**"ESP32-CAM ENHANCED SURVEILLANCE ROBOT SYSTEM USING CUSTOM MOBILE APPLICATION"**

A Dissertation submitted to **BHARATHIAR UNIVERSITY**, COIMBATORE

in partial fulfillment of the requirements for the award of the Degree of

**MASTER OF SCIENCE IN APPLIED ELECTRONICS**

Submitted by

**SANJAY KUMAR S (1P22EC003)**

Under the Guidance and supervision of

**Mr. S.ANANDA SARAVANAN M.Sc., M.Phil, NET.**

**Assistant Professor,**

Department of Electronics,

RVS College of Arts and Science,

Autonomous and Affiliated to Bharathiar University, Approved by AICTE

Re Accredited with ‘A+’ Grade by NAAC

Sulur, Coimbatore – 641 402



**DEPARTMENT OF ELECTRONICS**

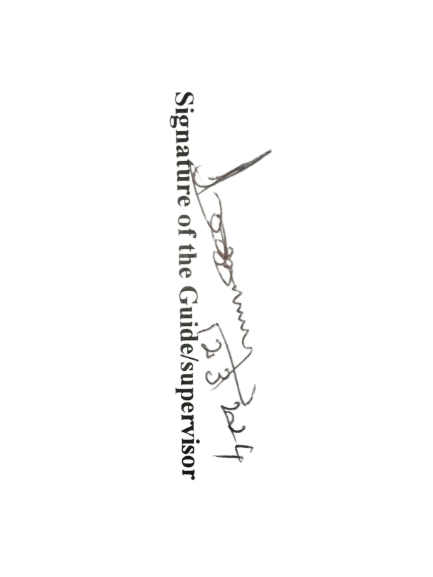
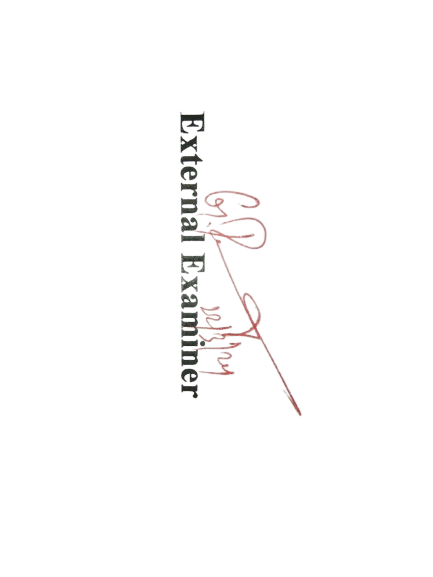
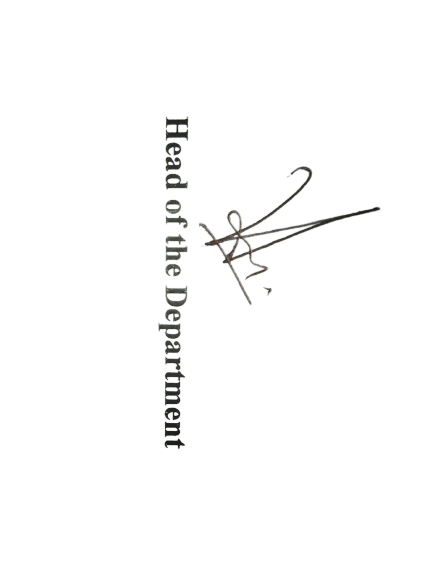
**RVS COLLEGE OF ARTS AND SCIENCE**

**(AUTONOMOUS) SULUR, COIMBATORE**

**MARCH – 2024**

**CERTIFICATE**

This is to certify that the dissertation entitled **“ESP-32 CAM ENHANCED SURVEILLANCE ROBOT SYSTEM USING CUSTOM MOBILE APPLICATION’’** submitted to the Bharathiar University in partial fulfillment of the requirements for the award of the Degree of **Master of Science** **in Applied Electronics** is a record of original research work done by **S.SANJAY KUMAR** during the period 2022 - 2024 of his study in the **Department of Electronics** at **RVS College of Arts and Science(Autonomous), Sulur, Coimbatore** under my supervision and guidance and the dissertation has not formed the basis for the award of any Degree / Diploma / Associate ship / Fellowship or other similar title to any candidate of any University.

 **  
**

**DECLARATION**

I, **S.SANJAY KUMAR** hereby declare that the dissertation, entitled **“ESP-32 CAM ENHANCED SURVEILLANCE ROBOT SYSTEM USING CUSTOM MOBILE APPLICATION’’**, submitted to the **Bharathiar University**, in partial fulfillment of the requirements for the award of the Degree of **Master of Science** **in Applied** **Electronics** is a record of original research work done by me during **2022 – 2024** under the Supervision and guidance of **Mr. S.ANANDA SARAVANAN M.Sc., M.Phil., NET., Assistant Professor, Department of Electronics, RVS College of Arts and Science, Sulur, Coimbatore – 641402** and it has not formed the basis for the award of any Degree / Diploma / Associate ship / Fellowship or other similar title to any candidate of any University.

**Date: 22/03/2024**  **Signature of the Candidate**

**Place: Sulur**  ­­­­­­ **SANJAY KUMAR.S**

**[IP22EC003]**

**ACKNOWLEDGEMENT**

I thank the Almighty and my parents for his marvelous blessings, support and all his guidance.

I thank our managing trustee **Mr. K.SENTHIL GANESH, RVS College of Arts and Science,** for helping attitude and constant encouragement which helped me a lot in successful completion of this project.

I immensely thank Principal **Prof. Dr. T.SIVAKUMAR M.Sc., M.Phil., Ph.D., RVS College of Arts and Science,**  for giving me a great opportunity to lug out this project.

I express my indebtedness to the Head, **Department of Electronics and communication, RVS College of Arts and Science, Mrs. Dr. P.VENKATESWARI M.SC., M.Phil., Ph.D.,** for his great support, encouragement and help throughout this project in a successful manner.

I also express my Sincere thanks to guide **Mr. S.ANANDA SARAVANAN M.Sc., M.Phil., NET.,** Assistant Professor, Department of Electronics, RVS College of Arts and Science, for his valuable technical guidance.

