A

PROJECT REPORT

ON

# "Mobile Robot Structure Design and Simulation for Surveillance of Outdoor Security"

Submitted in partial fulfillment of the requirements for the award of degree of

**BACHELOR OF TECHNOLOGY**

in

**Electrical & Electronics Engineering**

**Submitted by:**
Shreya Ojha (2016614)
Aditi Sharma (2016514)
Varnika Chaturvedi (2016627)

**Supervised by:**
Dr. S. K. Das

**SCHOOL OF AUTOMATION**

**BANASTHALI VIDYAPITH RAJASTHAN**

**May 2023**

**BANASTHALI VIDYAPITH RAJASTHAN**

**SCHOOL OF AUTOMATION**

## CERTIFICATE

We hereby submit the project entitled **"Mobile Robot Structure, Design and Simulation for Surveillance of Outdoor Security"** in the **School of Automation** of the Banasthali Vidyapith, under the supervision of **"Dr S. K. Das"**, School of Automation, Banasthali Vidyapith, Rajasthan, India.

Shreya Ojha (2016614)
Aditi Sharma (2016514)
Varnika Chaturvedi (2016627)

The project report is hereby approved for submission.

| | |
|---|---|
| Dr S. K. Das | Dr. Shailly Sharma |
| Mentor Name | (School of Automation) |

Date:

# ABSTRACT

There are various services available like surveillance and security for mobile robots that have been installed in public places like malls and hospitals. Designing a robust and fast mobile robot which can work in challenging known environment is a difficult task. That includes design of mechanical parts and control system of the mobile robot. The objective of this project is to integrate several current technologies along with software and hardware using controller into a robotic system that could carry out security-related duties. This project is designed to develop a mobile robot using vision sensor attached for detection and tracking purpose. The mobile robot along with vision sensor can process real time images for surveillance.

Keywords: Mobile Robot, Vision Sensor, Detection.

## ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us the possibility to complete the report. A special gratitude, we gave to our supervisor. Dr S. K. Das**,** whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project for this session. We sincerely thank our head of department Ms. Shailly Sharma**,** (Dean), School of Automation, Banasthali University for her guidance and encouragement in carrying out this project. We also wish to express our gratitude to other staff members who rendered their help.

Our thanks and appreciation also goes to our friends who helped us in our work of the project and people who have willingly helped us out with their abilities.

**Student Name**

Shreya Ojha
Aditi Sharma
Varnika Chaturvedi

# CONTENTS

# List of figures

CHAPTER-1

# INTRODUCTION

Surveillance is the process of monitoring a situation, an area or a person. Therefore, surveillance is needed within the areas like border areas, public places, offices and in industries. It is mainly used for observation activities. The act of surveillance can be performed each indoor also as in outside areas by humans or with the help of embedded systems like robots and alternative automation devices. A robot is nothing however an automatic electronic machine that can play programmed activities therefore replacing human work, providing extremely correct results, and simply overcoming the constraints of citizenry. The mechanism additionally consists of DC motors, wheel chassis, battery, Wi-Fi module and motor driver. During this project, we tend to use wireless transmission camera that gives video information which will be received at the user end.

## 1.1-BACKGROUND and MOTIVATION

Conventional surveillance for a security robot suffers from severe limitations, perceptual aliasing (e.g., different places/objects can appear identical), occlusion (e.g., place/object appearance changes between visits), illumination changes, significant viewpoint changes, etc. Human surveillance is achieved by deploying personnel near sensitive areas in order to constantly monitor for changes. But humans do have their limitations, and deployment in inaccessible places is not always possible. There are also added risks of losing personnel in the event of getting caught by the enemy. The human cannot record video safely in critical conditions and environments. These conditions and environments may be buildings where the fire breaks out, Areas with poisonous gases or harmful radiation and the places where there is an exchange of fire such as battlefield. With advances in technology over the years, however, it is possibly to remotely monitor areas of importance by using robots in place of humans. Apart from the obvious advantage of not having to risk any personnel, terrestrial and aerial robots can also pick up details that are not obvious to humans. By equipping them with high resolution cameras and various sensors, it is possible to obtain information about the specific area. In recent times, surveillance technology has become an area of great research interest. However, building a small robot for testing and research purposes proves to be extremely expensive. This project introduces the design and implementation of a surveillance robot based on Wi-Fi protocol. The robot can transmit real-time video to the intended recipient. The control mechanism is provided with a video transmission facility. The video transmission is practically achieved through high-speed image transmission. Our project is aimed at developing a surveillance system which can be controlled remotely by using web app. It includes a mechanism which has a wireless camera attach to it. This robot captures video and transmits it to the connected web app which is used to control the robot. The esp32 camera is used in video transmission with the help of web server and Wi-Fi.

