Module Code & Module Title

CC5068NI– Cloud Computing & IoT

"Smart IOT Object Detector"

Assessment Type

40% System Development Report

Semester 2025 Autumn

Group members

London Met ID	Student Name
23047380	Prashika Sharma
23048525	Archit Bhakta Shrestha
23047398	Sukhi Mali
23047337	Akshit Shah
23047537	Anuj Bhakta Shrestha

Assignment Due Date: 15 May 2025

Assignment Submission Date: 15 May 2025

Word Count: 4020

23047380 Prashika Sharma L2N4 Team2 .docx

Islington College,Nepal

Document Details

Submission ID

trn:oid:::3618:95993809

Submission Date

May 15, 2025, 11:55 AM GMT+5:45

Download Date

May 15, 2025, 11:56 AM GMT+5:45

File Name

23047380 Prashika Sharma L2N4 Team2 .docx

File Size

27.6 KB

23 Pages

4,020 Words

23,485 Characters

7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

28 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

3 Missing Quotations 1%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1%	Internet sources
2%	Publications
6%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Table of Contents

Table of Figures

Table of Tables

Acknowledgement

We would like to sincerely thank our module instructors, Mr. Jaganath Paudyal and Mr. Sugat Man Shakya, for their constant guidance and encouragement throughout the course of our project. Their support and insightful advice played a key role in helping us navigate challenges and successfully achieve our objectives.

We are also grateful to the Resources Department of our college for providing the essential hardware components and tools needed for the project. Their assistance ensured that both the development and testing phases went smoothly and efficiently.

We sincerely thank the developers of open-source tools like Arduino IDE, ESP32-CAM libraries, and YOLO for enabling us to build and test our Smart IoT Object Detection System efficiently and affordably.

Abstract

The Smart IoT Object Detector project uses an ESP32-CAM and an ultrasonic sensor to create a real-time object detection system aimed at enhancing safety and automation in areas such as smart vehicles and home automation. The system identifies nearby objects, measures their distance using the ultrasonic sensor, and provides details through an LCD screen. When an object comes within a certain range, a buzzer is triggered to give an audible warning. The ESP32-CAM also enables live video streaming, allowing remote monitoring.

This project brings together hardware like the Arduino Uno, sensors, and actuators, alongside software tools such as the Arduino IDE and Tinkercad for simulation and testing. The system's performance in detecting objects, triggering alerts, and streaming video was verified through practical testing. The object detection was implemented using YOLOv8 with Python and OpenCV, enabling fast and accurate real-time recognition of common objects without the need for custom training.

Looking ahead, the system could be further improved in the coming days, by adding cloud connectivity, using advanced machine learning for object recognition, and optimizing power usage. Overall, the project highlights how IoT technologies can be used to build smart, responsive systems for real-time, real-world application.

Smart IoT Object Detection using ESP32-CAM

Part 1: Introduction

The Internet of Things (IoT) connects commonplace objects to the Internet, transforming our interactions with the environment. These technologies can gather, analyze, and distribute data in real-time. The Internet of Things enhances efficiency, safety, and convenience in smart homes, self-driving vehicles, healthcare, and farms. According to IBM (2023), IoT-enabled systems can reduce the need for human interaction, improve decision-making processes, and lead to more reactive systems, facilitating real-time monitoring and control. (IBM, 2023)

1.1 Project Overview

The initiative, called "Smart IoT Object Detection System," provides a cost-effective and compact method for detecting nearby objects using an ESP32-CAM module in conjunction with an ultrasonic sensor. It calculates the space between itself and approaching objects and presents this information on a compact LCD display. concurrently, the ESP32-CAM takes pictures and can detect basic items like humans or bottles. When an object comes too near, a buzzer sounds like an alert, making the system perfect for scenarios that need a prompt reaction.

This approach can be successfully applied to:

- Autonomous vehicles to assist in avoiding barriers,
- Industrial configurations for ensuring safer machinery functioning,
- Systems for security and monitoring,
- Intelligent home devices such as garage door systems or robotic vacuum cleaners.

This system can be utilized in the following sectors:

- Real-Time Tracking: Information on object distance and detection is available in real time, including from remote locations.
- Automated Notifications: Functions such as automatic alarms aid in avoiding risks or mishaps.
- Cost-effective and Compact: It is constructed with commonly accessible and economical components such as the ESP32-CAM, HC-SR04 sensor, and a buzzer.

- Expandable Design: Additional sensors and sophisticated tools, such as YOLO, may be incorporated to enhance the accuracy and object identification moving forward. (Saiwa, 2025)

1.2 Current Scenario

With intelligent monitoring and automation leading the way in applications in sectors like transportation and home security, real-time object detection is crucial. Nevertheless, many current systems still utilize legacy devices or human monitoring, causing latency, human mistakes, and inefficiency. In high-speed environments like autonomous vehicles or robots, these failures increase the probability of accidents, breakdowns, and costly downtime, highlighting the need for intelligent and more efficient detection solutions now.

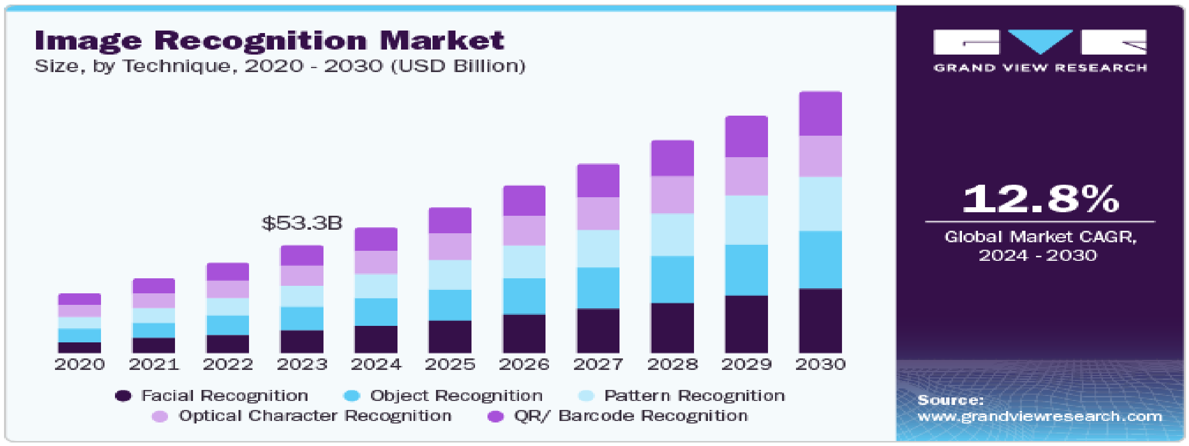


Figure 1: Object Recognition Market (GrandViewResearch, 2025)

The market size of image recognition was valued at USD 53.3 billion in 2023 across the world. The market is expected to increase from USD 62.3 billion in 2024 to USD 128.3 billion in 2030 throughout the forecasted period. It will grow at a CAGR of 12.8% during 2024–2030. The historic period considered is 2018–2022, and the forecast period considered is 2024–2030. (GrandViewResearch, 2025)

1.3 Problem Statement

The increasing need for real-time object detection in areas such as transportation and security faces obstacles including short range, low accuracy, and inadequate IoT compatibility, resulting in possible accidents and inefficiencies in systems

Despite technological developments, many object detection frameworks rely on expensive equipment such as LIDAR and Raspberry Pi boards, rendering them unavailable to students and individuals working on modest projects. Furthermore, many systems lack real-time connectivity and seamless interaction with modern smart surroundings, which limits their functionality and prevents remote operation. Their detection skills are often relatively simple, signaling whether something exists without providing much context about what the item actually is. (Li Yangyang, 2020)

1.4 Project as a Solution

To overcome the drawbacks of traditional systems, this project presents a low-cost, real-time object detection and monitoring system constructed with an ESP32-CAM and an ultrasonic sensor. The system was created with accessibility and affordability in mind, providing a less costly alternative to high-end technologies like LIDAR and Raspberry Pi, making it suitable for small projects, hobbyists, and students. Its live video feed and continuous distance reading support both visual recognition and obstacle detection. Unlike most basic systems, it enhances contextual awareness by complementing sensor information with real-time visuals. Additionally, the configuration enables smooth integration with IoT environments, enabling remote control and real-time alerts when an object crosses a predetermined threshold, thus improving responsiveness and intelligence of systems in smart environments. (Aniobi, 2024)

1.5 Aim and Objectives

The main aim of this project is to design and implement a smart IoT-based object detection system's prototype using ESP32-CAM and Ultrasonic sensor to recognize the object and measure its distance.

Objectives

- To build an affordable real-time object detection system using ESP32-CAM and Arduino UNO with a YOLO model, eliminating the need for a Raspberry Pi.
- To use an ultrasonic sensor for distance monitoring, LCD display and buzzer alerts when objects breach a set proximity limit.
- To test and evaluate the system's accuracy, responsiveness, and reliability in detecting and alerting obstacles in various scenarios.

Part 2: Background

2.1 System Overview

The proposed system is a Smart IoT Object Detection solution that uses an ESP32-CAM and an ultrasonic sensor to detect and identify nearby objects in real time. It is designed to improve safety and support automation in various settings.

The ESP32-CAM functions as the main controller provides video streaming, while object detection is handled via a Python-based YOLOv8 model, capable of recognizing items such as humans, bottles, and other everyday objects. The ultrasonic sensor measures how close an object is, and when something comes within a certain range, the system responds with both visual alerts on an LCD screen and sound alerts using a buzzer. This immediate feedback helps ensure quick awareness of any obstacles. An Arduino Uno is used to coordinate and manage the communication between sensors and alert components. The system follows a modular structure, making it simple to modify or expand in the future. Each component, including the ultrasonic sensor, LCD, and alert system, is connected through jumper wires, forming a complete and functional setup.

