



Graduation Project Object Detection System On Controlled Vehicle

Prepared by :

Mario Emad Edward	CS
Omar Mohammed Ahmed	CS
Ali Farouk Ali	CS
Moataz Ibrahim Gaber	CS
Nancy Khaled Sayed	CS
Yasmeen Mahdy Faheem	IS
Mohammed Eid Mahmoud	IS

**Supervised by
Prof.Dr. Mohamed Fouad**

**Co-Supervised by
Eng. Shady Ahmed
2023-2024**

Acknowledgment

Acknowledgment

Both scientifically and practically, we have put a lot of effort into our project. All of this wouldn't have been possible without Allah, Doctors, Assistants, and team members.

We all would like to extend our gratitude to the dean **Prof. Dr. Nabila Mohamed Hassan**.

We all have special thanks to **Dr. Mohamed Fouad** (The General supervisor) for his efforts and his guidance and constant supervision as well as for providing necessary information regarding the project and for his support in completing the project.

We all have special thanks to **Eng. Shady** for his continuous support with us and giving us comments and his kind cooperation and encouragement, which helped us in completing this project.

We would like to express my sincere appreciation to the team individuals and resources who contributed to the development of this documentation.

Table of Contents

Table of Contents

Table of Contents

Abstract.....	vii
List of Abbreviations.....	x
List of figures	xii
List of tables.....	xiv
<u>Chapter One: Introduction</u>	xv
1.1Project summary	1
1.2key features.....	2
1.3Motivation.....	4
1.4Project Objectives.....	6
1.5 Problems and solutions.....	11
<u>Chapter Two: Literature Review</u>	13
2.1Introduction.....	14
2.2Techniques Related to the literature.....	14
2.3Scability and Performance.....	16
<u>Chapter Three: Design and Methodology</u>	18
3.1Requirements and Analysis.....	19
3.1.1Hardware Process Model.....	19
3.1.2 Functional Requirements.....	20
3.1.3 Non-Functional Requirements.....	34
3.1.4 SOTWARE USED.....	35

3.2 Methodology.....	36
3.3 Analysis.....	38
<u>Chapter four 4- Result and Discussion</u>	43
4.1 Results.....	46
4.2 Discussion.....	47
 <u>Chapter five 5- Conclusion and Future Work</u>	48
5.1 Future Work	49
5.2 Conclusion.....	50
<u>References</u>	51
<u>Appendix</u>	52

Abstract

Abstract

A surveillance robot using ESP32-CAM is a system that combines the ESP32-CAM board and a robot chassis to create a mobile surveillance device. The ESP32-CAM is a low-cost development board that includes a small camera module and Wi-Fi connectivity, enabling it to capture and stream video. The robot chassis allows the device to move around and capture video in different locations, enhancing its surveillance capabilities. This system can be controlled through a web interface hosted on the ESP32-CAM board, allowing users to control the robot's movement, view live video streams, and take snapshots of the video feed. The integration of the camera module with the robot chassis enhances the mobility and flexibility of the system, making it capable of covering larger areas and navigating various environments.

Additionally, the system can be programmed to detect motion using computer vision algorithms, such as object detection and tracking, and send alerts to the user. This feature adds an extra layer of security by ensuring that users are promptly notified of any unusual activity. The surveillance robot using ESP32-CAM has potential applications in home security, monitoring remote locations, and industrial surveillance. Its low cost and easy-to-use interface make it an accessible solution for various surveillance needs.

Users can access the web interface from any device with internet connectivity, providing real-time monitoring and control over the robot. The ability to detect motion and send alerts enhances its effectiveness in ensuring security. This makes the surveillance robot a practical tool for both personal and professional use, offering an efficient and affordable way to enhance security and monitoring capabilities in various settings. The project stands out for its innovative approach, combining mobility, real-time video streaming, and advanced motion detection in a compact and cost-effective package.

Applications:

These projects have broad implications for understanding the surveillance robot using ESP32-CAM and the unmanned Arduino car have broad implications for understanding the foundational concepts of autonomous systems and surveillance technology. The simplicity and accessibility of the Arduino platform make the Unmanned Arduino Car an ideal educational tool for students and enthusiasts interested in robotics and autonomous technology. This project offers hands-on experience with basic programming, sensor integration, and autonomous navigation, providing valuable insights into the workings of self-driving vehicles.

Similarly, the ESP32-CAM Surveillance Robot offers a convenient solution for remote monitoring, with potential applications in home security, military, law enforcement, and space exploration. This robot allows users to control its movements and view live video streams through a web interface, enhancing its utility for real-time surveillance. Its ability to detect motion using computer vision algorithms and send alerts to users makes it an effective tool for ensuring security in various environments.

The low cost and user-friendly nature of these projects make them accessible to a wide audience, from hobbyists to professionals. The ESP32-CAM Surveillance Robot, in particular, stands out for its ability to navigate and monitor different locations, making it suitable for tasks that require mobility and real-time video streaming. These projects not only serve practical purposes but also contribute to the educational field by providing a hands-on approach to learning about robotics and surveillance technologies.

List of Abbreviation

List of Abbreviations

AI	Artificial Intelligence
DC	Direct Current
IOREF	Input /Output Reference
GND	Ground
UART	Universal Asynchronous Serial Receiver / Transmitter
SDA	Serial Data
SCL	Serial Clock
AREF	Analog Reference
ICSP	In-Circuit Serial Programmer
USB	Universal Serial Bus
LED	Light Emitting Diode
AC	Alternating current
IC	Integrated Circuit
SSD	Single Shot MultiBox Detect
MS COCO	Microsoft Common Objects in Context

List of Figures

List of Figures

● Figure 3.1	Arduino UNO R3 Board.....
● Figure 3.2	DC Motors.....
● Figure 3.3	L298N Driver.....
● Figure 3.4	Bluetooth Module HC-05.....
● Figure 3.5	Joystick.....
● Figure 3.6	ESP – 32 CAM
● Figure 3.7	RF Sender& Receiver.....
● Figure 3.8	SSD.....
● Figure 3.9	Object detection example.....
● Figure 3.10	Use case diagram
● Figure 3.11	Activity Diagram
● Figure 3.12	Sequence diagram
● Figure 3.13	Flowchart Diagram.....
● Figure 4.1	Reciver1.....
● Figure 4.2	Reciver2.....
● Figure 4.3	Sender.....
● Figure 4.4	WIFI CONNECTION.....
● Figure 4.5	AI.....

List of Tables

List of Tables

Table 3.1	Arduino UNO R3 Pins
Table 3.2	L298N Module Pinout Configuration
Table 3.3	HC-05 Bluetooth Module Pin
Table 3.4	Joystick Pinout

Chapter One

“Introduction”

1.1Project summary

The development of surveillance robots has been a growing area of interest due to its applications in various fields such as security, industrial monitoring, and home automation. The surveillance robot using the ESP32-CAM is a project that focuses on developing a low-cost and efficient surveillance robot for indoor and outdoor use. Surveillance robots are becoming increasingly popular in modern times due to their ability to monitor and collect information from a remote location. The use of the ESP32 CAM module, which is an integrated camera module with Bluetooth connectivity, has made it possible to create surveillance robots that can be controlled remotely. The ESP32 CAM module is a small-sized camera module that can be easily integrated into a robot. The module provides high quality images and can be controlled using the ESP32 microcontroller. The ESP32 CAM module also has Bluetooth connectivity, which allows for easy communication with remote control station or a computer. With the ESP32 CAM module, it is possible to create a surveillance robot that can be controlled remotely from a computer or a mobile device. The robot can be programmed to move around a specific area and capture images or videos of the surroundings. The images and videos can be transmitted wirelessly to the remote-control station, where they can be viewed and analyzed in real-time. The use of the ESP32 CAM module in surveillance robots has many advantages, including the ability to capture high quality images and videos, the ability to control the robot remotely, and the ability to transmit data wirelessly. Additionally, the ESP32 CAM module is easy to use and can be integrated into a robot without requiring any additional hardware. Overall, the ESP32 CAM module is an excellent tool for creating surveillance robots that can be used in a variety of applications, including security, monitoring, and inspection.

1.2 Key Features:

An object detection system for a controlled vehicle should incorporate several key features to ensure effective, reliable, and safe operation.

1. Real-time Object Detection

- **High Accuracy:** The system should accurately detect and classify various objects, including pedestrians, vehicles, obstacles, and road signs, in real time.
- **Low Latency:** Quick processing and response times are crucial for making timely decisions and ensuring safety.

2. Multi-Sensor Integration

- **Cameras:** High-resolution cameras to capture visual data for detecting and classifying objects.
- **LiDAR:** LiDAR sensors to provide precise distance measurements and 3D mapping of the environment.
- **Radar:** Radar sensors to detect objects in various weather conditions and provide speed information.
- **Ultrasonic Sensors:** For close-range object detection, especially useful for parking and low-speed maneuvering.

3. Sensor Fusion

- **Data Fusion Algorithms:** Combining data from multiple sensors to improve detection accuracy, reliability, and robustness.
- **Redundancy:** Ensuring that the system remains functional even if one sensor fails.

4. Advanced Machine Learning Algorithms

- **Deep Learning Models:** Using state-of-the-art deep learning models such as Convolutional Neural Networks (CNNs) for object detection and classification.
- **Pre-trained Models:** Utilizing pre-trained models for quick deployment and fine-tuning them with specific datasets relevant to the vehicle's operational environment.

5. Environmental Adaptability

- **Lighting Conditions:** Capable of operating in varying lighting conditions, including daylight, nighttime, and adverse weather.
- **Weather Resistance:** Robust detection performance in different weather conditions such as rain, fog, and snow.

6. Object Tracking

- **Real-time Tracking:** Continuously track detected objects to predict their movement and enhance situational awareness.
- **Trajectory Prediction:** Predict the future positions and paths of moving objects to aid in navigation and collision avoidance.

7. Robust Control Integration

- **Autonomous Navigation:** Integration with the vehicle's control system to enable autonomous or assisted navigation based on detected objects.
- **Emergency Braking:** Automatic braking in response to detected obstacles or imminent collisions.

8. User Interface and Alerts

- **Visual Feedback:** Providing real-time visual feedback to the driver or operator through a user-friendly interface.
- **Auditory and Visual Alerts:** Generating alerts for detected objects that pose potential hazards, improving safety and awareness.

9. Data Logging and Analysis

- **Event Logging:** Recording data from sensors and detection events for post-analysis and debugging.
- **Performance Metrics:** Monitoring and analyzing system performance metrics such as detection accuracy, false positives, and processing time.

10. Scalability and Upgradability

- **Modular Design:** Ensuring the system is modular and can be easily upgraded with new sensors or improved algorithms.
- **Software Updates:** Capability for over-the-air (OTA) software updates to enhance functionality and fix bugs.

11. Safety and Compliance

- **Standards Compliance:** Adherence to relevant safety and regulatory standards for autonomous and semi-autonomous vehicles.
- **Fail-Safe Mechanisms:** Implementing fail-safe mechanisms to ensure the vehicle can handle system failures without compromising safety.