## 1.2- PROCESS: -

**Step 1:** First assemble the pan tilt and fix servos.

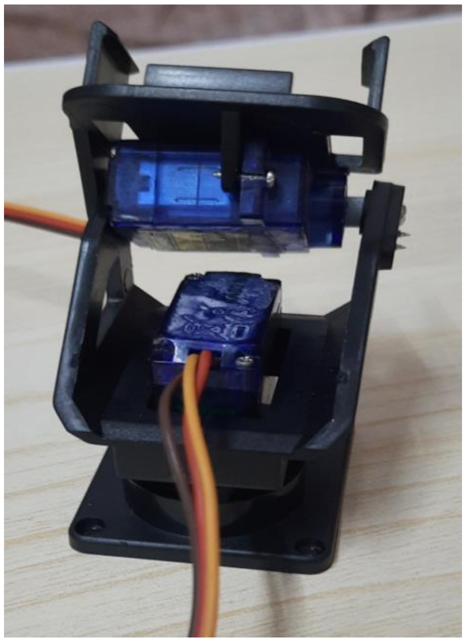

Figure 1.1: Pan tilt servo assembly

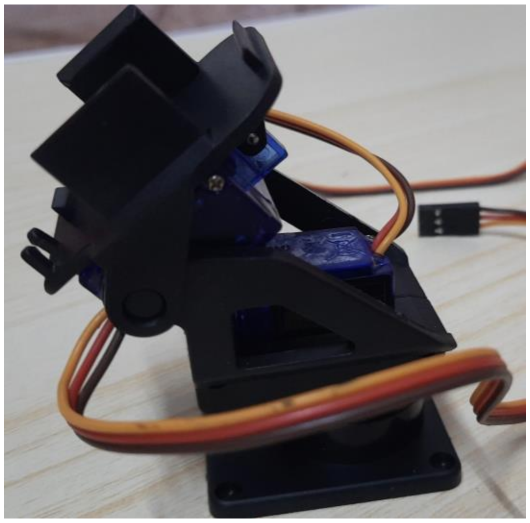

Figure 1.2: Pan tilt servo assembly

**Step 2:** For assemble the car, firstly solder the wires to gear motors. Then, mount all four motors on car chassis. Now attach both the car chassis plate using screws and make sure it is tightened properly so that we do not have any loose part. Now connect DC motors on each side.

