**Rashtreeya Sikshana Samithi Trust**

**RV UNIVERSITY**

**School of Computer Science and Engineering**

**Bengaluru – 560059**



**SURVEILLANCE AND OBSTACLE DETECTION ROBOT USING ESP32 CAM**

Submitted in partial fulfilment of the requirements for the

**Interdisciplinary Project (CS3710)**

**By**

|  |  |
| --- | --- |
| **NAME: Sumedha Vijayaram Kumar** | **USN: 1RVU22BSC101** |
| **NAME: K Akanksha Raju** | **USN: 1RVU22BSC039** |
| **NAME: Priyanka CM** | **USN: 1RVU22BSC075** |
| **NAME: Nishita Bankapur** | **USN: 1RVU22BSC063** |
| **NAME: Janak R** | **USN: 1RVU22BSC037** |

**Under the guidance of**

**Dr. T. Hongray Assistant Professor, Dr. K Sailaja Kumar, Associate Professor and Prof. Harish KR, Assistant Professor**

**A logo for a university

Description automatically generated**

**CERTIFICATE**

*This is to certify that the Report entitled*

**SURVEILLANCE AND OBSTACLE DETECTION ROBOT USING ESP32-CAM**

*is a bonafide work carried out by*

|  |  |
| --- | --- |
| **NAME: Sumedha Vijayaram Kumar** | **USN: 1RVU22BSC101** |
| **NAME: K Akanksha Raju** | **USN: 1RVU22BSC039** |
| **NAME: Priyanka CM** | **USN: 1RVU22BSC075** |
| **NAME: Nishita Bankapur** | **USN: 1RVU22BSC063** |
| **NAME: Janak R** | **USN: 1RVU22BSC037** |

In partial fulfillment for the completion of 6th semester Interdisciplinary Project (CS3710) under rules and regulations of RV University, Bengaluru during the period Jan – May 2025. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in respect of Interdisciplinary Project work.

|  |  |  |
| --- | --- | --- |
| **Guide Name** | **Dr. Merin Thomas** | **Dr. G Shobha** |
| **Designation, SoCSE** | **Assistant Dean, SoCSE** | **Dean, SoCSE** |

|  |  |
| --- | --- |
| **Name of the Examiner(s)** | **Signature with Date** |

**DECLARATION**

We, **Sumedha Vijayaram Kumar, 1RVU22BSC101, K Akanksha Raju 1RVU22BSC039, Priyanka CM, 1RVU22BSC075,** **Nishita Bankapur, 1RVU22BSC063,** **Janak R, 1RVU22BSC037,** hereby declare that the report entitled, ***‘*SURVEILLANCE AND OBSTACLE DETECTION ROBOT USING ESP32-CAM*’,*** is an original work done by us under the guidance of **Dr. T. Hongray,Assistant Professor, Dr. K Sailaja Kumar, Associate Professor and Prof. Harish KR, Assistant Professor School of Computer Science and Engineering, RV University**,is being submitted in partial fulfillment of the requirements for completion of 6th semester Interdisciplinary Project.

Further, we declare that the content of the report has not been submitted previously by anybody, anywhere. We also declare that any Intellectual Property Rights generated out of this project will be the property of RV University, Bengaluru and we will be one of the authors of the same.

**PLACE: Bengaluru**

**DATE: 06/05/2025**

|  |  |
| --- | --- |
| **NAME: Sumedha Vijayaram Kumar** | **- USN: 1RVU22BSC101** |
| **NAME: K Akanksha Raju** | **- USN: 1RVU22BSC039** |
| **NAME: Priyanka CM** | **- USN: 1RVU22BSC075** |
| **NAME: Nishita Bankapur** | **- USN: 1RVU22BSC063** |
| **NAME: Janak R** | **- USN: 1RVU22BSC037** |

**ACKNOWLEDGEMENT**

It gives us immense pleasure to express our heartfelt gratitude to all those who contributed to the successful completion of our project.

First and foremost, we extend our sincere thanks to the School of Computer Science and Engineering, RV University for granting us the opportunity to undertake our 6th Semester Interdisciplinary Project. The support and encouragement provided by the institution were instrumental in shaping the direction and execution of our work.

We would like to thank **Dr. Shobha G, Dean SoCSE and Dr. Merin Thomas, Assistant Dean, SoCSE** for providing the necessary facilities with guidance to carry out the project work.

A special note of thanks goes to **Dr. K. Sailaja Kumar, Associate Professor, SoCSE, and Prof. Harish K. R, Assistant Professor, SoCSE**, for their constant encouragement, technical insights, and invaluable support throughout the project duration.

We would also like to express our deep appreciation to our project guide, **Dr. T. Hongray** **Assistant Professor**, **SoCSE**, or their dedicated guidance, constructive feedback, and consistent support at every stage of our project, which played a crucial role in its successful completion.

We are equally thankful to our families and friends for their encouragement, understanding, and unconditional support throughout the project.

Finally, we extend our gratitude to all those who directly or indirectly contributed to the successful execution of our project.

|  |  |  |
| --- | --- | --- |
| **NAME: Sumedha Vijayaram Kumar** | **USN:1RVU22BSC101** | **SIGN** |
| **NAME: K Akanksha Raju** | **USN: 1RVU22BSC039** | **SIGN** |
| **NAME: Priyanka CM** | **USN: 1RVU22BSC075** | **SIGN** |
| **NAME: Nishita Bankapur** | **USN: 1RVU22BSC063** | **SIGN** |
| **NAME: Janak R** | **USN: 1RVU22BSC037** | **SIGN** |

**ABSTRACT**

This project involves designing and developing a Surveillance and Obstacle Detection Robot using the ESP32-CAM module. The robot is remotely driven over Wi-Fi using a web-based interface, allowing users to drive it manually while having the ability to watch a live video feed from the onboard RGB camera in real-time. The system consists of a four-wheel drive platform based on DC motors using an L298N motor driver, which is driven by signals from the ESP32-CAM.

An ultrasonic sensor attached on a servo motor keeps scanning the environment from 15° to 165° all the time. If there is an obstacle detected within 20 cm distance, a buzzer is triggered automatically to give a warning signal to the user. Distance reading is also graphed using a SONAR graph for real-time monitoring. The hardware and software of the robot were created with Arduino IDE and Embedded C++, and testing verified successful motor control, live video streaming, obstacle detection, and real-time proximity alerting.

The system illustrates a low-cost, versatile alternative to fixed surveillance systems, providing mobility, real-time feedback, and improved obstacle awareness. While current implementation is based on manual remote control, modular architecture allows future upgrades to include autonomous movement, artificial intelligence-based object recognition, and Internet of Things cloud connectivity, which open it up to further potential for smarter, more automated security surveillance uses.

