ST. XAVIER'S COLLEGE OF MANAGEMENT & TECHNOLOGY

NAAC Accredited with B++ Grade (1st Cycle)

Digha Ghat, Patna-800011, Bihar

PROJECT REPORT

On

Surveillance Car (Chitti 3.0)

Submitted to ARYABHATTA KNOWLEDGE UNIVERSITY

For the partial fulfillment for the award of the degree of
BACHELOR OF COMPUTER APPLICATION
SESSION-2021-2024

Under the Guidance – **Fr. Dr. Sebastian Alphonse S.J.**

COE & Guide (Department of Computer Science)

BY:

Zafer Eqbal [REG NO. 21303302012]

Table of Content

ACKNOWLEDGEMENT

We thank almighty God who has provided us the faculty of learning and understanding the phenomenon occurring around me. We wish to express our sincere thanks to our parents for their tremendous contribution and support both morally and financially towards the completion of this project. It gives us immense pleasure to express our sincere gratitude to our faculty guide **Fr. Dr. Sebastian Alphonse S.J.**, for helping us in shaping this project. His support and encouragement helped us to strive towards the best of our potentials.

We also take this opportunity to express our gratefulness to all the faculty members of the department of computer science at St. Xavier's College of Management & Technology, Patna for their help and support.

We gained immensely from doing this project. It has contributed greatly towards increasing our learning curve, both at the theoretical and practical level. we would also like to extend our gratitude to the principal **Fr. Dr. Martin Poras S.J.**, of St. Xavier's College of Management & Technology, Patna for providing us an opportunity to study in this premier institute of learning.

By:
Ayesha Firdous
Zafer Eqbal

CERTIFICATE

This is certified that the project entitled **Surveillance Car (Chitti 3.0)** submitted to Aryabhatta Knowledge in fulfillment of the requirements for the award of Degree of Bachelors in Computer Application **(BCA)**, the matter embodied in this project is a genuine work done by us.

……	……..……....
Signature of the Principal	Signature of the Project Guide
Fr. Dr. Martin Poras SJ	Fr.Dr. Sebastian Alphonse SJ
St. Xavier's College of	COE & Guide
Management & Technology	Dept. of Computer Science

……..…….

Signature of External Examiner

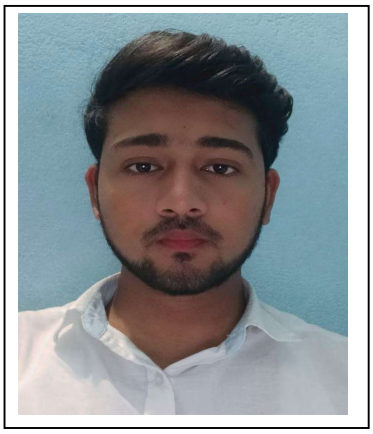
Aryabhatta Knowledge University

Patna, Bihar

DECLARATION

I hereby, declare that the work being presented in the project, titled **"Surveillance Car"** as a part of course curriculum of Bachelor of Computer Application (BCA), is an authentic record of our own work carried out under the guidance of Fr. Dr. Sebastian Alphonse SJ , COE , St. Xavier's College of Management & Technology, Patna.

STUDENT PROFILE



NAME	**Zafer Eqbal**
ROLL NUMBER	BCA2021045
UNIVERSITY REGISTRATION NO.	21303302012
EMAIL ID	zafereqbal753@gmail.com
CONTACT	+91-7870013573
COLLEGE	SXCMT, Patna

ABOUT COLLEGE

St. Xavier ‘s College of Management & Technology under Aryabhatta Knowledge University was established in 2012. It offers life-oriented and professional courses. St. Xavier ‘s College is a co-educational undergraduate college of Arts and Commerce under Magadh University, Bodh Gaya, started in July 2009 at its temporary College Campus at Digha. In June 2011, the college was shifted to its present location on a sprawling 36-acre campus at S.X.C.M.T on Digha Aashiyana Road.

Both colleges are managed by the Jesuits of Bihar. Jesuits are the members of a Christian religious Order called the Society of Jesus. Jesuit education is inspired by the life and teachings of Jesus Christ and on the principles of pedagogy elaborated by St. Ignatius of Loyola, the founder of the Society of Jesus.

The College aims to offer an all-round formation that is intellectual, cultural, social, emotional, physical, aesthetic, moral and spiritual It further aims at promoting values such as respect for common Indian cultural heritage, egalitarianism, democracy, secularism, equality of sexes, protection of environment, removal of social barriers, responsible use of cybernetics and mass media, transparency and

probity in private and public life, national unity and respect for religious and moral values.

The College Emblem contains the motto of the College: Pravahito Gyanganga Pravah: - Let the streams of Gyan Ganga keep on flowing. The College is situated near the river Ganga. Like the flowing river, the College is to ensure that the streams of Gyan keep on flowing and liberating people. The Sun with the letters IHS (first three letters for JESUS in Greek) is a symbol characterizing the Society of Jesus. The emblem within the emblem, containing crown, crescent moon, etc. is the coat of arms of the noble family of our patron, St. Francis Xavier.

Email : info@sxcpatna.edu.in
Website : sxcpatna.edu.in
Contact No : +91-8987262019 | +91-8877617734

ABOUT UNIVERSITY

The Aryabhatta Knowledge University Act 2008 provides for the establishment of university at Patna to conduct and facilitate affiliation of institutions in the conventional as well as frontiers of professional education. All colleges and instructions imparting professional education for example Engineering and Technology including Information Technology, Nano technology &Biotechnology, Management, Medicines, Health Technology, Public Health Pharmacy, Optometry, Nursing, Education, Law etc. is to be affiliated to this university.

Aryabhatta Knowledge University (AKU), Patna has been established by Government of Bihar for the Development and Management of Educational Infrastructure related to Technical, Medical, Management and allied professional education in the state. The objective of the university is to promote the professional education infrastructure of meet the national standard through well advanced Couse, infrastructure and quality faculty. Bihar lacks in the infrastructure both on the term of number of intuitions and the quality of education. As a result Bihar State has become the major hub of student migrating to the

other states for pursuing their education and carrier building for future growth.

The vision of this University is to mound the character, shape the career, and bring perfection in behavior and excellence in educating the young generation of today for future. Also, to bring up a vibrant knowledge university resonating with the mission of all round development of students in particular and the national and mankind in general by providing value-based creative, innovative quality education.

1. INTRODUCTION

1.1 About

This project introduces an ESP32-controlled surveillance car designed for remote monitoring applications. The surveillance car incorporates the ESP32 microcontroller, which serves as the central processing unit for controlling the car's movements and transmitting live video feed. Equipped with a camera module, the car captures real-time video footage of its surroundings, which is streamed wirelessly to a remote device for monitoring purposes.

The ESP32 microcontroller provides the necessary computational power and connectivity to facilitate seamless communication between the surveillance car and the remote monitoring station. Through the integration of Wi-Fi technology, the ESP32 establishes a reliable connection, enabling users to remotely control the car's movements and access live video feed from any location with Wi-Fi access.

Key features of the ESP32-controlled surveillance car include its compact design, maneuverability, and real-time video streaming capabilities. By leveraging the versatility and

performance of the ESP32 microcontroller, the surveillance car offers a cost-effective solution for various surveillance and monitoring applications, including home security, environmental monitoring, and remote inspection tasks.

In summary, the ESP32-controlled surveillance car represents a novel integration of hardware and software technologies, providing users with a powerful and flexible platform for remote surveillance and monitoring purposes.

1.2 Purpose

The purpose is to design a user friendly surveillance car that can serve various purposes depending on the design and the specific needs of its users. Following are some common purposes which we keep in mind while designing our project.

- Security Monitoring
- Remote Inspection
- Environmental Monitoring
- Surveillance in Hazardous Environments
- Educational Purposes

1.3 Scope

The scope of surveillance car spans security, industrial inspection , emergency response, environmental monitoring , and educational purposes. It serves as a versatile tool for patrolling, inspecting, and gathering data in various environments, including residential, commercial, industrial, and natural settings. Its flexibility and adaptability make it a valuable asset across various industries.

2. SYSTEM ANALYSIS

2.1 Existing System

In the existing system we have seen wall mounted CCTV (Closed-Circuit Television) cameras which consume more power, are costly to set up and require a standalone DVR to store the CCTV footage which ultimately ends up in an increased cost.

Another main limitation of installing a CCTV camera is that, once fitted in a direction they can only monitor a specific area.

These CCTV cameras need wired connection which increases the cost of setting up the surveillance cameras.
While recording a CCTV footage a Single CCTV camera can create video data which can result in GB's.

The output from the CCTV footage was visible only from a single monitor.
If there is a electricity – cutoff the CCTV cameras won't work and hence won't give 100% video footage guarantee.

2.2 Proposed System

Our proposed system consists of a ESP32 cam module mounted over a pan-tilt servo assembly which will provide us a live video feed on a web browser.

This setup would be fitted on the top of a car chassis which would contain four dc motors and enable us to move the surveillance car in all the directions.

To control the motors we would be using a L298N motor driver module which would provide the data output and external power to the motors. In order to protect our surveillance car from voltage fluctuation we would use a buck converter, to separate the power source of cam module and pan-tilt servo assembly.

The cam module will capture live data with regards to its surroundings and then send it to a desired device through wifi. The user will be observing this data on the device at the user end. According to the desired movement, the user will control the robotic vehicle through the webpage available at the user end.

2.3 Requirement Analysis

Requirement analysis is the process of identifying, documenting, and validating the needs and constraints of a system or project. It involves understanding the objectives, functionalities, and constraints to ensure that the final solution meets stakeholders' expectations. Key activities include gathering and prioritizing user requirements, defining system boundaries, and identifying dependencies. Requirement analysis helps in mitigating risks, managing project scope, and aligning the development efforts with stakeholders' needs. It serves as the foundation for system design, development, and testing phases. Effective requirement analysis facilitates communication between stakeholders and development teams, leading to the successful delivery of the desired solution.

Here's a breakdown of the requirement analysis process:

1. **Functional Requirements**
    - Remote Control
    - Real-time video streaming
    - Motor Control
    - Battery Management
    - Security Features

2. Non-functional Requirements

- Performance
- Reliability
- Scalability
- Portability
- User Interface Design

3. Constraints

- Cost
- Size and Weight
- Processing Power
- Wireless Connectivity
- Compatibility

4. Use Cases

- Home Security
- Remote Monitoring
- Educational Purposes
- Entertainment

5. Risk Assessment

- Hardware Failure
- Security Breach
- Environmental Factors
- Battery Life

2.4 Feasibility Study

A feasibility study is a structured evaluation of a proposed project's potential success. It assesses technical, economic, legal, operational, and scheduling factors to determine if the project is viable. It typically involves evaluating various factors such as technical, economic, legal, operational, and scheduling considerations to determine whether the project is feasible or not.

In short, a feasibility study provides a concise evaluation of the practicality and likelihood of achieving the desired outcomes of a proposed project.