12. Adaptive Learning and Improvement

- **Continuous Learning:** Utilizing adaptive learning techniques to improve object detection accuracy over time based on new data and environmental changes.
- **Feedback Loop:** Incorporating a feedback loop where the system learns from mistakes and enhances performance in subsequent operations.

13. Energy Efficiency

- **Low Power Consumption:** Designing the system to be energy-efficient, which is especially important for electric vehicles and drones.

1.3 Motivation:

1. Safety Enhancement

- **Collision Avoidance:** One of the primary motivations is to enhance vehicle safety by preventing collisions with pedestrians, other vehicles, and obstacles. An effective object detection system can detect hazards in real-time and initiate preventive actions such as emergency braking or evasive maneuvers.
- **Accident Reduction:** By improving the vehicle's ability to detect and respond to potential dangers, the system can significantly reduce the likelihood of accidents, thereby saving lives and reducing injuries.

2. Autonomous and Semi-Autonomous Navigation

- **Enabling Autonomy:** Object detection is a critical component of autonomous vehicles, enabling them to navigate complex environments without human intervention. Accurate object detection systems are essential for the reliable operation of self-driving cars.
- **Driver Assistance:** For semi-autonomous vehicles, the system provides advanced driver assistance features such as lane-keeping, adaptive cruise control, and automated parking, enhancing the overall driving experience.

3. Efficiency and Convenience

- **Traffic Management:** Object detection systems can help in managing traffic more efficiently by detecting and responding to traffic signals, signs, and road conditions, leading to smoother and faster journeys.
- **Parking Assistance:** Automated parking systems enabled by object detection can save time and reduce stress for drivers by precisely maneuvering the vehicle into tight spaces.

4. Technological Advancement

- **Innovation:** Developing cutting-edge object detection systems pushes the boundaries of current technology, leading to advancements in areas such as artificial intelligence, sensor technology, and data processing.
- **Competitive Edge:** For automotive manufacturers and tech companies, investing in advanced object detection systems can provide a competitive advantage in the rapidly evolving market for smart and autonomous vehicles.

5. Environmental Impact

- **Fuel Efficiency:** By optimizing driving patterns and reducing stop-and-go traffic through better detection and response to road conditions, object detection systems can contribute to improved fuel efficiency and reduced emissions.
- **Electric Vehicle Integration:** Object detection systems can enhance the functionality and appeal of electric vehicles (EVs), promoting their adoption and contributing to environmental sustainability.

6. Regulatory Compliance

- **Meeting Standards:** Increasingly, regulatory bodies are mandating the inclusion of advanced safety features in vehicles. An effective object detection system helps manufacturers meet these regulatory requirements and avoid penalties.
- **Insurance Incentives:** Vehicles equipped with advanced safety systems may qualify for lower insurance premiums, providing a financial incentive for both manufacturers and consumers to adopt these technologies.

7. Improved User Experience

- **Enhanced Comfort:** Advanced object detection systems can reduce driver fatigue by taking over routine tasks such as braking, accelerating, and steering in certain conditions.
- **User Trust and Adoption:** Reliable object detection systems build user trust in autonomous and semi-autonomous technologies, encouraging wider adoption and acceptance.

8. Data Collection and Analysis

- **Real-world Insights:** The data collected by object detection systems can provide valuable insights into traffic patterns, road conditions, and driver behavior, informing future improvements in vehicle design and urban planning.
- **Continuous Improvement:** Analysis of this data enables continuous refinement of detection algorithms, leading to incremental improvements in system performance and safety.

1.4 Project Objectives:

1. System Design

Requirement Analysis:

- **Identify Object Types:** Determine the specific objects the system needs to detect (e.g., pedestrians, cyclists, other vehicles, road signs, obstacles).
- **Operational Environment:** Define the environments where the vehicle will operate (e.g., urban, rural, highway, off-road).
- **Performance Metrics:** Establish target metrics for detection accuracy, processing speed, and system reliability.
- **Constraints:** Consider constraints such as power consumption, processing capability, and physical space on the vehicle.

Sensor Selection:

- **Cameras:** High-resolution cameras for visual data, considering field of view, frame rate, and low-light performance.
- **LiDAR:** LiDAR sensors for precise 3D mapping and distance measurement.
- **Radar:** Radar sensors for detecting objects in various weather conditions and measuring speed.
- **Ultrasonic Sensors:** For short-range detection, useful in parking and low-speed scenarios.
- **Sensor Placement:** Determine optimal sensor placement on the vehicle for maximum coverage and minimal blind spots.

Hardware Integration:

- **Processing Unit:** Select a powerful processing unit (e.g., GPU, TPU, or specialized AI chip) to handle data from multiple sensors.
- **Communication Protocols:** Ensure robust and low-latency communication protocols (e.g., CAN bus, Ethernet) between sensors and the processing unit.
- **Power Management:** Design an efficient power management system to ensure consistent operation of all sensors and processing units.

Software Architecture:

- **Data Acquisition:** Develop modules for continuous data collection from all sensors.
- **Preprocessing:** Implement preprocessing steps such as noise reduction, sensor calibration, and data normalization.
- **Detection Algorithms:** Choose or develop algorithms for object detection, such as YOLO, SSD, or custom CNN models.
- **Post-Processing:** Include post-processing steps like non-maximum suppression to refine detected objects.

2. Implementation

Algorithm Development:

- **Model Selection:** Choose a suitable deep learning architecture for object detection, like Faster R-CNN, YOLO, or SSD.
- **Dataset Preparation:** Collect and annotate a large dataset covering various scenarios, object types, and environmental conditions.
- **Training:** Train the model using high-performance computing resources, ensuring it generalizes well to new data.
- **Optimization:** Optimize the model for real-time inference using techniques like model pruning, quantization, and hardware-specific optimizations.

Sensor Data Fusion:

- **Fusion Algorithms:** Develop algorithms to combine data from multiple sensors, enhancing the reliability and accuracy of object detection.
- **Synchronization:** Ensure all sensors are time-synchronized to provide coherent data fusion.

Control System Integration:

- **Real-Time Communication:** Implement a robust communication protocol to relay detection data to the vehicle's control system with minimal delay.
- **Decision-Making Algorithms:** Develop algorithms for real-time decision-making based on detected objects, enabling actions like braking, steering, and acceleration.

User Interface Development:

- **Visualization Tools:** Create tools for real-time visualization of sensor data and detected objects, aiding in debugging and system monitoring.
- **Alert System:** Implement an alert system to notify the driver of detected objects that pose potential hazards.

3. Performance Evaluation

Testing and Validation:

- **Controlled Environments:** Conduct initial testing in controlled environments to fine-tune the system.
- **Real-World Scenarios:** Test the system in various real-world scenarios to ensure robustness and reliability.
- **Simulation Testing:** Use simulation platforms to test edge cases and scenarios difficult to reproduce in real life.

Performance Metrics:

- **Accuracy:** Measure the detection accuracy, precision, recall, and F1 score.
- **Latency:** Evaluate the system's response time from detection to action.
- **Robustness:** Assess system performance under different lighting, weather conditions, and speeds.

Safety and Compliance:

- **Regulatory Standards:** Ensure the system meets relevant safety standards and automotive regulations (e.g., ISO 26262 for functional safety).
- **Fail-Safe Mechanisms:** Implement fail-safe mechanisms to handle sensor failures or detection errors without compromising safety.

4. Deployment and Maintenance

Deployment Strategy:

- **Initial Setup:** Develop a detailed plan for the initial setup, calibration, and configuration of the system on the vehicle.
- **Field Testing:** Conduct extensive field testing to validate the system's performance post-deployment.
- **Scalability:** Design the system for easy scalability to different vehicle models and configurations.

Continuous Monitoring and Improvement:

- **Monitoring Tools:** Implement tools for continuous monitoring of system performance and health.
- **Software Updates:** Develop mechanisms for over-the-air (OTA) updates to improve the system and fix issues based on new data.

User Training and Documentation:

- **Training Programs:** Create training programs for users and operators, covering system operation, troubleshooting, and maintenance.
- **Comprehensive Documentation:** Provide detailed documentation, including user manuals, technical guides, and maintenance procedures.

5. Future Enhancements

Scalability:

- **Modular Architecture:** Design the system with a modular architecture to allow for easy integration of additional sensors and improved algorithms.
- **Performance Upgrades:** Plan for periodic hardware and software upgrades to keep the system up-to-date with the latest technologies.

● Object Detection

The integration of object detection systems within controlled vehicles stands as a pivotal milestone in the journey towards safer and more efficient transportation. These systems amalgamate cutting-edge sensor technologies with advanced algorithms to empower vehicles with the ability to perceive and respond to their surroundings in real-time. This project delves into the development of such a system, aimed at enhancing the capabilities of controlled vehicles to detect and react to various entities such as pedestrians, vehicles, obstacles, and road signs. Through meticulous design, implementation, and evaluation, the project strives to achieve a robust and reliable object detection framework that contributes significantly to the safety, efficiency, and autonomy of controlled vehicles.

1. **Comprehensive Sensor Integration:** Integrate a diverse array of sensors, including cameras, LiDAR, radar, and ultrasonic sensors, to provide comprehensive coverage of the vehicle's environment and facilitate accurate object detection in various driving conditions.
2. **Advanced Algorithm Development:** Develop and implement advanced object detection algorithms, leveraging state-of-the-art techniques such as deep learning and sensor fusion to achieve high levels of accuracy and reliability in detecting pedestrians, vehicles, and other relevant entities.
3. **Real-Time Processing Optimization:** Optimize the object detection algorithms and processing pipelines for real-time performance, minimizing latency and computational overhead to enable swift and precise responses to potential hazards.
4. **Training Data Acquisition and Annotation:** Gather and annotate a diverse dataset of annotated images or video sequences containing objects pertinent to driving scenarios, ensuring the robustness and generalization of the object detection models.
5. **Model Training and Evaluation:** Train and evaluate the object detection models using the annotated dataset, employing techniques such as transfer learning and data augmentation to enhance performance metrics such as accuracy, precision, and recall.

6. **Adaptive Environmental Awareness:** Develop mechanisms to adaptively adjust the object detection system's parameters and algorithms based on environmental factors such as lighting conditions, weather, and terrain, ensuring consistent performance across diverse driving scenarios.
7. **Integration with Vehicle Control System:** Seamlessly integrate the object detection system with the vehicle's control system, enabling automated responses and decision-making based on detected objects, such as collision avoidance, lane keeping, and adaptive cruise control.
8. **Safety and Regulatory Compliance:** Conduct thorough validation and testing to ensure the safety and reliability of the object detection system, complying with relevant automotive safety standards and regulations for deployment in controlled vehicles.

1.5 Problems and Solutions

Sensor Limitations:

Problem: Sensors such as cameras, LiDAR, and radar have limitations in certain conditions, like low light, fog, or heavy rain, which can degrade the quality of the data they collect.

Solution: Implement sensor fusion techniques that combine data from multiple sensors to enhance reliability and accuracy in adverse conditions. For example, combining LiDAR and radar can compensate for the limitations of cameras in low visibility.