**Table of Contents**

[1. Introduction 8](#_Toc197433903)

[1.1 Overview 8](#_Toc197433904)

[1.2 Problem Definition 8](#_Toc197433905)

[2. Literature Survey 10](#_Toc197433906)

[3. Hardware and Software Requirements 13](#_Toc197433907)

[3.1 Hardware Requirements 13](#_Toc197433908)

[3.2 Software Requirements 14](#_Toc197433909)

[4. Software Requirements Specification 16](#_Toc197433910)

[4.1 System Features 16](#_Toc197433911)

[5. System Design Description (SDD) 18](#_Toc197433912)

[5.1 System Overview 18](#_Toc197433913)

[5.2 Database Design/Data Set Description 20](#_Toc197433914)

[5.3 Functional Design 20](#_Toc197433915)

[5.3.1 Describe the functionalities of the system 20](#_Toc197433916)

[6. Implementation 22](#_Toc197433917)

[7. Testing 35](#_Toc197433918)

[7.1 Description of Testing 35](#_Toc197433919)

[7.2 Test Cases 36](#_Toc197433920)

[8.Results and Discussion 38](#_Toc197433921)

[9. Conclusion 40](#_Toc197433922)

[10. Scope for Further Enhancement 41](#_Toc197433923)

[11. Bibliography 43](#_Toc197433924)

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Title** | **Page No.** |
| **1.** | **ESP32-CAM Pinout Diagram Showing GPIO Assignments** | **13** |
| **2.** | **Wiring Diagram of ESP32-CAM, L298N Motor Driver, and DC Motors** | **19** |
| **3.** | **Robot system components and their connections** | **20** |
| **4.** | **Robot operation sequence** | **21** |
| **5.** | **ESP32-CAM Programming Setup using USB-to-Serial (FTDI) Module** | **36** |

# Introduction

## Overview

In the changing environment of monitoring and surveillance, traditional methods like static CCTV installations and manual patrols have major drawbacks. While manned patrols are subject to fatigue, inconsistency, and restricted area coverage, fixed surveillance systems tend to have blind spots and limited fields of view.

Additionally, the cost of running security personnel adds to the inefficiency of traditional methods. The recent breakthroughs in wireless communication, embedded systems, and robotics open new possibilities to tackle these issues.

Remotely controlled surveillance robots are one new solution that combines live video transmission, user-guided navigation, and real-time obstacle sensing into one mobile package. Such robots enhance flexibility, reduce the cost of operations, and provide faster responsiveness over static systems.

The "Surveillance and Obstacle Detection Robot using ESP32-CAM" project involves building a low-cost, portable robot system that may be manually driven through Wi-Fi while giving instant video feedback as well as obstruction detection alerts.

The robot comprises an ESP32-CAM module for real-time video streaming, DC motors for mobility, ultrasonic sensor coupled with servo motor for environmental scanning, and buzzer for alert on proximity. This system proves the viability of using small, remotely operated robots to improve surveillance effectiveness in offices, residences, campuses, and factories. The project highlights major disciplines like embedded system programming, wireless networking, robotics integration, and real-time sensor-based alerting, providing a basis for future development into autonomous operation and AI-based object detection.

## Problem Definition

Old methods of surveillance like manual security patrols and fixed CCTV cameras have several operational issues that decrease the effectiveness of security systems overall:

* **Limited Field of Vision:** Fixed CCTV cameras are static and can cover only a narrow area, meaning there are several blind spots where intrusions are not detected.
* **No Dynamic Tracking:** Cameras are not able to track moving objects or threats. The moment an object moves outside the line of sight, it is no longer tracked.
* **Human Error and Fatigue:** Human security officers can get exhausted or distracted over time, which results in missed detections and slow responses to incidents.
* **Delayed Incident Response:** Conventional systems tend to detect and respond to threats only after an incident has already taken place, extended response time and diminishing effectiveness.
* **High Operational Costs:** Recruitment and upkeep of trained human security officers for 24/7 surveillance and patrolling operations is costly and logistically challenging.
* **Lack of Real-Time Proximity Alerts:** Static surveillance does not have provisions to automatically alert operators of close proximity obstacles or intruders unless manually checked.
* **Limited Flexibility:** Conventional configurations are inflexible and cannot be easily modified to meet dynamic surveillance needs in evolving surroundings.

**Objective of this Project:**

The "Surveillance and Obstacle Detection Robot using ESP32-CAM" overcomes these limitations by:

* Enabling manual remote operation of a mobile robot via a Wi-Fi web interface,
* Streaming live video from an onboard RGB camera to smartphones and PCs,
* Sensing obstacles close to the robot with an ultrasonic sensor placed on a servo motor scanning the vicinity,
* Triggering an immediate buzzer alarm when an obstacle is sensed within a specified critical range (20 cm),
* Displaying sensor readings in a SONAR-like graph for real-time environmental comprehension.

Although the present implementation is user-controlled manually, the modular nature of the system allows future upgrades to autonomous navigation, AI-powered object recognition, and cloud-enabled surveillance systems, which will take it closer to becoming a completely intelligent, automated security solution.

# Literature Survey

The goal of this work was to design an efficient autonomous patrolling robot with real time object detection in a library survey. The fieldwork focused on existing mobile surveillance systems, approaches to obstacle detection, and advances in object detection using machine learning models such as YOLO, Faster R-CNN and other related approaches. The highlights of these works are presented in Table below.

**2.1 Review of Related Work**

**1.** **RGB camera based Fallen Person Detection System on a mobile platform (Sergio Lafuente-Arroyo et al., 2022)**

The research combines a RGB camera with YOLO based models for falling individuals. Recall rate of 98. 97% confirms the reliability of the system, and also highlights the contribution of single stage object detection to real-time applications and calls for future improvements involving multi-modal sensing. [1]

**2.** **On-line vision-based obstacle detection in a maritime environment (Duarte Nunes et al., 2022**

An embedded Raspberry Pi system with a Pi camera configuration was used for the real time obstacle detection optimized by OpenCV and TensorRT. Real time performance reached around 15–30 FPS which is acceptable for realtime operation, however in poor lighting performance tends to be reduced. [2]

**3.** **Mobile Robot for Security Applications in Remotely Operated Advanced Reactors (Ujjwal Sharma et al., 2024)**

The study used LiDAR, GPS, Ultrasonic sensor, real time wireless communication, and achieved efficient autonomous navigation and obstacle avoidance under complex environments suggesting modular sensor integration provides better patrolling performance. [3]

**4.** **Autonomous Surveillance for Indoor Security Robot (Min-Fan Ricky Lee et al., 2022)**

The indoor patrolling was carried out using a combination of computer vision approaches for intrusion detection, object recognition and localization, the results of which show that the indoor surveillance method achieves both real-time incident detection and efficient indoor surveillance. [4]

**5.** **Automatic Outdoor Patrol Robot with Sensor Fusion and Face Recognition (Wu-Chiang Chang et al., 2021)**

The paper presents an outdoor patrolling robot that is equipped with sensor fusion (LiDAR, GPS, IMU) for detection and face recognition. The results show that multimodal sensory integration increases the detection reliability dramatically. This work is promising for outdoor autonomous security applications. [5]

**6.** **Manipulator-based Autonomous Inspections at Road Checkpoints (Qing-yin Shi et al., 2022)**

The authors applied LF-YOLO for efficient object detection at checkpoints and found an impressive speedup and accuracy improvement compared to YOLOv3[7]. It also underscored the importance of lightweight models for real-time surveillance in mobile robotics. [6]