Assessing the feasibility of a new system entails ensuring its efficiency and affordability. Different types of feasibility must be evaluated, which include:

- Economical Feasibility
- Operational Feasibility
- Technical Feasbility

2.4.1 Economical Feasibility :

Economic feasibility evaluates if a proposed project or venture is financially viable. It involves analyzing projected costs and potential revenues to determine if the investment will generate sufficient returns. Factors such as market demand, pricing strategy, competition, and financial resources are considered to assess the project's profitability. If the projected benefits outweigh the costs and risks, the project is deemed economically feasible. This analysis helps stakeholders make informed decisions about whether to proceed with the project or pursue alternative options.

2.4.2 Operational Feasibility :

Operational feasibility determines if a proposed project can be implemented smoothly within existing organizational resources and processes. It assesses factors like staffing, training, and workflow adjustments to ensure the project aligns with operational capabilities. If the project doesn't disrupt core operations and can be integrated seamlessly, it's considered operationally feasible.

In this project there is no requirement of special trainings for the users to for operating the system. Any person can easily use the surveillance car.

2.4.3 Technical Feasibility :

The technical requirement for the system is economic and it does not use any other additional Hardware and software. It evaluates factors such as compatibility with existing systems, software requirements, and hardware capabilities. If the system can be developed and deployed using available technology resources without significant technical hurdles, it's considered technically feasible. This analysis ensures that the system can be effectively implemented and operated .

3. SYSTEM DESIGN

3.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

As use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by the other types of diagrams as well.
While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must do.

Due to their simplest nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed.

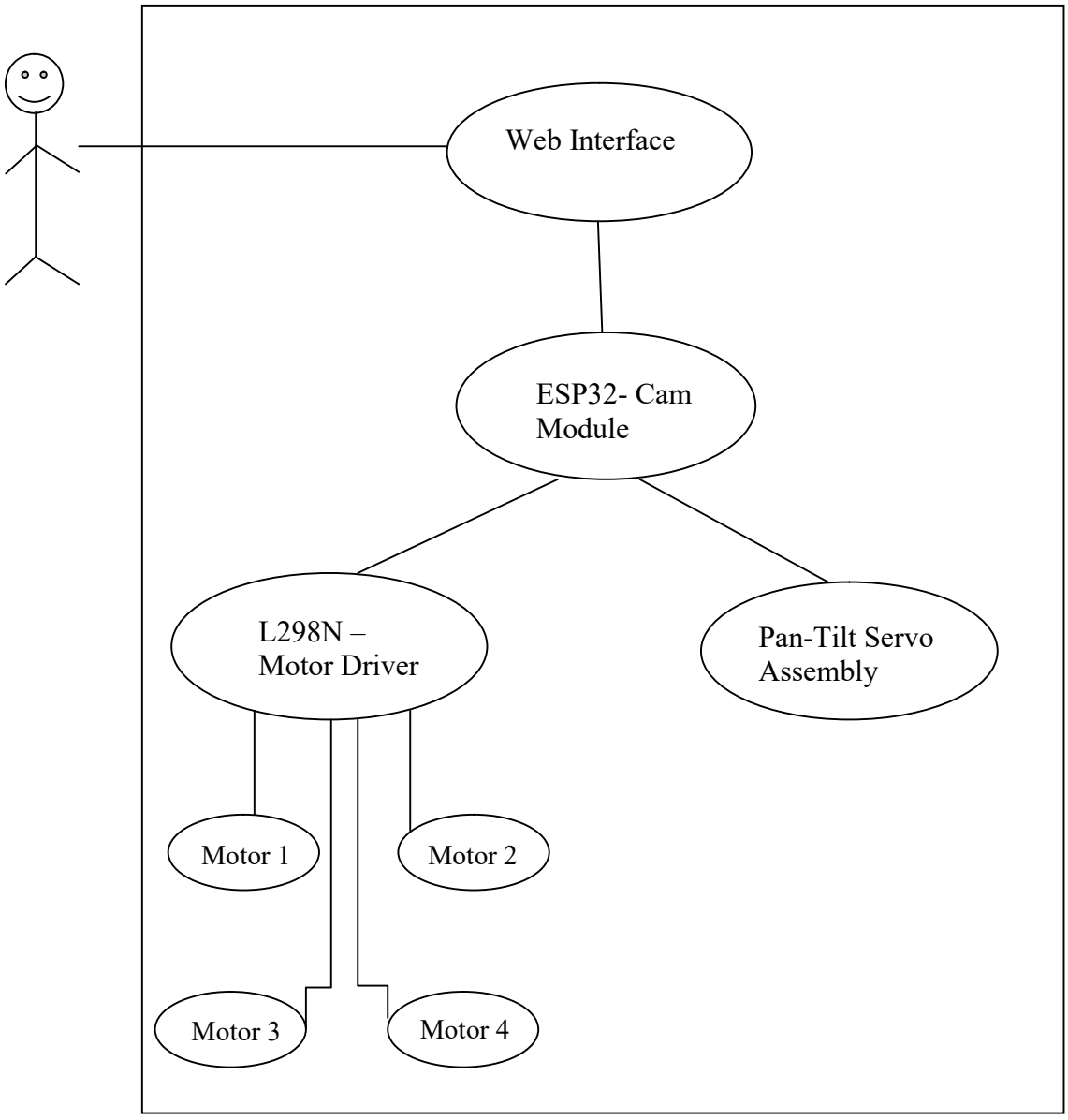


Image 1: Use Case Diagram of Surveillance Car

4. PROJECT DESCRIPTION

4.1 Perspective

The ESP32-controlled surveillance car project blends technical innovation with practical applications and societal considerations. From a technical perspective, it involves integrating hardware and software systems to enable functionalities such as remote control, live video streaming, and obstacle avoidance. Practically, the surveillance car offers solutions for remote monitoring and exploration in various scenarios, including home security, industrial inspections, and search and rescue missions. However, its deployment raises societal concerns regarding privacy, security, and ethical usage. Clear guidelines and regulations are needed to govern its deployment responsibly. The project showcases the innovative potential of IOT and robotics technologies to enhance security, safety, and efficiency in diverse environments.

4.2 Technologies Used for Development

The ESP32 Surveillance Car project involves a combination of hardware and software technologies to achieve its functionalities. Here are the main technologies used for developing the project:

4.2.1 HARDWARE TECHNOLOGIES:

All the hardware components that we have used in developing our project is cost effective. The major components which we have used in developing our surveillance car are as follows:

1. Arduino Uno R3 ATmega328P
2. ESP32 Camera Development Board Wifi+Bluetooth Module with OV2640 Camera Module
3. L298N 2A Based Motor Driver Module
4. 4WD Double Layer Car Chassis
5. Servo Bracket & SG90 Servo Motors
6. Buck Converter
7. Rechargeable Battery and few other minor components.

4.2.2 HARDWARE SPECIFICATION:

Arduino Uno R3 ATmega328P

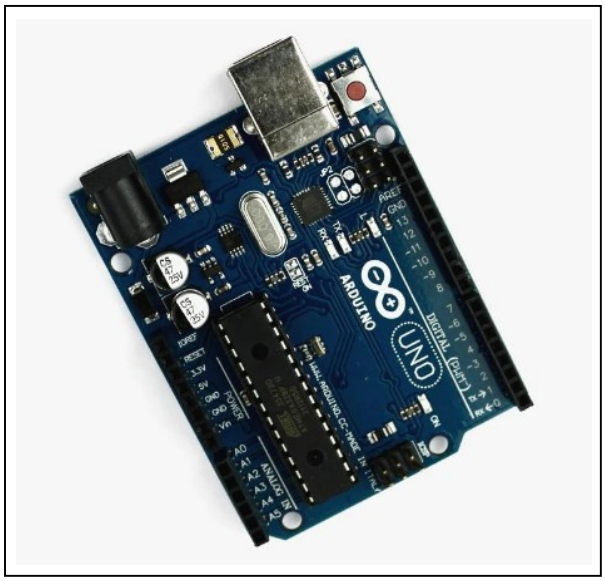


Image 1: An Arduino Uno Board

The Arduino Uno R3 is powered by the Atmel ATmega328P microcontroller. This microcontroller is an 8-bit AVR (Advanced Virtual RISC) processor with a clock speed of 16 MHz. The board has a total of 14 digital input/output pins, where 6 can be used as PWM (Pulse Width Modulation) outputs. Additionally, there are 6 analog input pins. A reset button is available for restarting the microcontroller. Pressing this button will restart the sketch (program) running on the Arduino. The Arduino Uno R3 has a USB interface (USB-B type) that allows it to connect to a computer for programming and serial communication. The USB connection also provides power to the board. The Arduino Uno R3 is part of the open-source Arduino platform, which means that the board's design files and software are freely available for users to modify.

ESP32 Camera Module

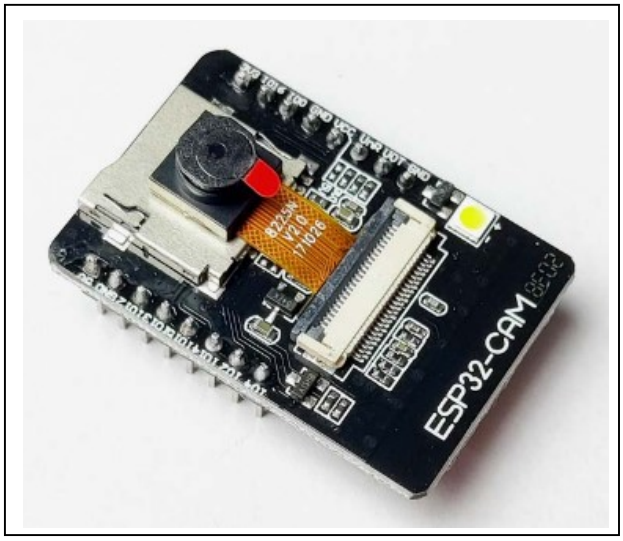


Image 2: ESP32 camera module

The ESP32 camera module is a versatile and compact hardware component that integrates a camera and wireless communication capabilities into a single device. The ESP32 itself is a powerful microcontroller and Wi-Fi/Bluetooth module developed by Espressif Systems. When coupled with a camera, it becomes an ideal solution for projects that require image and video capture, as well as wireless connectivity. The ESP32 camera module typically supports various communication interfaces, including Serial Peripheral Interface (SPI) and Inter-IC Communication (I2C), allowing it to communicate with with other devices and sensors. ESP32 camera modules include an SD card slot, allowing for local storage of captured images or video footage. One of the main advantages of the ESP32 is its built-in Wi-Fi and Bluetooth capabilities. This enables the camera module to connect to local networks or communicate with other devices.