I also express my sincere thanks to all my Teachers and Lab Instructor from the **Department of Electronics, RVS College of Arts and Science** and I thank silent support, encouragement and personal advice of my **Parents** for their valuable guidance, interaction and timely support throughout the project.

**ABSTRACT**

The proposed system in this Project entitled **“ESP32 – Cam Enhanced Surveillance Robot System using Custom Mobile Application”** is a surveillance robot using ESP32-CAM is a system that utilizes the ESP32-CAM board and a robot chassis with 4 DC gear motors and 65mm robot wheels to create a mobile surveillance device. The ESP32-CAM is a low-cost development board that integrates a small camera module with OV2640 2 Mega Pixel and Wi-Fi connectivity. The robot chassis allows the device to move around and capture video in different locations with sufficient clarity. The system can be controlled through a web interface hosted on the ESP32-CAM board. The web interface allows the user to control the robot's movement in all 360 degrees, view live video streams, and take snapshots of the video feed. Additionally, the system can also be programmed to detect motion using computer vision algorithms, such as object detection and tracking, and send alerts to the user. The surveillance robot using ESP32-CAM has potential applications in home security, monitoring of remote locations, and industrial surveillance. With its low cost and easy-to-use interface, it provides a convenient solution for anyone who needs to monitor their surroundings remotely.

**CONTENTS**

**1. INTRODUCTION**

**2. HARDWARE DETAILS**

1. **BLOCK DIAGRAM**
2. **CIRCUIT DIAGRAM**
3. **COMPONENTS DETAILS**
4. **CIRCUIT OPERATION**

**3. SOFTWARE DETAILS**

1. **FLOW CHART**
2. **PROGRAM**
3. **MIT APP INVENTOR**

**4. CONCLUSION**

**5. FUTURE ENHANCEMENT**

**6. REFERENCES**

**7. PHOTOGRAPHY**

**INTRODUCTION**

In the rapidly advancing landscape of smart technologies, the fusion of robotics and surveillance systems has become pivotal for achieving enhanced security and real-time monitoring. This project, titled "ESP32-CAM Enhanced Surveillance Robot System Using Custom Mobile Application," endeavors to create an intelligent and versatile surveillance robot leveraging the power of ESP32-CAM technology. The ESP32-CAM module, equipped with a camera, serves as the eyes of the robot, capturing and transmitting visual data wirelessly. The heart of the system lies in the integration of a custom-designed mobile application, providing users with intuitive manual control and advanced features for seamless interaction with the robot. This project aims not only to deliver a sophisticated surveillance solution but also to explore the possibilities of autonomous navigation, real-time video streaming, and smart functionalities, marking a significant stride towards intelligent and accessible robotics in the realm of surveillance applications.

**HARDWARE DETAILS**

**I. BLOCK DIAGRAM**

**POWER SUPPLY**

**L298N MOTOR DRIVER**

**BUCK CONVERTER**

**GEAR MOTOR 1**

**GEAR MOTOR 2**

**ESP-32 CAM**

**GEAR MOTOR 3**

­­­

**GEAR MOTOR 4**

**MOBILE APPLICATION**

**BLOCK DIAGRAM EXPLANATION**

The block diagram comprises 6 blocks. They are Power Supply block with Buck Converter, Esp32 – Cam block, L298N Motor Driver block with 4 gear motor and Mobile Application block.

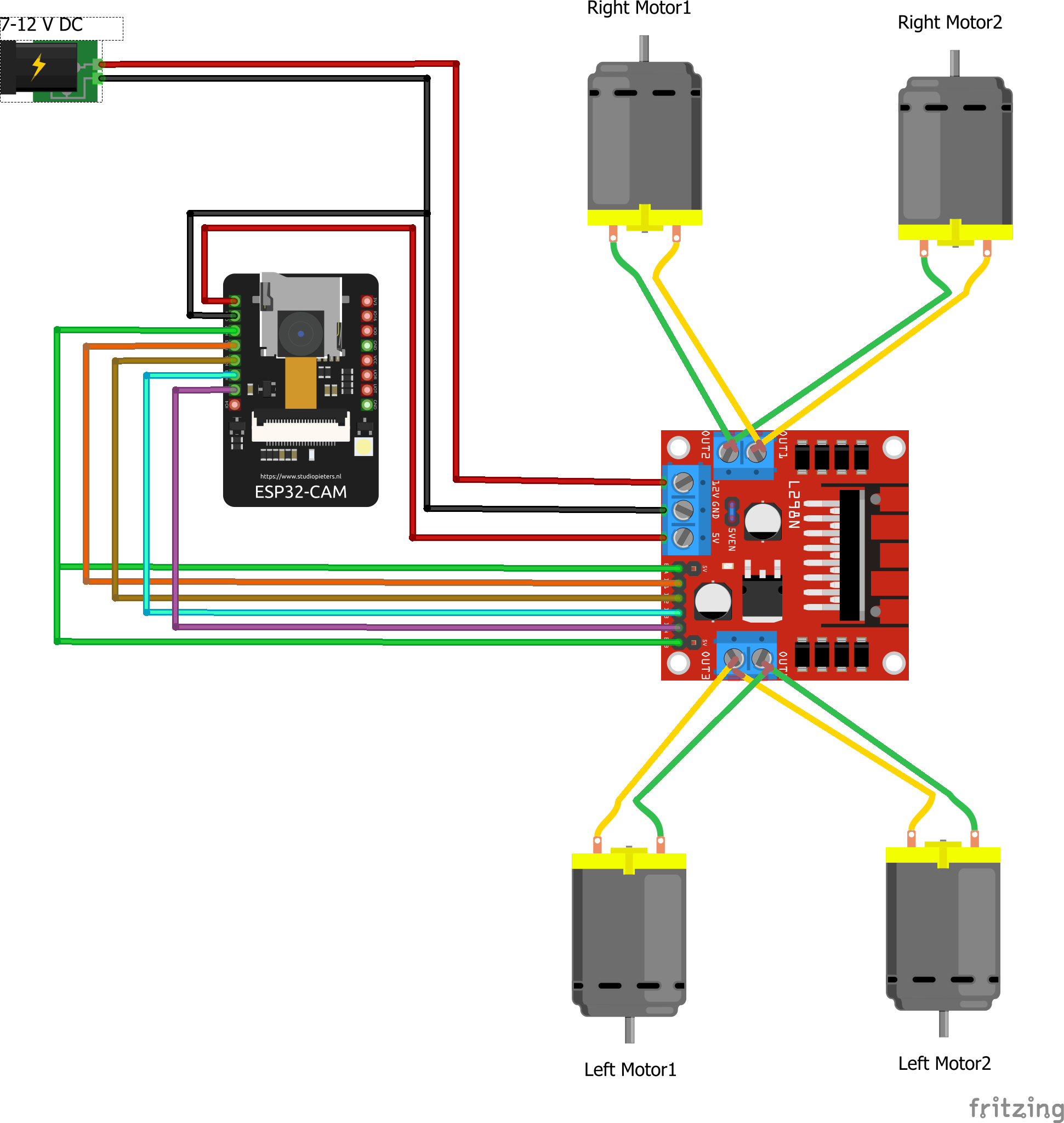
The Mobile Application block is a separate entity block which can send a command to the robot through Mobile Wifi and control based on command that given through the Mobile Application.