**7.** **Threat Detection and Patrolling Support Using Neural Networks and ROS (S.S. Maram et al., 2019)**

Kinect RGB-D sensors with YOLO-based object detection built into a ROS framework were utilized in this system. Despite being effective at identifying threats, Kinect-based sensing's drawbacks in outdoor settings were brought to light, which is why RGB-only solutions are better suited for basic security robots. [7]

**8.** **An inexpensive environment-based interactive patrol inspection system (Z. Zhang and L. Chen, 2025)**

This system used ROS-based navigation and obstacle avoidance algorithms to improve positional accuracy by 24.35% by combining LiDAR and RGB-D sensors. It proved that robot stability can be greatly increased with inexpensive sensor fusion. [8]

**9.** **WADOR: A Mobile Autonomous Robot for Monitoring (S. Mittal and M.K. Rai, 2016)**

GSM modules and PIR motion sensors were used by this robot to send out alerts. It was found to be appropriate for low-cost applications where simple motion detection is adequate, despite being less accurate than computer-vision-based techniques. [9]

**10.** **Surveillance Using Three-Dimensional Outdoor Object Detection (Zihan Tian et al., 2024)**

In order to detect outdoor objects with depth information, this work created a YOLO3D model that achieves high precision in challenging outdoor scenes. It underlined that three-dimensional detection models perform noticeably better than conventional 2D models in environments that demand high depth accuracy. [10]

**11.** **Creation of an Affordable and Completely Autonomous Patrol Robot (Yi-Tse Lin et al., 2023)**

Using RGB-D sensing and ultrasonic sensors, this system demonstrated a low-cost patrol robot. It demonstrated how crucial it is to strike a balance between cost effectiveness and adequate sensing dependability for patrol applications. [11]

**12.** **Research Projects to Assist in the Safety of Autonomous Mobile Robots (Hassan Almukhtar et al., 2024)**

The technological frameworks for mobile robots' safe autonomous navigation were the main focus of the review. It covered how important obstacle avoidance and object detection are to maintaining safety, particularly in dynamic or complex settings. [12]

**2.2 Main Conclusions and Relevance to Our Project**

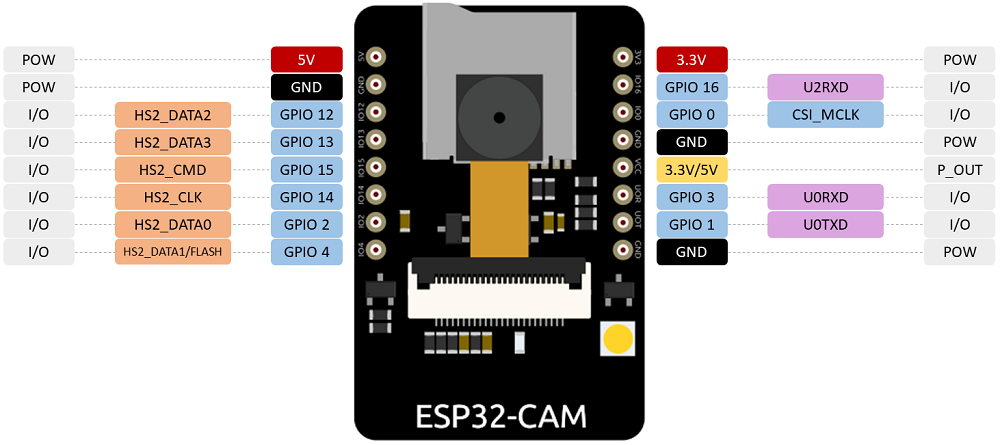
* YOLO and light versions (e.g., YOLO-tiny, LF-YOLO) provide the best balance between detection speed and accuracy and are therefore well-suited for real-time patrol robots.
* ESP32-CAM and Raspberry Pi based implementations have been shown to be economical platforms for constructing autonomous systems with vision capabilities.
* Sensor fusion methods (LiDAR + RGB-D + GPS) are improving navigation but can be overkill for a low-cost security robot designed for small-to-medium sized areas.
* Hardware reliability and thermal issues are common problems with running light embedded systems such as ESP32-CAM at high computational loads.
* Multi-robot coordination, integration with IoT for cloud monitoring, and using AI to classify threats (person, vehicle, unknown object detection) are future directions.

# Hardware and Software Requirements

## Hardware Requirements

**ESP32-CAM Controller:**

* Functions as the main processing unit of the robot.
* Controls movement, live video transmission, and remote control through a web interface.
* Built-in Wi-Fi module provides wireless communication capabilities.



**Fig 1. ESP32-CAM Pinout Diagram Showing GPIO Assignments**

**L298N Motor Driver:**

* Regulates speed and direction of the four DC motors.
* Receives command signals from the ESP32-CAM to power the motors appropriately.

**DC Motors and Wheels:**

* Four DC motors give mobility to the robot.
* Allow the robot to move forward, backward, left, and right according to user inputs.

**Chassis and Frame:**

* Acts as the physical frame to hold all electronic and mechanical parts.
* Provides stability, protection, and structured assembly.

**Power Supply:**

* Rechargeable battery pack with 7.4V output.
* Drives the ESP32-CAM module, L298N motor driver, DC motors, ultrasonic sensor, and buzzer.

**Connectors and Wires:**

* Utilized to form stable and secure electrical connections between all parts.

**Ultrasonic Sensor:**

* Utilized for object detection and measurement of distance.
* Provides obstacle information during the robot's patrol.

**Servo Motor:**

* Turns the ultrasonic sensor between 15° and 165° to sweep around.

**Buzzer:**

* Triggers when an obstacle is sensed in the 20 cm range.
* Gives an audio warning to signal close obstacles.

## Software Requirements

**Arduino IDE:**

* Utilized as the primary development environment.
* Is responsible for writing, compiling, and uploading code to the ESP32-CAM module.
* Supports ESP32 board setup and serial monitoring while testing.

**Embedded C++ Programming Language:**

* Utilized to develop:
* Four-motor movement control logic,
* Wi-Fi web server for live video streaming and command receiving,
* Ultrasonic distance measurement and servo motor scanning,
* Proximity detection buzzer activation.

.

# Software Requirements Specification

## System Features

The system features of the Surveillance and Obstacle Detection Robot are based on three main services: manual remote navigation, live video streaming, and real-time obstacle detection with alarm.

**4.1.1 System Feature: Manual Remote Navigation**

**Functional Requirements:**

**FR-1.1:** The robot must navigate on the basis of user instructions received through the Wi-Fi-based web interface.

**FR-1.2:** The ESP32-CAM must send movement control commands to the four DC motors through the L298N motor driver.

**FR-1.3:** The robot ought to enable fundamental motion operations — going forward, going backward, turning left, turning right, and stopping.

**FR-1.4:** The robot ought to react promptly and consistently to remote movement commands.

**FR-1.5:** The robot ought to be stable when moving over various surfaces.

**4.1.2 System Feature: Live Video Streaming**

**Functional Requirements:**

**FR-2.1:** The ESP32-CAM should take live frames continuously with a built-in RGB camera.

**FR-2.2:** The video captured by the camera should be streamed to a web interface which can be accessed using smartphone or PC.