By integrating sensors, actuators, microcontroller and wired communication, the system enables real-time monitoring, automatic responses, and improved safety. Its flexibility and ability to adapt to different needs make it a practical and scalable option for a wide range of smart technology applications. This IoT-based solution is well-suited for use in smart vehicles, robotics, and home automation, where detecting and avoiding obstacles is essential.

2.2 Design Diagram

This section includes visual diagrams that help explain the design and working process of the Smart IoT Object Detection system. These visuals are useful for understanding how the different components are connected and how the system detects nearby objects and sends out alerts.

2.2.1 System Architecture Diagram

The system architecture diagram presents the overall layout of the IoT-based object detection system. It shows how key components like the ESP32-CAM, ultrasonic sensor, LCD, Arduino and buzzer are connected and interact to perform the intended functions.

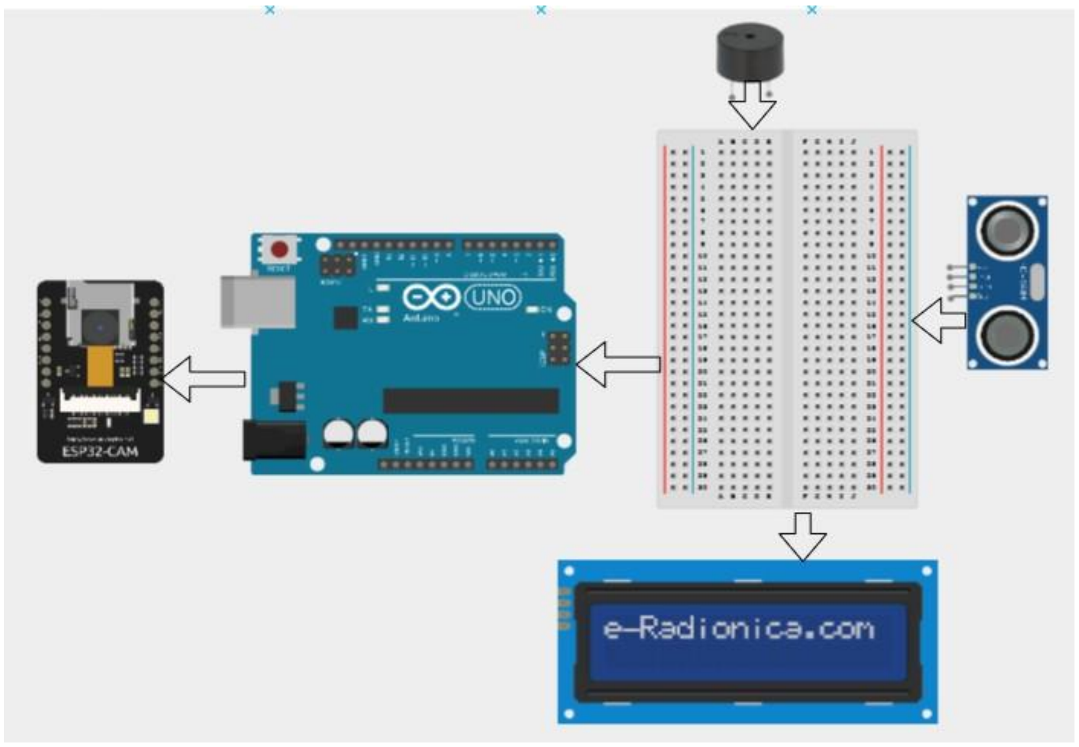


Figure 2: System Architecture

2.2.2 Block Diagram

This diagram illustrates the functional flow of the system.

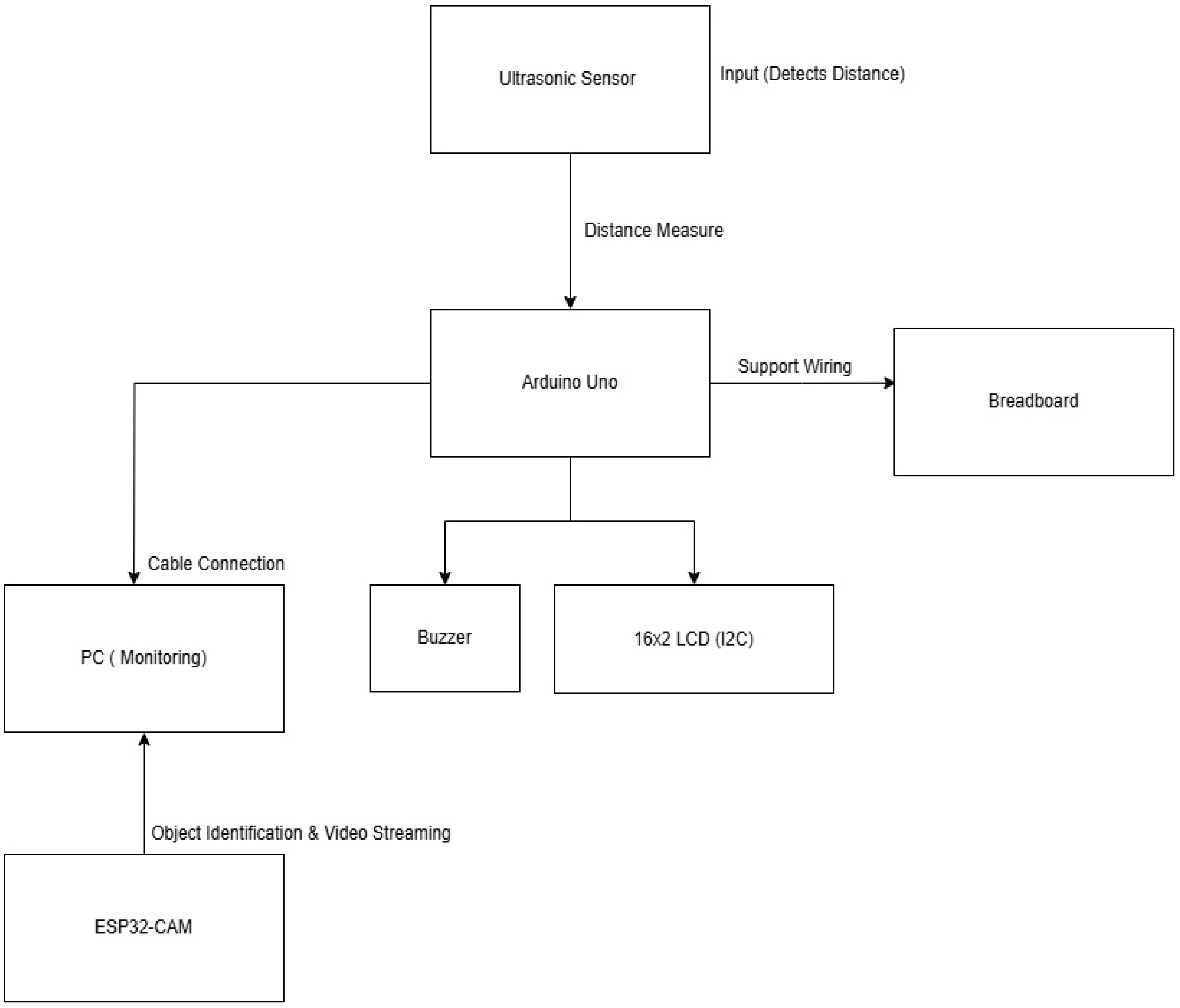


Figure 3: Block Diagram

2.2.3 Flowchart

The flowchart represents the step-by-step logic behind how the system operates. This visual guide helps clarify the system's decision-making process and overall workflow.