L298N 2A Motor Driver Module

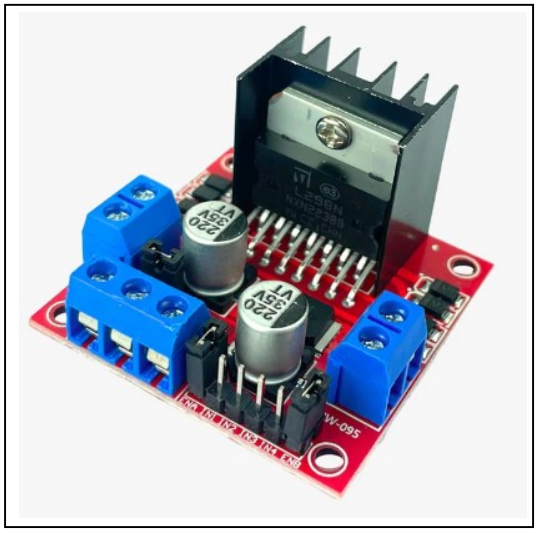
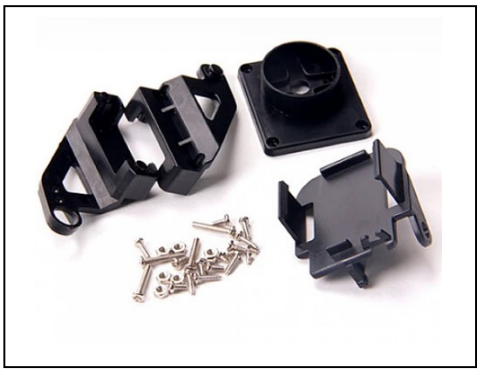


Image 3: Motor Driver Module

The L298N is a popular motor driver IC (Integrated Circuit) that is commonly used to control DC motors and stepper motors. The L298N 2A-based motor driver module is a ready-made module that incorporates the L298N IC along with additional components to make it easier to interface with microcontrollers and control motors in various applications.

The heart of the module is the L298N dual H-bridge motor driver IC. This IC is capable of controlling two DC motors or one stepper motor. It provides bidirectional control, meaning it can drive the motors forward or backward. Each motor channel has an enable pin that can be controlled to turn the motor on or off. This feature allows for PWM (Pulse Width Modulation) control to adjust the motor speed. The module includes built-in diodes to protect the circuit from voltage spikes generated when the motor is turned off. These diodes help prevent damage to the circuitry.

Servo Bracket & SG90 Servo Motor



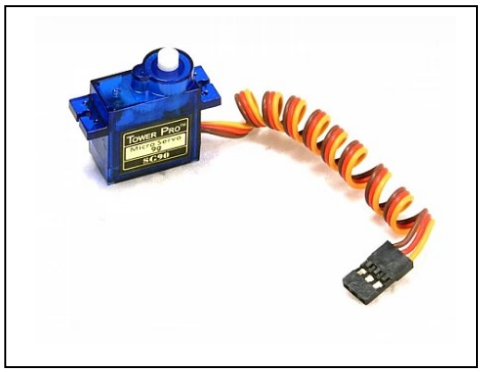


Image 4: Servo Bracket & SG90 Servo Motor

Servo Bracket :

- A servo bracket serves the purpose of providing a stable and secure mounting platform for servo motors.
- It allows for the proper positioning and orientation of the servo motor in a variety of projects.
- Servo brackets come in various shapes and sizes to suit different servo motor models and applications.
- They may have additional features like slots for sensors, cameras, or other accessories.

SG90 Servo Motor:

- Widely used in robotics, remote-controlled vehicles, model airplanes, and other projects that require controlled and precise movement.
- The SG90 has a rotation range of approximately 180 degrees, making it suitable for applications requiring limited rotation.
- The SG90 servo motor is compact and lightweight, making it easy to integrate into various projects without adding significant weight.
- SG90 servo motors are cost-effective, making them popular choices for hobbyists and students working on small-scale projects.

4WD Double Layer Smart Car Robot Chassis

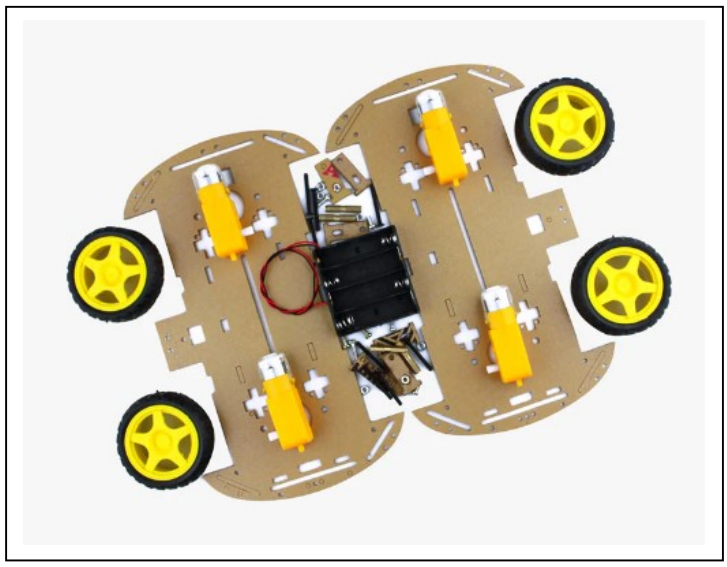


Image 5: Car Chassis

The 4WD Double Layer Smart Car Robot Chassis is a versatile platform designed for building customizable robotic projects. With four-wheel drive capabilities, it offers enhanced stability and maneuverability. The double-layer design provides ample space for mounting components such as microcontrollers, sensors, and batteries. Its robust construction ensures durability and reliability in various environments. The chassis features multiple mounting holes for easy attachment of additional modules and accessories. It supports a wide range of sensors and actuators, making it suitable for diverse applications such as obstacle avoidance, line following, and remote control. The car robot chassis is compatible with popular microcontroller platforms like Arduino and Raspberry Pi, allowing for easy integration with existing projects.Its sleek and modern design adds aesthetic appeal to robotic projects.

LM2596 DC-DC Buck Converter

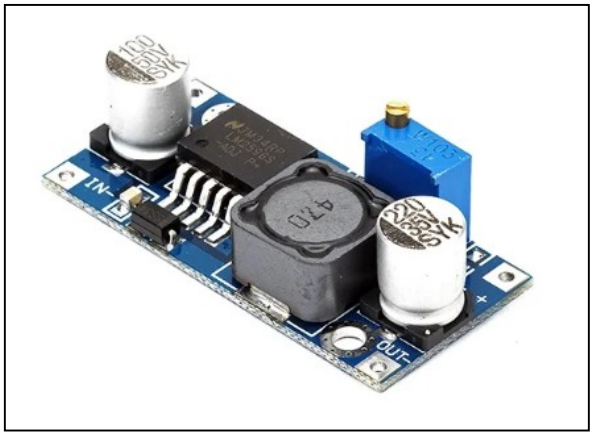


Image 6: Buck Converter

The LM2596 DC-DC buck converter is a versatile voltage regulator module commonly used in electronic projects. It efficiently steps down input voltage to a lower output voltage, providing stable power for various applications. With its adjustable output voltage and high efficiency, it is ideal for powering microcontrollers, sensors, and other electronic components. The module features a potentiometer for precise voltage adjustment, offering flexibility in voltage regulation. Its compact size and simple design make it easy to integrate into circuits with limited space. The LM2596 converter operates over a wide input voltage range, making it suitable for various power sources. It includes built-in overcurrent and thermal protection features for added safety and reliability.

The main purpose to use this in our project is to avoid the voltage fluctuation.

Li-ion Rechargeable Battery

Image 7: Li-ion 18650 Battery

The Li-ion 18650 rechargeable battery is a popular and versatile power source commonly used in portable electronic devices. It features a cylindrical shape with dimensions of 18mm in diameter and 65mm in length. With a nominal voltage of 3.7 volts, it provides reliable power for various applications. The 18650 battery has a high energy density, offering long-lasting performance in a compact size. It can be recharged hundreds of times, making it cost-effective and environmentally friendly. Proper handling and storage are essential to prevent damage and ensure optimal performance. The 18650 battery is known for its stability, reliability, and high discharge rate, making it suitable for demanding applications.

4.2.3 SOFTWARE TECHNOLOGIES:

All the software technologies that we have used in developing our project are listed below along with there specification:

1. **Arduino IDE (Integrated Development Environment)**: Used for programming the ESP32 microcontroller with the Arduino framework, providing an easy-to-use platform for developing embedded applications.

2. **ESP32 Libraries:** Utilized to interface with hardware peripherals (such as camera modules, sensors, and motors) and implement communication protocols (e.g., Wi-Fi) on the ESP32 platform.

3. **HTML & CSS**: Html & Css is used for the development of the webpage which will be used at the user end. This webpage is embedded and uploaded in the esp32 cam module.

4. **Video and Audio Streaming Protocols**: Implement protocols for streaming video and audio data captured by the camera module and microphone, facilitating real-time

monitoring and communication with the surveillance car.

5. **User Interface Development**: Develop a user interface, either in the form of a mobile application or web interface, to enable remote control of the surveillance car and access to live video/audio streams.

6. **Security Protocols and Encryption**: Implement encryption protocols and security measures to secure the communication between the surveillance car and the user's device, ensuring the privacy and integrity of the data transmitted.

4.2.4 SOFTWARE SPECIFICATION:

Arduino IDE

Image 1: Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software tool used for programming Arduino microcontroller boards. It offers a user-friendly interface for writing, compiling, and uploading code to Arduino devices. The IDE supports various programming languages, including C and C++, making it accessible to both beginners and experienced developers. It includes a built-in text editor with features like syntax highlighting and auto-completion, enhancing code readability and efficiency. The IDE provides a seamless workflow for managing libraries, uploading sketches, and monitoring serial communication.The Arduino IDE is cross-platform, compatible with Windows, macOS, and Linux operating systems. It is open-source software, allowing users to contribute to its development and customize it according to their needs. The IDE integrates seamlessly with Arduino hardware, simplifying the process of programming and debugging Arduino-based projects.

4.3 CONSTRAINTS:

Several constraints may influence the development and deployment of an ESP-32 controlled surveillance car project:

1. Power Limitations
2. Size and Weight Restrictions
3. Communication Range
4. Processing Power
5. Sensor Limitations
6. Environmental Factors
7. Regulatory Compliance
8. Cost Constraints
9. Integration Challenges
10. Security and Privacy Concerns

5. SYSTEM LAYOUT

5.1 PARTS ASSEMBLY

Image 1: All the hardware parts

In the above image we can see all the parts that we have used in making our project. All these parts contains car chassis, motors & wheels, esp32-cam module, motor driver, bread board, rechargeable batteries, servo motors, servo brackets etc . Further on we will assemble these parts and then upload the code in the ESP 32-CAM module to smoothly run our surveillance car. All the description of these parts where already mentioned above.