**FR-2.3:** The video feed should update at a frequency adequate for surveillance and navigation tasks.

**FR-2.4:** The system should have an uninterrupted Wi-Fi connection so that the live video streaming does not get interrupted.

**4.1.3 System Feature: Real-Time Obstacle Detection and Alert**

**Functional Requirements:**

**FR-3.1:** The ultrasonic sensor on the servo motor needs to keep scanning the environment by rotating from 15° to 165°.

**FR-3.2:** The system needs to measure the distance of objects close to it.

**FR-3.3:** In case there is an obstacle detected within 20 cm, the buzzer automatically needs to be turned on to notify the user.

**FR-3.4:** The system needs to turn off the buzzer in case there is no obstacle close by.

**FR-3.5:** The measurements of distance must be processed for both left and right scanning movements.

# System Design Description (SDD)

## System Overview

The Surveillance and Obstacle Detection Robot system is comprised of hardware and software components that collaborate to facilitate live video streaming, wireless navigation, and real-time obstacle detection.

The ESP32-CAM module acts as the main controller, controlling robot movement, recording live video frames, and facilitating wireless communication with the user.

DC motors and L298N motor driver enable the robot to navigate in different directions according to user inputs.

The servo motor-mounted ultrasonic sensor can detect close-proximity obstacles, triggering a buzzer as appropriate to avoid crashes.

The power supply system ensures reliable operation of the ESP32-CAM module, the motor driver, the motors, and the obstacle detecting modules.

**System Architecture (Deployment Overview):**

**ESP32-CAM Module:**

* Drove the four DC motors with the L298N motor driver.
* Grabs real-time video frames using its built-in RGB camera.
* Has a Wi-Fi-based web server for remote control and video streaming.
* Handles wireless communication between the robot and the user's smartphone or PC.

**L298N Motor Driver:**

* Acts as the interface between the ESP32-CAM and DC motors.
* Accepts GPIO control signals to drive the motors and facilitate the mobility of the robot (forward, backward, left, right, and stop).

**DC Motors and Wheels:**

* Impart mobility for the robot according to the user commands received via the web interface. Move.
* Facilitate easy navigation and movement.

**Ultrasonic Sensor and Servo Motor:**

* Scan the environment continuously by rotating between 15° and 165°.
* Take measurements to nearby obstacles.
* Activate the buzzer alert when an obstacle is found within 20 cm.

**Buzzer:**

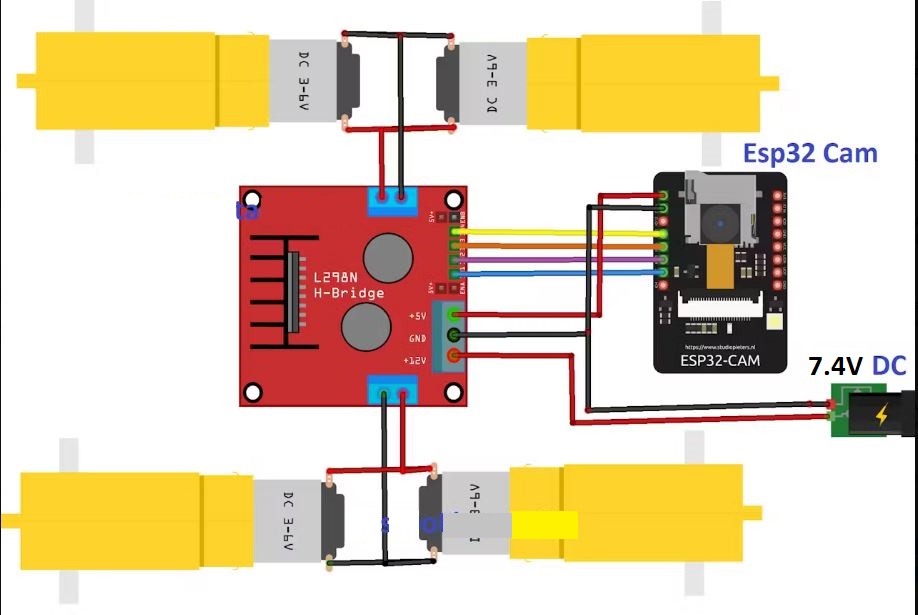
* Give an audible warning when an obstacle is found too close to the robot.
* Improve operating safety during robot movement.

**Battery Pack:**

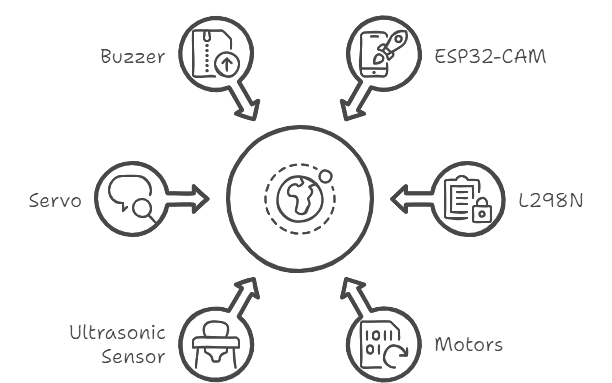
* Power the ESP32-CAM, motor driver, and DC motors with regulated power.
* Enables the robot to function continuously while patrolling and navigating.

**Chassis/Frame:**

* Offers mechanical support for all the mounted components.
* Guarantees stability, toughness, and protection during movement.



**Fig 2. Wiring Diagram of ESP32-CAM, L298N Motor Driver, and DC Motors**



**Fig 3. Robot system components and their connections**

## Database Design/Data Set Description

* No traditional database system (like MySQL or MongoDB) is utilized.
* No local image storage is used.
* Live video streaming is offered using the ESP32-CAM via Wi-Fi.
* No dataset is generated or stored during robot runtime.

## Functional Design

### Describe the functionalities of the system

**Navigation Module:**

* Accepts manual movement commands through the Wi-Fi web interface.
* Operates motors to move the robot forward, backward, left, right, and stop.
* Obstacle Detection Module:
* Uses an ultrasonic sensor supported by a servo motor to detect obstacles.
* Triggers a buzzer when an obstacle is found within a specified range (20 cm).

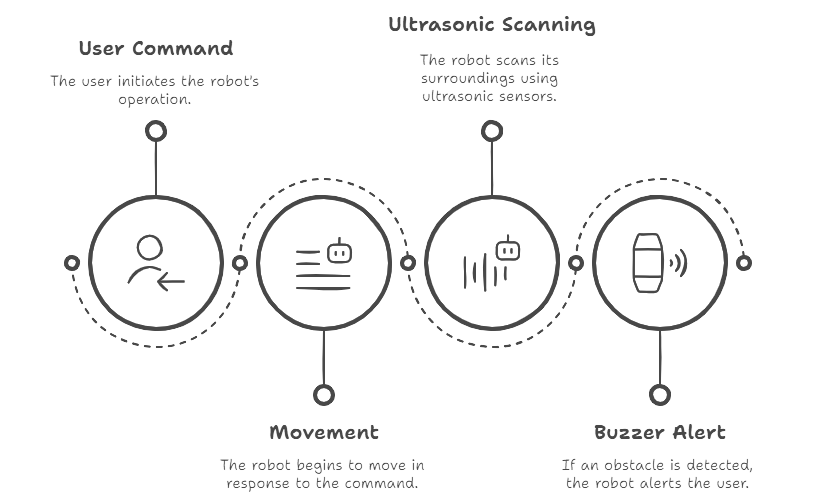
**Vision Module:**

* Streams and captures live video from the ESP32-CAM's RGB camera onto the user's device.
* Does no image processing or object detection whatsoever.