The ESP- 32 Cam is a Universal Micro-Controller with Camera module works based on the command and also control the movement of the DC gear motor.

The Power Supply block which supplies necessary power to the ESP32-Cam module as well as the DC gear motor through L298N Motor driver.

The ESP32-CAM Enhanced Surveillance Robot System block diagram outlines a cohesive architecture for seamless integration and operation. At the core of the system is the ESP32-CAM module, equipped with a camera for image and video capture. This module interfaces with a custom-designed PCB that acts as the central hub, facilitating connections and interactions between various components. The motorized chassis, comprising wheels and motor drivers, enables controlled movement of the robot. Optional sensors such as infrared and ultrasonic devices provide enhanced functionality for obstacle detection, while a gyroscope or accelerometer ensures stability during operation. The entire system is powered by a dedicated energy source, ensuring consistent and reliable performance. The custom mobile application, developed for iOS and Android platforms, forms a pivotal component in this project. It enables users to remotely control the robot, offering functionalities like live video streaming, camera manipulation (pan and tilt), and direct commands for robot movement. The integration of secure communication protocols enhances user data protection. This block diagram succinctly captures the interconnected hardware components, illustrating the streamlined communication and collaboration necessary for the ESP32-CAM Enhanced Surveillance Robot System to function as an efficient and user-friendly surveillance solution.

**II. CIRCUIT DIAGRAM**

****

**CIRCUIT DIAGRAM EXPLANATION**

A circuit diagram for the ESP32-CAM Enhanced Surveillance Robot System involves integrating various components for seamless functionality. The ESP32-CAM module serves as the brain of the system, incorporating an ESP32 microcontroller and a camera module. Connected to the ESP32 are motor drivers that control the movement of the robot, allowing it to navigate in different directions. The wheels are attached to DC motors, and their speed and direction are manipulated by the motor drivers.

Power distribution is crucial, and a battery pack supplies power to the ESP32-CAM module and the motors. Voltage regulators ensure a stable power supply to the components, preventing voltage fluctuations that could potentially damage the electronics. The ESP32-CAM communicates with the custom mobile application via Wi-Fi, providing a wireless interface for remote control and video streaming. The mobile application, designed specifically for this project, includes features for controlling the robot's movements, adjusting the camera angles, and receiving real-time video feeds. The system is designed with user-friendly interfaces, enabling seamless interaction between the user and the robot.

The entire circuit is carefully laid out to optimize performance, minimize interference, and ensure the reliability of the surveillance robot system. Proper grounding and signal routing are considered to reduce noise and enhance the overall stability of the circuit. The integration of sensors, motors, and communication modules requires meticulous attention to detail to create a well-functioning and robust surveillance robot system. This comprehensive circuit design forms the foundation for the successful implementation of the ESP32-CAM Enhanced Surveillance Robot System with a custom mobile application, providing users with an advanced and intuitive solution for surveillance applications.

**III. COMPONENT DESCRIPTION**

**3.1. ESP-32 CAM**

The ESP32-CAM shown in Fig.3.1.1 is a small-sized camera module that is based on the ESP 32 microcontroller and OV2640 sensor. It is capable of capturing images, streaming video, and performing various image processing tasks. The ESP32CAM module also features Wi-Fi and Bluetooth connectivity, making it ideal for IoT and surveillance applications. Here are some key features of the ESP32- CAM: ESP32 microcontroller, OV2640 2 MP camera sensor, Wi-Fi and Bluetooth connectivity, GPIO pins for interfacing with other devices, Micro SD card slot for storage, 5 V DC power supply. To use the ESP32-CAM module, it is needed to have some knowledge of programming, especially with the Arduino IDE, as it is commonly used to program the ESP32-CAM. It is also needed to have some basic electronics knowledge, such as how to connect wires and components to the GPIO pins of the module. There are many application based projects that can be done with the ESP32CAM, such as building a smart security camera, a remote-controlled car with a camera, or even a face recognition system. With its small size and powerful features, the ESP32-CAM is a versatile and powerful tool for makers and hobbyists.

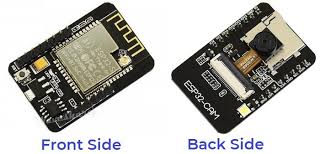
****

Fig.3.1.1 ESP32 cam

**3.2. MOTOR DRIVER**

The L298N is a popular motor driver IC (integrated circuit) that can control the direction and speed of DC motors and stepper motors. It can handle up to 2 amps of continuous current per channel, and has a wide input voltage range of 5V to 35V. The L298N has two H-bridge circuits, which are used to control the direction of the motor. Each H-bridge consists of four transistors that can be controlled independently. By switching these transistors on and off in the correct sequence, the motor can be driven in either direction.To use the L298N, it is required to connect it to the microcontroller or other control circuitry. There are several pins on the L298N shown in Fig 3.2.1 that it is needed to connect to: ENA and ENB: These are the enable pins for channels A and B, respectively. The digital output pins has been connected to the microcontroller to enable or disable the motor channels. IN 1, IN2, IN3, and IN4: These are the control pins for the H-bridge circuits. By setting the correct combination of high and low signals on these pins, you can control the direction and speed of the motor. OUTI, OUT2, OUT3, and OUT4: These are the output pins that connect to the motor. To connect the L298N to a motor, you need to connect the motor to the OUT pins, and also connect the motor power supply to the L298N's V+ and GND pins. It's important to note that the L298N can get quite hot when driving high-current motors, so you should use a heat sink to dissipate the heat. You should also be careful not to exceed the maximum current ratings of the L298N, or you may damage the IC.