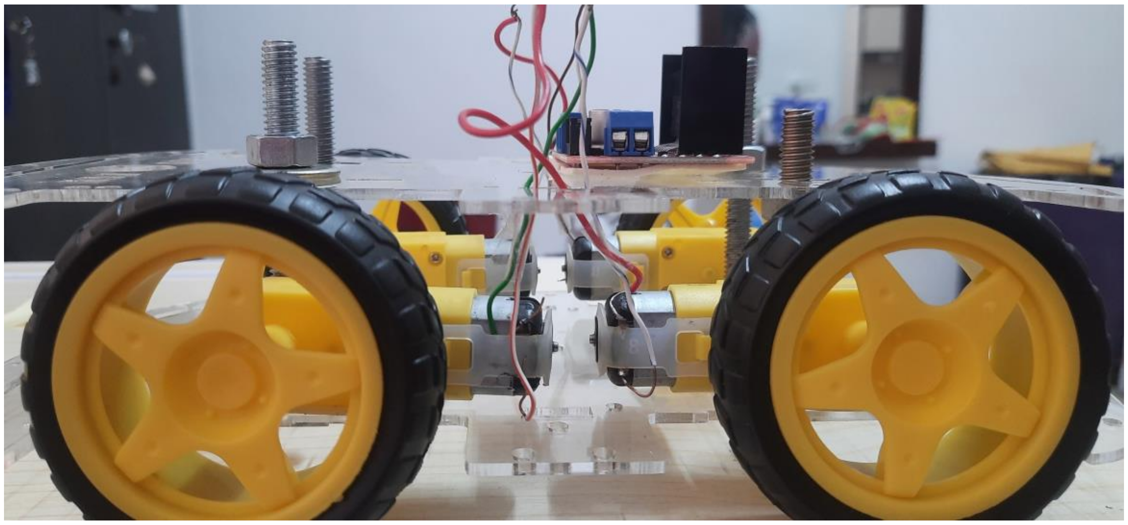

Figure 1.3: Car chassis with DC motors

**Step 3:** Attach L298N motor driver module on car chassis.

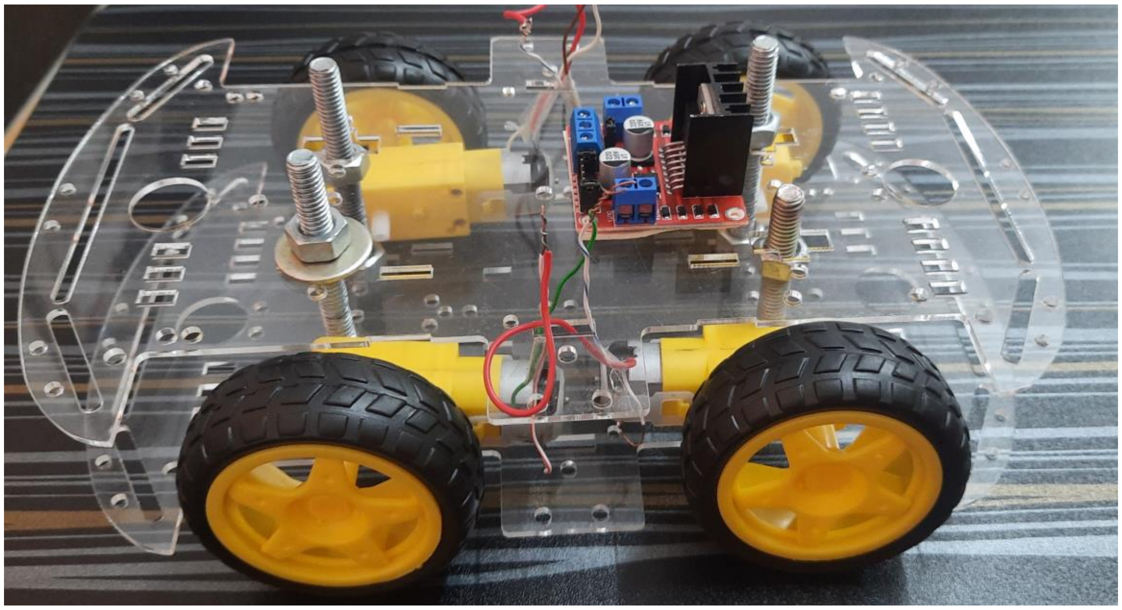

Figure 1.4: Car chassis with L298N motor driver

**Step 4:** Now, make the connections as per circuit diagram:

Connect right side motors to out 1 and out 2 pins of L298N motor driver module. Then, connect L298N motor driver module to esp32 cam pins as per table below.

| **ESP32 cam pins** | **L298N driver pins** |
|---|---|
| IO2 | enA |
| IO12 | IN1 |
| IO13 | IN2 |
| VOT/IO1 | IN3 |
| VOR/IO3 | IN4 |
| IO2 | enB |
| 5V | 5V |
| GND | GND |

Connect both enable A and B pins to I02 to esp32 cam. We will take +5V and ground from L298N motor driver module and provide to esp32 cam module.

**Step 5:** Fix esp32 cam module on structure. Now, mount pan tilt assembly on car chassis.

**Step 6:** Connect pan servo and tilt servo to I014 and I015 respectively.

**Step 7:** Connect USB to battery supply and provide +5V and ground to servos.

**Step 8:** Attach DC battery power connectors to motor driver module to +5V pin and ground pin.

**Step 9:** Attach wheels to car.

**Step 10:** Now we use Arduino uno for uploading code.

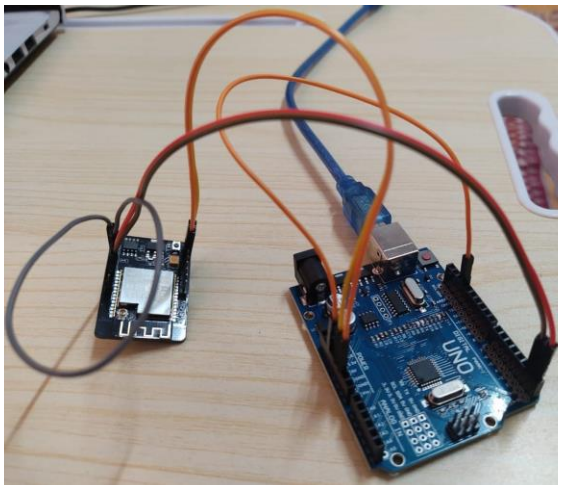
Figure 1.5: Arduino UNO

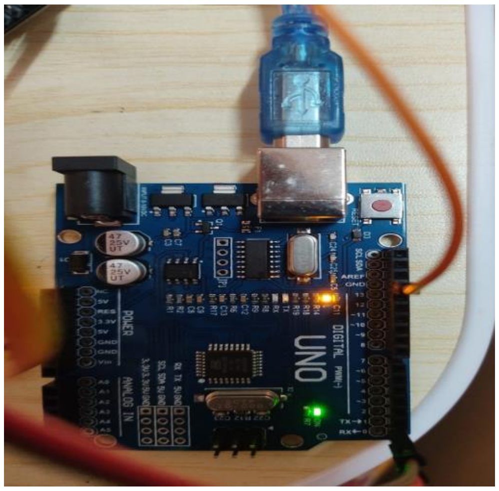
Figure 1.6: Arduino UNO

**Step 11:** Connect battery to car.

**Step 12:** Let us open mobile app now. Go to wi-fi settings and connect to my wi-fi car ssid using password. Open google chrome browser on mobile and type 192.168.4.1 . It will open our camera car control app.

## 1.3- ORGANISATION OF REPORT: -

The organisation of report is as follows, first chapter contains introduction of the topic, background, and motivation behind the project; the process and steps involved in making the robot. The second chapter contains the purpose of the robot and introduction. Then the specific components part contains the components used in making the robot. Then we discuss about robotics and the flowchart of the process. Now we come to the circuit diagram of the connections done in our robot. Then there is the coding process, installation, and setup of some libraries. After that in the code part there is code on which our robot works. The third chapter contains result and discussions regarding our project. At last, the fourth chapter contains conclusion and future scope of this robot.