**Alert Module:**

* Triggers a buzzer to notify the user when there is an obstacle nearby.

**5.3.2 Behavioural Design**

****

**Fig 4. Robot operation sequence**

# Implementation

1. **Ultrasonic sensor Code for the Robot:**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Set the LCD address to 0x27 for 16 chars and 2 line display

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Includes the Servo library

#include <Servo.h>

// Defines Tirg and Echo pins of the Ultrasonic Sensor

const int trigPin = 2;

const int echoPin = 3;

const int Buzzer=13;

// Variables for the duration and the distance

long duration;

int distance;

Servo myServo; // Creates a servo object for controlling the servo motor

void setup()

{

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

pinMode(Buzzer,OUTPUT);

lcd.init(); /\* Initialize 16x2 LCD \*/

lcd.backlight(); // Turn on the blacklight and print a message

Serial.begin(9600);

myServo.attach(9); // Defines on which pin is the servo motor attached

}

void loop()

{

Display\_Reading();

// rotates the servo motor from 15 to 165 degrees

for(int i=15;i<=165;i++){

myServo.write(i);

delay(30);

distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

if(distance>0 && distance<20)

{digitalWrite(Buzzer,HIGH);}

else

{digitalWrite(Buzzer,LOW);}

Serial.print(i); // Sends the current degree into the Serial Port

Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

Serial.print(distance); // Sends the distance value into the Serial Port

Serial.print("."); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

}

//delay(500); //pause time

// Repeats the previous lines from 165 to 15 degrees

for(int i=165;i>15;i--)

{

myServo.write(i);

delay(30);

distance = calculateDistance();

if(distance>0 && distance<20)

{digitalWrite(Buzzer,HIGH);}

else

{digitalWrite(Buzzer,LOW);}

Serial.print(i);

Serial.print(",");

Serial.print(distance);

Serial.print(".");

}

//delay(500); //pause time

}

// Function for calculating the distance measured by the Ultrasonic sensor

int calculateDistance()

{

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); // Reads the echoPin,

distance= duration\*0.034/2;

return distance;

}

/////////////////////////

void Display\_Reading()

{

lcd.setCursor(0,0); /\* Set cursor to column 0 row 0 \*/

lcd.print("SONAR Project"); /\* Print data on display \*/

lcd.setCursor(0,1); /\* Set cursor to column 0 row 0 \*/

lcd.print("Scanning...");

}

**2. Code for the ESP32 CAM Module:**

/\*\*\*\*\*\*\*\*\*

Rui Santos

Complete instructions at https://RandomNerdTutorials.com/esp32-cam-projects-ebook/

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files.

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

\*\*\*\*\*\*\*\*\*/

#include "esp\_camera.h"

#include <WiFi.h>

#include "esp\_timer.h"

#include "img\_converters.h"

#include "Arduino.h"

#include "fb\_gfx.h"

#include "soc/soc.h" // disable brownout problems

#include "soc/rtc\_cntl\_reg.h" // disable brownout problems

#include "esp\_http\_server.h"

// Replace with your network credentials

const char\* ssid = "shreyastechno";

const char\* password = "praveen.15";

#define PART\_BOUNDARY "123456789000000000000987654321"

#define CAMERA\_MODEL\_AI\_THINKER

//#define CAMERA\_MODEL\_M5STACK\_PSRAM

//#define CAMERA\_MODEL\_M5STACK\_WITHOUT\_PSRAM

//#define CAMERA\_MODEL\_M5STACK\_PSRAM\_B

//#define CAMERA\_MODEL\_WROVER\_KIT

#if defined(CAMERA\_MODEL\_WROVER\_KIT)

#define PWDN\_GPIO\_NUM -1

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 21

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 19

#define Y4\_GPIO\_NUM 18

#define Y3\_GPIO\_NUM 5

#define Y2\_GPIO\_NUM 4

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

#elif defined(CAMERA\_MODEL\_M5STACK\_PSRAM)

#define PWDN\_GPIO\_NUM -1

#define RESET\_GPIO\_NUM 15

#define XCLK\_GPIO\_NUM 27

#define SIOD\_GPIO\_NUM 25

#define SIOC\_GPIO\_NUM 23

#define Y9\_GPIO\_NUM 19

#define Y8\_GPIO\_NUM 36

#define Y7\_GPIO\_NUM 18

#define Y6\_GPIO\_NUM 39

#define Y5\_GPIO\_NUM 5

#define Y4\_GPIO\_NUM 34

#define Y3\_GPIO\_NUM 35

#define Y2\_GPIO\_NUM 32

#define VSYNC\_GPIO\_NUM 22

#define HREF\_GPIO\_NUM 26

#define PCLK\_GPIO\_NUM 21

#elif defined(CAMERA\_MODEL\_M5STACK\_WITHOUT\_PSRAM)

#define PWDN\_GPIO\_NUM -1

#define RESET\_GPIO\_NUM 15

#define XCLK\_GPIO\_NUM 27

#define SIOD\_GPIO\_NUM 25

#define SIOC\_GPIO\_NUM 23

#define Y9\_GPIO\_NUM 19

#define Y8\_GPIO\_NUM 36

#define Y7\_GPIO\_NUM 18

#define Y6\_GPIO\_NUM 39

#define Y5\_GPIO\_NUM 5

#define Y4\_GPIO\_NUM 34

#define Y3\_GPIO\_NUM 35

#define Y2\_GPIO\_NUM 17

#define VSYNC\_GPIO\_NUM 22

#define HREF\_GPIO\_NUM 26

#define PCLK\_GPIO\_NUM 21

#elif defined(CAMERA\_MODEL\_AI\_THINKER)

#define PWDN\_GPIO\_NUM 32

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

#elif defined(CAMERA\_MODEL\_M5STACK\_PSRAM\_B)

#define PWDN\_GPIO\_NUM -1

#define RESET\_GPIO\_NUM 15

#define XCLK\_GPIO\_NUM 27

#define SIOD\_GPIO\_NUM 22

#define SIOC\_GPIO\_NUM 23

#define Y9\_GPIO\_NUM 19

#define Y8\_GPIO\_NUM 36

#define Y7\_GPIO\_NUM 18

#define Y6\_GPIO\_NUM 39

#define Y5\_GPIO\_NUM 5

#define Y4\_GPIO\_NUM 34

#define Y3\_GPIO\_NUM 35

#define Y2\_GPIO\_NUM 32

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 26

#define PCLK\_GPIO\_NUM 21

#else

#error "Camera model not selected"