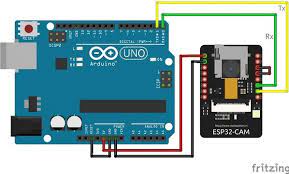
****

Fig.3.2.1 L298N Motor driver

**3.3 Upload Program to ESP32 – CAM with Arduino Uno**

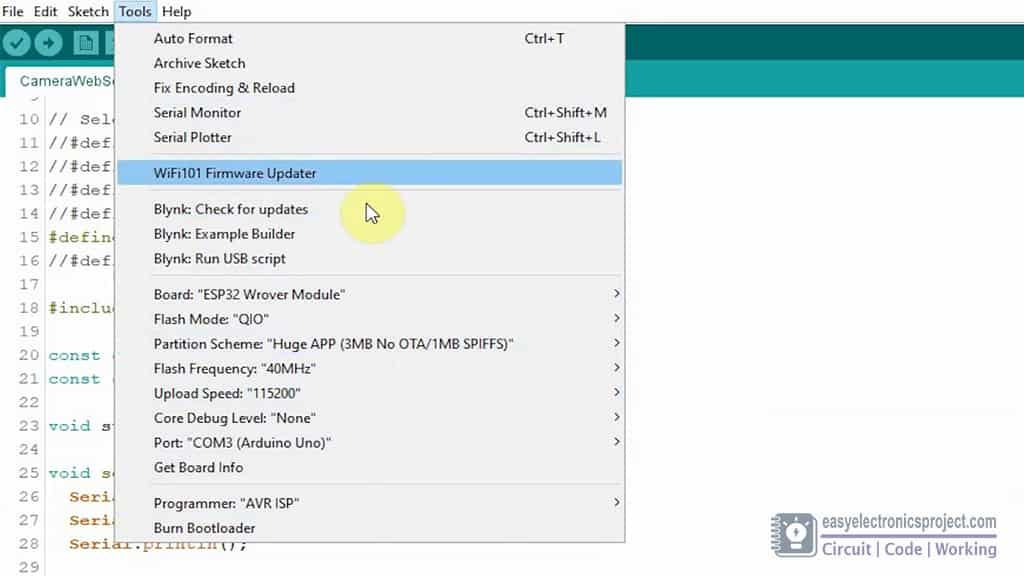
ESP32CAM Arduino UNO Connection:

|  |  |
| --- | --- |
| **ESP32CAM** | **Arduino UNO** |
| U0T | TX |
| U0R | RX |
| 5V | 5V |
| GND | GND |
| GPIO-0 –> GND | Reset –> GND |

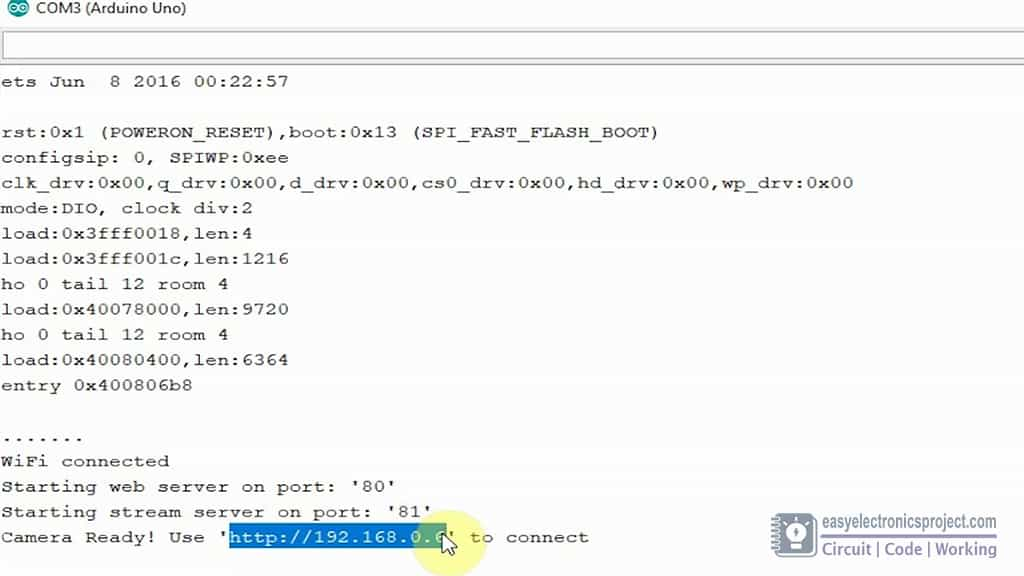


**Before uploading the code to ESP32-CAM** module, please check the following setting:

* Update the**Preferences** –> **Additional boards Manager URLs**: https://dl.espressif.com/dl/package\_esp32\_index.json, http://arduino.esp8266.com/stable/package\_esp8266com\_index.json
* **Board Settings:**
  + Board: “ESP32 Wrover Module”
  + Flash Mode: “QIO”
  + Partition Scheme: “Hue APP (3MB No OTA/1MB SPIFFS)”
  + Flash Frequency: “40MHz”
  + Upload Speed: “115200”
  + Core Debug Level: “None”
  + Programmer: “AVR ISP”
* COM Port: Depends On the System
* GPIO 0 must be connected to GND pin while uploading the sketch
* After connecting GPIO 0 to GND pin, press ESP32 CAM on-board RESET button to put the board in the flashing mode.

****

After uploading the code disconnect the GPIO-0 pin from GN.

****

Now to get the IP address of ESP32 camera module:

* Open Serial Monitor
* Set the Baud rate to 115200
* Press the ESP32 CAM on-board RESET button

Now the IP address can be copied and pasted it to any browser to start the video streaming.