Real-time Processing:

Problem: Object detection systems require substantial computational power to process data in real-time, which can be challenging for onboard processors.

Solution: Utilize edge computing and hardware acceleration techniques, such as GPUs and FPGAs, to improve processing speed. Additionally, optimize algorithms to reduce computational load without compromising accuracy.

False Positives and Negatives:

Problem: Object detection systems can generate false positives (detecting nonexistent objects) and false negatives (failing to detect actual objects), which can lead to unsafe driving decisions.

Solution: Enhance the accuracy of detection algorithms using advanced machine learning techniques and extensive training on diverse datasets. Implement confidence scoring and cross-validation with multiple sensor data to reduce errors.

Dynamic and Complex Environments:

Problem: Controlled vehicles often operate in dynamic and complex environments with unpredictable elements such as pedestrians, animals, and other vehicles.

Solution: Develop robust algorithms capable of handling dynamic environments. This includes predictive modeling and adaptive learning techniques that allow the system to anticipate and react to changes in real-time.

Integration with Vehicle Control Systems:

Problem: Integrating object detection systems with vehicle control systems to ensure smooth and safe operation can be complex.

Solution: Create well-defined interfaces and communication protocols between the detection system and the vehicle's control units. Implement rigorous testing and validation processes to ensure seamless integration and operation.

Cost and Power Consumption:

Problem: High-quality sensors and powerful processing units can be expensive and consume significant power, impacting the overall feasibility of the system.

Solution: Research and develop cost-effective sensor technologies and energy-efficient processing solutions. Explore alternative power sources, such as advanced batteries or energy harvesting techniques, to mitigate power consumption issues.

Ethical and Privacy Concerns:

Problem: The deployment of object detection systems raises ethical and privacy concerns, particularly regarding data collection and usage.

Solution: Establish clear guidelines and regulations for data handling to protect user privacy. Incorporate privacy-preserving techniques such as data anonymization and secure data transmission protocols.

Chapter 2

“Literature Review”

2.1Introduction

Unmanned Arduino cars represent a fascinating intersection of robotics, automation, and embedded systems, offering a platform for experimentation and development in the field of autonomous vehicles. This literature review explores key themes and advancements in the realm of Unmanned Arduino Cars, highlighting the evolution of technology, challenges faced, and potential applications. The development of these cars has been driven by advancements in sensor technology, machine learning algorithms, and real-time processing capabilities. Key areas of focus include the integration of various sensors for navigation and obstacle avoidance, the implementation of path-planning algorithms, and the use of wireless communication for remote control and monitoring. Additionally, the review addresses the challenges such as ensuring reliability, scalability, and handling dynamic environments. Potential applications range from educational tools and research platforms to practical uses in logistics, surveillance, and even agricultural automation.

2.2Techniques related to the literature:

2.2.1 Artificial Intelligence:

Artificial Intelligence (AI) is the ability of a computer or a computer-controlled robot to think and perform tasks like humans. An artificially intelligent machine is one which is capable of learning, reasoning, planning, perception, problem solving. This field was discovered on the claim that human intelligence can be sufficiently described to the point that a machine can simulate it.

2.2.2 Robotics and Robots:

Robotics is an interdisciplinary research area at the interface of computer science and engineering. Robotics involves the design, construction, operation, and use of robots. The goal of robotics is to design intelligent machines to help human beings in their daily activities. This technology has resulted in automated machines that can replace humans in manufacturing processes or dangerous environments.

These robots have numerous structures depending on their functions. Generally, robots are grouped into:

- Manipulator robots (industrial robots)
- Mobile robots (autonomous vehicles)
- Self-reconfigurable robots, which are robots that can adjust themselves based on the task to be performed.

2.2.3 Robot Learning:

Robot learning involves the combination of robotics and machine learning by the learning of different algorithms by the robot to apply techniques which will enable it to obtain skills and adjust to its surroundings. Robot learning can take place by imitating humans or by self-learning guided by a human.

2.2.4 Autonomous Robot:

An Autonomous robot is one that can perform activities with a high level of self-control and without any form of external control. This kind of robot is achieved by integrating artificial intelligence, robotics, and information engineering.

Autonomous robots, just like human beings, additionally can settle on their own choices and afterward respond accordingly. A genuinely autonomous robot is one that can sense and recognize its environment, settle on choices dependent on what it sees and afterwards, actuate a movement in response. Regarding mobility, for instance, the decision-based activities incorporate but however are not restricted to the following: starting, stopping, and maneuvering around obstructions that are in their path.

Some mobile robots utilize ultrasound sensors to detect obstacles or infrared. These sensors work in a similar fashion to animal echolocation. The robot sends a beam of light, or a sound signal out and observes the reflection distance by identifying how long it takes the signal to bounce back. In some advanced robots, stereo vision is used. This method uses two cameras for depth perception and image recognition to detect and categorize different objects.

2.3 Scalability and Performance

Increasing Sensor Inputs:

- As controlled vehicles evolve, there is a need to accommodate a growing number of sensor inputs for comprehensive environmental perception. This scalability requirement involves integrating additional sensors such as LiDAR, radar, and ultrasonic sensors to enhance detection capabilities.

1. Expanding Detection Capabilities:

- Scalability also pertains to expanding the detection capabilities beyond basic object detection. Future systems may incorporate advanced features such as semantic segmentation, instance segmentation, and 3D object detection to provide richer environmental understanding.

2. Adapting to Varied Environments:

- A scalable object detection system should be adaptable to diverse environmental conditions and driving scenarios, including urban, rural, highway, and off-road environments. This involves developing algorithms and models that can generalize well across different conditions.

3. Integration with Autonomous Features:

- As vehicles progress towards higher levels of autonomy, scalability entails integrating object detection systems with other autonomous features such as path planning, decision-making, and control systems. This integration allows for seamless coordination and execution of driving tasks.

Performance

1. Real-time Processing:

- Performance is crucial for object detection systems on controlled vehicles, particularly in terms of real-time processing. Systems must meet stringent latency requirements to enable timely detection and response to dynamic environments and potential hazards.

2. Accuracy and Reliability:

- High performance entails achieving high levels of accuracy and reliability in object detection. Systems must minimize false positives and negatives to ensure safe and reliable driving behavior, even in challenging conditions.

3. Scalable Computational Infrastructure:

- Performance also encompasses the scalability of computational infrastructure to handle the increasing complexity of object detection algorithms and models. This may involve leveraging parallel processing, distributed computing, or cloud-based solutions to efficiently process large volumes of sensor data.

4. Energy Efficiency:

- Energy efficiency is a critical aspect of performance, particularly for onboard systems with limited power resources. Object detection systems should optimize energy consumption while maintaining high levels of computational throughput and accuracy.

Chapter Three

“Design and Methodology”

3.1 Requirements and Analysis

3.1.1 Hardware Process Model:

The first step is to assemble the hardware components required for the project. This includes the ESP32-CAM module, motor drivers, motors, wheels, chassis, battery, and infrared sensor. The ESP32-CAM module is connected to the motor drivers and the infrared sensor using jumper wires. The motors are connected to the motor drivers, and the wheels are attached to the motors. The battery is connected to the ESP32-CAM module and the motor drivers provide power to the robot.

- **Arduino UNO R3**
- **4 Wheel Robot Car Chassis (Includes Motors)**
- **Bluetooth Module HC-05**
- **L293D Motor Driver Shield**
- **Joystick**
- **3.7 V lithium battery**
- **Jumper wires**
- **ESP32-CAM**
- **RF Sender& Receiver**

3.1.2 Functional Requirements:

Functional requirements define what a system or product is supposed to do, Here's a list of functional requirements.

➤ Arduino UNO R3

Arduino Uno is a microcontroller board based on Atmega328. Arduino Uno already contains everything needed to support the microcontroller and easily connects it to a computer with a USB cable or supplies it with an AC to DC adapter or uses a battery to get it started. ATmega328 on Arduino Uno comes with a bootloader that allows us to upload new code to ATmega328 without using an external hardware program.

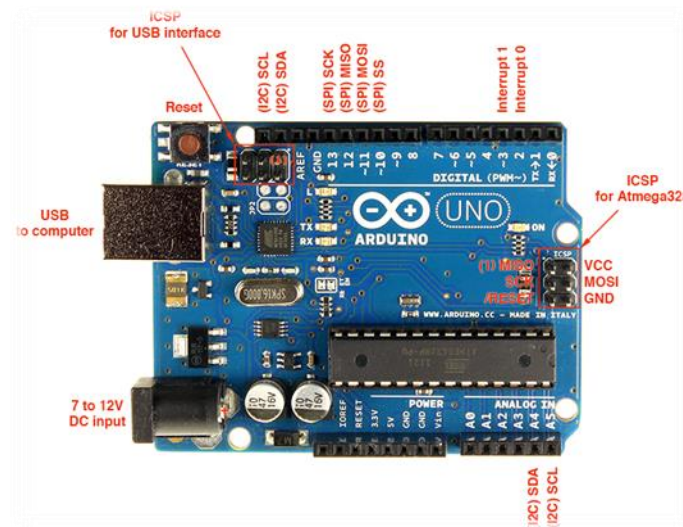


Figure 3.1 Arduino UNO R3 Board

▪ Arduino UNO Power Overview

- External power supply voltage: 7V to 12V DC
- USB power or externally via barrel jack connector

Table 3.1: Arduino UNO R3 Pins

Pin	Function
Vin	Voltage from External power jack
5V	5V output from on-board Voltage regulator chip
3.3V	3.3V output from on-board Voltage regulator chip
GND	3 pins for ground
IOREF	Tied to 5V, tells Arduino shields voltage level from which Arduino board operates
Reset	From RESET pin on MCU, tied to VCC through 10K resistor, pull to GND to reset

➤ Arduino UNO Digital Input/ Output Overview

- It has 14 Digital Inputs and Outputs.
- It has a logic level of 5V.
- Its Sink/Source is 20mA per pin.
- It utilizes Universal Asynchronous Serial Receiver/Transmitter (UART) for which pin(0) is for receiving and pin 1 is for transmitting.
- It has 8-bit PWM.
- It has a Serial Peripheral Interface.

➤ Arduino UNO Analog Inputs Overview

- Analog Input pins are A0, A1, A2, A3, A4, A5
- Its sampled input is from 0V to 5V, and it has 10 bits resolution.
- The AREF pin can be used to adjust the upper limit of the input range.
- Pin A4 is SDA pin for data.
- Pin A5 is SCL pin for clock.

➤ **Arduino UNO R3 Communication Overview**

- It can communicate with a computer, another Arduino, shields, sensors.
- Asynchronous communication (No clock):
 - UART TTL (5V)
 - On board ATmegaU16 programmed to function as USB to serial chip
 - ATmegaU16 uses standard USB COM drivers so that no external drivers are required for communication with the board.
- Synchronous communication (Clock) uses SPI Pins 10, 11, 12, 13, and TWI pins A4 and A5.