#endif

#define MOTOR\_1\_PIN\_1 14

#define MOTOR\_1\_PIN\_2 15

#define MOTOR\_2\_PIN\_1 13

#define MOTOR\_2\_PIN\_2 12

static const char\* \_STREAM\_CONTENT\_TYPE = "multipart/x-mixed-replace;boundary=" PART\_BOUNDARY;

static const char\* \_STREAM\_BOUNDARY = "\r\n--" PART\_BOUNDARY "\r\n";

static const char\* \_STREAM\_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

httpd\_handle\_t camera\_httpd = NULL;

httpd\_handle\_t stream\_httpd = NULL;

static const char PROGMEM INDEX\_HTML[] = R"rawliteral(

<html>

<head>

<title>ESP32-CAM Robot</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body { font-family: Arial; text-align: center; margin:0px auto; padding-top: 30px;}

table { margin-left: auto; margin-right: auto; }

td { padding: 8 px; }

.button {

background-color: #2f4468;

border: none;

color: white;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 18px;

margin: 6px 3px;

cursor: pointer;

-webkit-touch-callout: none;

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

-webkit-tap-highlight-color: rgba(0,0,0,0);

}

img { width: auto ;

max-width: 100% ;

height: auto ;

}

</style>

</head>

<body>

<h1>ESP32-CAM Robot</h1>

<img src="" id="photo" >

<table>

<tr><td colspan="3" align="center"><button class="button" onmousedown="toggleCheckbox('forward');" ontouchstart="toggleCheckbox('forward');" onmouseup="toggleCheckbox('stop');" ontouchend="toggleCheckbox('stop');">Forward</button></td></tr>

<tr><td align="center"><button class="button" onmousedown="toggleCheckbox('left');" ontouchstart="toggleCheckbox('left');" onmouseup="toggleCheckbox('stop');" ontouchend="toggleCheckbox('stop');">Left</button></td><td align="center"><button class="button" onmousedown="toggleCheckbox('stop');" ontouchstart="toggleCheckbox('stop');">Stop</button></td><td align="center"><button class="button" onmousedown="toggleCheckbox('right');" ontouchstart="toggleCheckbox('right');" onmouseup="toggleCheckbox('stop');" ontouchend="toggleCheckbox('stop');">Right</button></td></tr>

<tr><td colspan="3" align="center"><button class="button" onmousedown="toggleCheckbox('backward');" ontouchstart="toggleCheckbox('backward');" onmouseup="toggleCheckbox('stop');" ontouchend="toggleCheckbox('stop');">Backward</button></td></tr>

</table>

<script>

function toggleCheckbox(x) {

var xhr = new XMLHttpRequest();

xhr.open("GET", "/action?go=" + x, true);

xhr.send();

}

window.onload = document.getElementById("photo").src = window.location.href.slice(0, -1) + ":81/stream";

</script>

</body>

</html>

)rawliteral";

static esp\_err\_t index\_handler(httpd\_req\_t \*req){

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, (const char \*)INDEX\_HTML, strlen(INDEX\_HTML));

}

static esp\_err\_t stream\_handler(httpd\_req\_t \*req){

camera\_fb\_t \* fb = NULL;

esp\_err\_t res = ESP\_OK;

size\_t \_jpg\_buf\_len = 0;

uint8\_t \* \_jpg\_buf = NULL;

char \* part\_buf[64];

res = httpd\_resp\_set\_type(req, \_STREAM\_CONTENT\_TYPE);

if(res != ESP\_OK){

return res;

}

while(true){

fb = esp\_camera\_fb\_get();

if (!fb) {

Serial.println("Camera capture failed");

res = ESP\_FAIL;

} else {

if(fb->width > 400){

if(fb->format != PIXFORMAT\_JPEG){

bool jpeg\_converted = frame2jpg(fb, 80, &\_jpg\_buf, &\_jpg\_buf\_len);

esp\_camera\_fb\_return(fb);

fb = NULL;

if(!jpeg\_converted){

Serial.println("JPEG compression failed");

res = ESP\_FAIL;

}

} else {

\_jpg\_buf\_len = fb->len;

\_jpg\_buf = fb->buf;

}

}

}

if(res == ESP\_OK){

size\_t hlen = snprintf((char \*)part\_buf, 64, \_STREAM\_PART, \_jpg\_buf\_len);

res = httpd\_resp\_send\_chunk(req, (const char \*)part\_buf, hlen);

}

if(res == ESP\_OK){

res = httpd\_resp\_send\_chunk(req, (const char \*)\_jpg\_buf, \_jpg\_buf\_len);

}

if(res == ESP\_OK){

res = httpd\_resp\_send\_chunk(req, \_STREAM\_BOUNDARY, strlen(\_STREAM\_BOUNDARY));

}

if(fb){

esp\_camera\_fb\_return(fb);

fb = NULL;

\_jpg\_buf = NULL;

} else if(\_jpg\_buf){

free(\_jpg\_buf);

\_jpg\_buf = NULL;

}

if(res != ESP\_OK){

break;

}

//Serial.printf("MJPG: %uB\n",(uint32\_t)(\_jpg\_buf\_len));

}

return res;

}

static esp\_err\_t cmd\_handler(httpd\_req\_t \*req){

char\* buf;

size\_t buf\_len;

char variable[32] = {0,};

buf\_len = httpd\_req\_get\_url\_query\_len(req) + 1;

if (buf\_len > 1) {

buf = (char\*)malloc(buf\_len);

if(!buf){

httpd\_resp\_send\_500(req);

return ESP\_FAIL;

}

if (httpd\_req\_get\_url\_query\_str(req, buf, buf\_len) == ESP\_OK) {

if (httpd\_query\_key\_value(buf, "go", variable, sizeof(variable)) == ESP\_OK) {

} else {

free(buf);

httpd\_resp\_send\_404(req);

return ESP\_FAIL;

}

} else {

free(buf);

httpd\_resp\_send\_404(req);

return ESP\_FAIL;

}

free(buf);

} else {

httpd\_resp\_send\_404(req);

return ESP\_FAIL;

}

sensor\_t \* s = esp\_camera\_sensor\_get();

int res = 0;

if(!strcmp(variable, "forward")) {

Serial.println("Forward");

digitalWrite(MOTOR\_1\_PIN\_1, 1);

digitalWrite(MOTOR\_1\_PIN\_2, 0);

digitalWrite(MOTOR\_2\_PIN\_1, 1);

digitalWrite(MOTOR\_2\_PIN\_2, 0);

}

else if(!strcmp(variable, "left")) {

Serial.println("Left");

digitalWrite(MOTOR\_1\_PIN\_1, 0);

digitalWrite(MOTOR\_1\_PIN\_2, 1);

digitalWrite(MOTOR\_2\_PIN\_1, 1);

digitalWrite(MOTOR\_2\_PIN\_2, 0);

}

else if(!strcmp(variable, "right")) {

Serial.println("Right");

digitalWrite(MOTOR\_1\_PIN\_1, 1);

digitalWrite(MOTOR\_1\_PIN\_2, 0);

digitalWrite(MOTOR\_2\_PIN\_1, 0);

digitalWrite(MOTOR\_2\_PIN\_2, 1);

}

else if(!strcmp(variable, "backward")) {

Serial.println("Backward");

digitalWrite(MOTOR\_1\_PIN\_1, 0);

digitalWrite(MOTOR\_1\_PIN\_2, 1);

digitalWrite(MOTOR\_2\_PIN\_1, 0);

digitalWrite(MOTOR\_2\_PIN\_2, 1);