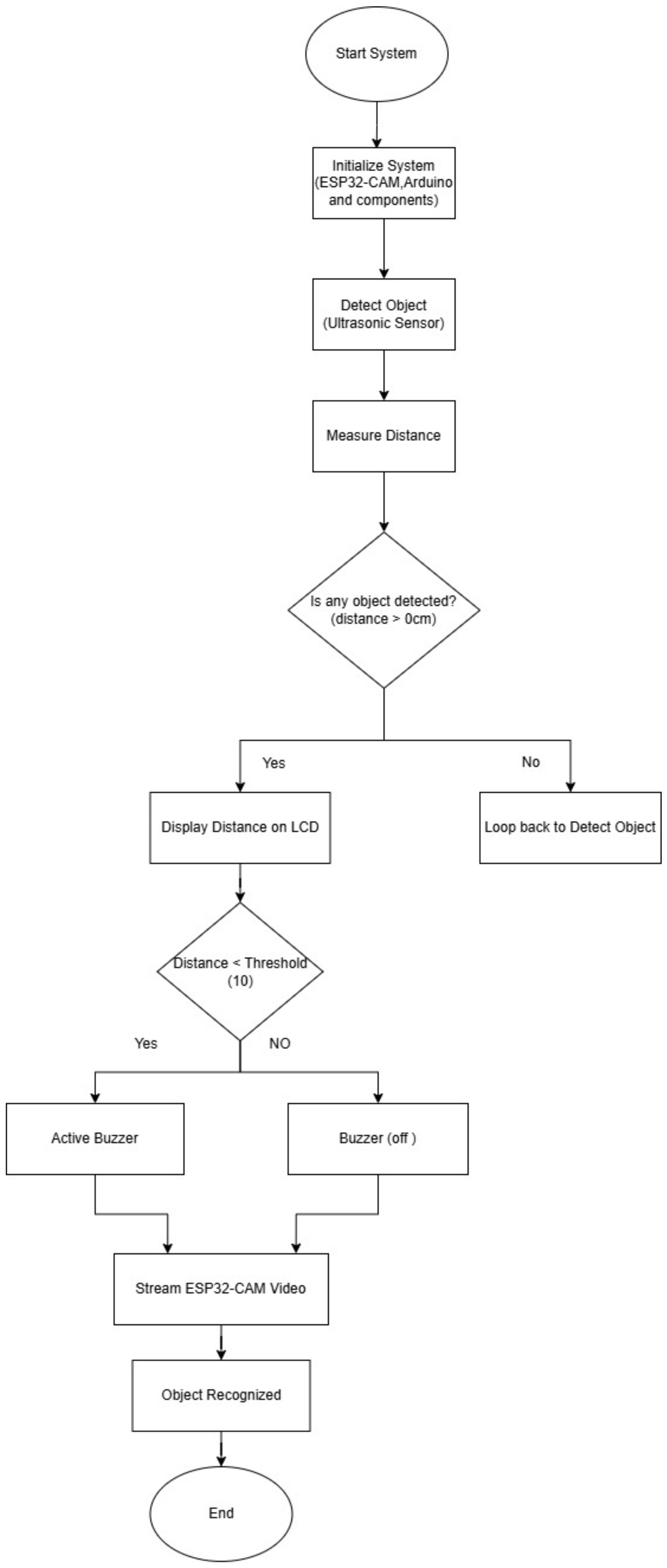

Start System
Initialize System (ESP32-CAM,Arduino and components)
Detect Object (Ultrasonic Sensor)
Measure Distance
Is any object detected? (distance > 0cm)
Yes
No
Display Distance on LCD
Loop back to Detect Object
Distance < Threshold (10)
Yes
NO
Active Buzzer
Buzzer (off)
Stream ESP32-CAM Video
Object Recognized
End

Figure 4: Flowchart

2.2.4 Circuit Diagram

The circuit diagram illustrates how all the components of the system are wired together showing the physical setup.

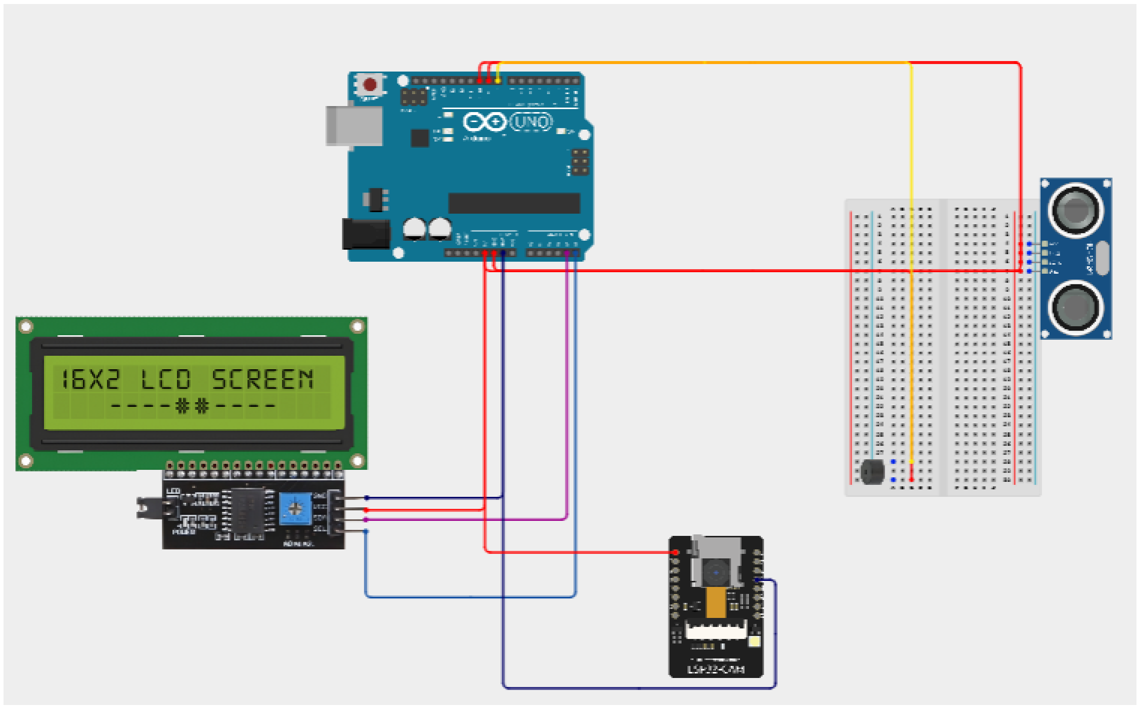


Figure 5: Circuit Diagram

2.3 Requirement Analysis

This section lists the key hardware and software components needed to build the IoT-based object detection system.

2.3.1 Hardware Requirements

- ESP32-CAM: A flexible microcontroller with built-in Wi-Fi and an integrated camera module, the ESP32-CAM enables live video streaming and remote monitoring. It also acts as the main controller, handling object detection and processing tasks within the system.

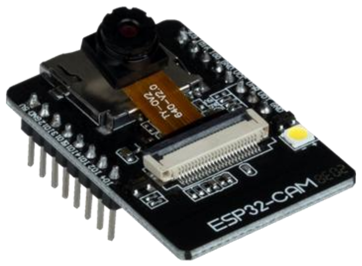


Figure 6: ESP32-CAM

- Arduino Uno: It acts as the central microcontroller for managing the system's sensors and actuators. It handles input from the ultrasonic sensor and controls the LCD and buzzer, processing the data and delivering the appropriate outputs .

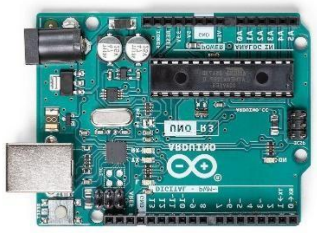


Figure 7: Arduino Uno

- Breadboard: A prototyping platform that allows components to be connected without soldering. It offers a flexible and temporary setup, making it ideal for testing and refining the system before the final assembly.

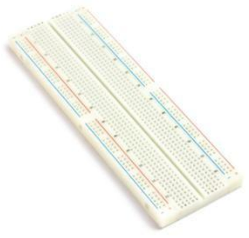


Figure 8: Breadboard

- Ultrasonic Sensor (HC-SR04): It is a sensor that measures distance by emitting ultrasonic waves and calculating the time it takes for the echo to bounce back. This allows the system to detect nearby objects and determine how far they are (SparkFun Electronics , 2025).

Figure 9: Ultrasonic Sensor

- 16x2 LCD Display: A liquid crystal display used to present the real-time distance measured by the ultrasonic sensor, providing users with a clear and easy-to-read visual interface.

Figure 10: LCD

- Buzzer: A compact audio device that emits a sound alert when an object is detected within a certain range, signaling the presence of an obstacle.

Figure 11: Buzzer

- Jumper Wires: They are used to connect the microcontroller, sensors, and other components on the breadboard. They enable reliable data transmission and power delivery throughout the system.

Figure 12: Jumper Wires

2.3.2 Software Requirements

Arduino IDE: The main development platform used for writing, compiling, and uploading code to both the Arduino Uno and ESP32-CAM. It allows for easy integration of libraries to control sensors and other hardware components (Mohamed Fezari, 2018).

Figure 13: Arduino IDE

- Tinkercad: An online platform for designing, prototyping, and testing electronic circuits virtually, helping users visualize, experiment, and troubleshoot before building the physical system.
- Python: It is used to run the YOLO algorithm for real-time object detection with the ESP32-CAM, handling video processing, streaming, and visualization efficiently (Nur Adila Husin, 2022).
- MS Word: It is used for creating comprehensive project documentation, including designs, diagrams, code details, and system architecture reports.
- Draw.io: It is used to visually represent the system's architecture, flow, and component interactions for clear understanding of the IoT object detection setup.

Part 3: Development

This section outlines the step-by-step process followed to build the Smart IoT Object Detection system. It includes everything from the initial planning and design stages to the assembly of components, programming, and final system integration.

3.1 Planning and Design

The objective of the project was to develop an intelligent IoT system that could perform real-time video monitoring and object detection. For this purpose, the ESP32-CAM module was used because it is inexpensive, small in size, comes with a built-in camera, and provides Wi-Fi connectivity, making it suitable for edge-level monitoring applications. It was made to connect to a local Wi-Fi network and provide video streaming using an IP address, enabling real-time monitoring on any device on the network.

The object detection feature was built using Python, OpenCV, and Ultralytics's' YOLOv8 system. YOLOv8 was utilized due to its real-time capabilities and capacity to identify more than 80 object classes from a pre-trained COCO dataset model. Python virtual environment was used to encapsulate dependencies. The system analyses every video frame from the stream, detects objects, and provides bounding boxes as well as class labels and confidence scores with OpenCV.

During the planning stage, a simulation of the circuit was done using Tinkercad, making use of available components such as an ultrasonic sensor, Arduino Uno, LCD, and buzzer to demonstrate the intended configuration. Even though Tinkercad lacked support for the ESP32-CAM, this virtual simulation assisted in the visualization of the system. The modular approach allows for scalability and future upgrades, for example, adding alerts, object category expansion, or transitioning to cloud processing.

3.2 Resource Collection

All the necessary hardware parts i.e., the Arduino Uno, HC-SR04 ultrasonic sensor, ESP32-CAM, 16x2 LCD display, buzzer, and battery were obtained from the Islington College Resource Department. These parts constituted the basis of the physical configuration of the system.

Simultaneously, the team recognized and incorporated the requisite software libraries and utilities to enable development. These were uploading code, monitoring serial data, and simulating circuits where necessary. The team worked in tandem in exercises like wiring parts, soldering, and hardware testing to guarantee seamless integration and operation during the project.