# CHAPTER-02

# PROBLEM FORMULATION

## 2.1 PURPOSE

It serves the following purpose:-

- Video Surveillance Robot in Agriculture: - Some areas need to be placed under video surveillance for a short time only, and therefore equipping them with stationary CCTV systems is not viable. For example, the fields might only need to be placed under video surveillance when the crops are reaching maturity. Surveillance of pastures is advantageous only while cattle are roaming around them. Conveniently, robot delivery is followed by setting up position specifications (if we add GPS to it) to include operational shift times.
- To Catch a Thief: - Pity the poor security guard. In busy environments such as prisons and crowded shopping malls, they need to be everywhere at once. And in a more isolated setting—on the night shift in a lonely warehouse, for instance—they may struggle to stay engaged and alert. Security robots do not have these problems. Instead, they have artificial intelligence that works without getting tired or bored and streaming video to broadcast everything they see, bringing others on the scene no matter where they might be. Artificial intelligence, streaming video, and other connected technologies are giving us security guards that can work 24 hours a day, can see in the dark, and have eyes—literally—in the backs of their heads.
- Robotics in Security and Military: Mobile robotics plays an increasingly important role in military matters, from patrol to dealing with potential explosives. "With suitable sensors and cameras to perform different missions, mobile robots are operated remotely for reconnaissance patrol and relay back video images to an operator. Robots can neutralize suspicious objects that may explode. The platform has a robot arm to pick up explosives or suspected hazards in military or civilian settings. The mobile robotic platform is mounted on a rectangular box with electronic equipment. The platform moves on wheels or tracks, or both, and is usually battery-powered. Communication equipment and sensors can detect images, sounds, gases, and other hazards. The communication systems read sensors and relay that information to the operator.
- Security robots for mobile patrols: - A security patrol robot is equipped with a 180-degree video surveillance system. It consists of pan tilt shift ESP 32 camera for panoramic

observation and a camera to track objects at larger distances. In automatic mode, the video surveillance system directs the camera at any movement source that is discernible. In manual mode, a human operator can control the camera. A built-in, high-power Wi-Fi unit is used to view the images, control the camera, and control the robot's movement along its route. Wireless communication within the robot's systems allows its status and position to be displayed on a terrain map on a security officer's tablet or on desktop computers at a central monitoring station. Areas along the patrolling route having an unstable Wi-Fi network can cause restricted online video surveillance functionality.

- Virtual security robots: - Virtual robots are essentially AI algorithms that can identify and protect your data and devices against malware. Although not conventional robotics, virtual security robots can be invaluable to large corporations that run on collected data, as well as to everyday people who wish to protect themselves from fraud.
- Search and rescue robots: - In disaster zones that are hard to reach and a danger to life, robots provide invaluable support to human search and rescue teams. These include rugged small vehicles with tracks, cameras and sensors that can search inside rubble and climb over obstacles.

## 2.2 INTRODUCTION

### 2.2.1 Specific Components:

**Esp 32 camera**

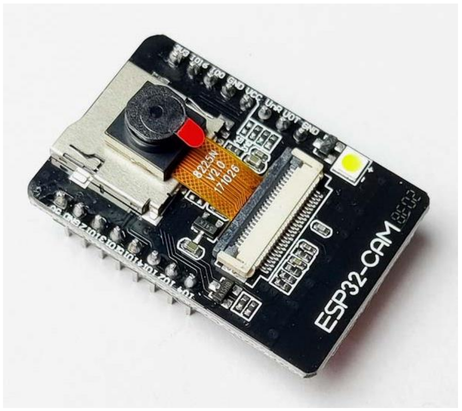

Fig 2.1: Esp 32 cam

ESP32-CAM is a low-cost ESP32-based development board with onboard camera, small.

It consists of:-

1. OV2640 camera
2. ESP32-CAM-MB micro USB to serial port adapter
3. WIFI module: ESP-32S

4. Processor: ESP32-D0WD-V3
5. Built-in Flash: 32Mbit
6. RAM: Internal 512KB + External 4M PSRAM
7. Antenna: Onboard PCB antenna
8. Bluetooth: Bluetooth 4.2 BR/EDR and BLE
9. WIFI mode: Station / SoftAP / SoftAP+Station
10. Security: WPA/WPA2/WPA2-Enterprise/WPS
11. Output image format: JPEG (OV2640 support only), BMP, GRAYSCALE
12. Supported TF card: up to 4G
13. Peripheral interface: UART/SPI/I2C/PWM
14. IO port: 9
15. UART baudrate rate: default 115200bps
16. Power supply: 5V

APPLICATIONS: - The ESP32-CAM suit for IOT applications such as:

- Smart home devices image upload
- Wireless monitoring
- Intelligent agriculture
- QR wireless identification
- facial recognition

In our project, The ESP32-CAM hosts a web server that shows video streaming and buttons to control the servo motors to move the camera.

