cv.detect_common_objects(im)
        im = draw_bbox(im, bbox, label, conf)

        cv2.imshow('detection',im)
        key=cv2.waitKey(5)
        if key==ord('q'):
            break

    cv2.destroyAllWindows()



if __name____== '__main__':
    print("started")
    with concurrent.futures.ProcessPoolExecutor() as executer:
            f1= executer.submit(run1)
            f2= executer.submit(run2)
```

A01.3. Libraries

ESP32 Libraries:

- **`WiFi.h`**:
  - Enables Wi-Fi connectivity for the ESP32, allowing the module to connect to a local network.
  - Facilitates image transmission from the ESP32-CAM to a server.
- **`esp_camera.h`**:
  - Provides the interface for controlling the camera module on the ESP32-CAM.
  - Allows image capture, initialization, and frame fetching.
- **`WebServer.h`**:
  - Implements a simple web server on the ESP32.
  - Handles HTTP requests to capture images and send them over the network.
- **`Arduino.h`**:
  - The core library for programming the ESP32-CAM with the Arduino IDE.
  - Provides basic functions for digital I/O, timing, and data handling.

OpenCV Libraries:

- **`opencv-python`**:
  - Python bindings for OpenCV, used for image processing, object detection, and analysis.
  - Essential for tasks like object counting and contour detection.
- **`numpy`**:
  - Handles image data as arrays, allowing manipulation of pixel data for processing.

Appendix 02

A02.1. Project Proposal Form

The project proposal form was prepared and duly signed from our Faculty-in-Charge Dr. Biswaranjan Swain. The same is attached at the last of this report.

A02.2. Project Management

#	Component	Individual Contributions in %				Total
		Abhishek	Archit	Jaydev	Debadutta	
1.	Planning	25%	25%	25%	25%	100%
2.	Background Research and Analysis	25%	25%	25%	25%	100%
3.	Hardware design	70%	10%	10%	10%	100%
4.	Software design	40%	20%	20%	20%	100%
5.	Testing	70%	10%	10%	10%	100%
6.	Final Assembling	25%	25%	25%	25%	100%
7.	Project report writing	10%	40%	40%	10%	100%
8.	Presentation	25%	25%	25%	25%	100%
9.	Logistics	25%	25%	25%	25%	100%

A02.3. Bill of Material

Table 1. Component listing.

#	Component	Specification	Unit Cost	Quantity	Total
1.	ESP32DEVBOARD38PIN	Versatile	370	1	370
2.	ESP CAMERA	Compact	510	1	510
3.	USBTOTTLCONVERTOR	Adapter	170	1	170
4.	ESP32CAMMPUSBBOARDBLACK	Development	110	1	110
5.	MICRO SDCARD8GB	Storage	200	1	200
6.	JUMPERWIREF-F	Connector	2	20	40
7.	USB TO USB CONNECTOR	Adapter	350	1	350
				Grand Total	**1750**

Appendix 03

A03.1. Data Sheets

1. ESP32DEVBOARD38PIN

Datasheet: The ESP32 DEVBOARD 38-pin features dual-core processor, Wi-Fi, Bluetooth, 38 GPIO pins, ADC, DAC, and various interfaces.

Specifications:

- Dual-core 32-bit processor (XTensa)
- 38 GPIO pins
- Wi-Fi and Bluetooth connectivity
- 4MB Flash memory
- Integrated camera interface (OV2640)
- Supports I2C, SPI, UART, PWM
- Onboard voltage regulator
- ADC, DAC support
- Compatible with Arduino IDE and ESP-ID

2. ESP CAMERA

Datasheet: The ESP32-CAM features a 2MP OV2640 camera, Wi-Fi/Bluetooth, 4MB Flash, 38 GPIO pins, supports Arduino/ESP-IDF development.

Specifications:

- 2MP OV2640 camera module
- ESP32 microcontroller with Wi-Fi and Bluetooth
- 4MB Flash memory
- 38 GPIO pins
- Supports UART, SPI, I2C, PWM
- Onboard microSD card slot
- Supports Arduino IDE and ESP-IDF programming
- Integrated antenna for wireless connectivity

3. USB TO TTL CONVERTOR

Datasheet: The USB to TTL converter enables serial communication between USB and TTL devices, supporting 3.3V/5V levels, with FTDI chipset.

Specifications:

- Converts USB to TTL serial signals
- Supports 3.3V and 5V logic levels

- FTDI or CH340 chipset
- Standard USB-A to micro USB or USB-B cable
- Compatible with Windows, Linux, and macOS
- Used for serial communication/debugging
- Baud rates up to 115200 bps

4. ESP32CAMMPUSBBOARDBLACK

Datasheet: ESP32-CAM MP USB Board Black features an ESP32 chip, 2MP camera, USB interface, supports Wi-Fi/Bluetooth, 4MB Flash memory.

Specifications:

- ESP32 microcontroller with Wi-Fi and Bluetooth
- 2MP OV2640 camera module
- 4MB Flash memory
- Micro USB interface for power and programming
- 38 GPIO pins
- Onboard microSD card slot
- Supports Arduino IDE and ESP-IDF development
- Compatible with OpenCV for image processing

5. MICRO SDCARD8GB

Datasheet: The Micro SD Card 8GB provides storage for data, supports high-speed transfer, compatible with ESP32-CAM for image storage.

Specifications:

- 8GB storage capacity
- Compatible with microSD card slots
- Supports high-speed data transfer
- FAT32 file system
- Ideal for storing images, videos, and logs
- Durable and reliable for embedded systems
- Low power consumption
- Easy integration with ESP32-CAM