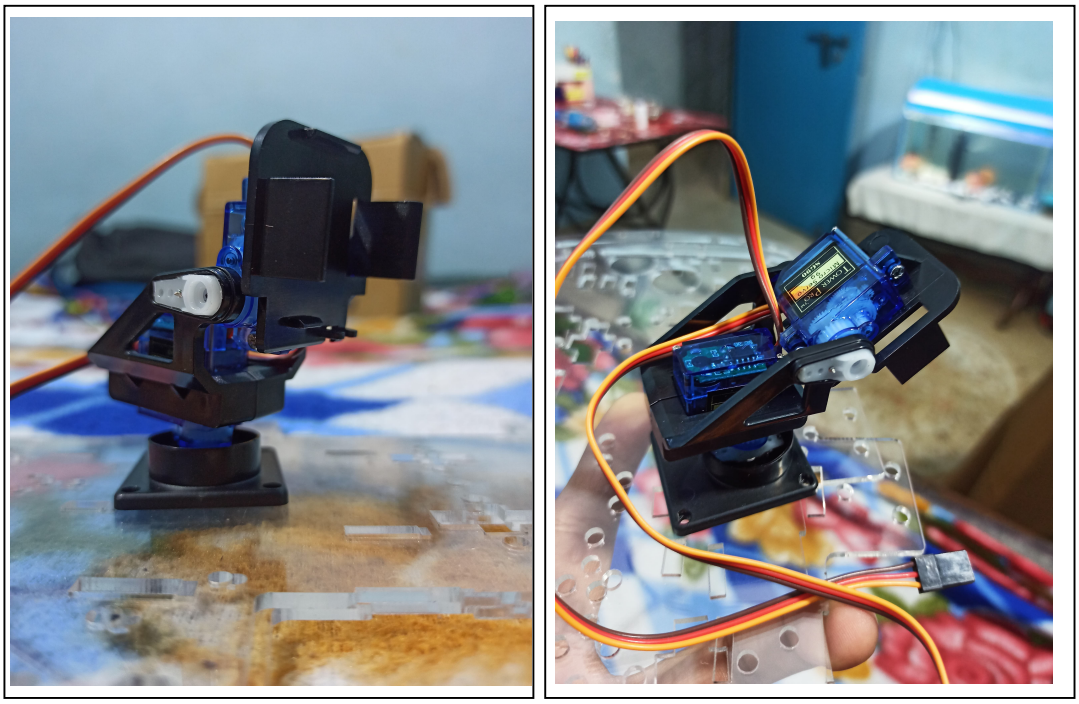


Image 2: Pan-tilt servo assembly

We have started our assembling journey with pan-tilt servo assembly, where we assembled the servo motors in the servo brackets to make our pan-tilt module. Firstly we fix one servo motor in the base of servo bracket to make a pan (right-left) motion. Then we took the second servo motor and assembled it to make tilt (upward-downward) motion.

With the help of this pan-tilt servo assembly we can move the camera into left-right and upward-downward direction to enhance our view sight.

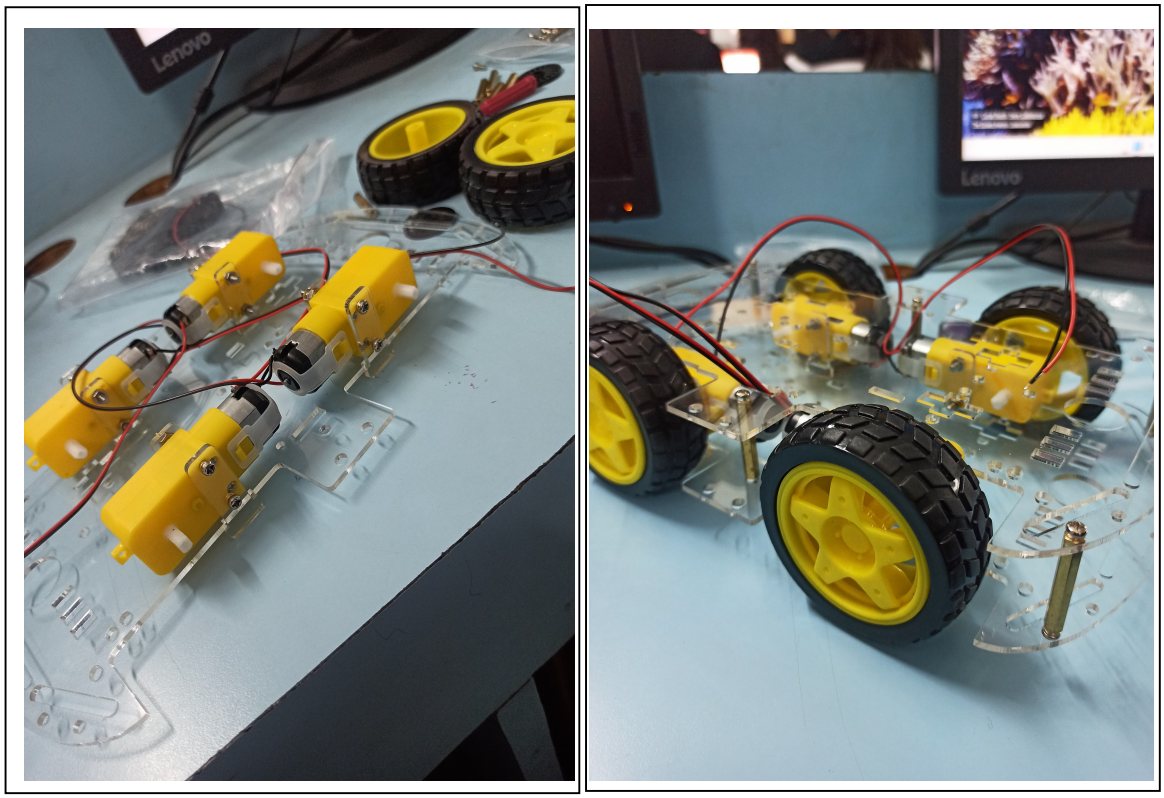


Image 3: Car chassis assembly

Here we assembled the car chassis, and after assembling the car chassis we will mount the pan-tilt assembly on the top of the car chassis.

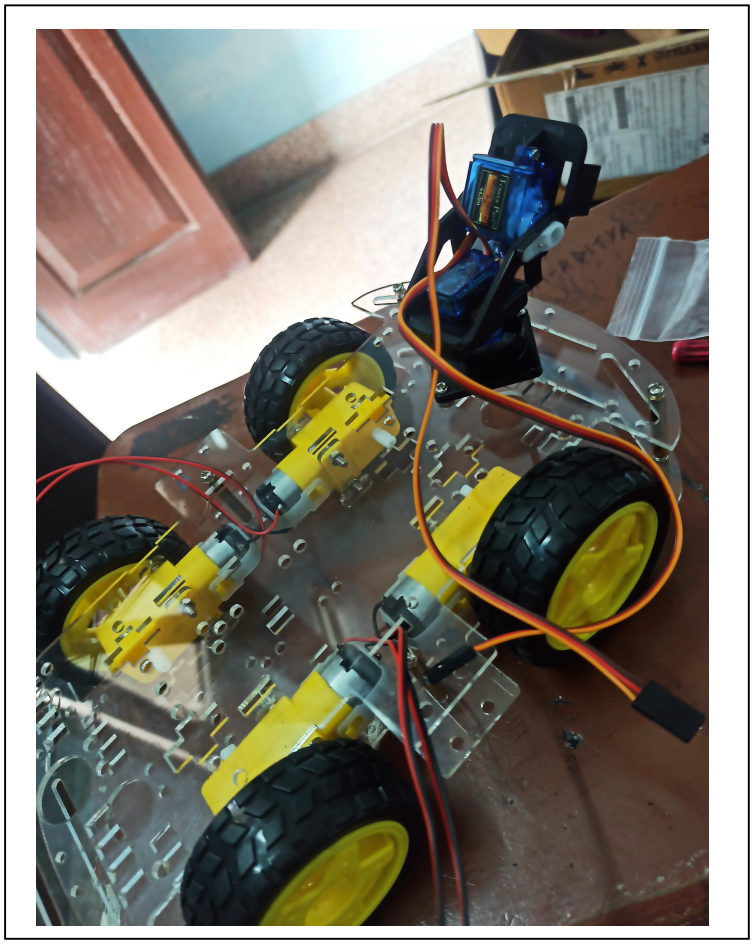


Image 4: Car chassis along with pan-tilt assembly

After assembling these parts our overall structure of the car is ready. And now it's time to upload the code in our ESP 32-CAM module.

5.2 CONNECTION DIAGRAM – I

(For uploading the code)

As if till now the overall body of our surveillance car is ready. So now we will upload the code in the ESP 32- CAM module using the ARDUINO.

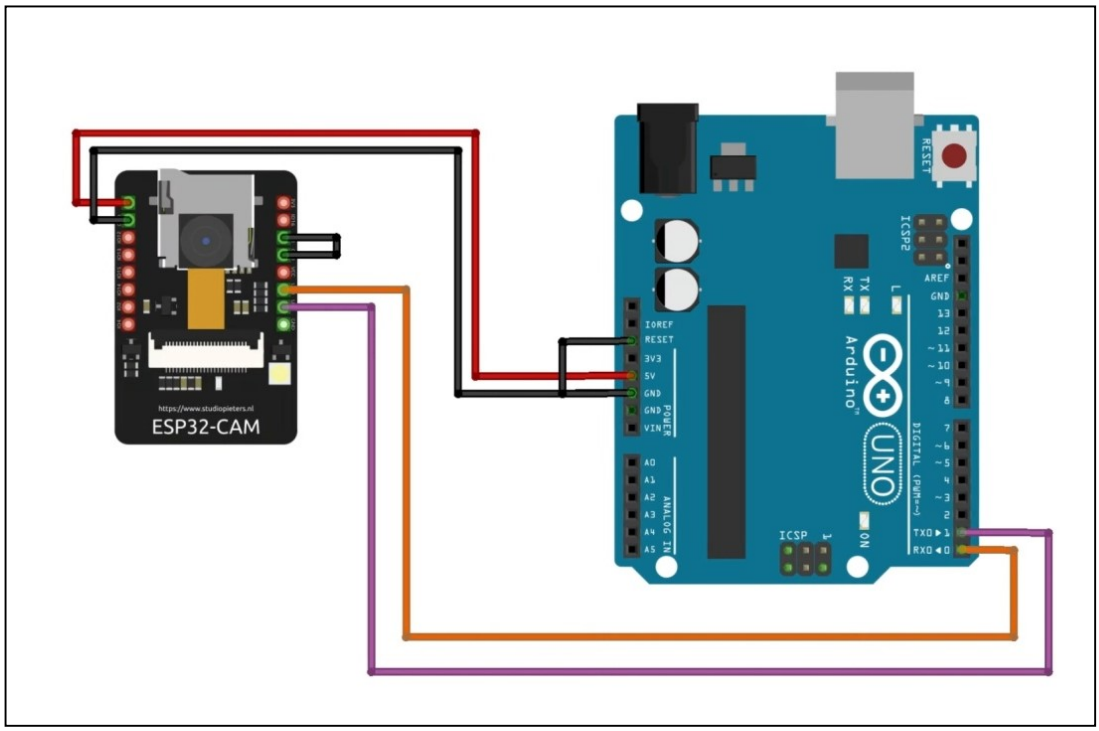


Image 1: Connection between ESP32 Camera Module and Arduino

In-order to upload the desired code of our project in the ESP 32-CAM module we have used Arduino Uno R3 ATmega328P.