**3.4 DC Gearbox Motor**

Perhaps you've been assembling a new robot friend, adding a computer for a brain and other fun personality touches. Now the time has come to let it leave the nest and fly on its own wings– err, wheels! These durable also affordable plastic gearbox motors (also known as 'TT' motors) are an easy, low-cost way to get your projects moving. This is a TT DC Gearbox Motor with a gear ratio of 1:48, and it comes with 2 x 200mm wires with breadboard-friendly 0.1" male connectors. Perfect for plugging into a breadboard or terminal blocks. You can power these motors with 3VDC up to 6VDC, they'll of course go a little faster at the higher voltages. We grabbed one motor and found these stats when running it from a bench-top supply

• At 3VDC we measured 150mA @ 120 RPM no-load, and 1.1 Amps when stalled

• At 4.5VDC we measured 155mA @ 185 RPM no-load, and 1.2 Amps when stalled

• At 6VDC we measured 160mA @ 250 RPM no-load, and 1.5 Amps when stalled

Note that these are very basic motors, and have no built-in encoders, speed control or positional feedback. Voltage goes in, rotation goes out! There will be variation from motor to motor, so a separate feedback system is required if you need precision movement. You cannot drive these directly from a microcontroller; a high-current motor driver is required! We recommend our DRV8833 motor driver for these motors, as it works well down to 3V and can be set up with current limiting since the stall current on these can get high. The TB6612 can also be used, it's on our shields and wings, but you'll need to supply at least 4.5V - which is what you'll likely want to run these motors at anyhow! We have a range of wheels, add-ons and accessories for these motors so you can bring out your bot just the way you like.

**TECHNICAL DETAILS**

• Rated Voltage: 3~6V

• Continuous No-Load Current: 150mA +/- 10%

• Min. Operating Speed (3V): 90+/- 10% RPM

• Min. Operating Speed (6V): 200+/- 10% RPM

• Torque: 0.15Nm ~0.60Nm

• Stall Torque (6V): 0.8kg.cm

• Gear Ratio: 1:48

• Body Dimensions: 70 x 22 x 18mm

• Wires Length: 200mm

• & 28 AWG Weight: 30.6g

• Product Weight: 30.6g / 1.1oz

****

**3.5 Robot Wheels**

****

BO motor wheels are used to convert the rotational motion of the motor into linear motion. They are typically made of rubber or plastic and come in a variety of sizes and styles.

**3.6 Buck Converter**

****

LM2596 is a voltage regulator mainly used to step down the voltage or to drive load under 3A. It is also known as DC-to-DC power converter or buck converter which is used to step down the voltage from its input supply to the output load. The current goes up during this voltage step down process. LM2596 comes with a remarkable load and line regulation. It is available in both versions: fixed output voltage version with 3.3V, 5V, 12V, and customized output version where you can choose the output as per your requirement. This regulator is incorporated with a fixed-frequency oscillator and an internal frequency compensation method. Frequency compensation is applied by adjusting both phase and gain characteristics of the open-loop output to avoid oscillation and vibration in the circuit.  This is achieved with the help of resistance-capacitance networks. A minimum number of external components are required for this regulator that works at a fixed frequency of 150 kHz.

**3.6.1 LM2596 Pin Out**

****

The LM2596 comes with total five pins as follow: **Vin** = I = this is the input supply pin interlaced with the input bypass capacitor to provide the switching current and to reduce voltage transients. **Output** = O = this is the internal switch with voltage switches between (Vin – Vsat) and -0.5V. It comes with a duty cycle of Vout /Vin. The PCB copper area attached to this pin is used to minimize the coupling. **Ground** = this pin is connected to the ground. **Feedback** = I = this pin identifies the regulated output voltage for the feedback loop. **ON/OFF** = I = this pin is used to shut down the voltage regulator circuit with input supply current reducing to 80uA. When the voltage on this pin goes below the threshold voltage of 1.3V, it turns on the buck converter. And when the voltage goes above the 1.3V, it turns off the converter. You can get rid of this shutdown feature by attaching the pin to the ground or leaving it open. In both cases, the regulator remains ON.

**3.6.2 LM2596 Features**

* Fixed versions i.e. 3.3-V, 5-V, 12-V, and customizable output versions
* Customizable output version with voltage range: 1.2-V to 37-V ±4% maximum over load and line conditions
* Available in two packages including TO-263 and TO-220 packages.
* Can drive load under 3A.
* 40 V is the input voltage range
* 4 external components are needed
* Remarkable load and line regulations
* Internal oscillator with a fixed frequency of 150 kHz
* TTL shutdown capability
* Comes with low power standby mode, commonly 80 µA
* High efficient and readily available
* Protection against thermal shutdown and current

**3.6.3 LM2596 Power Rating**

| **Absolute Maximum Ratings LM2596** | | | | |
| --- | --- | --- | --- | --- |
| No. | Rating | Value | Unit |  |
| 1 | Maximum Supply Voltage | 45 | V |  |
| 2 | SD/SS Input Voltage | 6 | V |  |
| 3 | Delay Pin Voltage | 1.5 | V |  |
| 4 | Flag Pin Voltage | 45 | V |  |
| 5 | Feedback Pin Voltage | 25 | V |  |
| 6 | Output Voltage to Ground | -1 | V |  |
| 7 | Storage Temperature | -65 to 150 | C |  |

* While working with this component, make sure stresses don’t exceed the absolute maximum ratings, else they can permanently damage the component.
* Plus, if stresses are applied for more than the required time, they can affect device reliability.

**3.6.4 LM2596 Applications**

* Used to step down voltage
* Can drive load under 3A
* Provide remarkable load and line regulation

**3.7 Cooling Fan**

Cooling fans are essential in electronic circuits to dissipate heat generated by components such as microprocessors and transistors during operation. By maintaining optimal temperatures, these fans prevent overheating, ensuring efficient performance and extending the lifespan of electronic devices. They play a crucial role in preventing thermal throttling, improving reliability, and enabling compact designs in confined spaces. In summary, cooling fans are critical for managing heat and maintaining the proper functioning of electronic circuits.