➤ **Arduino Programming Overview**

- The bootloader firmware comes already preinstalled with the Arduino UNO
- New sketches can be Uploaded via the USB
- External programmer can be used to upload sketches via the In-circuit serial programmer pins (ICSP)
- It has 32KB of Flash memory where the sketches are stored.
- It has 2KB of SRAM where the variables are stored until power cycles.
- It has 1KB of EEPROM which stores long term information.

DC Motor



Figure 3.2 DC Motors.

➤ DC Motor Overview

A DC motor is a device that converts electrical energy into mechanical energy as it rotates. A magnetic field is produced by an inductor inside the DC motor and this magnetic field creates a rotary motion as DC voltage is applied to its terminal. There is an Iron shaft inside the motor which is wrapped in a coil of wire. This shaft contains two fixed North and South magnets on both sides which causes both an attractive and repulsive force thereby producing a torque.

➤ Types of DC Motors:

- Brushless Motors
- Planetary Gear Motors
- Spur Gear Motors
- Stepper Motors

➤ Applications:

DC motors are used in various applications, including:

Electric vehicles and robotics.

Fans, blowers, and household appliances.

Conveyors and industrial machinery.

Power tools and toys.

L298N Driver

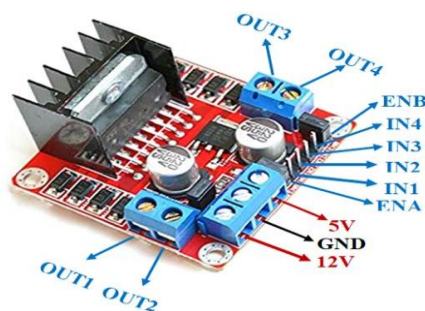


Figure 3.3 L298N Driver.

➤ **L298N Driver**

This L298N Motor Driver Module is a high-power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

Table 3.2: L298N Module Pinout Configuration overview:

Pin Name	Description
IN1 & IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 & IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 & OUT2	Output pins of Motor A
OUT3 & OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

➤ **Applications overview:**

- Drive DC motors.
- Drive stepping motors
- In Robotics

➤ **Bluetooth Module HC-05**

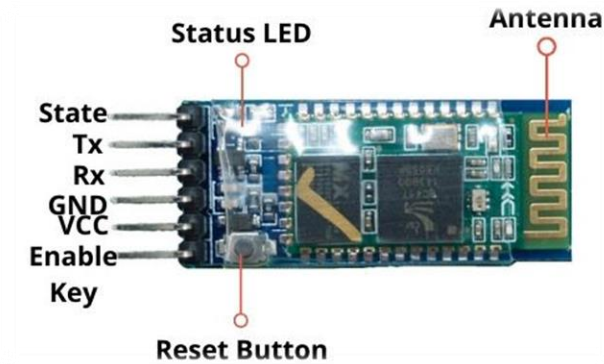


Figure 3.4 Bluetooth Module HC-05

HC-05 is a Bluetooth module which is designed for wireless communication. This module can be used in a master or slave configuration.

HC-05 Bluetooth Module Specifications Overview:

- Voltage: 3.6 V-6 V connected to the VCC pin
- Communication leads are working with voltage 3.3 V (tolerate 5 V)
- Current consumption: approx. 50 mA
- Class 2 – maximum transmitter power + 4 dBm
- Range: up to 10 m
- The password for pairing: 1234
- Standard: Bluetooth 2.0 + EDR
- The SPP profile with possibility to configure via AT commands.
- Communication: UART (RX, TX)



Table 3.3: HC-05 Bluetooth Module Pin Description:

Pin Name	Description
EN	It is used to bring Bluetooth module in AT commands mode.
VCC	Connect 5 V or 3.3 V to this Pin.
GND	Ground Pin of module.
RXD	received data will be transmitted wirelessly by Bluetooth module.
TXD	Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin).
State	It tells whether module is connected or not.

➤ Joystick

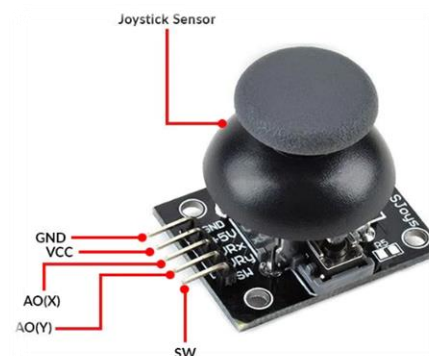


Figure 3.5 Joystick

A joystick is an input device that consists of a stick that pivots on a base and reports its angle or direction to the device it is controlling. Joysticks are commonly used for controlling video games, flight simulation, and various other applications that require precise and variable control.

Joystick Overview:

The joystick is composed of two potentiometers square with each other, and one push button.

- An analog value (from 0 to 1023) corresponding to the horizontal position (called X- coordinate).
- An analog value (from 0 to 1023) corresponding to the vertical position (called Y- coordinate).
- The digital value of a pushbutton (HIGH or LOW).

Table 3.4: Joystick Pinout:

Pin Name	Description
GND	needs to be connected to GND (0V)
VCC	needs to be connected to VCC (5V)
VRX	outputs an analog value corresponding to the horizontal position (called X- coordinate).
VRY	outputs an analog value corresponding to the vertical position (called Y- coordinate).
SW	The SW tip is high when not pressed, and low when pressed

➤ ESP – 32 CAM

The ESP-32 is a versatile microcontroller widely used in IoT (Internet of Things) projects. It features built-in Wi-Fi and Bluetooth capabilities, making it suitable for wireless communication applications. With its powerful processing capabilities and low power consumption, the ESP-32 is popular for projects ranging from home automation to wearable devices.



Figure 3.6 ESP – 32 CAM

ESP – 32 CAM Overview:

- 1-Camera Module: Captures images and video.
- 2-Wi-Fi Module: Provides Wi-Fi connectivity.
- 3-Bluetooth Module: Enables Bluetooth communication.
- 4-Flash Memory: Stores firmware and program data.
- 5-MicroSD Card Slot: Allows for external storage.
- 6-Voltage Regulator: Regulates voltage for stable operation.
- 7-Crystal Oscillator: Generates clock signals for timing.
- 8-LEDs: Indicator lights for power and status.
- 9-Reset Button: Resets the board.
- 10-Boot Button: Used for programming.
- 11-Headers/Pins: GPIO pins for connecting external devices.
- 12-ESP32 Microcontroller: Main microcontroller unit, providing processing power and GPIO interfaces.

➤ RF Sender& Receiver



Figure 3.7 RF Sender& Receiver

RF Sender Components:

- 1-Power Supply: Provides the necessary energy to the transmitter.
- 2-Oscillator: Generates the carrier signal at a specific frequency.
- 3-Modulator: Modulates the carrier signal with the input data (audio, video, digital data, etc.).
- 4-Amplifier: Boosts the modulated signal to a higher power level for transmission.
- 5-Antenna: Converts the electrical signals into electromagnetic waves and radiates them into the air.

RF Receive components:

- Antenna: Captures the electromagnetic waves from the air and converts them back into electrical signals.
- RF Amplifier: Boosts the weak signals received by the antenna.
- Demodulator: Extracts the original data from the modulated carrier signal.
- Filter: Removes any unwanted frequencies and noise from the received signal.
- Decoder: Converts the demodulated signal into a usable format (analog audio, video, or digital data).
- Power Supply: Provides the necessary energy to the receiver.

➤ SSD object detection

SSD is designed for object detection in real-time. Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second. Far below what real-time processing needs. SSD speeds up the process by eliminating the need for the region proposal network. To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN's accuracy using lower resolution images, which further pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN. (Accuracy is measured as the mean average precision mAP: the precision of the predictions.)

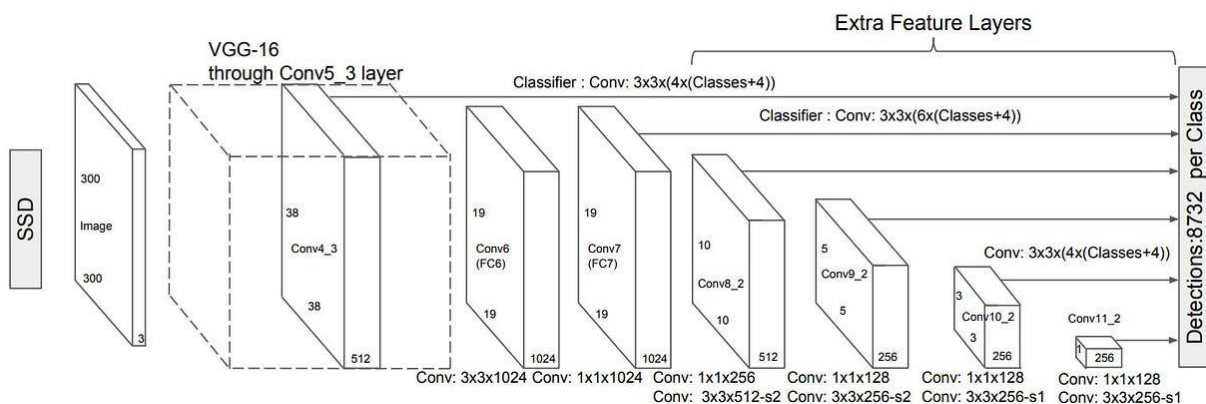


Figure 3.8 SSD

In object detection tasks, the model aims to sketch tight bounding boxes around desired classes in the image, alongside each object labeling. See Figure 3.9