Components	Quantity	Remarks
Arduino Uno	1	Microcontroller
HC-SR04 Ultrasonic Sensor	1	Sensor
Breadboard	1	Other (Prototyping)
16x2 LCD (I2C)	1	Actuator (Visual Output)
Buzzer	1	Actuator (Audio Output)
ESP32-CAM	1	Microcontroller + Sensor
Jumper Wires	7-14	Others (Connectors)

Table 1: Resources

3.3 System Development

Step 1: Integration of Ultrasonic Sensor

Component Used: Ultrasonic Sensor (HC-SR04)

In the early stages of development, the Arduino Uno was interfaced with the Ultrasonic Sensor HC-SR04 to serve as the foundation for the object detection system. The sensor can measure distance without making contact because it is an active digital proximity sensor. The sensor's transmitter sends out ultrasonic waves. The sensor's receiver receives the reflected waves when they strike an object in front of it. The time between sending and receiving is measured by the Arduino. The Arduino uses the following formula to determine the object's distance:

Distance (cm) = (Time × Speed of Sound) / 2

The object detection capability needed for later system development was established during this phase.

Step 2: Integration of LCD Display

Component Used: 16x2 Liquid Crystal Display (LCD)

At this stage, the Arduino Uno was interfaced with a 16x2 LCD to provide a visual output of the distance measured by the ultrasonic sensor in real time. By serving as both a user interface module and a digital output actuator, the module improves the system's usability and interactivity.

Depending on proximity thresholds, the LCD showed either the actual distance (in centimeters) or context-dependent messages like "Object Detected." This made it possible for users to read sensor data straight from the Arduino without having to look at the serial monitor or connect it to a computer.

The Arduino IDE's LiquidCrystal library was used to power the LCD. The feedback was accurate and timely because the display was continuously updated with new data from the ultrasonic sensor. Because of this integration, the system is now more intuitive and suitable for standalone operation.

Step 3: Integration of Buzzer

Component Used: Buzzer

During this stage, the Arduino Uno was also equipped with a buzzer to serve as an auditory indication system, which further enhanced the system's capability to report object detection events as sound. When an object was detected within a predetermined proximity threshold of less than 10 cm, the buzzer, which functioned as a digital output actuator, was set to sound.

When it is not possible to visually inspect the LCD, the buzzer's ability to alert users to nearby objects is particularly helpful. Real-time situational awareness is greatly enhanced by this feature, which alerts the user without requiring them to look at the device.

The Arduino was able to properly activate the buzzer by routinely comparing the distance measured by the ultrasonic sensor with the threshold. this phase consequently being a valuable enhancement for responsiveness and accessibility overall in the system.

Step 4: Integration of ESP32-CAM

During this stage, the ESP32-CAM module was configured to connect to a nearby Wi-Fi network in order to stream live video within the network. Smart monitoring was made possible by the fact that users could watch the live stream from any device linked to the same network.

A Python-based object detection system using OpenCV and Ultralytics's YOLOv8 model was developed for optimal performance. Without further training, YOLOv8, which has been pre-trained on the COCO dataset, can identify more than 80 common object classes, including people, cars, bottles, and bicycles.

Dependencies were managed using a Python virtual environment, and the ESP32-CAM's IP stream or webcam video feed was processed using OpenCV. The script processed each video frame, displaying bounding boxes, class labels, and confidence on each. For clarity, it used different colors for each object class.

This marked a significant advancement from basic monitoring to an intelligent, AI-powered IoT system aligned with Smart IoT principles.

Step 5: Final Hardware Assembly

At the project's conclusion All of the independently developed modules were combined into a complete and operational object detection and monitoring system at the project's conclusion. Physical component integration and perfect connectivity between components were the main priorities at this stage.

Roles of Components:

- Breadboard: Serves as the framework for adaptable and modular merging with all of the components. Non-permanent, solderless connections made with a breadboard make it easier to test, troubleshoot, and reconfigure connections throughout the development cycle.
- Arduino Uno: The Arduino Uno served as the main controller for the LCD, buzzer, and ultrasonic sensor and was powered by the Arduino IDE. It ran continuously, using an ultrasonic sensor to measure distance, updating the LCD in real time with distance information or detection messages, and setting off the buzzer when an object within a predetermined range (less than 10 cm, for example) was detected. This provided synchronized audio-visual alerts along with precise detection.
- ESP32-CAM: In this phase, the ESP32-CAM module was utilized primarily for object detection and recognition. The programmed with a separate sketch and integrated with external Python-based processing.

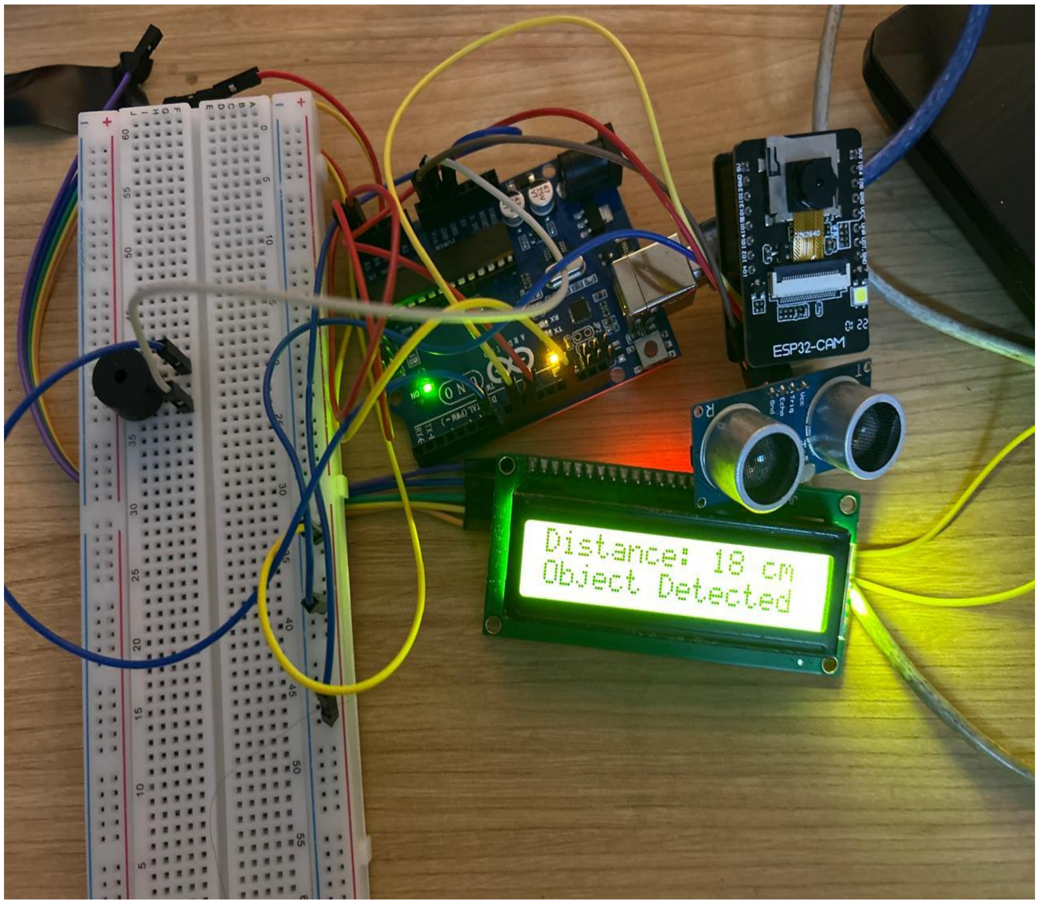


Figure 14: Hardware Assembly

Overall, the ultrasonic sensor, LCD, and buzzer were interfaced directly with the Arduino Uno, while the ESP32-CAM operated on an independent power circuit. This configuration ensured stable voltage supply, minimized power interference between modules, and enabled seamless coordination among sensing, alerting, and video monitoring components.