The connections are as follows:

Arduino UNO	**ESP 32-CAM**
RX	VOR
TX	VOT
5V	5V
GND	GND
RESET ↔ GND	IO0 ↔ GND

After making all the connections according to the wiring diagram, now we have to select all the parameters in the arduino ide before uploading the code in the esp 32-cam module.

These parameters include selecting the board type, port, flash frequency, flash mode, partition scheme, and at last upload speed. After selecting all these parameters according to our need then we can upload our code in the esp 32-cam using arduino.

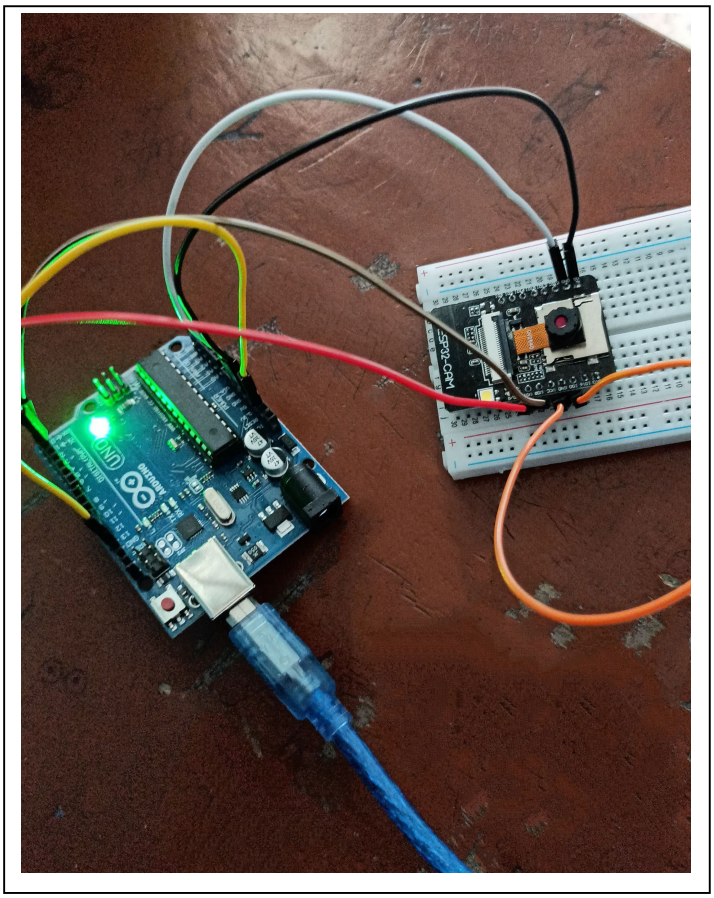


Image 2: Connection between ESP32 Camera Module and Arduino

As we can see above how we made a connection between the arduino uno and esp 32-cam module in order to upload the code of our project. Afterwards then we made the selected changes in the arduino ide before uploading the code.

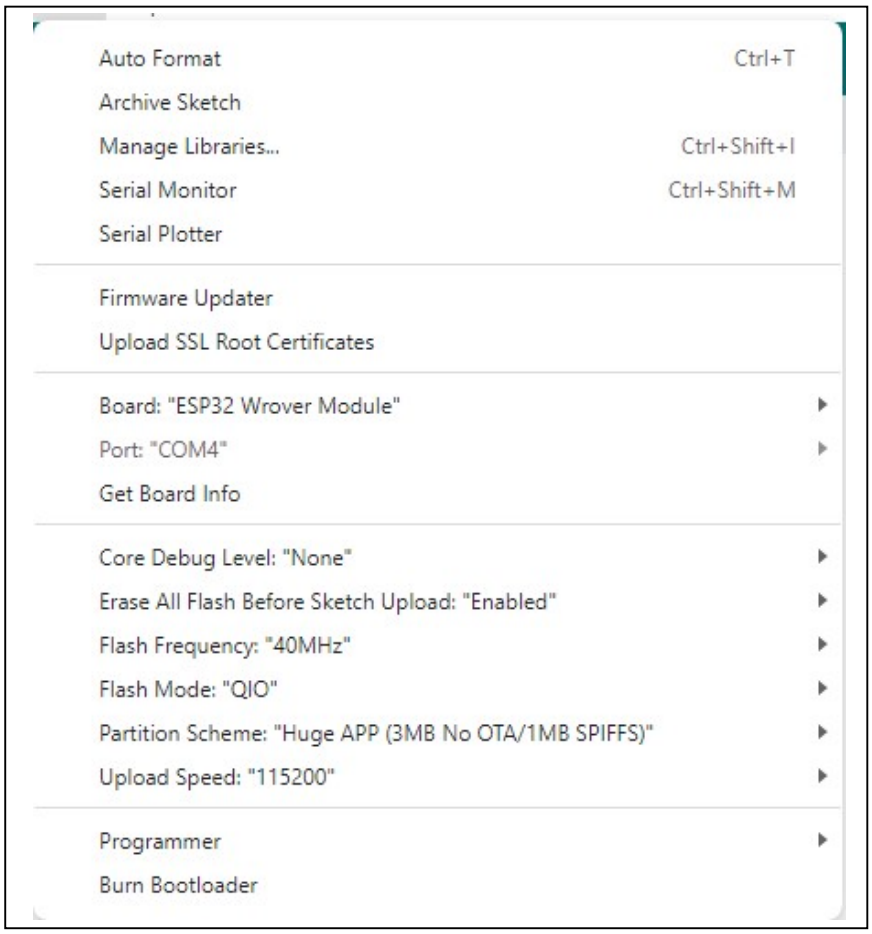


Image 3: Specific changes in arduino ide

After making these changes we can simply click the upload button to upload the code.

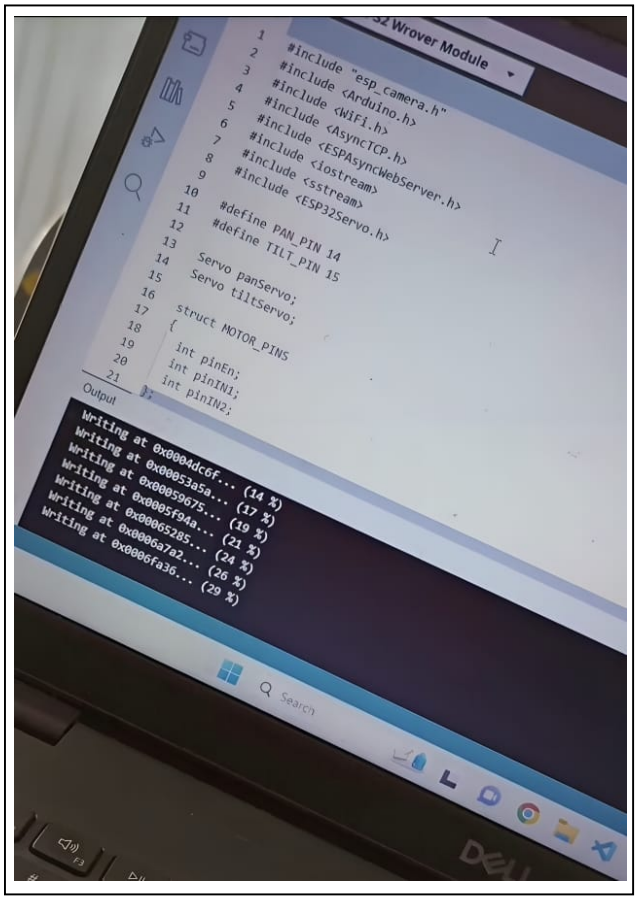


Image 4: Codes getting uploaded

As we can see in the above image all the codes were being uploaded on the ESP32-CAM module.

In the code we were already set the IP address and the wifi name and password for the same, which we will use to connect the car with our device. And by entering the IP address we will be able to control the car and watch whatever is going around the car.

Now our ESP 32- CAM module is ready to be assembled along with car chassis.

5.3 CONNECTION DIAGRAM – II

(For connecting esp 32 module with car)

Firstly we will connect all the DC motors with L298N DC motor driver.

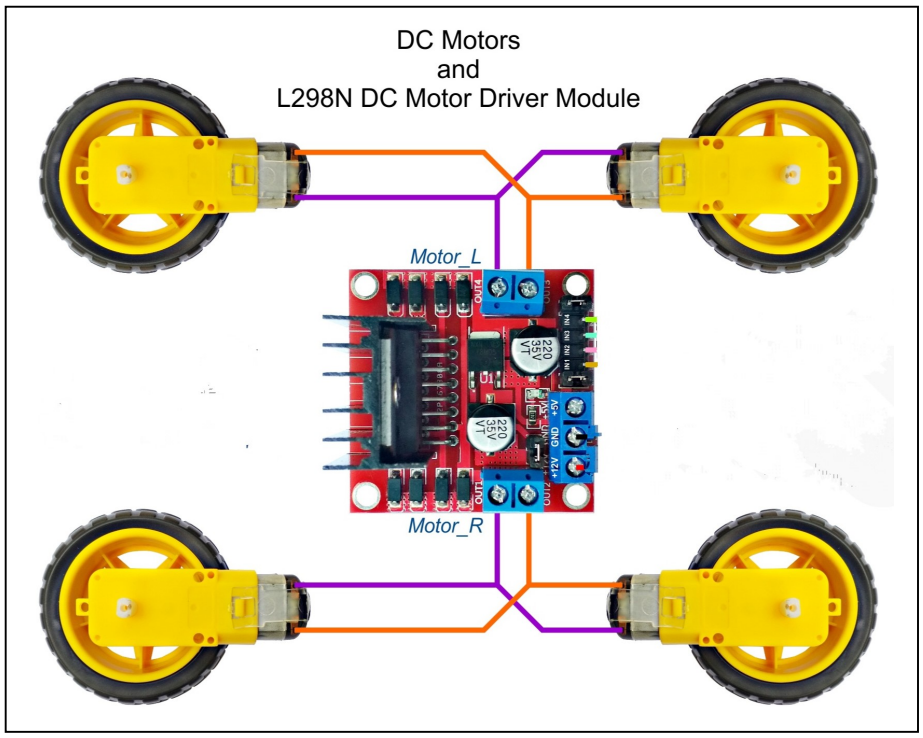


Image 1: Connection between DC motors and motor driver

After making the connection between the motor driver and the motor, now we will connect the ESP32- CAM module with the motor and the pan-tilt servo assembly.

As ESP32 module will receive the instructions from our device and then sends back the instructions to the motor driver and servo motors to work as per the instructions.

We can say that the ESP32- CAM module is the main processing unit of our surveillance car as it hosts the webserver, receives instructions from our devices, sends instructions to the motor driver and the servo motors for the proper working of the car.

Now we will the see the connection between the ESP32 module with the motor driver and the servo motors.