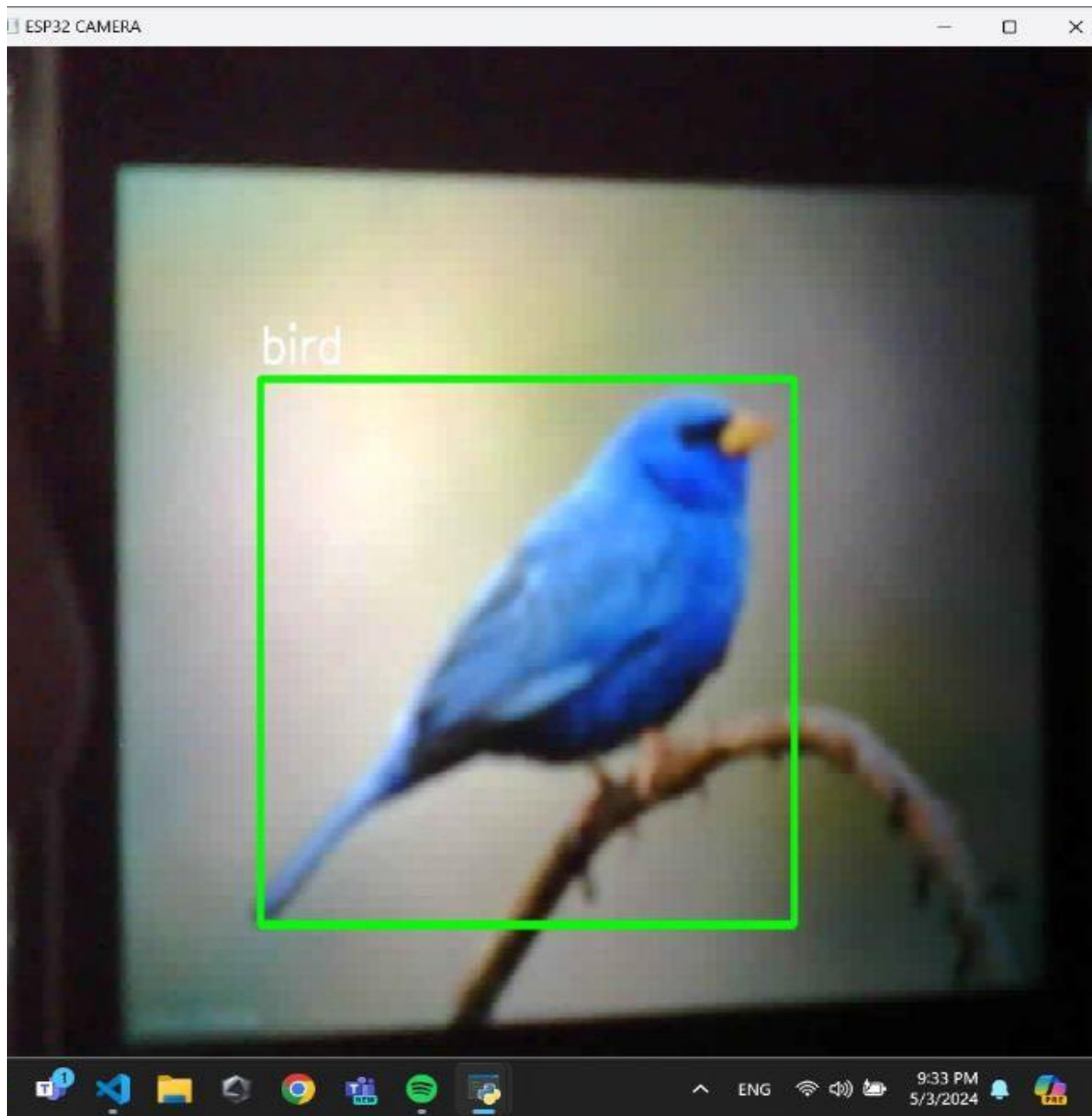


Figure 3.9: Object detection example

▪ **Implementation Plan**

Phase 1: Initial Setup

Install necessary libraries and frameworks (TensorFlow, PyTorch).

Set up data pipeline and initial SSD architecture.

Phase 2: Model Training

Train the model on selected datasets.

Fine-tune hyperparameters and model architecture.

Phase 3: Optimization and Testing

Optimize the trained model for deployment.

Test the model on validation and test sets to ensure performance.

Phase 4: Deployment

Deploy the model on edge devices or cloud platforms.

Monitor performance in real-world scenarios and iterate as needed.

▪ **Applications**

Autonomous Vehicles: Real-time object detection for obstacle avoidance and navigation.

Surveillance Systems: Monitoring and detecting suspicious activities or objects.

Retail: Analyzing customer behavior and managing inventory.

Healthcare: Detecting abnormalities in medical images for diagnostic purposes.

➤ MS COCO

The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.

Splits: The first version of MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 additional test set of 81K images was released, including all the previous test images and 40K new images.

Based on community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. The new split uses the same images and annotations. The 2017 test set is a subset of 41K images of the 2015 test set. Additionally, the 2017 release contains a new unannotated dataset of 123K images.

Annotations: The dataset has annotations for

- object detection: bounding boxes and per-instance segmentation masks with 80 object categories,
- captioning: natural language descriptions of the images (see MS COCO Captions),
- keypoints detection: containing more than 200,000 images and 250,000 person instances labeled with keypoints (17 possible keypoints, such as left eye, nose, right hip, right ankle),
- stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff),
- panoptic: full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road),
- dense pose: more than 39,000 images and 56,000 person instances labeled with DensePose annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D model. The annotations are publicly available only for training and validation images.

3.1.3 Non-Functional Requirements:

➤ Performance:

- Remote control commands should have minimal latency.
- Motor response time should be within an acceptable range.

➤ Reliability:

- The car should operate reliably for an extended period without unexpected failures.
- The remote-control connection should be stable.

➤ Usability:

- The user interface for remote control should be intuitive.
- The car's behavior should be predictable for users.

3.1.4 SOFTWARE USED

Arduino IDE (Integrated Development Environment) is a software platform used to program and develop applications for the Arduino microcontroller boards. Arduino is an opensource hardware and software platform used to build electronics projects, which is designed for makers, hobbyists, and professionals.

The Arduino IDE provides a simple and user-friendly interface to write, compile, and upload code to Arduino boards. It supports the C and C++ programming languages, and also provides a range of libraries and functions that can be used to interface with various sensors, actuators, and other components. With the Arduino IDE, you can easily create custom programs for your Arduino board, such as controlling an LED, reading data from sensors, and communicating with other devices. The IDE provides a built-in serial monitor to debug your code and view the output from your Arduino board.

The Arduino IDE is available for free and can be downloaded from the Arduino website. It is compatible with various operating systems, including Windows, Mac, and Linux.

The Arduino IDE is also open-source, which means that the source code is available for anyone to view and modify. The Arduino IDE (Integrated Development Environment) is a software tool used to write and upload code to a variety of microcontrollers, including the ESP32-CAM. The ESP32-CAM is a development board that combines an ESP32 S chip, a camera module, and other components to create a versatile platform for building Internet of Things (IoT) projects.

To program the ESP32-CAM using the Arduino IDE, you will need to install the ESP32 board package in the IDE and select the appropriate board and port settings. You will also need to install any necessary libraries for your project. Once you have everything set up, you can use the Arduino IDE to write code for the ESP32-CAM and upload it to the board via a USB cable. The IDE provides a range of useful features, including a code editor, a serial monitor, and a debugger, which can help you develop and debug your code

3..4 Methodology:

Given the components you've listed for your unmanned Arduino car project, the functionalities and interactions related to these specific components. Here is user story based on the components you've mentioned:

The ESP32-CAM is a low-cost microcontroller that integrates a camera module and Bluetooth capabilities. In this project, we use the ESP32-CAM to control the movement of the robot and capture video and images that can be viewed remotely. The robot is equipped with a set of motors that enable it to move in different directions. The methodology for developing a surveillance robot using ESP32-CAM can be divided into several steps:

Programming:

The next step is to program the ESP32-CAM module to control the movement of the robot and capture video and images. The programming is done using the Arduino IDE and the ESP32-CAM library. The code includes functions to control the motors, capture video and images, and send them to a remote server using Bluetooth.

Testing:

Once the programming is completed, the robot is tested to ensure that it is functioning as intended. The robot is placed in different environments to test its movement and obstacle detection capabilities. The video and images captured by the robot are also checked for quality and clarity.

Remote access:

The final step is to enable remote access to the robot. This is done by setting up a server that can receive the video and images captured by the robot. The server can be accessed using a smartphone or a computer, allowing users to view the surveillance footage in real-time. In summary, the methodology for developing a surveillance robot using ESP32-CAM involves assembling the hardware components, programming the ESP32-CAM module, testing the robot, and enabling remote access to the surveillance footage.

3.5 Analysis

➤ Use Case Diagram.

It is a set of scenarios that describe the interaction between the user and the system as it is produced.

Also, the relationship between (actors) and (use cases) which are the two main components of a use case.

Here it is used to represent the last user or system that will interact with the resulting system.

➤ Activity Diagram

It is a diagram that shows the progress of activities and work in a gradual manner, with support for selection and repetition.

Synchronization diagrams aim to model both computational and organizational processes.

➤ Sequence Diagram

Explains the order of exchange time in software development activities,

It also explains how a use case works during a certain time.

➤ Flowchart Diagram

is a graphical representation of a process or algorithm. It uses various symbols to denote different types of actions or steps, and arrows to show the flow of control from one step to the next.