In my project, I strategically incorporated 5cm x 5cm cooling fans to address the heat generated by the motor driver. Recognizing the significance of temperature regulation in optimizing the performance and longevity of electronic components, I selected cooling fans that fit the specific dimensions of 5cm x 5cm to ensure a precise and efficient cooling solution. These fans effectively dissipate the excess heat generated during the motor driver's operation, preventing potential overheating issues and enhancing the overall reliability of the system. The deliberate choice of cooling fans tailored to the dimensions of the motor driver not only demonstrates a meticulous approach to thermal management but also contributes to the successful implementation of a robust and temperature-controlled electronic setup.

**IV.CIRCUIT OPERATION**

The ESP32-CAM Enhanced Surveillance Robot System is designed as a comprehensive mobile surveillance solution. At its core is the ESP32-CAM module, integrating a camera for capturing images and video. This module is mounted on a motorized chassis featuring wheels for controlled movement. The hardware setup includes a custom PCB to organize and connect various components, ensuring efficient power distribution and motor control. The system's capabilities extend with optional sensors such as infrared and ultrasonic devices for obstacle detection and a gyroscope or accelerometer for stability. The entire robotic platform is powered by a dedicated energy source. A crucial element of this project is the development of a custom mobile application 0020 for Android platforms, facilitating seamless remote control and surveillance. This application offers functionalities such as live video streaming, camera manipulation with pan and tilt options, and direct commands for robot movement. The hardware design prioritizes a user-friendly interface, secure communication protocols, and effective power management to enhance the robot's operational endurance. The amalgamation of these hardware components aims to deliver an advanced and adaptable surveillance robot system controlled through a dedicated mobile application.

**SOFTWARE DETAILS**

**I. FLOW CHART**

**ESP32 – Cam Enhanced Surveillance Robot System**

**using Custom Mobile Application**

**START**

**Initialize ESP32-Cam**

**Wait a While**

**ESP32 - Cam module connected to the network**

**Get an IP address to broadcast**

**Capture a Video**

**Wait a While**

**Encrypt and send to Web Server connected with the Mobile Application**

**Web Server decrypts all the payload and combines them in order**

**Send decrypted whole buffer to the client**

**FINISH**

**Yes** **No**

**Yes**

**No**

**II.PROGRAM**

#include "esp\_camera.h"

#include <Arduino.h>

#include <WiFi.h>

#include <AsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <iostream>

#include <sstream>

struct MOTOR\_PINS

{

int pinEn;

int pinIN1;

int pinIN2;

};

std::vector<MOTOR\_PINS> motorPins =

{

{12, 13, 15}, //RIGHT\_MOTOR Pins (EnA, IN1, IN2)

{12, 14, 2}, //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 = "sanjay";

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:40px;

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">

<!--h2 style="color: teal;text-align:center;">Wi-Fi Camera &#128663; Control</h2-->

<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>

</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)

{

var speedButton = document.getElementById("Speed");

sendButtonInput("Speed", speedButton.value);

var lightButton = document.getElementById("Light");

sendButtonInput("Light", lightButton.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 RIGHT:

rotateMotor(RIGHT\_MOTOR, FORWARD);

rotateMotor(LEFT\_MOTOR, BACKWARD);

break;

case LEFT:

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);

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);

}

}

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\_0;

config.ledc\_timer = LEDC\_TIMER\_0;

config.pin\_d0 = Y2\_GPIO\_NUM;

config.pin\_d1 = Y3\_GPIO\_NUM;

config.pin\_d2 = Y4\_GPIO\_NUM;

config.pin\_d3 = Y5\_GPIO\_NUM;

config.pin\_d4 = Y6\_GPIO\_NUM;

config.pin\_d5 = Y7\_GPIO\_NUM;

config.pin\_d6 = Y8\_GPIO\_NUM;

config.pin\_d7 = Y9\_GPIO\_NUM;

config.pin\_xclk = XCLK\_GPIO\_NUM;

config.pin\_pclk = PCLK\_GPIO\_NUM;

config.pin\_vsync = VSYNC\_GPIO\_NUM;

config.pin\_href = HREF\_GPIO\_NUM;

config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

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

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

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

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

config.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()

{

//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());

}

**II.MIT APP INVENTER**

Creating a Surveillance Robot Car using MIT App Inventor involves several steps to design the mobile app for remote control. Here's a brief overview of the process using the IP address 192.168.4.1 that I used in my project:

**1. Design Interface:**

- Opening MIT App Inventor and create a new project.

- Designing the user interface with buttons for controlling the robot car (forward, backward, left, right, stop and voice control).

- Including additional components like sliders or joysticks for smoother (speed and light).

**2. Connect to Wi-Fi Module:**

- Used the "Web" component in MIT App Inventor to establish a connection to the Wi-Fi module on my robot car.

- Setting the Web component's `URL` property to the IP address of my robot car (192.168.4.1 in this case) and the `Port` property to the port number your robot is listening on (commonly 80).

**3. Control Commands:**

- Implement blocks to send control commands to the robot car when the user interacts with the interface components.

- For each control button, use the Web component's `Get` method to send specific commands (e.g., "forward," "backward," "left," "right", “stop” and “voice”) to the specified IP address and port.

**4. Receive Feedback:**

- My robot car has camera, I use the Web component to send requests for camera to process a video.

- Implement blocks to process the received data and update the app interface accordingly.

**5. Handling Responses:**

- I used the Web component's `GotText` and `GotFile` events to handle responses from the robot car.

- Process feedback from the robot (camera stream) and update the app interface dynamically.

**6. Testing:**

- Connected my mobile device to the same Wi-Fi network as the robot car.

- And installed the MIT AI2 Companion app on my mobile and use it to test the app in real-time.

**APPLICATION SCREEN**

**Screen 1**



In the MIT App Inventor development environment, a 'Start' button has been integrated into Screen 1, featuring a programmed event handler. Uponactivation of the 'Start' button, the application seamlessly transitions to Screen 2, providing a user-friendly and responsive navigation experience.

**Screen 2**

****

Screen 2 is equipped with a control button interface designed to manage the movements of a robot car. Additionally, the screen facilitates real-time video streaming through the incorporation of an IP address, enhancing the overall functionality of the application by providing users with dynamic control over both the robotic platform and live video feed.