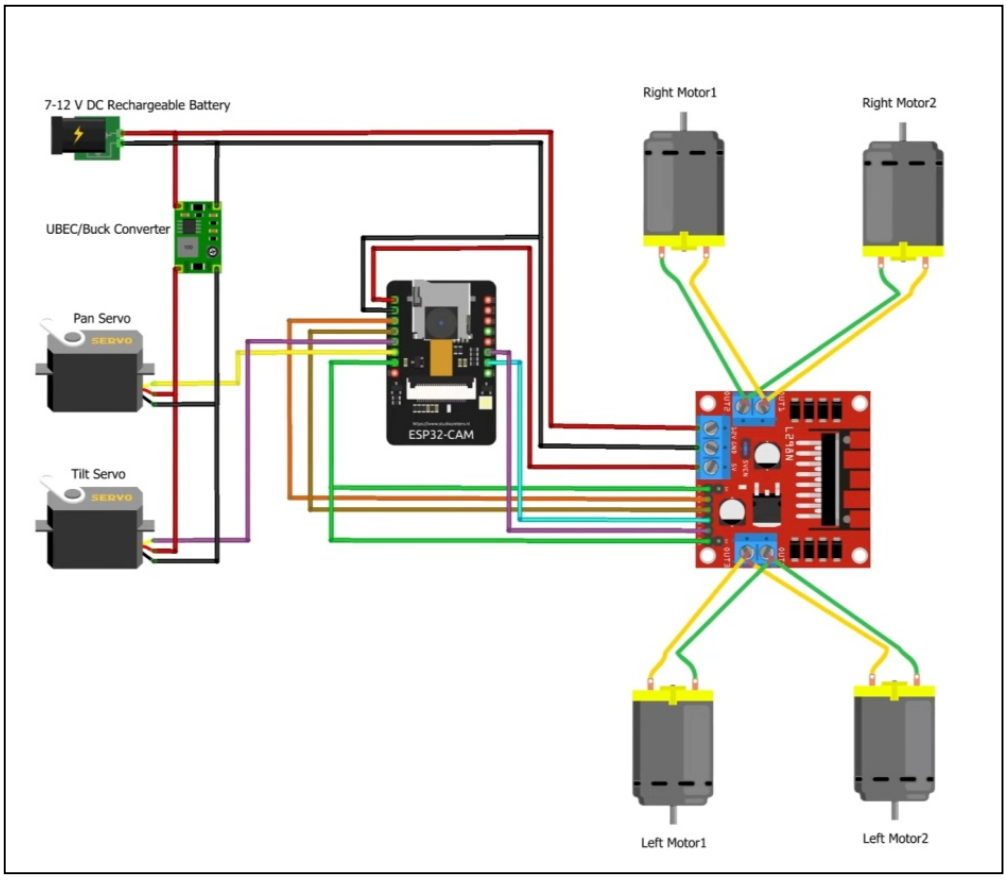


Image 2: Connection of ESP 32 module with motor driver & servo motors

What we see above is the actual wiring diagram for the car, where we have connected the ESP32-CAM module with the motor driver, servo motors, buck converter and the battery pack.

The connections for the cam module with motor driver are as follows :

ESP32-CAM	**L298N Motor Driver**
VOR	IN4
VOT	IN3
IO13	IN2
IO12	IN1

The connections for the cam module with servo motors are as follows :

ESP32-CAM	**Servo Motors**
IO14	PAN SERVO (Left-Right)
IO15	TILT SERVO (Up-Down)

After making all the connections perfectly according to the connection diagram & chart mentioned above, now it's time to attach the battery pack with the motor driver and the dc-dc buck converter.

The connections are as follows :

Battery Pack (12V) *(Output)*	**Motor Driver (12V)** *(Input)*	**Buck Converter (12V)** *(Input)*
Positive (+)	Positive (+)	Positive (+)
Negative (-)	Negative (-)	Negative (-)

Here we have connected the output of power supply with the input of motor driver and the buck converter. And then we will connect the output of the motor driver and buck converter with the esp32-cam module and servo motors respectively.

The connections are as follows :

Motor Driver (5V) *(Output)*	**ESP32-Cam (5V)** *(Input)*
Positive (+)	Positive (+)
Negative (-)	Negative (-)

The connections are as follows :

Buck Converter (5V) *(Output)*	**Servo Motors (5V)** *(Input)*
Positive (+)	Positive (+)
Negative (-)	Negative (-)

The main purpose to use a buck converter here is to avoid the voltage fluctuation in our project. As the motor driver alone is not able to power supply both the servo motors and the camera module, so to maintain the proper voltage flow we have used a dc-dc buck converter which will supply 5V output to the servo motors separately without any fluctuation.

In order to maintain a 5V output from the buck converter we have to adjust the power supply with a screw present on the buck converter. The output voltage from the buck converter depends on the clockwise and anti-clockwise movement of the screw.

And to measure the proper output voltage we have to use a multimeter.

5.4 SYSTEM FLOWCHART

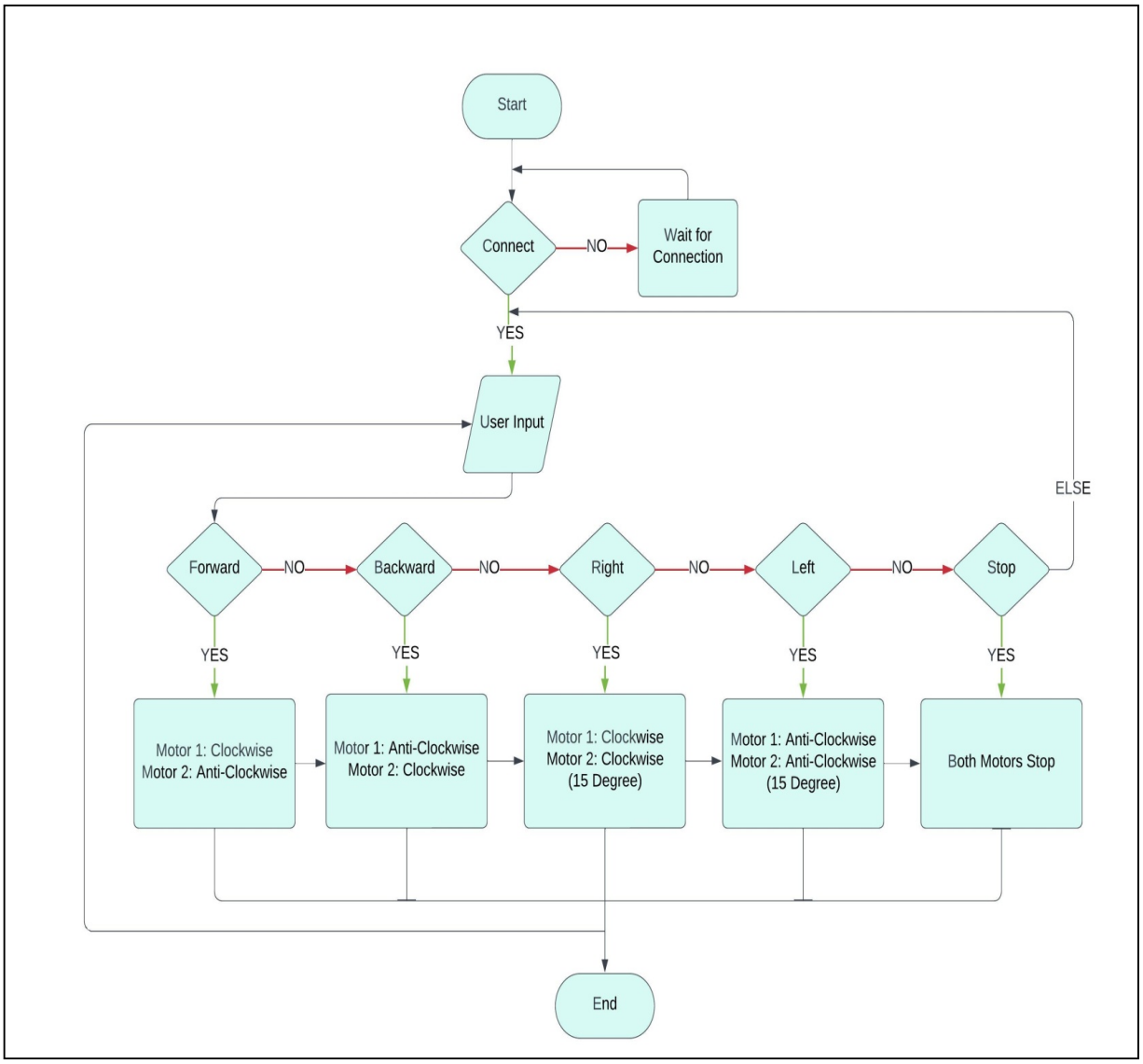


Image 1: System Flowchart

5.5 FINAL LOOK

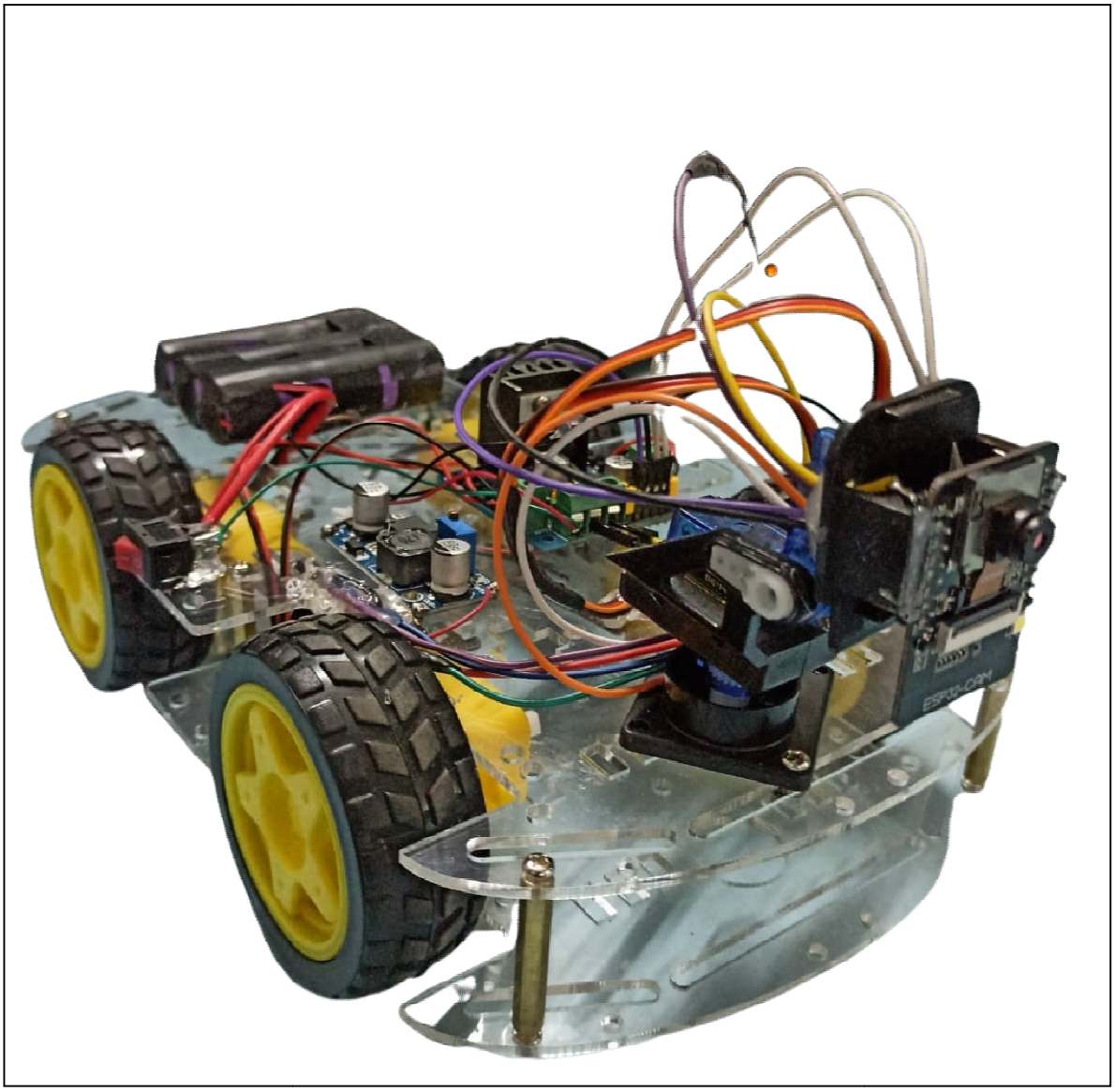


Image 1: Surveillance Car

6. SCREENSHOT

In order to use the surveillance car firstly we have to connect it to our device via WiFi.