➤ Use Case Diagram

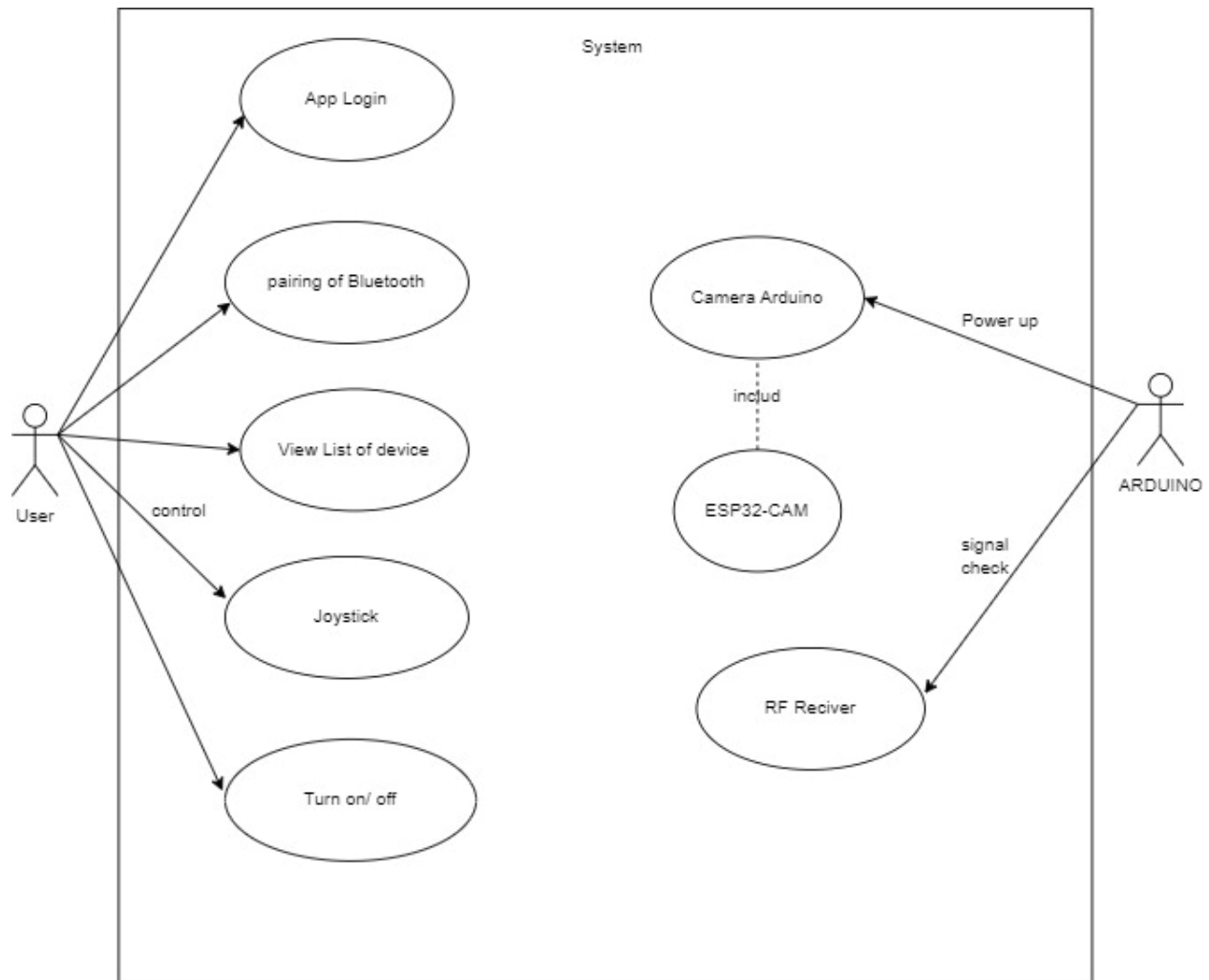


Figure 3.10 use case diagram

➤ Activity Diagram

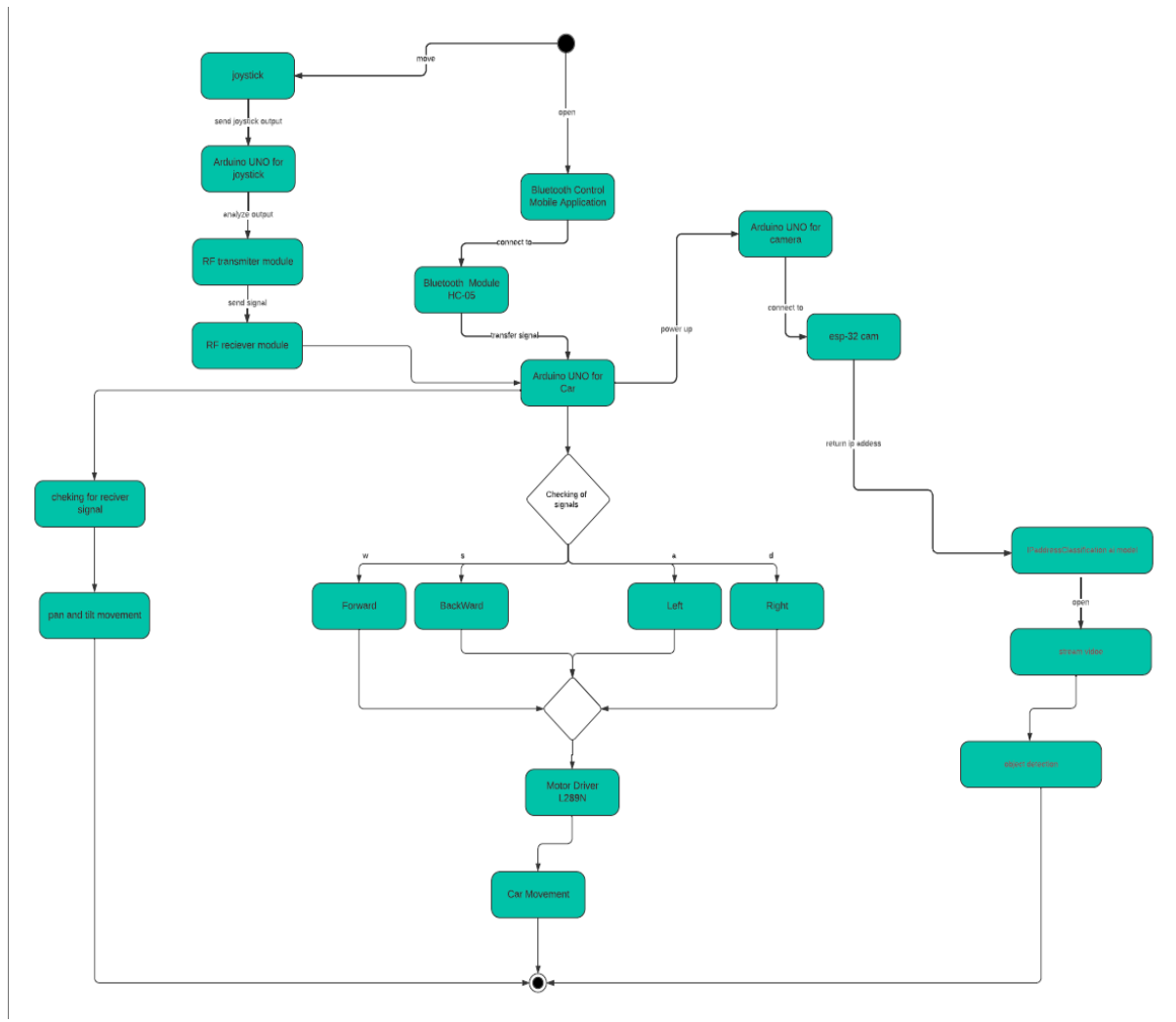


Figure 3.11 Activity Diagram

• Sequence Diagram

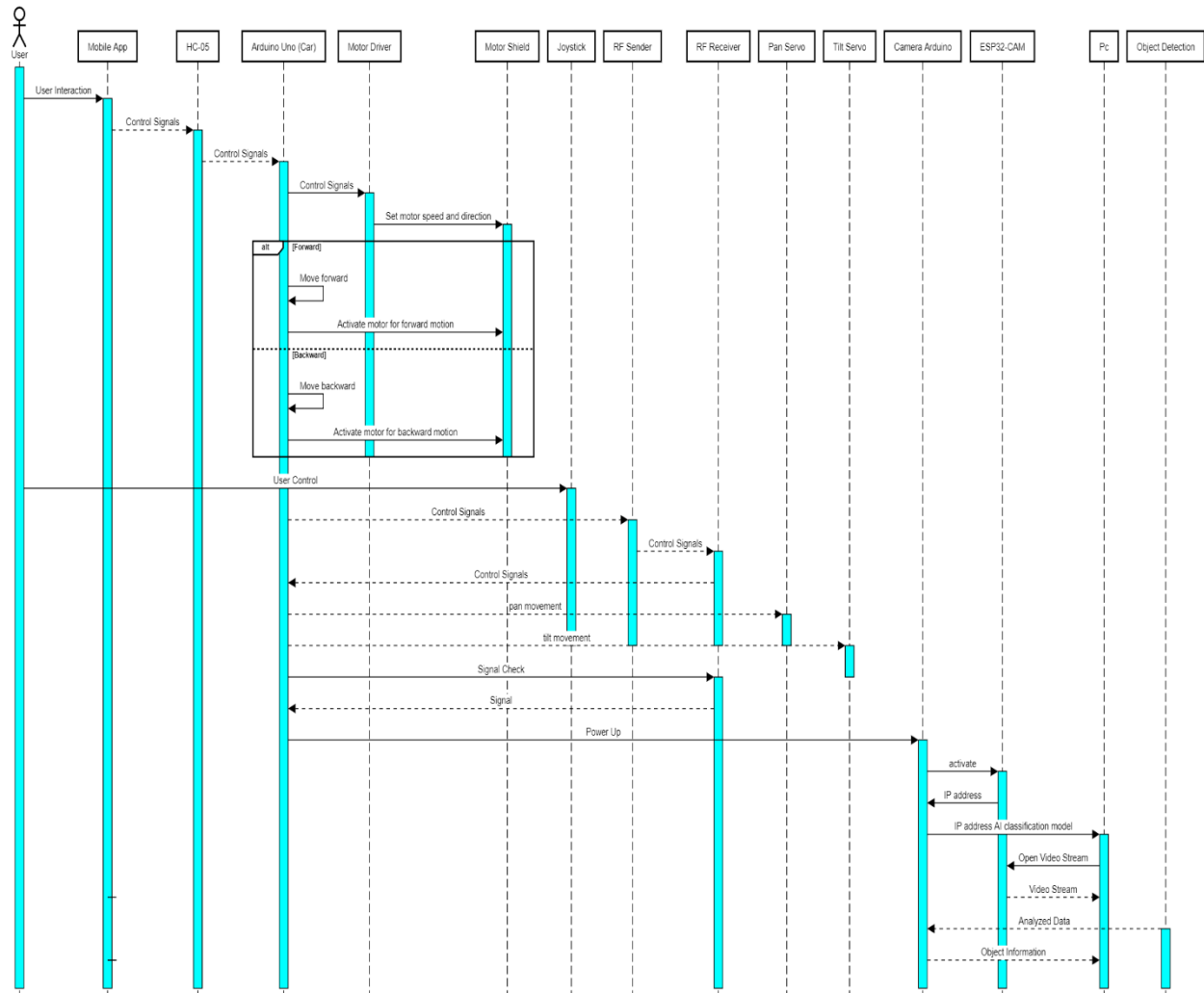


Figure 3.12 sequence diagram

- **Flowchart Diagram**

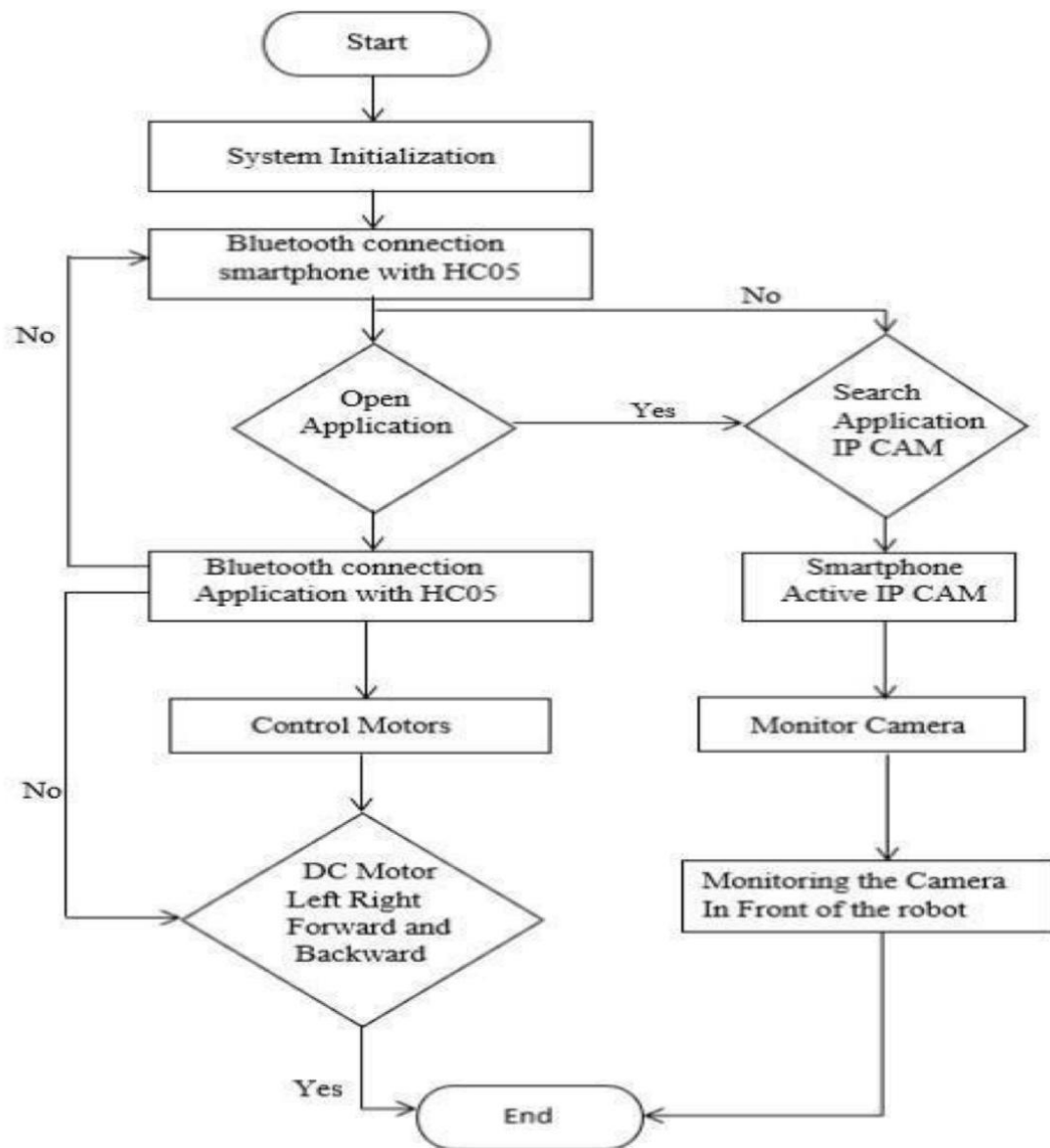


Figure 3.13 Flowchart Diagram

Chapter Four

“Result and Discussion”