**CONCLUSION**

The "ESP32-CAM Enhanced Surveillance Robot System Using Custom Mobile Application" project has successfully realized a dynamic fusion of cutting-edge technologies to create an intelligent and user-friendly surveillance solution. The integration of the ESP32-CAM module as the visual core of the robot, coupled with a meticulously crafted custom mobile application, has empowered users with seamless manual control and advanced functionalities. The project not only demonstrates the potential for real-time monitoring and enhanced security through the ESP32-CAM's imaging capabilities but also opens doors to future developments in autonomous navigation and smart features. As technology continues to evolve, this project serves as a testament to the possibilities of creating accessible and sophisticated robotics solutions that cater to the growing demand for intelligent surveillance systems. With its blend of hardware and software innovation, the ESP32-CAM Enhanced Surveillance Robot System stands as a testament to the project's success in pushing the boundaries of surveillance technology and its potential applications in various real-world scenarios.

**FUTURE ENHANCEMENT**

**1. Autonomous Navigation:**

- Implement computer vision algorithms to enable autonomous navigation for the robot. Use the ESP32-CAM's camera to recognize and avoid obstacles or follow predefined paths.

**2. Object Recognition and Tracking:**

- Integrate machine learning models for object recognition and tracking. This can help the robot identify and follow specific objects or individuals.

**3. Night Vision:**

- Integrate infrared sensors or an additional camera module with night vision capabilities to enhance the robot's functionality in low-light or dark environments.

**4. Communication Upgrades:**

- Enhance communication capabilities by integrating long-range communication modules, like LoRa or GSM, to extend the robot's range and accessibility.

**5. AI-Based Behavior:**

- Implement artificial intelligence algorithms to give the robot more sophisticated behavior, such as recognizing specific people or learning and adapting to its environment.

**6. Environmental Sensors:**

- Include additional sensors like temperature, humidity, or gas sensors to provide environmental data along with surveillance capabilities.

**REFERENCES**

1. A. Arthi., et al. “Smart Spy Surveillance Robot System” International Journal Research & Technology (IJERT) ISSN:2278- 0181, Special Issue-2018.

2. “Design and Implementation of SPY Robot”, IJARIIE-ISSN(0)-4396, Vol- 5 Issue-5 2019.

3. Liu, C., et al. "Design and implementation of a wireless surveillance robot based on ESP32- CAM." IEEE Access, vol. 9, 2021, pp. 14582-14591.

4. Kumar, S., et al. "Design and implementation of a low-cost surveillance robot for indoor applications." International Journal of Computer Applications, vol. 180, no. 18, 2018, pp. 9-14.

5. Wang, Y., et al. "Design and implementation of a wireless robot for environmental monitoring." Sensors, vol. 21, no. 1, 2021, p. 36.

6. Nagendra, N., et al. "Design and implementation of a low-cost surveillance robot using Raspberry Pi." International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 12, 2019, pp. 46-50.

7. Noh, J., et al. "Development of a surveillance robot with obstacle avoidance capability." International Journal of Advanced Robotic Systems, vol. 15, no. 1, 2018, pp. 1- 8.

8. Singh, A., et al. "Smart security system using ESP32 and Raspberry Pi." International Journal of Engineering and Advanced Technology, vol. 9, no. 1, 2019, pp. 35-38.

9. Waghmare, D. "Implementation of a low-cost security robot using ESP32- CAM." International Journal of Engineering Research and SURVEILLANCE ROBOT USING ESP32 CAM Section A-Research paper Eur. Chem. Bull. 2023,12(Special Issue 1, Part-B), 3148-3157 3157 Applications, vol. 10, no. 2, 2020, pp. 34-38.

10. Zhao, S., et al. "Design and implementation of a remote monitoring system for intelligent robots." Journal of Physics: Conference Series, vol. 1761, 2021, p. 012026.

11. Chen, W., et al. "Design of a mobile robot for environmental monitoring." IEEE Transactions on Instrumentation and Measurement, vol. 67, no. 8, 2018, pp. 1899-1908.

12. Kambli, R., et al. "Implementation of a wireless

13. surveillance robot using ESP32- CAM." International Journal of Advanced Research in Computer Engineering & Technology, vol. 7, no. 5, 2018, pp. 93-96.

14. Zou, J., et al. "A novel design of wireless monitoring robot." Journal of Physics: Conference Series, vol. 1749, 2021, p. 012015.

15. Yarlagadda, P., et al. "Design and implementation of a wireless robot for industrial monitoring." Journal of Robotics and Mechatronics, vol. 33, no. 2, 2021, pp. 399-404.