**Pan Tilt Servo Assembly**

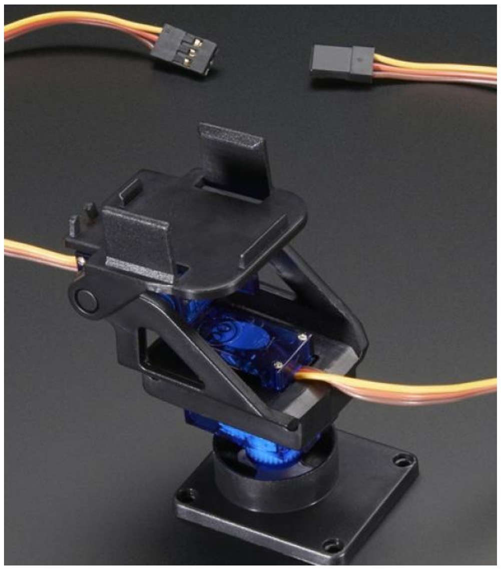

Fig 2.2 pan tilt

The Servo Tilt / Pan Kit for SG-90 Servos is a DIY Pan-Tilt chassis that can rotate up to 180 degrees vertically and horizontally.

It’s features are: -

- Perfect for cameras or sensors up to 500gms
- The most rigid and precise direct drive pan and tilt available
- Servos sold separately, buy here - SG90 9g servo
- Base: 37mm x 33mm x 3mm / 1.5" x 1.3" x 0.1"
- Mounting Hole Distances: 30mm x 26mm / 1.2" x 1"
- Standing Height (zero tilt): 67mm / 2.6"
- Top Platform: 38mm x 36mm / 1.5"x 1.4"
- Weight: 37g

Applications: - widely used in surveillance applications for high accuracy positioning of sensor payloads such as cameras and laser pointers. In our project the pan-tilt rotate roughly 180° from side-to-side and can tilt up & downwards around 150°. It is 38mm x 36mm space mount the camera or sensor.

**SG 90 Servo MOTOR**

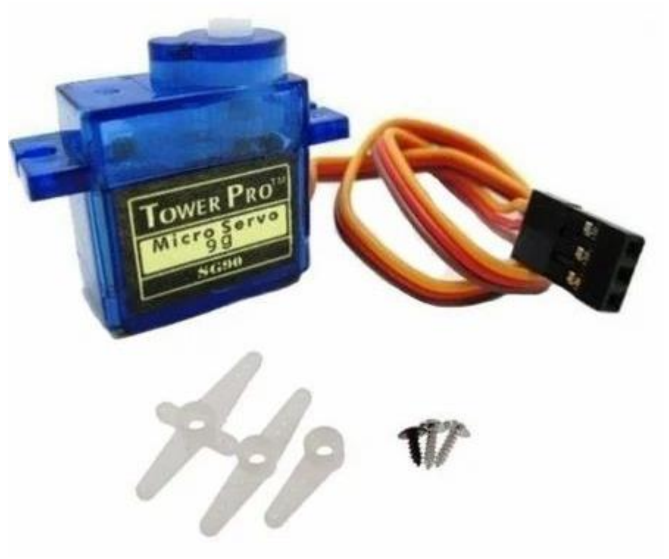

Fig 2.3 SG 90 Servo motor

SG90 is a popular micro servo motor commonly used in DIY projects. It is a small, low-cost servo motor that can rotate 180 degrees with a maximum torque of 1.8 kg-cm.

It consists: - Electronic Assembly. PWM is the electronic board that can control the part of the speed in the servo.

- Protective Case.
- Drive Gears.

- Output Spline.

Applications: -

1. Robotics
2. Telescope
3. Industrial production
4. Machinery works
5. Production industry

In our project, it is used control its speed.

**L298N Motor Driver**

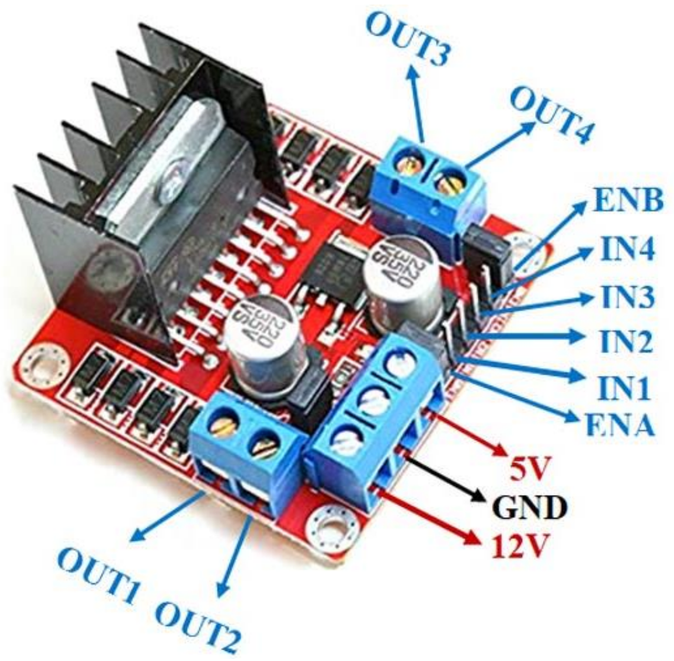

Fig 2.4 L298N Motor Driver

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

It consist:-

- L298 Motor Driver IC
- 78M05 Voltage Regulator
- Resistors
- Capacitor
- Power LED
- 5V jumper in an integrated circuit.