Part 4: Results and Findings

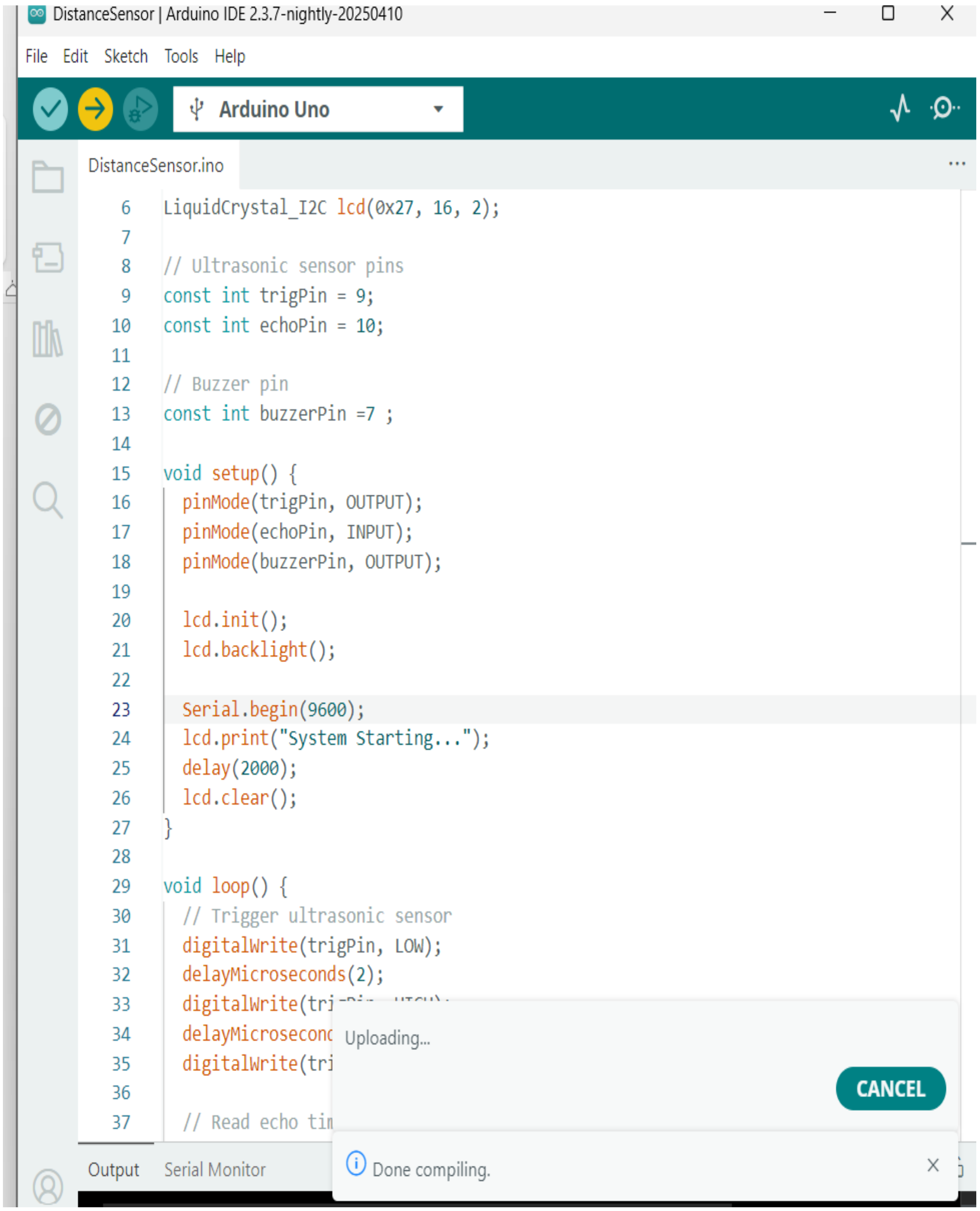
4.1 Testing

To ensure the system was functioning accurately and reliably, we carried out several tests. This included checking each component on its own and as part of the full setup to confirm everything worked together as expected.

Test 1: Code Execution and Upload

Test No	1
Objective	To show the execution of the code.
Action	Code was written, verified, and uploaded onto the Arduino using the Arduino IDE application.
Expected Result	The code would compile and upload successfully to the boards without any errors.
Actual Result	The code was successfully compiled and uploaded to both the ESP32-CAM and Arduino Uno, with no errors during the process, confirming proper setup and readiness for execution.

Table 2: Test1 (Code Execution)



```
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Ultrasonic sensor pins
const int trigPin = 9;
const int echoPin = 10;

// Buzzer pin
const int buzzerPin =7 ;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);

  lcd.init();
  lcd.backlight();

  Serial.begin(9600);
  lcd.print("System Starting...");
  delay(2000);
  lcd.clear();
}

void loop() {
  // Trigger ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(tri
  delayMicrosecond
  digitalWrite(tri

  // Read echo tin
```

Figure 15: Test1 (Code)

Test 2: Object Detection and LCD Display

Test No	2
Objective	To verify that the ultrasonic sensor detects objects accurately and displays the distance on the LCD.
Action	The ultrasonic sensor was connected to the Arduino, and the code was uploaded. Objects at various distances were placed to test the sensor, and the LCD was monitored for accurate distance readings.
Expected Result	The LCD should display the correct distance to the detected object.
Actual Result	The LCD displayed accurate distance values, and the readings changed accordingly as the object was moved closer or farther from the sensor.

Table 3: Test 3 (Object Detection -LCD Display)

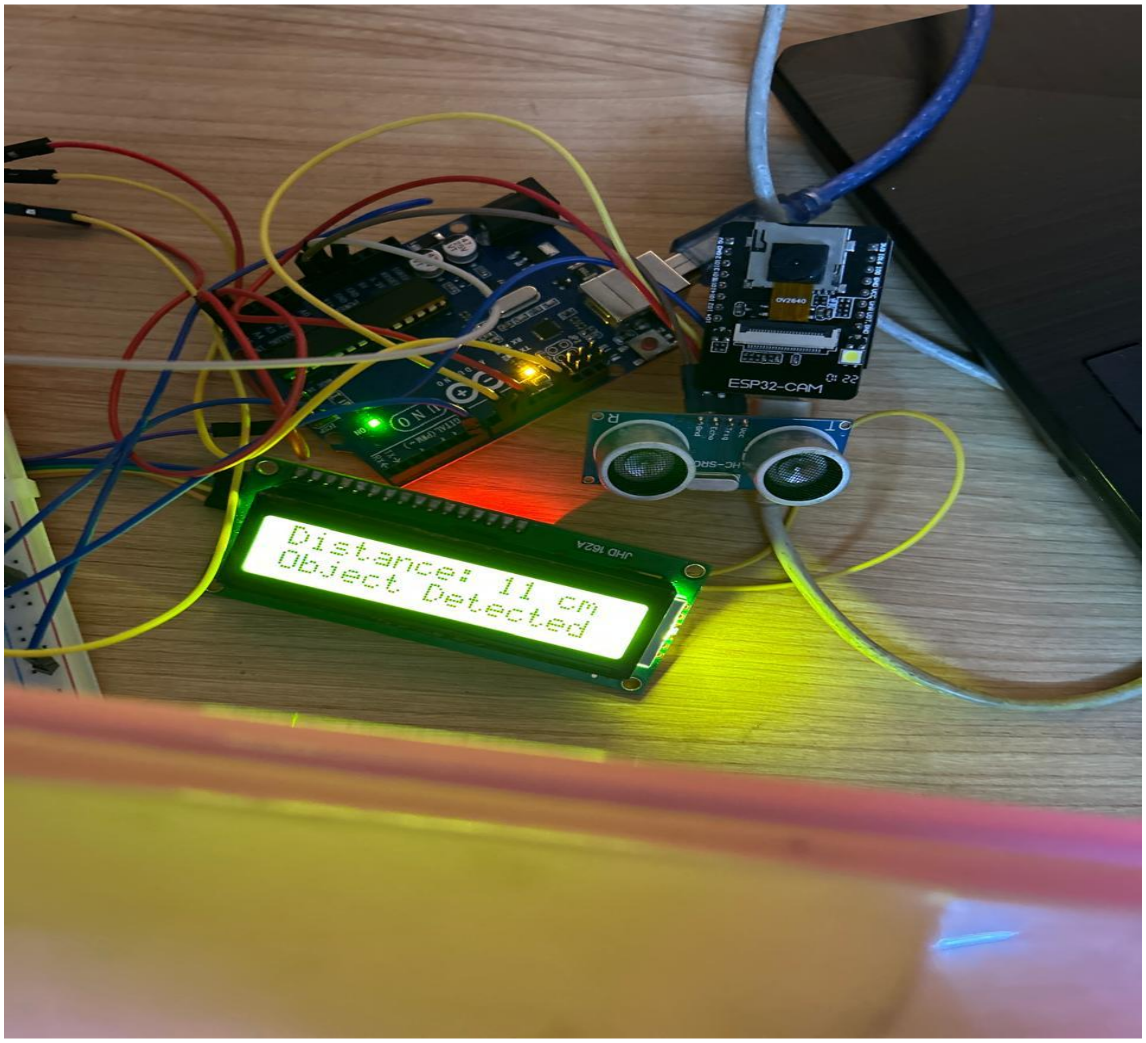


Figure 16: Distance measure (LCD display)

Test 3: Buzzer Alert

Test No	3
Objective	To verify the buzzer activates when an object is detected within the threshold distance.
Action	The threshold distance was set to 10 cm. The system was tested by placing objects within and beyond this range.
Expected Result	The buzzer should activate when an object is within 10 cm and remain silent when the object is beyond this range.
Actual Result	The buzzer successfully activated when an object was detected within the threshold distance and remained silent when no object was within range.

Table 4: Test 3 (Buzzer Alert)