Result and Discussion

4- Result and Discussion

Imagine a vehicle equipped with an camera, and communication systems, capable of monitoring its surroundings in real-time. From urban streets to remote locations, the Surveillance Car provides a versatile and adaptable solution for comprehensive surveillance needs. With its high-resolution cameras strategically positioned, the Surveillance Car offers panoramic views and detailed imagery, capturing crucial information to aid in security monitoring and threat detection. Integrated sensors detect anomalies such as suspicious activity, alerting operators to potential risks in the vicinity. Not limited to passive monitoring, the Surveillance Car is equipped with advanced communication technologies, enabling seamless integration with command centers and security networks. Operators can remotely access live feeds, control and respond swiftly to emerging situations, enhancing situational awareness and response capabilities. Privacy and compliance are prioritized through meticulous design, ensuring adherence to regulatory requirements and ethical standards. The Surveillance Car operates with transparency and accountability, leveraging data encryption and access controls to safeguard sensitive information.

- **System Development:** Design and implement a comprehensive object detection system tailored for integration into controlled vehicles, ensuring compatibility with existing vehicle control systems and hardware.
- **Sensor Integration:** Integrate a diverse range of sensors, including cameras, LiDAR, radar, and ultrasonic sensors, to provide comprehensive coverage of the vehicle's surroundings and enable accurate object detection.
- **Algorithm Development:** Research and develop advanced object detection algorithms, utilizing techniques such as deep learning, sensor fusion, and real-time processing optimization to achieve high accuracy and reliability.
- **Training Data Collection:** Gather and annotate a diverse dataset of annotated images or video sequences containing objects relevant to driving scenarios, ensuring the effectiveness and robustness of the object detection models.
- **Model Training and Evaluation:** Train and evaluate the object detection models using the annotated dataset, employing techniques such as transfer learning and data augmentation to optimize performance metrics.
- **Real-time Processing Optimization:** Optimize the object detection algorithms and processing pipelines for real-time performance, minimizing latency and computational overhead to ensure timely detection and response to potential hazards.

- **Integration with Vehicle Control System:** Seamlessly integrate the object detection system with the vehicle's control system, enabling automated responses and decision-making based on detected objects, such as collision avoidance and lane keeping.
- **Safety and Reliability Assurance:** Conduct thorough validation and testing of the object detection system to verify its safety, reliability, and effectiveness in real-world driving conditions.
- **Regulatory Compliance:** Ensure that the object detection system complies with relevant safety standards, regulations, and certification requirements applicable to automotive systems.
- **Documentation and Knowledge Transfer:** Document the design, implementation, and testing processes comprehensively, providing detailed technical documentation, user manuals, and training materials to facilitate knowledge transfer and future maintenance.

➤ 4.1 Results

A. Performance of Wireless Joystick Control

The wireless joystick control demonstrated high responsiveness and reliability in various conditions. Key findings include:

- Latency: Minimal delay between joystick movement and car response, typically under 100 milliseconds.
- Range: Effective control ranges up to 30 meters in an open area without significant interference.
- Battery Life: The joystick operated continuously for approximately 8 hours on a single charge.

B. Performance of Mobile Application Control

The mobile application provided a user-friendly interface with additional functionalities. Key findings include:

- Responsiveness: Similar latency to the wireless joystick, maintaining under 100 milliseconds.
- Range: Dependent on Bluetooth connectivity, with effective control up to 50 meters in an open space.
- User Interface: Intuitive design allowing for easy control and access to advanced features like speed adjustments and sensor data monitoring.

C. Microcontroller and Car Unit Integration

The integration of the microcontroller with the car unit was seamless, ensuring precise execution of commands. Key aspects observed include:

- Command Execution: Accurate and reliable interpretation of both joystick and mobile app inputs.
- Sensor Feedback: Real-time data from sensors (e.g., obstacle detection) processed effectively, enhancing navigation.
- Power Consumption: Efficient power usage, allowing for extended operation periods.

➤ 4.2. Discussion

A. Comparison of Control Methods

While both control methods showed excellent performance, there are notable differences:

- **Ease of Use:** The mobile application was found to be more user-friendly, especially for beginners, due to its graphic interface and additional control options.
- **Portability:** The wireless joystick offered a more traditional control feel, preferred by users accustomed to physical controllers.

B. Range and Interference

The effective range for both control methods was satisfactory, though environmental factors such as walls and other electronic devices caused minor interferences. Bluetooth connectivity in the mobile application provided a slightly extended range compared to RF signals used in the wireless joystick.

C. Reliability and Redundancy

The dual control system ensures reliability and redundancy. If one control method fails (e.g., due to battery depletion or signal interference), the other can seamlessly take over, ensuring continuous operation.

Chapter 5

“Conclusion and Future Work”

Conclusion and Future Work

5.2Future Work

1. Enhanced Object Detection Algorithms

Future iterations of the project could implement more advanced object detection algorithms, such as those based on deep learning, to improve accuracy and reliability. Leveraging pre-trained models on more powerful hardware or cloud-based processing could significantly enhance the system's ability to detect and classify a wider range of objects under diverse conditions.

2. Integration of Additional Sensors

Incorporating additional sensors like ultrasonic sensors, LiDAR, or infrared sensors can provide complementary data to enhance object detection and navigation. This multi-sensor approach would improve obstacle avoidance and environmental awareness, making the system more robust and reliable.

3. Edge Computing with Enhanced Hardware

Upgrading to more powerful edge computing platforms, such as the NVIDIA Jetson series, could allow for real-time processing of more complex object detection algorithms on-board. This would reduce latency and dependence on external processing, leading to faster and more accurate responses.

6. Machine Learning for Adaptive Control

Integrating machine learning techniques for adaptive control systems can help the vehicle learn from its environment and improve its performance over time. This includes learning from past navigation data to optimize movement patterns and improve object detection accuracy.

7. Enhanced Mobile Application Features

Future development of the mobile application could include advanced features such as real-time video streaming, augmented reality (AR) for enhanced user interaction, and detailed telemetry data. This would improve the user experience and provide more comprehensive control over the vehicle.

8. Energy Efficiency Improvements

Optimizing the power consumption of the vehicle and its components is essential for longer operational times. Future work could involve implementing energy-efficient algorithms, low-power hardware, and renewable energy sources such as solar panels to extend the vehicle's operational life.

5.2CONCLUSION

In conclusion, The surveillance robot using the ESP32-CAM is a promising project that can be applied in various scenarios such as monitoring and surveillance of homes or offices. The ESP32-CAM module offers a compact and cost-effective solution for capturing images and streaming video. By integrating it with a robotic platform, it is possible to remotely control the robot and view live video footage from the camera. With the increasing demand for remote monitoring and surveillance, this project has the potential to provide a practical and affordable solution for many different scenarios.

The ESP32-CAM's versatility allows it to be used in numerous environments, from personal home security systems to professional office monitoring setups. Additionally, integrating the ESP32-CAM with a mobile robotic platform enhances its functionality by enabling the robot to navigate and monitor larger areas more effectively. This system is particularly beneficial in situations where constant human presence is impractical, offering a reliable alternative for continuous surveillance. Furthermore, the project's cost-effectiveness makes it accessible to a wider audience, including small businesses and individual users who need efficient security solutions without significant financial investment.

The combination of live video streaming and remote control capabilities ensures that users can keep an eye on their properties in real-time from any location. As the need for remote surveillance grows, this project stands out as a viable and innovative solution, demonstrating significant potential in the field of security and monitoring technology. The adaptability and affordability of this surveillance robot make it a compelling choice for a variety of surveillance needs, paving the way for future advancements and broader adoption

References

1. Beginning Arduino Book by Michael McRoberts.
2. Arduino UNO R3 Tutorials :
(<https://docs.arduino.cc/hardware/uno-rev3/>)
3. L298N Motor Driver – Arduino Interface, How It Works, Codes and Schematics.
(<https://www.arduino.cc/reference/en/libraries/smartcar-shield/>)
4. The Complete Guide to DC Motors :
(<https://byjus.com/physics/dc-motor/>)
5. RF Transmitter and Receiver Module Features :
(<https://robu.in/what-is-rf-transmitter-and-receiver/>)
6. Getting Started With the Wireless module (ESP826) on Arduino IDE:
(<https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01/>)
7. Bluetooth RC Car mobile app tutorial
(<https://bluetooth-rc-car.en.softonic.com/android>)
8. Analog Joystick with Arduino Basics and Mapping:
(http://exploreembedded.com/wiki/Analog_Joystick_with_Arduino)
9. Servo Motor Basics with Arduino:
(<https://docs.arduino.cc/learn/electronics/servo-motors/>)
10. Robotics and Robots :
(<https://builtin.com/robotics>)
- 11.Components
([Arduino Uno Pinout, Specifications, Pin Configuration & Programming \(components101.com\)](#))
- 12.(<https://clear.ml/blog/the-battle-of-speed-accuracy-single-shot-vs-two-shot-detection>)
13. (<https://paperswithcode.com/dataset/coco>)
14. ([1905.02244] Searching for MobileNetV3 (arxiv.org))

“Appendix”