Applications:-

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

In our project, it uses the H-Bridge technique to control the direction of rotation of a DC motor. In this technique, H-Bridge controlled DC motor rotating direction by changing the polarity of its input voltage.

**UBEC/buck Converter**

Fig 2.5 Buck converter

A buck converter or step-down converter is a DC-to-DC converter which steps down voltage (while stepping up current) from its input (supply) to its output (load). It is a class of switched-mode power supply. Switching converters (such as buck converters) provide much greater power efficiency as DC-to-DC converters than linear regulators, which are simpler circuits that lower voltages by dissipating power as heat, but do not step up output current.

It consists:-

- two semiconductors (a diode and a transistor, although modern buck converters frequently replace the diode with a second transistor used for synchronous rectification)
- one energy storage element (a capacitor, inductor, or the two in combination).

Applications:- Buck converter is basically used for conversion of high dc voltage to low dc voltage. It is mostly used in the following applications:

- It is used in battery power systems such as battery chargers for mobile phones, laptops, and power banks
- It is used in self-regulating power systems such as SMPS
- It is used point of load converters for PCs and motherboards

- It is used in solar chargers
- It is used in adaptive control applications
- It is used in power audio amplifier applications

In our project, it performs step-down conversion of the applied dc input signal.

**Arduino UNO**

The Arduino Uno is an open-source microcontroller board and is developed by Arduino.cc and initially released in 2010. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

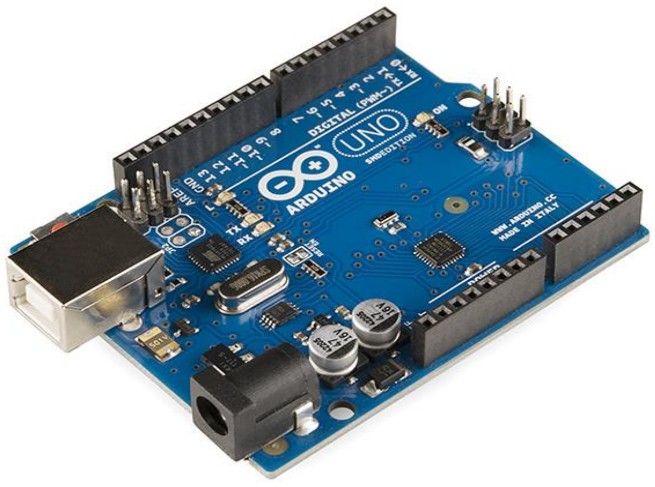

Fig 2.6 Arduino UNO

It consists:-

- 6 analog pin inputs
- 14 digital pins
- a USB connector
- a power jack
- an ICSP (In-Circuit Serial Programming) header.

Applications:-

- Weighing Machines.
- Traffic Light Count Down Timer.
- Parking Lot Counter.
- Embedded systems.
- Home Automation.
- Industrial Automation.
- Medical Instrument.

- Emergency Light for Railways.

In our project, it is used to program ESP 32 board.

### 2.2.2 Robotics: -

- Robotics: It is the branch of technology that deals with the design, construction, operation, and application of robots.
- Robot: A robot is a machine, especially one programmable by a computer that can carry out a complex series of actions automatically.
- Types of robots:
- There are mainly two types of robots, that are
- Fixed Robots
- Mobile robots
- Fixed Robots: Fixed robots can be defined as articulated arm robots perform stationary tasks. These robots are the backbone of many industrial processes and have not changed a tremendous amount in the time that they have been used in manufacturing and logistics.
- Mobile Robots: An autonomous mobile robot (AMR) is a type of robot that can understand and move through its environment without being overseen directly by an operator or limited to a fixed, predetermined path.

Control scheme for mobile robot systems:

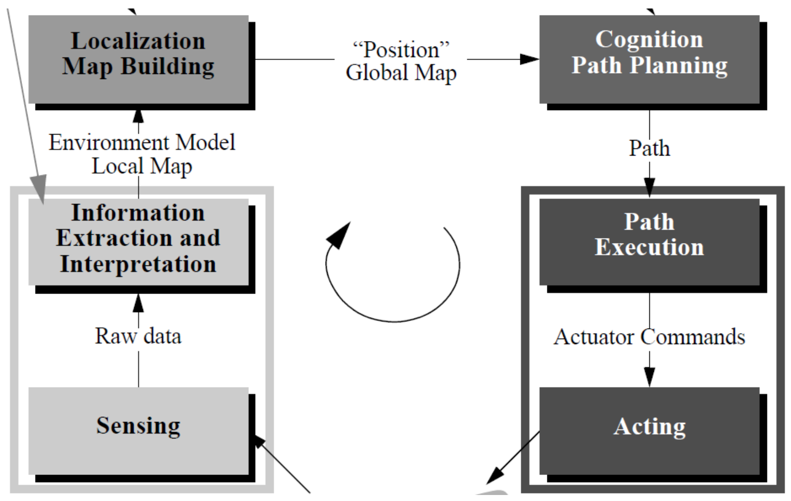

Fig 2.7- Control scheme for mobile robot

- Perception: It is the part where robot senses from the real world environment, extracts and interprets the information received from the raw data.