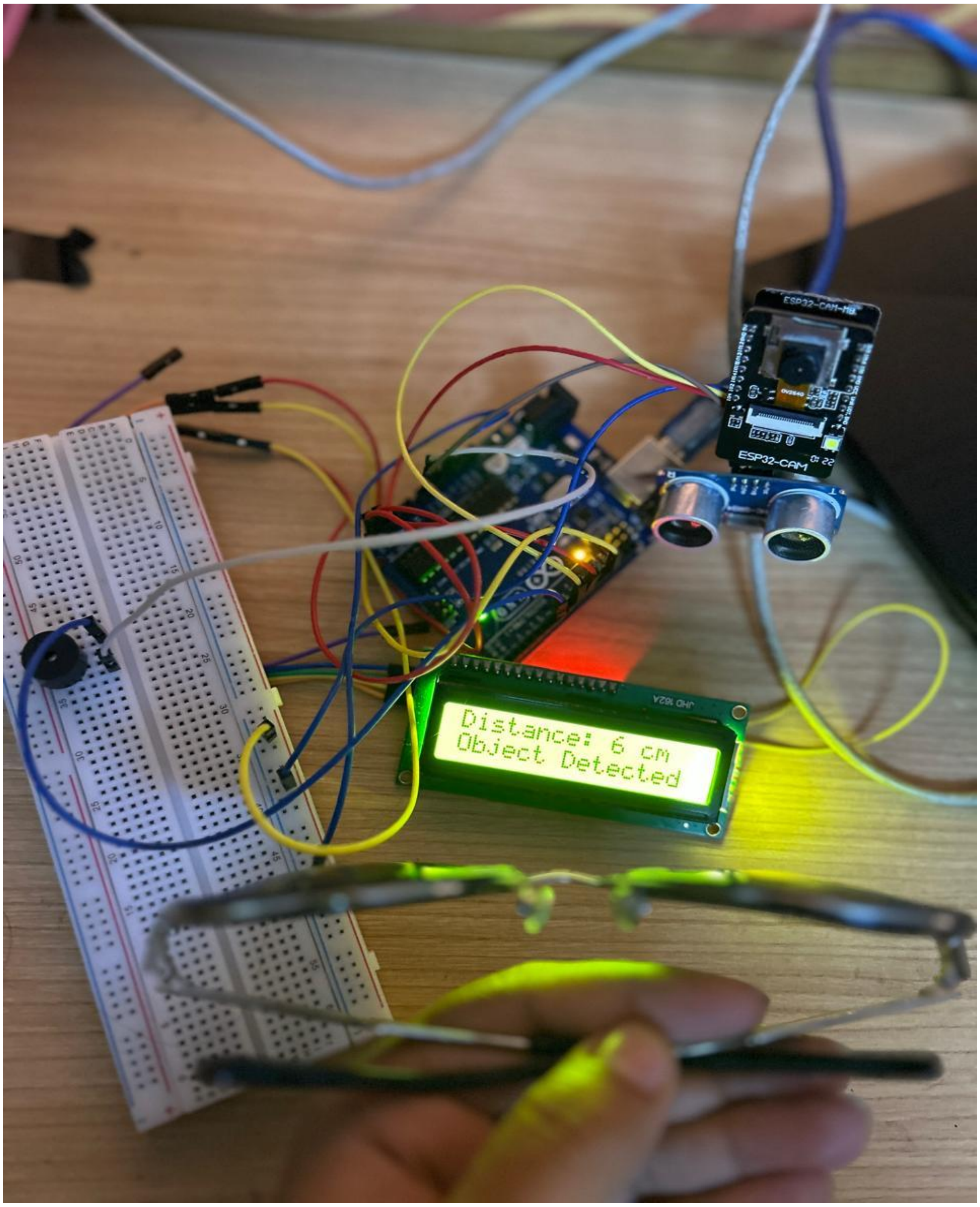


Figure 17: Buzzer Alert

Test 4: ESP32-CAM Object Recognition

Test No	4
Objective	To verify that the ESP32-CAM successfully detects objects and processes them using the object detection algorithm.
Action	The ESP32-CAM module was connected and programmed with the object detection code. The camera was aimed at various objects, and the system was tested for its ability to recognize and process them.
Expected Result	The ESP32-CAM should detect objects accurately, and the processed results should be displayed or logged correctly.
Actual Result	The ESP32-CAM successfully detected and processed objects, displaying the expected results without errors.

Table 5: Test 4 (Esp32-cam -Object Recognition)

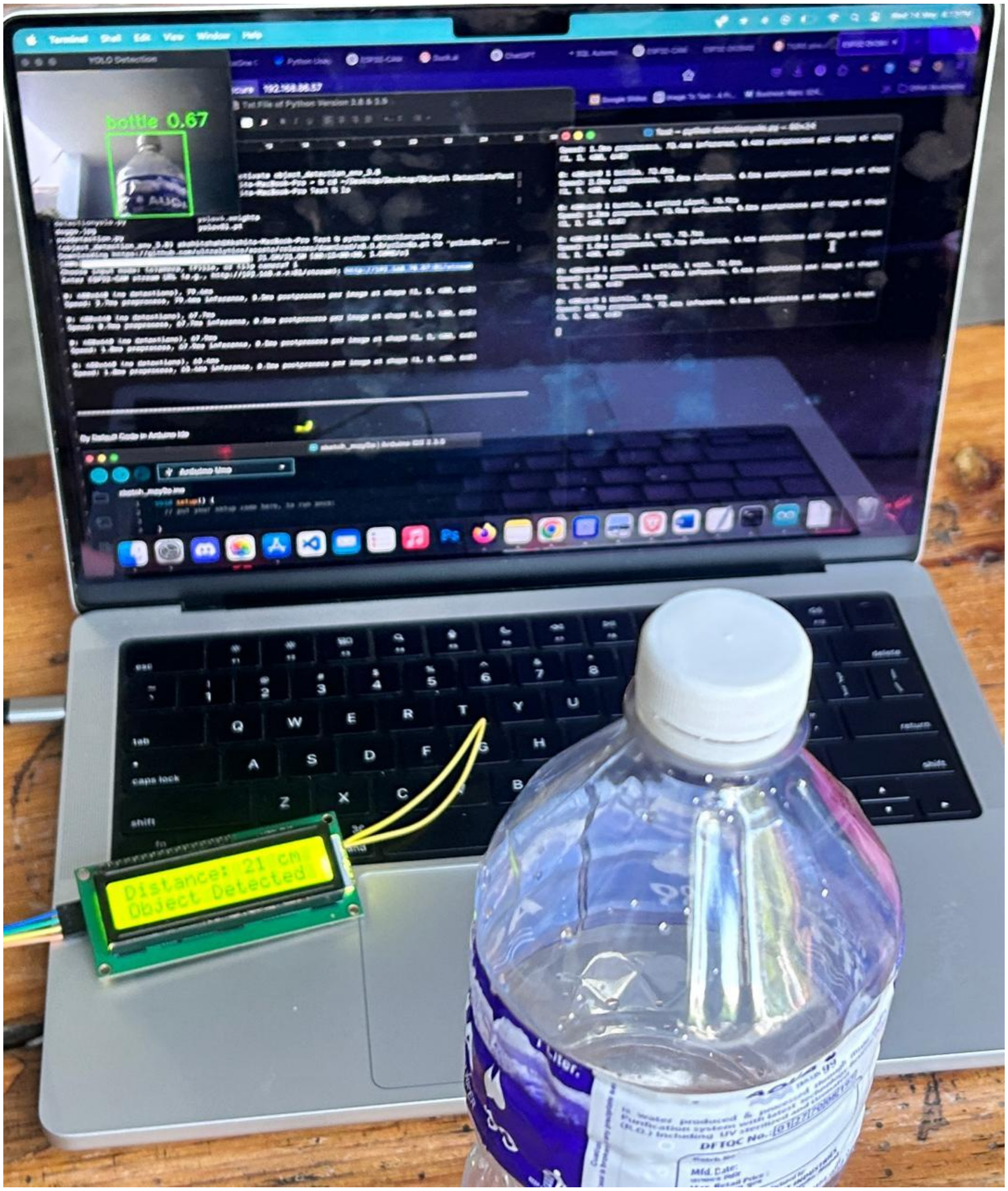
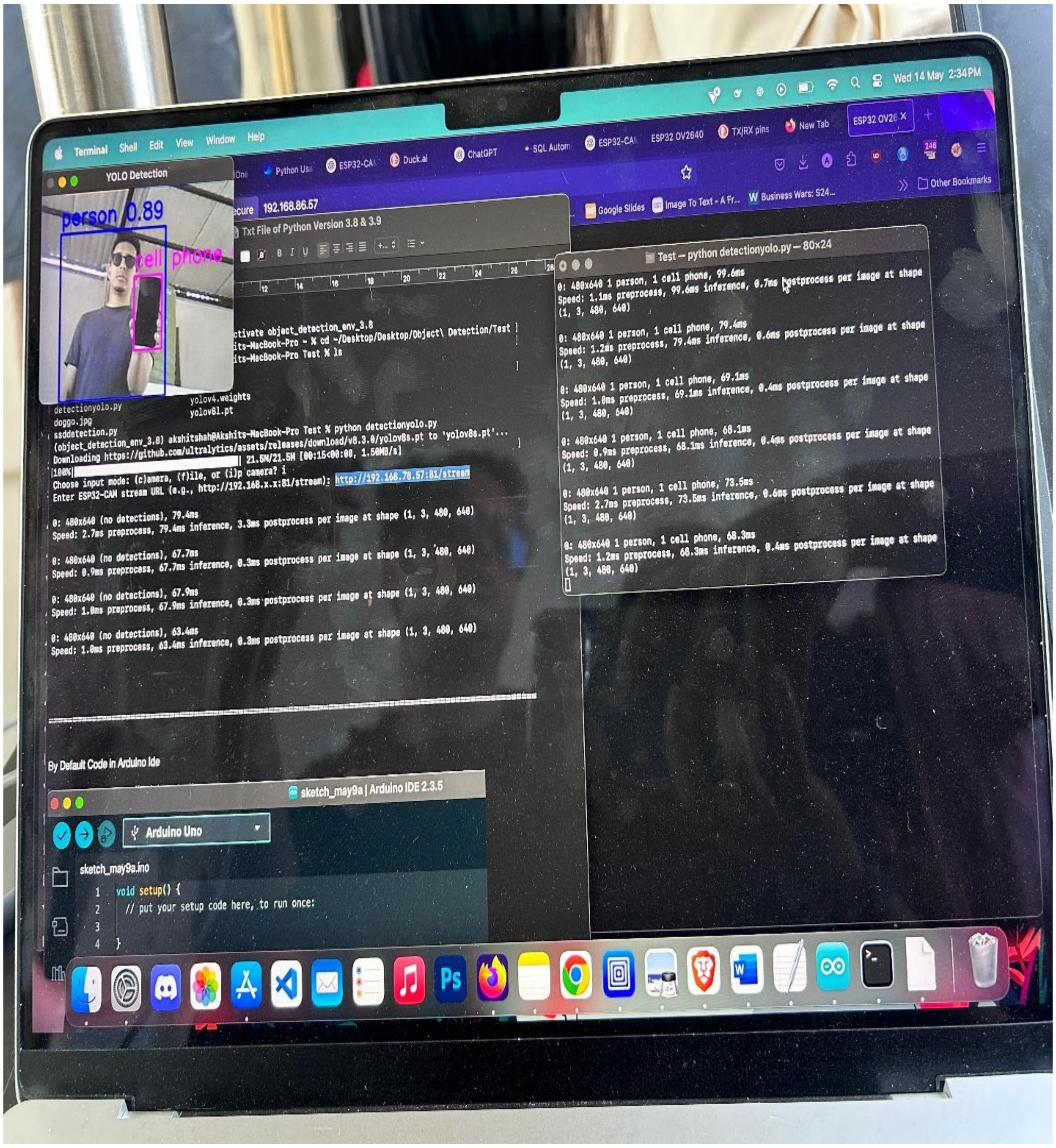