Reciver1

```
1  #include <Servo.h>
2  #include <RCSwitch.h>
3
4  Servo servo,servo2;
5  RCSwitch mySwitch = RCSwitch();
6  const int SERVO_SPEED = 8;
7
8  char t;
9  void setup() {
10     Serial.begin(9600);
11     mySwitch.enableReceive(0);
12     servo.attach(3);
13     servo2.attach(4);
14
15
16     pinMode(5,OUTPUT);
17     pinMode(6,OUTPUT);
18     pinMode(13,OUTPUT); //left motors forward
19     pinMode(12,OUTPUT); //left motors reverse
20     pinMode(11,OUTPUT); //right motors forward
21     pinMode(10,OUTPUT); //right motors reverse
22     Serial.begin(9600);
23
24 }
25
26 void loop() {
27     if(Serial.available()){
28         t = Serial.read();
29         Serial.println(t);
30     }
31
32     if(t == 'F'){ //move forward(all motors rotate in forward direction)
33
34
35         analogWrite(5, 255);
36         analogWrite(6, 255);
37
38         digitalWrite(10,LOW);
39         digitalWrite(11,HIGH);
40         digitalWrite(12,LOW);
41         digitalWrite(13,HIGH);
42
43     }
44
45     else if(t == 'B'){ //move reverse (all motors rotate in reverse direction)
46         analogWrite(5, 255);
47         analogWrite(6, 255);
48
49         digitalWrite(10,HIGH);
50         digitalWrite(11,LOW);
51         digitalWrite(12,HIGH);
52         digitalWrite(13,LOW);
53
54     }
55
56     else if(t == 'L'){ //turn left
57         analogWrite(5, 255);
58         analogWrite(6, 255);
59
60         digitalWrite(10,LOW);
61         digitalWrite(11,HIGH);
62         digitalWrite(12,LOW);
63         digitalWrite(13,LOW);
64     }
65
66     else if(t == 'R'){ //turn right
67         analogWrite(5, 255);
68         analogWrite(6, 255);
69
70         digitalWrite(10,LOW);
71         digitalWrite(11,LOW);
72         digitalWrite(12,LOW);
73         digitalWrite(13,HIGH);
74     }
75
76     else if(t == 'I'){ // forward right
77
78         analogWrite(5, 120);
79         analogWrite(6, 255);
80
81
82         digitalWrite(10,LOW);
83         digitalWrite(11,HIGH);
84         digitalWrite(12,LOW);
85         digitalWrite(13,HIGH);
86     }
```

Reciver2

```
1  else if(t == 'G'){ // forward left
2
3      analogWrite(5, 255);
4      analogWrite(6, 120);
5
6
7  digitalWrite(10,LOW);
8  digitalWrite(11,HIGH);
9  digitalWrite(12,LOW);
10 digitalWrite(13,HIGH);
11 }
12
13 else if(t == 'J'){ // backward right
14
15     analogWrite(5, 120);
16     analogWrite(6, 255);
17
18
19 digitalWrite(10,HIGH);
20 digitalWrite(11,LOW);
21 digitalWrite(12,HIGH);
22 digitalWrite(13,LOW);
23 }
24 else if(t == 'H'){ // backward left
25
26
27     analogWrite(5, 255);
28     analogWrite(6, 120);
29
30
31 digitalWrite(10,HIGH);
32 digitalWrite(11,LOW);
33 digitalWrite(12,HIGH);
34 digitalWrite(13,LOW);
35 }
36
37
38
39
40 else if(t == 'S'){ //STOP (all motors stop)
41 digitalWrite(13,LOW);
42 digitalWrite(12,LOW);
43 digitalWrite(11,LOW);
44 digitalWrite(10,LOW);
45 }
46
47
48
49 if (mySwitch.available()) {
50     int receivedValue = mySwitch.getReceivedValue();
51     Serial.println(receivedValue); // Print the received value for debugging
52     if (receivedValue > 0 && receivedValue < 180) {
53         moveServo(servo, receivedValue);
54         Serial.println("Servo 1 moved");
55     }
56     else if (receivedValue > 200 && receivedValue < 380) {
57         int targetAngle = map(receivedValue, 200, 380, 0, 180);
58         moveServo(servo2, targetAngle);
59         Serial.println("Servo 2 moved");
60     }
61     else {
62         Serial.println("Value not in the range");
63     }
64     mySwitch.resetAvailable();
65 }
66 }
67
68 void moveServo(Servo& servo, int targetAngle) {
69     int currentAngle = servo.read();
70     if (currentAngle != targetAngle) {
71         if (currentAngle < targetAngle) {
72             for (int angle = currentAngle; angle < targetAngle; angle++) {
73                 servo.write(angle);
74                 delay(SERVO_SPEED);
75             }
76         } else {
77             for (int angle = currentAngle; angle > targetAngle; angle--) {
78                 servo.write(angle);
79                 delay(SERVO_SPEED);
80             }
81         }
82     }
83 }
84
85
86 }
```

Sender



```
1  #include <RCSwitch.h>
2  RCSwitch mySwitch = RCSwitch();
3
4  void setup() {
5      Serial.begin(9600);
6      mySwitch.enableTransmit(10);
7  }
8
9  void loop() {
10
11     int x_data = analogRead(A0);
12     int y_data = analogRead(A1);
13     int value_X = x_data;
14     int value_Y = y_data;
15     int value ;
16     if((x_data>400 && y_data>600 )||(x_data>400 && y_data<400)){
17         value = map(value_Y, 0, 1024, 0, 180);
18     }
19     else if((x_data<400 && y_data>400 )||(x_data>600 && y_data>400)){
20         value = map(value_X , 0, 1024, 200, 380);
21     }
22
23
24     mySwitch.send(value, 30);
25
26
27     Serial.print("x_data:");
28     Serial.print(x_data);
29     Serial.print("\t");
30     Serial.print("y_data:");
31     Serial.print(y_data);
32     Serial.println("\t");
33     Serial.println(value);
34
35
36
37 }
```

Ai Model

```
1 import cv2
2 import urllib.request
3 import numpy as np
4
5 url = 'http://172.20.10.4/cam-hi.jpg'
6 winNameCamera = 'ESP32 CAMERA'
7 winNameCounter = 'Object Counter'
8
9 cv2.namedWindow(winNameCamera, cv2.WINDOW_NORMAL)
10 cv2.namedWindow(winNameCounter, cv2.WINDOW_NORMAL)
11
12 classNames = []
13 classFile = 'coco.names'
14 with open(classFile, 'rt') as f:
15     classNames = f.read().rstrip('\n').split('\n')
16
17 configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
18 weightsPath = 'frozen_inference_graph.pb'
19
20 net = cv2.dnn_DetectionModel(weightsPath, configPath)
21 net.setInputSize(320, 320)
22 net.setInputScale(1.0 / 127.5)
23 net.setInputMean((127.5, 127.5, 127.5))
24 net.setInputSwapRB(True)
25
26 # List of objects to count
27 objects_to_count = [
28     'person', 'laptop', 'keyboard', 'mouse', 'tv', 'desk', 'chair',
29     'book', 'cell phone', 'potted plant', 'cup', 'bottle', 'backpack',
30     'window', 'door'
31 ]
32
33 # Initialize the object count dictionary
34 objectCounts = {obj: 0 for obj in objects_to_count}
35
36 while True:
37     imgResponse = urllib.request.urlopen(url)
38     imgNp = np.array(bytearray(imgResponse.read()), dtype=np.uint8)
39     img = cv2.imdecode(imgNp, -1)
40
41     classIds, confs, bbox = net.detect(img, confThreshold=0.5)
42
43     # Reset object counts for the new frame
44     objectCounts = {obj: 0 for obj in objects_to_count}
45
46     # Draw bounding boxes around detected objects and count them
47     if len(classIds) != 0:
48         for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
49             className = classNames[classId - 1]
50             if className in objects_to_count:
51                 cv2.rectangle(img, box, color=(0, 255, 0), thickness=3)
52                 cv2.putText(img, className, (box[0], box[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
53                 objectCounts[className] += 1
54
55     # Display the camera feed
56     cv2.imshow(winNameCamera, img)
57
58     # Determine window size based on the number of objects
59     num_objects = len(objects_to_count)
60     window_height = 40 * num_objects # 40 pixels per object line
61     window_width = 300 # Fixed width for the counter window
62
63     # Create a suitable counter image
64     counterImg = np.zeros((window_height, window_width, 3), dtype=np.uint8)
65
66     # Display the object counts
67     for i, (obj, count) in enumerate(objectCounts.items()):
68         cv2.putText(counterImg, f'{obj.capitalize()}: {count}', (10, 30 * (i + 1)), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)
69
70     # Display the counter image
71     cv2.imshow(winNameCounter, counterImg)
72
73     key = cv2.waitKey(5) & 0xFF
74     if key == 27:
75         break
76
77 cv2.destroyAllWindows()
78
```


WIFI ESP-32 CAM

```
1 #include <WebServer.h>
2 #include <WiFi.h>
3 #include <esp32cam.h>
4
5 const char* WIFI_SSID = "Moataz's iPhone";
6 const char* WIFI_PASS = "123456789";
7
8 WebServer server(80); //servidor en el puerto 80
9
10 static auto loRes = esp32cam::Resolution::find(320, 240); //baja resolucio
11 static auto hiRes = esp32cam::Resolution::find(800, 600); //alta resolucio
12 //static auto hiRes = esp32cam::Resolution::find(640, 480); //alta resolucio (para tazas de fps) (IP CAM APP)
13
14 void
15 serveJpg() //captura imagen .jpg
16 {
17     auto frame = esp32cam::capture();
18     if (frame == nullptr) {
19         Serial.println("CAPTURE FAIL");
20         server.send(503, "", "");
21         return;
22     }
23     Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
24                   static_cast<int>(frame->size()));
25
26     server.setContentLength(frame->size());
27     server.send(200, "image/jpeg");
28     WiFiClient client = server.client();
29     frame->writeTo(client); // y envia a un cliente (en este caso sera python)
30 }
31
32 void
33 handleJpgLo() //permite enviar la resolucio
34 {
35     if (!esp32cam::Camera.changeResolution(loRes)) {
36         Serial.println("SET-LO-RES FAIL");
37     }
38     serveJpg();
39 }
40
41 void
42 handleJpgHi() //permite enviar la resolucio
43 {
44     if (!esp32cam::Camera.changeResolution(hiRes)) {
45         Serial.println("SET-HI-RES FAIL");
46     }
47     serveJpg();
48 }
49
50
51
52
53 void
54 setup()
55 {
56     Serial.begin(115200);
57     Serial.println();
58
59     {
60         using namespace esp32cam;
61         Config cfg;
62         cfg.setPins(pins::AiThinker);
63         cfg.setResolution(hiRes);
64         cfg.setBufferCount(2);
65         cfg.setJpeg(80);
66
67         bool ok = Camera.begin(cfg);
68         Serial.println(ok ? "CAMARA OK" : "CAMARA FAIL");
69     }
70
71     WiFi.persistent(false);
72     WiFi.mode(WIFI_STA);
73     WiFi.begin(WIFI_SSID, WIFI_PASS); //nos conectamos a la red wifi
74     while (WiFi.status() != WL_CONNECTED) {
75         delay(500);
76     }
77
78     Serial.print("http://");
79     Serial.print(WiFi.localIP());
80     Serial.println("/cam-lo.jpg"); //para conectarnos IP res baja
81
82     Serial.print("http://");
83     Serial.print(WiFi.localIP());
84     Serial.println("/cam-hi.jpg"); //para conectarnos IP res alta
85
86     server.on("/cam-lo.jpg", handleJpgLo); //enviamos al servidor
87     server.on("/cam-hi.jpg", handleJpgHi);
88
89     server.begin();
90 }
91
92 void loop()
93 {
94     server.handleClient();
95 }
```