- Locomotion: Robot locomotion is the collective name for the various methods that robots use to transport themselves from place to place.
- Cognition: It means that robots can learn from experience and from others, commit relevant knowledge and skills to memory, retrieve them as the context requires, and flexibly use this knowledge to select appropriate actions in the pursuit of their goals, while anticipating the outcome of those actions when doing so.
- Path Planning: Robotic path planning is instead of removing or adding material to fabricate an object, robotic path planning determines how an object can navigate through a space with known or unknown obstacles while minimizing collisions.
- Motion Control: Motion control is a sub-field of robotics, encompassing the systems or sub-systems involved in moving parts of machines in a controlled manner. It consists of Path Execution with the help of actuator commands and the robots act accordingly in real world environment.

## 2.3 FLOWCHART

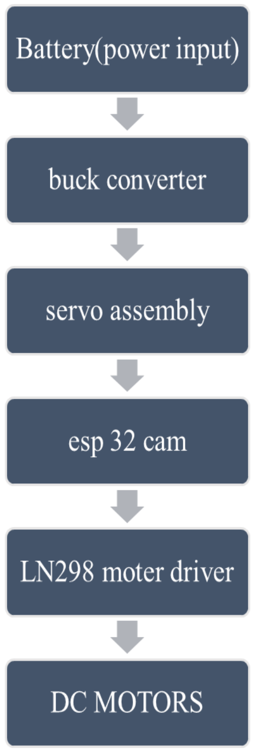

Fig 2.8 Flowchart

## 2.4 CIRCUIT DIAGRAM

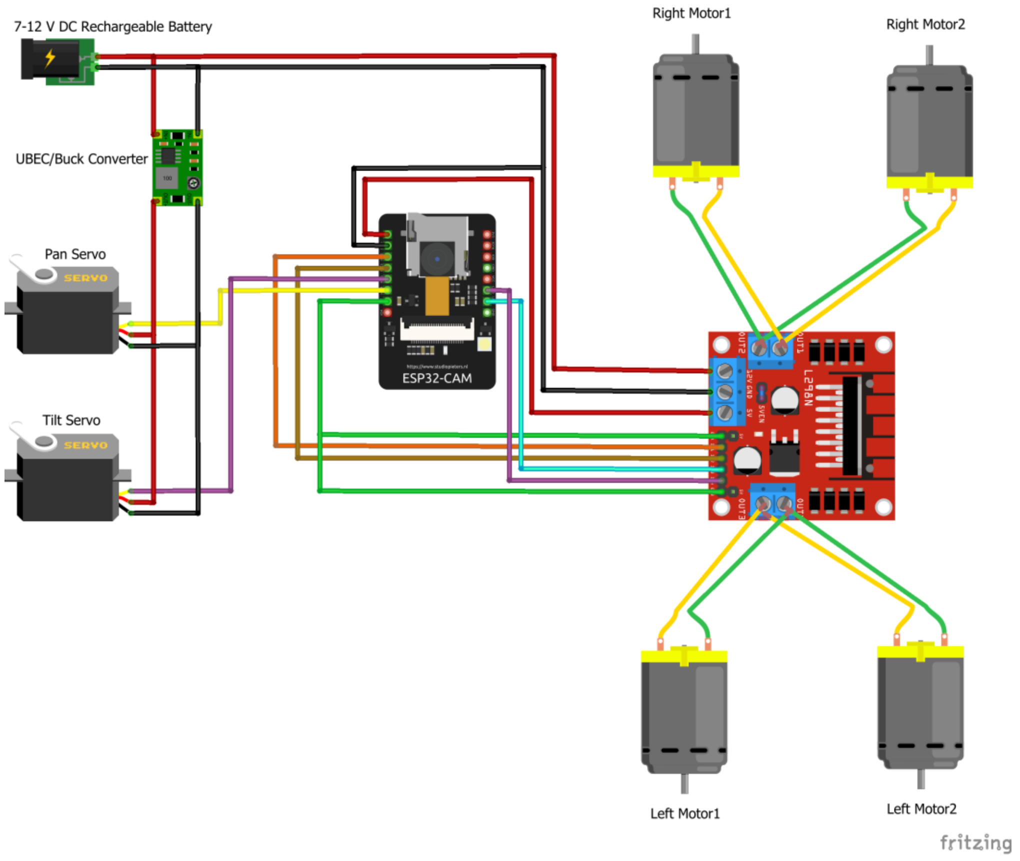

Fig 2.9 Circuit Diagram

## 2.5 – CODING PROCESS: -

### 2.5.1- INSTALLATIONS AND SETUP: -

1. Install the Arduino IDE.
2. Post installation, click on file then preferences, then in Additional Board Manager URLs add the ESP 32 board Link that is

   https://dl.espressif.com/dl/package_esp32_index.json
3. Click on tools, then Board manager, then search ESP32 and install it
4. Download Async TCP

   https://tinyurl.com/2k76xaee
5. download Async Webserver library

   https://tinyurl.com/25majfdh
6. Click on Sketch, then Include Library, then add zip file and add the downloaded files.