Figure 18: Bottle Detected

Test 5: LCD display of Multiple Detected Object at the same time

Test No	5
Objective	To verify that the ESP32-CAM can detect multiple objects simultaneously and correctly display their distances on the LCD.
Action	The ESP32-CAM was programmed to detect multiple objects in the camera's field of view. The ultrasonic sensor was used to measure the distance for each object. Multiple objects were placed at different distances, and the distance values were displayed on the LCD.
Expected Result	The ESP32-CAM should accurately detect multiple objects and display all their respective distances correctly on the LCD without confusion.
Actual Result	The system successfully detected multiple objects but displayed the distance of only one object, firstly detected by ultrasonic, on the LCD due to sensor overlapping or processing limitations.

Table 6: Test 5 (Failed Test)


YOLO Detection
person 0.89
cell phone
Txt File of Python Version 3.8 & 3.9
By Default Code in Arduino Ide
sketch_may9a | Arduino IDE 2.3.5
Arduino Uno
sketch_may9a.ino

```
ctivate object_detection_env_3.8
its-MacBook-Pro ~ % cd ~/Desktop/Desktop/Object\ Detection/Test
its-MacBook-Pro Test % ls
detectionyolo.py                yolov4.weights
doggo.jpg                       yolov8l.pt
ssddetection.py
(object_detection_env_3.8) akshitshah@Akshits-MacBook-Pro Test % python detectionyolo.py
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8s.pt to 'yolov8s.pt'...
100%|████████████████████| 21.5M/21.5M [00:15<00:00, 1.50MB/s]
Choose input mode: (c)amera, (f)ile, or (i)p camera? i
Enter ESP32-CAM stream URL (e.g., http://192.168.x.x:81/stream): http://192.168.78.57:81/stream

0: 480x640 (no detections), 79.4ms
Speed: 2.7ms preprocess, 79.4ms inference, 3.3ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 67.7ms
Speed: 0.9ms preprocess, 67.7ms inference, 0.3ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 67.9ms
Speed: 1.0ms preprocess, 67.9ms inference, 0.3ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 63.4ms
Speed: 1.0ms preprocess, 63.4ms inference, 0.3ms postprocess per image at shape (1, 3, 480, 640)
```

```
Test — python detectionyolo.py — 80x24
0: 480x640 1 person, 1 cell phone, 99.6ms
Speed: 1.1ms preprocess, 99.6ms inference, 0.7ms postprocess per image at shape
(1, 3, 480, 640)

0: 480x640 1 person, 1 cell phone, 79.4ms
Speed: 1.2ms preprocess, 79.4ms inference, 0.6ms postprocess per image at shape
(1, 3, 480, 640)

0: 480x640 1 person, 1 cell phone, 69.1ms
Speed: 1.0ms preprocess, 69.1ms inference, 0.4ms postprocess per image at shape
(1, 3, 480, 640)

0: 480x640 1 person, 1 cell phone, 68.1ms
Speed: 0.9ms preprocess, 68.1ms inference, 0.4ms postprocess per image at shape
(1, 3, 480, 640)

0: 480x640 1 person, 1 cell phone, 73.5ms
Speed: 2.7ms preprocess, 73.5ms inference, 0.6ms postprocess per image at shape
(1, 3, 480, 640)

0: 480x640 1 person, 1 cell phone, 68.3ms
Speed: 1.2ms preprocess, 68.3ms inference, 0.4ms postprocess per image at shape
(1, 3, 480, 640)
```

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

}
```

Figure 19: Multiple Object Detected

This experiment showed that the HC-SR04 sensor can only measure the distance to the nearest object, however the ESP32-CAM can visually recognize many things. As a result, the LCD displayed just one distance. Additional sensors or sophisticated techniques must be use to accurately display more than one distance.

4.2 Results

After completing the hardware and software integration of the Smart IoT Object Detection system, comprehensive testing was conducted to validate the performance and functionality of all components. The following key features were evaluated:

- Object Detection: The HC-SR04 ultrasonic sensor accurately detected objects within the predefined range. The measured distance was displayed in real time on the 16x2 LCD display.
- Alert Mechanism: When an object was detected within a critical distance threshold (e.g., 10 cm), the buzzer was successfully activated, providing immediate auditory feedback.
- ESP32-CAM Streaming: The ESP32-CAM module streamed real-time video over the local Wi-Fi network without issues, allowing remote visual monitoring and object recognition.
- Serial Monitor Output: The Arduino IDE's serial monitors reliably displayed real-time distance values, confirming accurate sensor readings and system logic.

Overall, all integrated components including the ESP32-CAM, Arduino Uno, ultrasonic sensor (HC-SR04), 16x2 LCD, and buzzer performed as intended. The system demonstrated reliable object detection, accurate feedback, and effective communication between hardware and software, validating the success of the implementation.

Part 5: Future Enhancements and Opportunities

While the current system effectively demonstrates the integration of object detection, real-time monitoring, and alert mechanisms, there are several promising directions for future development to enhance its performance, scalability, and usability.

- Cloud Integration: Incorporating platforms like ThingSpeak, Blynk, or Firebase would enable remote monitoring, real-time data visualization, and cloud-based logging. This would allow users to access system status and historical data from anywhere (G, 2024).
- Machine Learning and AI: Integrating lightweight AI models can enable advanced features such as object classification (distinguishing between humans, pets, or objects) and motion prediction. This would significantly increase the intelligence and automation of the system (Yutong Liu, 2021).
- Power Efficiency: Optimizing the system with low-power microcontrollers or implementing sleep modes during idle periods would make the solution more suitable for battery-powered and off-grid applications (Ivey, 2011).
- Mobile App Integration: Developing a dedicated Android or iOS application could provide users with instant alerts, live camera feeds from the ESP32-CAM, and distance tracking in a user-friendly interface (Abdelsalam A. Helal, 2012).

These enhancements would help transform the current prototype into a more intelligent, scalable, and reliable IoT solution ready for deployment in diverse smart environments.

Part 6: Conclusion

The Smart IoT Object Detection System shows an effective integration of different electronic components and microcontrollers to apply an effective, real-time obstacle detection and alerting system. With the utilization of the Arduino Uno for fundamental control, ultrasonic sensor for precise distance measurements, LCD display for real-time feedback, and buzzer for sound signals, the system can detect the closeness of objects in its vicinity and alert users accordingly. In addition, the ESP32-CAM offers object identification and video streaming on a local network, offering visual monitoring to the system without additional and expensive hardware.

This project has shown how low-cost components can be successfully combined to create an intelligent, functional IoT solution compatible with applications in home automation, robotics, assistive technology, and smart vehicles. It highlights the way real-time communication among hardware and software elements can enhance safety, awareness and responsiveness in automated systems.

The test and development cycle validated the effectiveness and stability of the system, ensuring its capacity to identify objects correctly and deliver quick visual and audio feed. The design of the system is modularity and simplicity oriented, making it applicable to several practical uses and easy to upgrade or reconfigure.

Future enhancements like premium features like integration with the cloud and intelligent processing can be done. Overall, the project offers a good foundation for developing more advanced smart systems that combine sensing, automation, and IoT connectivity in beneficial combinations.

References

Abdelsalam A. Helal, S. H. R. B. W. L., 2012. *Mobile Platforms and Development Environments.* s.l.:Morgan & Claypool Publishers.

Aniobi, A., 2024. *Sensor Fusion for Real-Time Object Detection and Spatial Positioning in Unmanned Vehicles Using YOLOv8 and ESP32-Cam.* [Online]
Available at:
https://www.researchgate.net/publication/385703919_Sensor_Fusion_for_Real-Time_Object_Detection_and_Spatial_Positioning_in_Unmanned_Vehicles_Using_YOLOv8_and_ESP32-Cam
[Accessed May 2025].

G, R., 2024. *Advanced Applcatons in Osmotic Computing.* s.l.:IGI Global Scientific Publishing.

GrandViewResearch, 2025. *Image Recognition Market Size & Trends.* [Online]
Available at: https://www.grandviewresearch.com/industry-analysis/image-recognition-market#:~:text=The%20global%20image%20recognition%20market,solutions%20further%20supports%20market%20expansion
[Accessed May 2025].

IBM, 2023. *What is the Internet of Things (IoT)?.* [Online]
Available at: https://www.ibm.com/think/topics/internet-of-things
[Accessed May 2025].

Ivey, B., 2011. *Low-Power Design Guide (AN1416),* s.l.: Microchip Technology Inc..

Li Yangyang, Q. H. X. P. L. J., 2020. *RADet: Refine Feature Pyramid Network and Multi-Layer Attention Network for Arbitrary-Oriented Object Detection of Remote Sensing Images.* [Online]
Available at:
https://www.researchgate.net/publication/338847053_RADet_Refine_Feature_Pyramid_Network_and_Multi-Layer_Attention_Network_for_Arbitrary-Oriented_Object_Detection_of_Remote_Sensing_Images
[Accessed May 2025].

Mohamed Fezari, A. A. D., 2018. *Integrated Development Environment (IDE) For Arduino,* s.l.: ResearchGate.

Nur Adila Husin, M. R. D. M. A., 2022. Journal of Counseling and Educational Technology. *Tinkercad simulation software to optimize online teaching and learning in embedded internet of things.*

Saiwa, 2025. *Exploring The Real-World Applications of Object Detection.* [Online] Available at: https://saiwa.ai/blog/real-world-applications-of-object-detection/ [Accessed May 2025].

SparkFun Electronics , 2025. *"Ultrasonic Distance Sensor - 5V (HC-SR04).* [Online] Available at: https://www.sparkfun.com/ultrasonic-distance-sensor-hc-sr04.html [Accessed 2025].

Yutong Liu, L. K. G. C. F. X. Z. W., 2021. Journal of Systems Architecture. *Light-weight AI and IoT collaboration for surveillance video pre-processing,* Volume 114.

Appendix

Arduino Code:

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>



// LCD setup (I2C address 0x27, 16 columns, 2 rows)

LiquidCrystal_I2C lcd(0x27, 16, 2);



// Ultrasonic sensor pins

const int trigPin = 9;

const int echoPin = 10;



// Buzzer pin

const int buzzerPin =7 ;



void setup() {

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  pinMode(buzzerPin, OUTPUT);



  lcd.init();

  lcd.backlight();
  Serial.begin(9600);
  lcd.print("System Starting...");
  delay(2000);
  lcd.clear();
}

void loop() {
  // Trigger ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read echo time
  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2;

  // Display on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" cm");


  if (distance < 10 && distance > 0) {

    digitalWrite(buzzerPin, HIGH);

    lcd.setCursor(0, 1);

    lcd.print("Object Detected");

  } else {

    digitalWrite(buzzerPin, LOW);

    lcd.setCursor(0, 1);

    lcd.print("No Object       ");

  }


delay(500);

}
```