SSID : Spy_Car

Password : 12345678

We can easily connect to our car by following up the above credentials, which we have already declared in the code.

After connecting it to our device now we will enter the IP address in our web browser i.e : **192.168.4.1**

As soon as we will enter that IP address our web page to control & watch the surrounding of the car will appear.

The overall interface to control the car is very user friendly, so anyone can control it easily and able to watch the their surrounding by sitting at their comfort place.

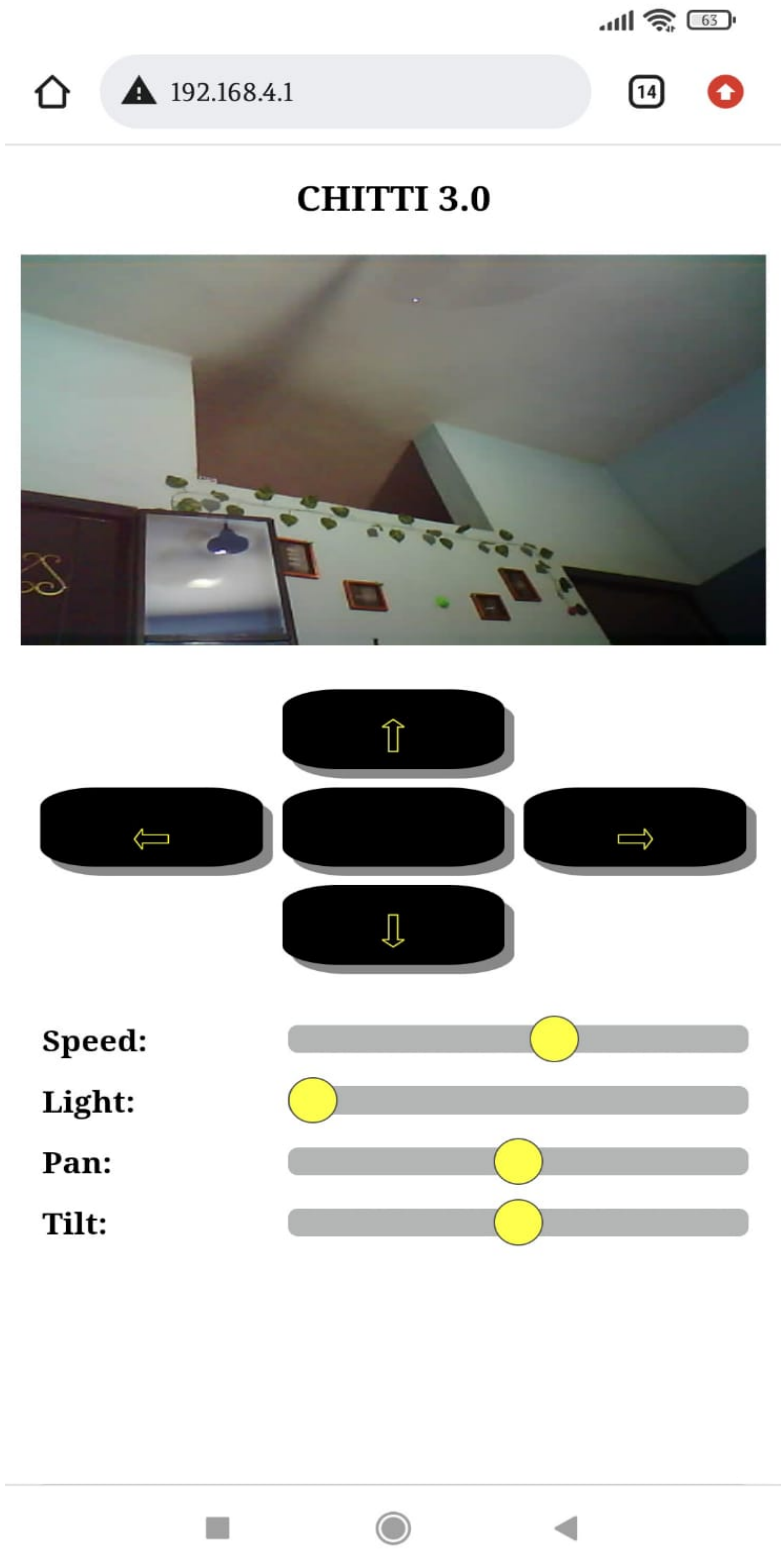


Image 1: Interface of our surveillance car

7. SOURCE CODE

```
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <iostream>
#include <sstream>
#include <ESP32Servo.h>

#define PAN_PIN 14
#define TILT_PIN 15

Servo panServo;
Servo tiltServo;

struct MOTOR_PINS
{
  int pinEn;
  int pinIN1;
  int pinIN2;
};

std::vector<MOTOR_PINS> motorPins =
{
  {2, 12, 13}, //RIGHT_MOTOR Pins (EnA, IN1,
IN2)
  {2, 1, 3},  //LEFT_MOTOR  Pins (EnB, IN3,
IN4)
};
#define LIGHT_PIN 4
```

```
#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define STOP 0

#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1

#define FORWARD 1
#define BACKWARD -1

const int PWMFreq = 1000; /* 1 KHz */
const int PWMResolution = 8;
const int PWMSpeedChannel = 2;
const int PWMLightChannel = 3;

//Camera related constants
#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
```

```
const char* ssid     = "Spy_Car";
const char* password = "12345678";

AsyncWebServer server(80);
AsyncWebSocket wsCamera("/Camera");
AsyncWebSocket wsCarInput("/CarInput");
uint32_t cameraClientId = 0;

const char* htmlHomePage PROGMEM =
R"HTMLHOMEPAGE(
<!DOCTYPE html>
<html>
  <head>
  <meta name="viewport" content="width=device-
width, initial-scale=1, maximum-scale=1, user-
scalable=no">
    <style>
    .arrows {
      font-size:30px;
      color: yellow;
    }
    td.button {
      background-color:black;
      border-radius:25%;
      box-shadow: 5px 5px #888888;
    }
    td.button:active {
      transform: translate(5px,5px);
      box-shadow: none;
    }

    .noselect {
```

```
      -webkit-touch-callout: none; /* iOS
Safari */
        -webkit-user-select: none; /* Safari */
         -khtml-user-select: none; /* Konqueror
HTML */
           -moz-user-select: none; /* Firefox
*/
            -ms-user-select: none; /* Internet
Explorer/Edge */
                user-select: none; /* Non-
prefixed version, currently
                                     supported
by Chrome and Opera */
    }

    .slidecontainer {
      width: 100%;
    }

    .slider {
      -webkit-appearance: none;
      width: 100%;
      height: 15px;
      border-radius: 5px;
      background: #949697;
      outline: none;
      opacity: 0.7;
      -webkit-transition: .2s;
      transition: opacity .2s;
    }

    .slider:hover {
      opacity: 3;
    }
```

```
    .slider::-webkit-slider-thumb {
      -webkit-appearance: none;
      appearance: none;
      width: 25px;
      height: 25px;
      border-radius: 50%;
      background: yellow;
      cursor: pointer;
      border: 1px solid black;
      border-radius: 25px;
    }

    .slider::-moz-range-thumb {
      width: 25px;
      height: 25px;
      border-radius: 50%;

      background:yellow;
      cursor: pointer;
    }

    </style>

  </head>
  <body class="noselect" align="center"
style="background-color:white">
      <table id="mainTable"
style="width:100%;max-
width:400px;margin:auto;table-layout: fixed;"
CELLSPACING=10>
      <tr>
        <h3>CHITTI 3.0</h3>
```

```
        <img id="cameraImage" src=""
style="width:100%;height:210px;max-
width:400px;max-height:
210px;margin:auto;table-layout: fixed;"></td>
      </tr>
      <tr>
        <td></td>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","1")'
ontouchend='sendButtonInput("MoveCar","0")'><sp
an class="arrows" >&#8679;</span></td>
        <td></td>
      </tr>
      <tr>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","3")'
ontouchend='sendButtonInput("MoveCar","0")'><sp
an class="arrows" >&#8678;</span></td>
        <td class="button"></td>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","4")'
ontouchend='sendButtonInput("MoveCar","0")'><sp
an class="arrows" >&#8680;</span></td>
      </tr>
      <tr>
        <td></td>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","2")'
ontouchend='sendButtonInput("MoveCar","0")'><sp
an class="arrows" >&#8681;</span></td>
        <td></td>
      </tr>
      <tr/><tr/>
      <tr>
```

```
        <td style="text-
align:left"><b>Speed:</b></td>
        <td colspan=2>
         <div class="slidecontainer">
            <input type="range" min="0"
max="255" value="150" class="slider" id="Speed"
oninput='sendButtonInput("Speed",value)'>
          </div>
        </td>
      </tr>
      <tr>
        <td style="text-
align:left"><b>Light:</b></td>
        <td colspan=2>
          <div class="slidecontainer">
            <input type="range" min="0"
max="255" value="0" class="slider" id="Light"
oninput='sendButtonInput("Light",value)'>
          </div>
        </td>
      </tr>
      <tr>
        <td style="text-
align:left"><b>Pan:</b></td>
        <td colspan=2>
         <div class="slidecontainer">
            <input type="range" min="0"
max="180" value="90" class="slider" id="Pan"
oninput='sendButtonInput("Pan",value)'>
          </div>
        </td>
      </tr>
      <tr>
```

```
        <td style="text-
align:left"><b>Tilt:</b></td>
        <td colspan=2>
          <div class="slidecontainer">
            <input type="range" min="0"
max="180" value="90" class="slider" id="Tilt"
oninput='sendButtonInput("Tilt",value)'>
          </div>
        </td>
      </tr>
    </table>

    <script>
      var webSocketCameraUrl = "ws:\/\/" +
window.location.hostname + "/Camera";
      var webSocketCarInputUrl = "ws:\/\/" +
window.location.hostname + "/CarInput";
      var websocketCamera;
      var websocketCarInput;

      function initCameraWebSocket()
      {
        websocketCamera = new
WebSocket(webSocketCameraUrl);
        websocketCamera.binaryType = 'blob';
        websocketCamera.onopen    =
function(event){};
        websocketCamera.onclose   =
function(event){setTimeout(initCameraWebSocket,
2000);};
        websocketCamera.onmessage =
function(event)
        {
```

```
          var imageId =
document.getElementById("cameraImage");
          imageId.src =
URL.createObjectURL(event.data);
        };
      }

      function initCarInputWebSocket()
      {
        websocketCarInput = new
WebSocket(webSocketCarInputUrl);
        websocketCarInput.onopen    =
function(event)
        {
          sendButtonInput("Speed",
document.getElementById("Speed").value);
          sendButtonInput("Light",
document.getElementById("Light").value);
          sendButtonInput("Pan",
document.getElementById("Pan").value);
          sendButtonInput("Tilt",
document.getElementById("Tilt").value);

        };
        websocketCarInput.onclose   =
function(event){setTimeout(initCarInputWebSocke
t, 2000);};
        websocketCarInput.onmessage =
function(event){};
      }

      function initWebSocket()
      {
        initCameraWebSocket ();
```

```
        initCarInputWebSocket();
      }

      function sendButtonInput(key, value)
      {
        var data = key + "," + value;
        websocketCarInput.send(data);
      }

      window.onload = initWebSocket;

document.getElementById("mainTable").addEventLi
stener("touchend", function(event){
        event.preventDefault()
      });
    </script>
  </body>
</html>
)HTMLHOMEPAGE";


void rotateMotor(int motorNumber, int
motorDirection)
{
  if (motorDirection == FORWARD)
  {
    digitalWrite(motorPins[motorNumber].pinIN1,
HIGH);
    digitalWrite(motorPins[motorNumber].pinIN2,
LOW);
  }
  else if (motorDirection == BACKWARD)
  {
```

```
    digitalWrite(motorPins[motorNumber].pinIN1,
LOW);
    digitalWrite(motorPins[motorNumber].pinIN2,
HIGH);
  }
  else
  {
    digitalWrite(motorPins[motorNumber].pinIN1,
LOW);
    digitalWrite(motorPins[motorNumber].pinIN2,
LOW);
  }
}

void moveCar(int inputValue)
{
  Serial.printf("Got value as %d\n",
inputValue);
  switch(inputValue)
  {

    case UP:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);

      break;

    case DOWN:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case LEFT:
      rotateMotor(RIGHT_MOTOR, FORWARD);
```

```
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case RIGHT:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case STOP:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      break;

    default:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      break;
  }
}

void handleRoot(AsyncWebServerRequest *request)
{
  request->send_P(200, "text/html",
htmlHomePage);
}

void handleNotFound(AsyncWebServerRequest
*request)
{
    request->send(404, "text/plain", "File Not
Found");
}
```

```
void onCarInputWebSocketEvent(AsyncWebSocket
*server,
                      AsyncWebSocketClient
*client,
                      AwsEventType type,
                      void *arg,
                      uint8_t *data,