}

else if(!strcmp(variable, "stop")) {

Serial.println("Stop");

digitalWrite(MOTOR\_1\_PIN\_1, 0);

digitalWrite(MOTOR\_1\_PIN\_2, 0);

digitalWrite(MOTOR\_2\_PIN\_1, 0);

digitalWrite(MOTOR\_2\_PIN\_2, 0);

}

else {

res = -1;

}

if(res){

return httpd\_resp\_send\_500(req);

}

httpd\_resp\_set\_hdr(req, "Access-Control-Allow-Origin", "\*");

return httpd\_resp\_send(req, NULL, 0);

}

void startCameraServer(){

httpd\_config\_t config = HTTPD\_DEFAULT\_CONFIG();

config.server\_port = 80;

httpd\_uri\_t index\_uri = {

.uri = "/",

.method = HTTP\_GET,

.handler = index\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t cmd\_uri = {

.uri = "/action",

.method = HTTP\_GET,

.handler = cmd\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t stream\_uri = {

.uri = "/stream",

.method = HTTP\_GET,

.handler = stream\_handler,

.user\_ctx = NULL

};

if (httpd\_start(&camera\_httpd, &config) == ESP\_OK) {

httpd\_register\_uri\_handler(camera\_httpd, &index\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &cmd\_uri);

}

config.server\_port += 1;

config.ctrl\_port += 1;

if (httpd\_start(&stream\_httpd, &config) == ESP\_OK) {

httpd\_register\_uri\_handler(stream\_httpd, &stream\_uri);

}

}

void setup() {

WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //disable brownout detector

pinMode(MOTOR\_1\_PIN\_1, OUTPUT);

pinMode(MOTOR\_1\_PIN\_2, OUTPUT);

pinMode(MOTOR\_2\_PIN\_1, OUTPUT);

pinMode(MOTOR\_2\_PIN\_2, OUTPUT);

Serial.begin(115200);

Serial.setDebugOutput(false);

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\_sscb\_sda = SIOD\_GPIO\_NUM;

config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

config.pin\_pwdn = PWDN\_GPIO\_NUM;

config.pin\_reset = RESET\_GPIO\_NUM;

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

//config.pixel\_format = PIXFORMAT\_RGB565; // for face detection/recognition

if(psramFound()){

config.frame\_size = FRAMESIZE\_VGA;

config.jpeg\_quality = 10;

config.fb\_count = 2;

} else {

config.frame\_size = FRAMESIZE\_SVGA;

config.jpeg\_quality = 12;

config.fb\_count = 1;

}

// 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;

}

// Wi-Fi connection

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.print("Camera Stream Ready! Go to: http://");

Serial.println(WiFi.localIP());

// Start streaming web server

startCameraServer();

}

void loop() {

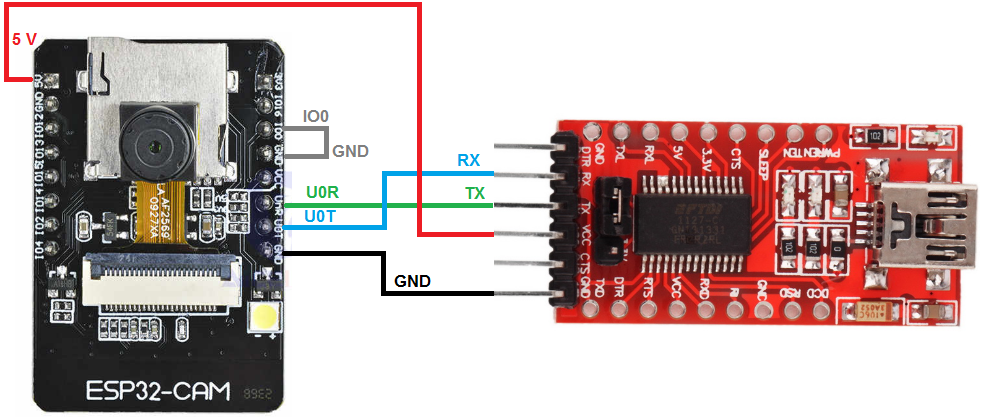
}

# Testing

## Description of Testing

**7.1 Testing Description**

* Testing was carried out in the environment of the Arduino IDE.
* A basic Blink program was initially loaded onto the ESP32-CAM to verify that it was working and would accept code.
* The Blink test test was successful, demonstrating that the ESP32-CAM could be powered and programmed correctly.
* The motor control program was subsequently loaded onto the ESP32-CAM.
* The motor control code loaded and ran without compilation or runtime errors.
* The robot movement (forward, backward, left, right, and stop) was tested via the web interface and confirmed successfully.
* The live video streaming feature was tested by opening the ESP32-CAM's IP address in a smartphone and PC.
* The video feed was smooth, and control commands were fast.
* The ultrasonic sensor and buzzer module was tested by creating obstacles in front of the robot.
* The buzzer was triggered properly when an object was found within 20 cm.
* Furthermore, the SONAR readings of the ultrasonic sensor were checked using the SONAR graph visualization.
* The graph showed distance readings in accordance with the scanning angle of the servo, confirming proper sensor behavior and detection of obstacles.



**Fig 5. ESP32-CAM Programming Setup using USB-to-Serial (FTDI) Module**

## Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case #**  (Number each test case) | **Test Case Name** | **Test Case Description**  (Describe reason for test or functionality being tested.) | **Inputs**  (What actions or data needs to be provided?) | **Expected Output**  (Describe what should be the result of the input actions) | **Actual Output** | **Status** |
| 1. | Blink Test | Check basic function of ESP32-CAM | Upload Blink program | LED blinks at regular intervals | LED blinked successfully | Pass |
| 2. | Motor Control Code Upload Test | Check upload and execution of motor control program | Upload motor control code | Program is uploaded without errors; robot moves through web controls | Code uploaded and robot moved successfully | Pass |
| 3. | Live Video Stream Test | Verify live video streaming from ESP32-CAM | Connect to ESP32-CAM IP address via Wi-Fi | Videos feed shown properly on smartphone/PC | Live video feed operational successfully | Pass |
| 4. | Ultrasonic Obstacle Detection Test | Verify obstacle detection and buzzer alert function | Position obstacle at 30 cm | Buzzer turns on when object found between 20 cm | Buzzer turned on properly | Pass |
| 5. | SONAR Graph Visualization Test | Check ultrasonic sensor reading and plotting during servo scanning | Watch serial sonar graph in Arduino IDE | Smoothly changing distance readings based on obstacle position | Sonar graph shown correctly | Pass |

# 8. Results and Discussion

**8. Results and Discussion**

The Surveillance and Obstacle Detection Robot was conceptualized, built, and programmed based on the specified project requirements.

Testing was done in phases, and the following major observations were recorded.

**8.1 Results**

**Successful Setup and Programming:**

* Arduino IDE was successfully installed and set up for the ESP32-CAM board.
* Required libraries and board definitions for ESP32 were installed without any problem.
* The ESP32-CAM was recognized and communicated correctly through the CH340 USB driver.