**PHOTOGRAPHY**

**Front View:**

****

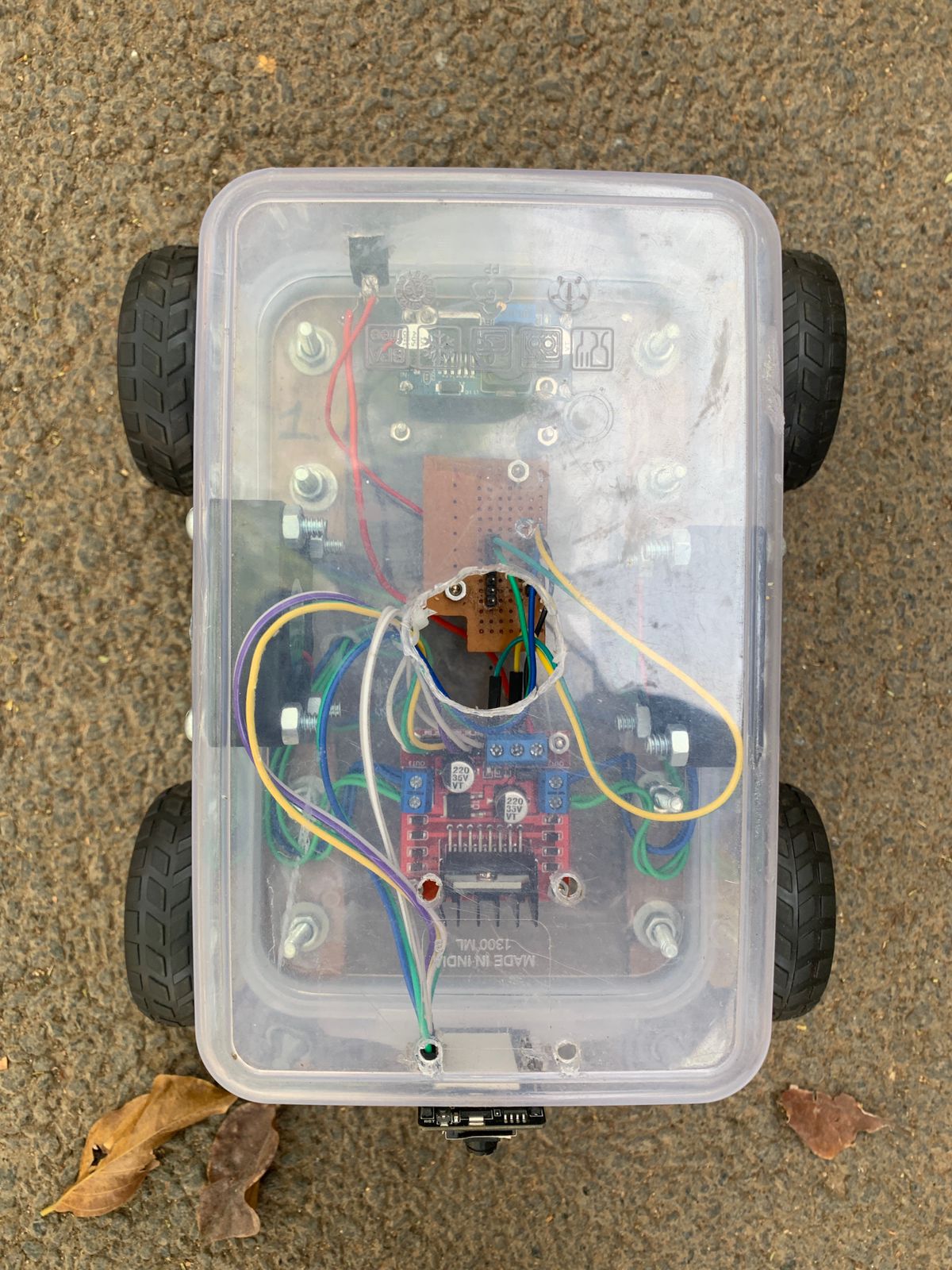
**Back View:**

****

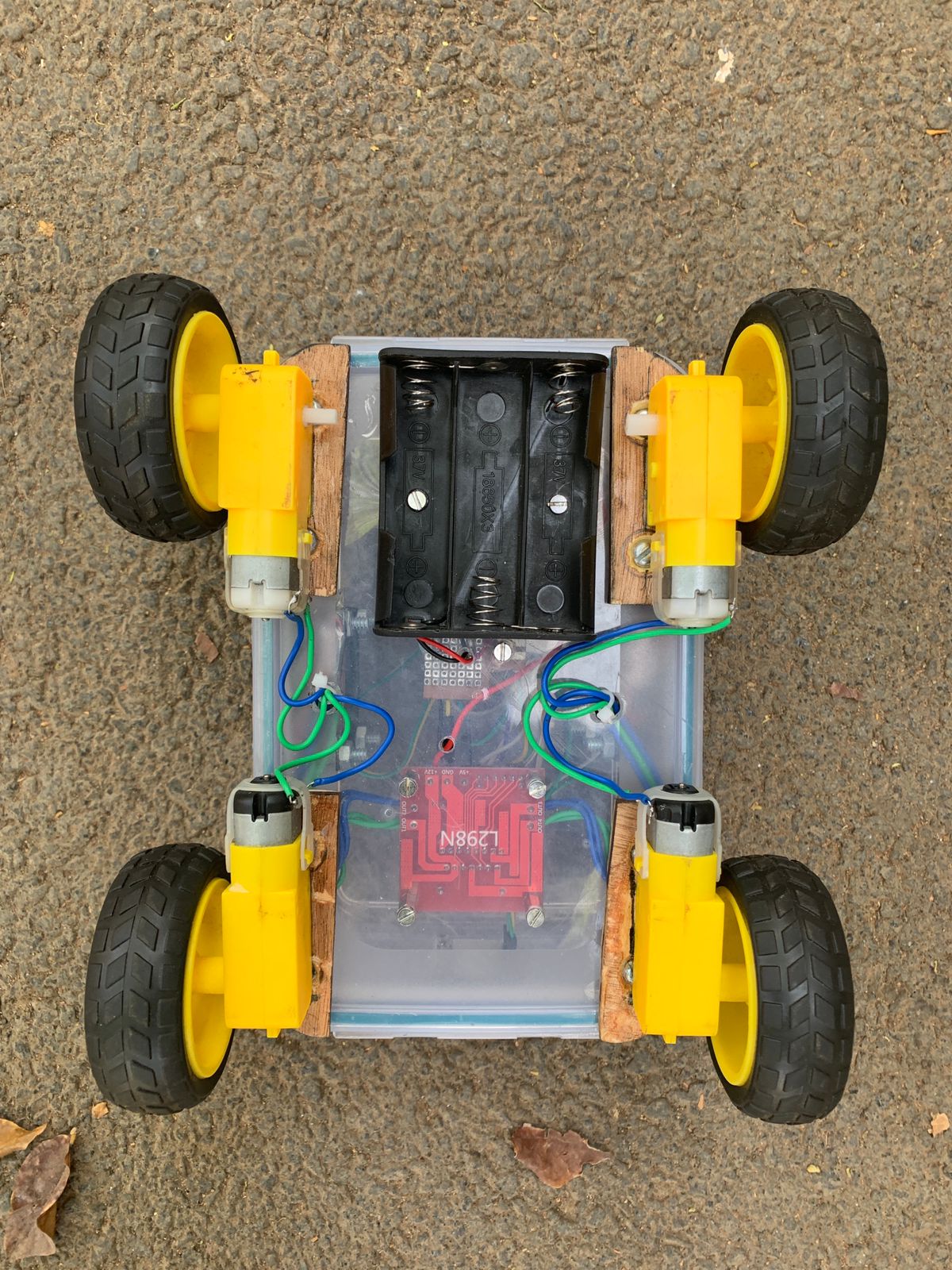
**Side View:**

****

**Top View:**

****

**Bottom View:**

****