                      size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u
connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u
disconnected\n", client->id());
      moveCar(0);
      ledcWrite(PWMLightChannel, 0);
      panServo.write(90);
      tiltServo.write(90);
      break;
    case WS_EVT_DATA:
      AwsFrameInfo *info;
      info = (AwsFrameInfo*)arg;
      if (info->final && info->index == 0 &&
info->len == len && info->opcode == WS_TEXT)
      {
        std::string myData = "";
        myData.assign((char *)data, len);
        std::istringstream ss(myData);
        std::string key, value;
```

```
        std::getline(ss, key, ',');
        std::getline(ss, value, ',');
        Serial.printf("Key [%s] Value[%s]\n",
key.c_str(), value.c_str());
        int valueInt = atoi(value.c_str());
        if (key == "MoveCar")
        {
          moveCar(valueInt);
        }
        else if (key == "Speed")
        {
          ledcWrite(PWMSpeedChannel, valueInt);
        }
        else if (key == "Light")
        {
          ledcWrite(PWMLightChannel, valueInt);

        }
        else if (key == "Pan")
        {
          panServo.write(valueInt);
        }
        else if (key == "Tilt")
        {
          tiltServo.write(valueInt);
        }
      }
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
```

```
}

void onCameraWebSocketEvent(AsyncWebSocket
*server,
                      AsyncWebSocketClient
*client,
                      AwsEventType type,
                      void *arg,
                      uint8_t *data,
                      size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u
connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
      cameraClientId = client->id();
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u
disconnected\n", client->id());
      cameraClientId = 0;
      break;
    case WS_EVT_DATA:
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}
```

```
void setupCamera()
{
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_4;
  config.ledc_timer = LEDC_TIMER_2;
  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.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  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;
  }

  if (psramFound())
  {
    heap_caps_malloc_extmem_enable(20000);
    Serial.printf("PSRAM initialized. malloc to
take memory from psram above this size");
  }
}

void sendCameraPicture()
{
  if (cameraClientId == 0)
  {
    return;
  }
  unsigned long  startTime1 = millis();
  //capture a frame
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb)
  {
      Serial.println("Frame buffer could not be
acquired");
      return;
  }

  unsigned long  startTime2 = millis();
  wsCamera.binary(cameraClientId, fb->buf, fb-
>len);
  esp_camera_fb_return(fb);

  //Wait for message to be delivered
  while (true)
```

```
  {
    AsyncWebSocketClient * clientPointer =
wsCamera.client(cameraClientId);
    if (!clientPointer || !(clientPointer-
>queueIsFull()))
    {
      break;
    }
    delay(1);
  }

  unsigned long  startTime3 = millis();
  Serial.printf("Time taken Total:
%d|%d|%d\n",startTime3 - startTime1, startTime2
- startTime1, startTime3-startTime2 );
}

void setUpPinModes()
{
  panServo.attach(PAN_PIN);
  tiltServo.attach(TILT_PIN);

  //Set up PWM
  ledcSetup(PWMSpeedChannel, PWMFreq,
PWMResolution);
  ledcSetup(PWMLightChannel, PWMFreq,
PWMResolution);

  for (int i = 0; i < motorPins.size(); i++)
  {
    pinMode(motorPins[i].pinEn, OUTPUT);
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);
```

```
    /* Attach the PWM Channel to the motor enb
Pin */
    ledcAttachPin(motorPins[i].pinEn,
PWMSpeedChannel);
  }
  moveCar(STOP);

  pinMode(LIGHT_PIN, OUTPUT);
  ledcAttachPin(LIGHT_PIN, PWMLightChannel);
}


void setup(void)
{
  setUpPinModes();
  //Serial.begin(115200);

  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);

  wsCamera.onEvent(onCameraWebSocketEvent);
  server.addHandler(&wsCamera);

  wsCarInput.onEvent(onCarInputWebSocketEvent);
  server.addHandler(&wsCarInput);

  server.begin();
  Serial.println("HTTP server started");
```

```
  setupCamera();
}


void loop()
{
  wsCamera.cleanupClients();
  wsCarInput.cleanupClients();
  sendCameraPicture();
  Serial.printf("SPIRam Total heap %d, SPIRam
Free Heap %d\n", ESP.getPsramSize(),
ESP.getFreePsram());
}
```

8. MAINTENANCE

Maintenance of the ESP32-controlled surveillance car project involves several key aspects to ensure its continued functionality and effectiveness:

1.Regular Inspections: Conduct routine checks of hardware components, including motors, sensors, and the ESP32 board, to identify any signs of wear, damage, or malfunction.

2.Battery Management: Monitor the battery health and charging cycles to maintain optimal battery life and prevent power-related issues during operation.

3.Remote Monitoring: Set up remote monitoring capabilities to track the surveillance car's status, receive alerts for critical events (e.g., low battery, system errors), and perform diagnostics or troubleshooting remotely.

4.Component Replacement: Plan for the timely replacement of consumable components (e.g., batteries) and worn-out parts (e.g., motors, sensors) to prevent system downtime and ensure continuous operation.

5.User Support: Provide user support and documentation to assist users in troubleshooting common issues, operating the surveillance car effectively, and accessing support resources when needed.

6.Security Updates: Stay vigilant against security threats by applying security patches, implementing access controls, and periodically reviewing and updating security measures to protect against unauthorized access or data breaches.

7.Feedback and Improvement: Gather feedback from users and stakeholders to identify areas for improvement, new features, or enhancements that can be incorporated into future iterations of the surveillance car.

9. LIMITATIONS

- Due to their small size and weight constraints, ESP-32 controlled surveillance car may have limited capacity to carry heavy or bulky equipment, restricting the types of sensors and other devices that can be integrated.
- Surveillance cars may have limited battery life, necessitating frequent recharging or replacement of batteries, which could affect their operational efficiency.
- ESP-32 modules operates over Wi-Fi or Bluetooth, which have limited range compared to other communication technologies.
- ESP-32 microcontrollers may not be sufficient for handling complex algorithms or large datasets required for advanced analytics.
- Surveillance car may face challenges operating in extreme environmental conditions such as heavy rain, snow, high temperatures, which could affect their performance and durability.
- The use of surveillance technology raises legal and ethical concerns regarding privacy, consent, surveillance abuse, and data protection.

- Building and deploying surveillance cars with advanced features can be costly.
- Surveillance cars require regular maintenance, updates, and technical support to ensure optimal performance and reliability.
- Requires careful adherence to avoid legal risks and penalties.
- Lack of security features.

10. FUTURE SCOPE

The project has a very vast scope in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed design our car is capable for monitoring the surrounding. Yet we can improve the performance of our designed car. The following are the future development for the project.

- In further development the distance range can be increased.
- The number of sensors can be increased to get more data.
- Surviving in different weather conditions can be developed.
- Implement features such as zooming and image stabilization to enhance surveillance capabilities.
- Implement AI algorithms that enable the surveillance car to learn and adapt its behavior over time.
- Implement algorithms that allow the surveillance car to identify and track specific objects or individuals in its surroundings.

11. CONCLUSION

Being the Final year students of Computer Application , we were tried to develop this surveillance car which is quite different of what we learn in our classes, yet we were able to attain our set objectives, and this helped us gain confidence in writing our own code and our own applications.

We worked as a team, and gained some experience on this new technology.

There is always room for improvement, and this application we created can also be improved. The main intention of this research to raise awareness on Robotics. We tried to develop a Surveillance Car that can be used for our daily life. We dreamed, a simple Car that is controlled wirelessly by android or ios devices on the web server.

Our project is extremely economical and price effective that replaces human work and reduces human labor while performing observance works in a well effective manner.

THANK YOU

12. BIBLIOGRAPHY

- https://opensource.com/resources/what-arduino
- https://www.arduino.cc/en/software
- https://www.w3schools.com/html/html_css.asp
- https://www.youtube.com/watch?v=7MdEl0om70w&t=685s
- https://www.youtube.com/watch?v=DdybJZ58mlI
- https://github.com/un0038998/CameraCar