Esp32-Cam Code:

```
#include "esp_camera.h"

#include <WiFi.h>

#include "camera_pins.h"


// Enter your WiFi credentials

const char *ssid = "Test";
const char *password = "12345678";


void startCameraServer();

void setupLedFlash(int pin);


void setup() {

  Serial.begin(115200);

  Serial.setDebugOutput(true);

  Serial.println();


  camera_config_t config;

  config.ledc_channel = LEDC_CHANNEL_0;

  config.ledc_timer = LEDC_TIMER_0;

  config.pin_d0 = Y2_GPIO_NUM;

  config.pin_d1 = Y3_GPIO_NUM;

  config.pin_d2 = Y4_GPIO_NUM;

  config.pin_d3 = Y5_GPIO_NUM;

  config.pin_d4 = Y6_GPIO_NUM;

  config.pin_d5 = Y7_GPIO_NUM;

  config.pin_d6 = Y8_GPIO_NUM;

  config.pin_d7 = Y9_GPIO_NUM;

  config.pin_xclk = XCLK_GPIO_NUM;

  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG;  // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;


  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                      for larger pre-allocated frame buffer.
  if (config.pixel_format == PIXFORMAT_JPEG) {
    if (psramFound()) {
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  sensor_t *s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);        // flip it back
    s->set_brightness(s, 1);   // up the brightness just a bit
    s->set_saturation(s, -2);  // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif


#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif
```

```
// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  Serial.print("WiFi connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}
```

```
void loop() {

  // Do nothing. Everything is done in another task by the web server

  delay(10000);

}
```

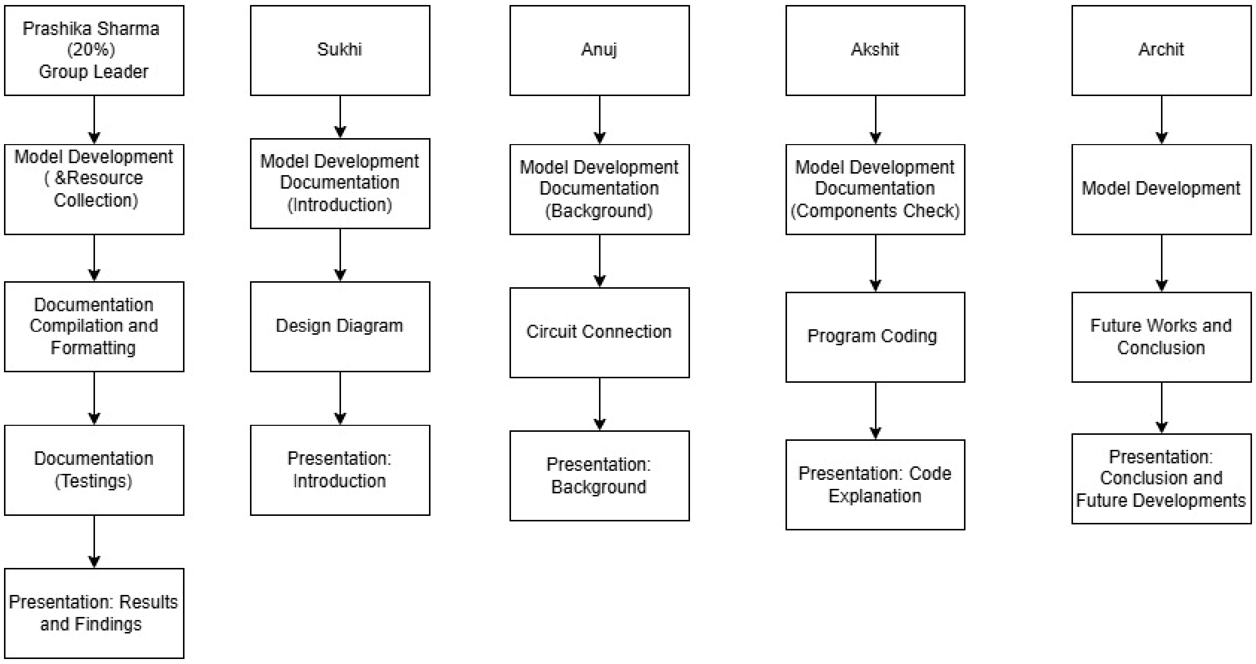
Python Code

```
import cv2

from ultralytics import YOLO

import os



# Load YOLOv8 model

yolo = YOLO('yolov8s.pt')



# Function to generate distinct colors for classes

def getColours(cls_num):

    base_colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255)]

    color_index = cls_num % len(base_colors)

    increments = [(1, -2, 1), (-2, 1, -1), (1, -1, 2)]

    color = [

        (base_colors[color_index][i] + increments[color_index][i] * (cls_num // len(base_colors)))
% 256

        for i in range(3)

    ]

    return tuple(color)
```

```
# Ask user to choose input mode
mode = input("Choose input mode: (c)amera, (f)ile, or (i)p camera? ").strip().lower()


if mode == 'c':
    cap = cv2.VideoCapture(0)


elif mode == 'i':
    ip_url = input("Enter ESP32-CAM stream URL (e.g., http://192.168.x.x:81/stream): ").strip()
    cap = cv2.VideoCapture(ip_url)


elif mode == 'f':
    file_path = input("Enter image or video file path: ").strip()
    if not os.path.isfile(file_path):
        print("Error: File not found.")
        exit()


    # If image
    if file_path.lower().endswith(('.png', '.jpg', '.jpeg')):
        frame = cv2.imread(file_path)
        results = yolo.track(frame, stream=True)


        for result in results:
            classes_names = result.names
            for box in result.boxes:
                if box.conf[0] > 0.4:
                    x1, y1, x2, y2 = map(int, box.xyxy[0])
                    cls = int(box.cls[0])
                    color = getColours(cls)
                    label = f'{classes_names[cls]} {box.conf[0]:.2f}'
                    cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
                   cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color,
2)

        cv2.imshow('YOLO Image Detection', frame)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()

    # If video
    elif file_path.lower().endswith(('.mp4', '.avi', '.mov', '.mkv')):
        cap = cv2.VideoCapture(file_path)

    else:
        print("Unsupported file format.")
        exit()
```

```
else:
    print("Invalid input. Choose 'c', 'f', or 'i'.")
    exit()


# Check if capture is open
if not cap.isOpened():
    print("Error: Could not open video source.")
    exit()


# Run detection loop
while True:
    ret, frame = cap.read()
    if not ret:
        break


    results = yolo.track(frame, stream=True)


    for result in results:
        classes_names = result.names
        for box in result.boxes:
            if box.conf[0] > 0.4:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                cls = int(box.cls[0])
            color = getColours(cls)

            label = f'{classes_names[cls]} {box.conf[0]:.2f}'

            cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)

             cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color,
2)


   cv2.imshow('YOLO Detection', frame)

   if cv2.waitKey(1) & 0xFF == ord('q'):

      break


cap.release()

cv2.destroyAllWindows()
```

Work Contribution Breakdown:


Prashika Sharma (20%) Group Leader
Model Development (&Resource Collection)
Documentation Compilation and Formatting
Documentation (Testings)
Presentation: Results and Findings
Sukhi
Model Development Documentation (Introduction)
Design Diagram
Presentation: Introduction
Anuj
Model Development Documentation (Background)
Circuit Connection
Presentation: Background
Akshit
Model Development Documentation (Components Check)
Program Coding
Presentation: Code Explanation
Archit
Model Development
Future Works and Conclusion
Presentation: Conclusion and Future Developments