7. Click on sketch, then Library and search ESP Async Web server folder, then open SRC folder, then open Asyncwebserver.h file and change WS_MAX_QUEUED_MESSEGES from 32 to 1 and save it.
8. Install ESP 32 servo library by clicking on tools, then manage libraries and search ESP 32 servo then install it.

### 2.5.2- CODE: -

```
#include <AsyncEventSource.h>
#include <AsyncJson.h>
#include <AsyncWebSocket.h>
#include <AsyncWebSynchronization.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFSEditor.h>
#include <StringArray.h>
#include <WebAuthentication.h>
#include <WebHandlerImpl.h>
#include <WebResponseImpl.h>
#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     = "MyWiFiCar";
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:red;
    }
    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: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}
.slider:hover {
  opacity: 1;
}
.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
```

```
      border-radius: 50%;
      background: red;
      cursor: pointer;
    }
    .slider::-moz-range-thumb {
      width: 25px;
      height: 25px;
      border-radius: 50%;
      background: red;
      cursor: pointer;
    }
    </style>
  </head>
  <body class="noselect" align="center" style="background-color:white">
    <table id="mainTable" style="width:400px;margin:auto;table-layout:fixed"
CELLSPACING=10>
      <tr>
        <img id="cameraImage" src="" style="width:400px;height:250px"></td>
      </tr>
      <tr>
        <td></td>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","1")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8679;</span></td>
        <td></td>
      </tr>
      <tr>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","3")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8678;</span></td>
        <td class="button"></td>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","4")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8680;</span></td>
      </tr>
      <tr>
        <td></td>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","2")'
ontouchend='sendButtonInput("MoveCar","0")'><span 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(initCarInputWebSocket,
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").addEventListener("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());
}
```

# CHAPTER-3

# RESULTS AND DISCUSSION

We have successfully implemented the working of the wireless video surveillance robot controlled using android mobile device. The completed project is a small land-based robot that can be controlled from a web app using a Wi-Fi Direct connection. The app allows the user to send commands to the robot for moving forward and backwards, turning left and right, and stopping. Even the real time video feel is successfully achieved using the Wi-Fi technology on our designed webs application. Surveillance is needed in almost every field. It could be a great solution to various problems or situation where wireless Surveillance is needed our project has tremendous scope as it uses the latest technology in the market. The Surveillance robot can be controlled remotely using the web application; this gives it a huge scope for future application.

# CHAPTER-04

# CONCLUSIONS AND FUTURE SCOPE

## 4.1: - CONCLUSION: -

We have studied an efficient and effective way of monitoring and inspecting the defined environment. Previous generation of surveillance bot does not have any autonomous capabilities and does not have sensors like vision sensor for the safety of the robot. By having both manual control and autonomous capability the robot can be accessed for various security related applications. The user-friendly web interface makes it easier to operate from any part of the world. Further this robot can be altered with a specific set of sensors depending upon the terrain and function to be carried. The system has been tested and implemented practically. In future using port forwarding techniques it can be made a universally accessed robot. Later the robot can be accessed using a website by connecting it to an external web server. Here we conclude that the surveillance robot can be used for security purpose in industries, homes, in agriculture lands to monitor the fields and security purpose, in military to monitor the border areas, to surveillance the disaster affected areas where a person cannot reach. Here we can stream live video.

## 4.2: -FUTURE SCOPES:-

The project future scope has numerous openings that could be prosecuted for various future applications for monitoring and control, etc. In domestic applications such as home security can also be implemented.

- It could be used for surveillance in a designated area:
- It can work as a night vision surveillance robot:
- It could be used in multiple terrains:

## REFERENCES

[1]. I. R. Nourbakhsh & R. Siegwart, "Introduction to Autonomous Mobile Robots", MIT Press, 2004, pp.1-336

[2]. A. Arthi, G. Kalpana, M. Kavith & Jaya Surya "Smart Spy Surveillance Robotic System" International Journal of Engineering Research & Technology (IJERT), pp. 1-8

[3]. D. Chandrakala, K. A. Venkatesh, N. Balaji, B. M. bharathi, A. M. S. Hussain "Design Of Mobile Surveillance And Security Bot For Home Safety" European Journal of Molecular & Clinical Medicine, Vol. 7, (4), 2020, pp. 2586-2591

[4].K. Damodhar, B. Vanathi and K. Shanmugam, "A Surveillance Robot For Real Time Monitoring And Capturing Controlled Using Android Mobile" Middle-East Journal of Scientific Research, Vol. 24 (S1), 2016, pp. 155-166

[5].N.Pugazhenthi, K.VinuLakshmi, V.Preneeth, K.Shrivani "Design and Fabrication of Robot for Surveillance using Arduino" International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.8, (10), 2019 , pp. 3691-3693,

[6].S. Kumar, N. Kumari, P. Shukla, S. Srivastava & M. Aggarwal, "Wireless surveillance robot" International Journal of Advance Research, Ideas and Innovations in Technology" (Volume 4, Issue 2), pp. 1964-1966

[7].Anas f. Ahmed, Ruaa h. Ahmeed, Tamara z. Fadhil, "Design and Implementation Surveillance Robot Using ATmega328 Microcontroller" Iraqi Journal of Information Technology, Vol. 8 (4). 2018, pp.93-107.