**Successful Blink Test:**

* A simple LED blink software was loaded on to the ESP32-CAM.
* The internal LED blinked from time to time, verifying the working of ESP32-CAM and its availability for further code upload.

**Successful Motor Control Programming:**

* The motor control software upload on the ESP32-CAM was successful.
* The four DC motors reacted suitably to commands remotely sent on Wi-Fi web pages for move forward, reverse, left turn, right turn, and halt directions.
* Successful Live Video Streaming
* Live video was streamed from ESP32-CAM to smartphones and PCs through the allocated IP address.
* Video streaming was stable, with good clarity and refresh rate for remote navigation.

**Successful Obstacle Detection and Buzzer Activation:**

* The ultrasonic sensor attached on the servo motor scanned the area around it.
* Obstacles at 20 cm were detected precisely.
* The buzzer fired appropriately when obstacles were found in close proximity, giving an audio notification.

**Successful SONAR Graph Visualization:**

* Ultrasonic sensor distance readings were graphed in a SONAR-type graph.
* The graph validated sensor and servo motor rotation through 15° to 165° correct functioning.

**8.2 Discussion**

* All key functionalities of the system, such as ESP32-CAM configuration, motor control, Wi-Fi live video streaming, and ultrasonic-based obstacle detection, were implemented and tested successfully.
* The motor movement in all directions was silky and active under real-time control instructions through the web interface.
* The live video streaming function gave the user real-time visual feedback, helping with navigation and control.
* The obstacle detection system based on ultrasonic scanning and buzzer indication increased the safety of the robot during operation.
* The SONAR graph visualization offered an intuitive real-time display of obstacle distances, confirming the accuracy of the scanning.
* The project achieved its goals of developing a manually controlled surveillance robot with real-time obstacle detection and live video feedback, employing low-cost and readily available parts.

# Conclusion

The "Surveillance and Obstacle Detection Robot using ESP32-CAM" project was intended to develop a remotely controlled, low-cost robot for real-time video surveillance and obstacle detection.

The system was planned, built, and implemented effectively with the ESP32-CAM as the central controller, accompanied by the L298N motor driver, four DC motors, a servo-mounted ultrasonic sensor, and a buzzer module.

Initial testing phases, such as the Blink test and motor control programming, were accomplished successfully without any problems. The robot showed smooth navigation according to user commands through a Wi-Fi-based web interface.

Live video streaming was accomplished effectively, giving real-time visual feedback to the user. The ultrasonic sensor properly detected obstacles up to a range of 20 cm, and buzzer warnings functioned as expected. SONAR graph visualization also verified the accuracy of distance measurements during scanning.

This project gave useful hands-on experience in embedded systems, robotics, Wi-Fi communication, and real-time remote control.

# Scope for Further Enhancement

Although the main objectives of developing a manually controlled surveillance and obstacle detection robot were fully attained, a number of areas have been determined as a scope for further enhancement to enhance the performance, reliability, and capabilities of the system:

**Autonomous Navigation:**

* Future major enhancement is to shift from manual control to autonomous navigation.
* The use of sensors, algorithms, and path-planning methods can be used to make the robot automatically patrol the set paths without any intervention.
* Self-learning obstacle avoidance and decision-making function will enable the robot for authentic security uses.

**Lightweight Advanced Object Detection Models:**

* Including compact deep learning models like YOLO-Tiny or MobileNet-SSD may enable real-time object detection and classification within the robot.
* Tuned-up AI models may enhance surveillance without taxing system resources.

**Cloud Accessibility and IoT Compatibility:**

* Including cloud storage can enable video frames, range readings, or alarm alerts to be remotely accessed.
* IoT integration can facilitate the transmission of real-time alerts (e.g., obstacle detection alarms) to mobile apps or cloud boards.

**Obstacle Avoidance Upgrades:**

* Placing multiple ultrasonic sensors in various angles will enable broader and more accurate obstacle detection.
* Simple path-planning algorithms can help the robot adjust routes dynamically and avoid collisions automatically.

**Multi-Robot Cooperation (Future Enhancement):**

* Creating a network of several patrol robots that could communicate and coordinate would allow coverage of large areas like campuses, shopping centres, or industrial parks.
* Communication and task-sharing protocols can enhance system scalability and dependability.

**Autonomous Charging:**

* Adding an autonomous charging station will permit the robot to charge itself when the battery level is low, allowing continuous operation.

**User Interface Improvements:**

* Developing a mobile app to control the robot, view the live video stream, and get real-time alerts will improve user experience.
* Additional advanced features like remote manual override, cloud connectivity, and push notifications can be included.

# Bibliography

|  |  |
| --- | --- |
| [1] | S. M.-M. P. I.-I. C. M.-B. S. &. A.-R. F. J. Lafuente-Arroyo, "RGB camera-based fallen person detection system embedded on a mobile platform," *Expert Systems with Applications,,* vol. 197, 2022. |
| [2] | D. F. J. D. B. &. V. R. Nunes, "Real-time vision based obstacle detection in maritime environments," *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC),* pp. 243-248, 2022. |
| [3] | U. M. U. S. D. T. M. M. &. Y. V. Sharma, "Mobile Robot for Security Applications in Remotely Operated Advanced Reactors," *Applied Sciences,* vol. 14, 2024. |
| [4] | M.-F. R. &. S. Z.-S. Lee, " Autonomous Surveillance for an Indoor Security Robot. Processes," vol. 10, 2022. |
| [5] | W.-C. &. J. J.-G. Chang, "Automatic Outdoor Patrol Robot Based on Sensor Fusion and Face Recognition Methods," *Applied Sciences,* vol. 11, 2021. |
| [6] | Q. X. L. C. S. G. B. Q. W. Y. G. T. H. Y. W. H. .. &. D. X. G. Shi, "Manipulator-based autonomous inspections at road checkpoints: Application of faster YOLO for detecting large objects.," *Defence Technology,* vol. 18, pp. 937-951, 2022. |
| [7] | S. S. V. T. &. P. S. Maram, "Neural network and ROS based threat detection and patrolling assistance," *In 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP),* pp. 1-5, 2019. |
| [8] | Z. &. C. L. Zhang, " A Low-cost Environment-Interactive Patrol Inspection System with Navigation Based on Sensor-Fusion and Robotic Arm Contact Pose Feedback.," 2025. |
| [9] | S. &. R. J. K. Mittal, "Wadoro: An autonomous mobile robot for surveillance," *In 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES),* pp. 1- 6, 2016. |
| [10] | M. H. F. Z. M. H. R. T. &. V. R. C. Tanveer, "hree-dimensional outdoor object detection in quadrupedal robots for surveillance navigations," *In Actuators,* vol. 13, no. 10, p. 422, 2024. |
| [11] | C. W. &. C. C. Y. Lan, "Development of a low cost and path-free autonomous patrol system based on stereo vision system and checking flags," *Applied Sciences,* vol. 10, no. 3, 2020. |
| [12] | M. A. A. A. A. A. S. A. S. A. A. .. &. K. F. Faisal, "Robot-based solution for helping Alzheimer patients," *SLAS technology,* vol. 29, no. 3, 2024. |