Index

Abstract

The main idea of the project is to create a surveillance car and monitor it using local up addressing. The surveillance car is equipped with the ESP32 cam allowing it to do real time video streaming and remote control is done according to the code that we have written in HTML and that code is compiled using Arduino. Then the code is dumped into the Cam board using Arduino Uno (we use it as a programmer module instead of FTDI connecter). The remote control of the car enables flexible manoeuvring and navigation in diverse environments. It is also integrated by IOT and so can be accessed when connected to the same Wi-Fi using the URL we can monitor the same. These include features such as encrypted video transmission, motion detection and user authentication.

List of figures

Components list

- ESP32 CAM
- Arduino UNO
- L298N motor driver
- Buck converter 12 to 5V
- Chassis and Wheels
- Dc motors
- Servo motors SG90
- Pan tilt
- Power supply

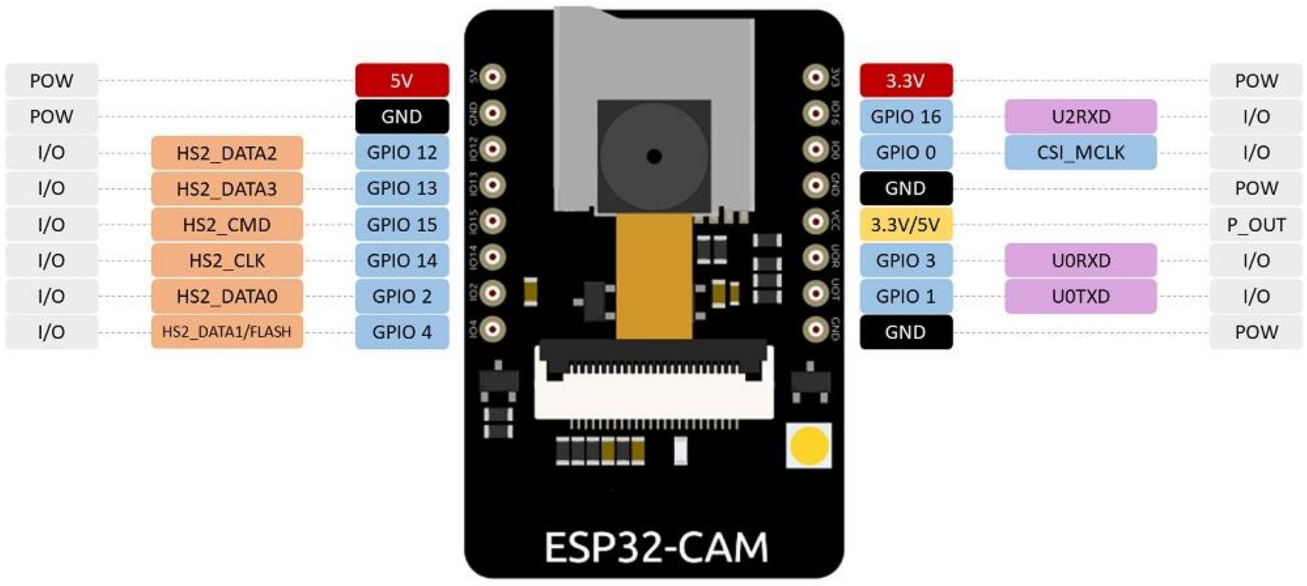
Description of components

- ➢ **ESP32 CAM MODULE**

The ESP32-CAM is a small-size, low-power camera module based on ESP32. It comes with an OV2640 camera and provides an onboard TF card slot. This board has 4MB PSRAM which is used for buffering images from the camera into video streaming or other tasks and allows you to use higher quality in your pictures without crashing the ESP32. It also comes with an onboard LED for flash and several GPIOs to connect peripherals. The ESP32-CAM AI-Thinker comes with 10 exposed GPIOs.

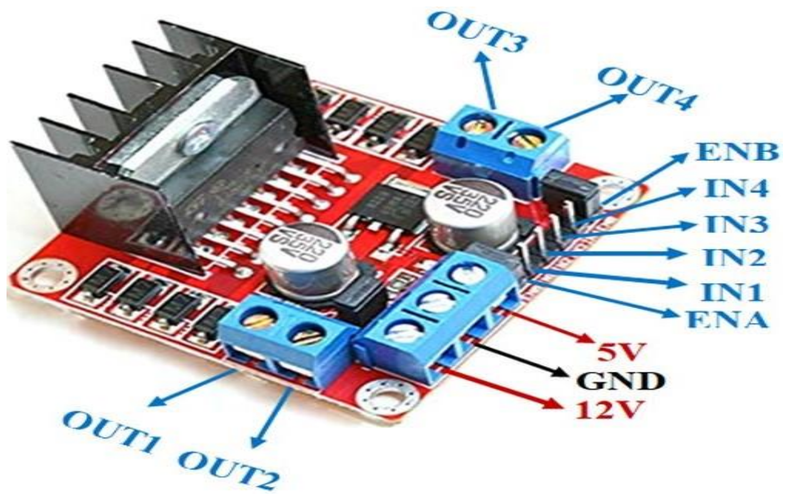
- *Features and specifications-*

i. Onboard ESP32-S module, supports Wi-Fi + Bluetooth
ii. OV2640 camera with flash which has 2mp resolution
iii. RAM; Internal 512KB + External 4M PSRAM
iv. Antenna; Onboard PCB antenna
v. Supports Wi-Fi video monitoring and Wi-Fi image upload
vi. Supports multi sleep modes, deep sleep current as low as 6mA
vii. Control interface is accessible via pin-header, easy to be integrated and embedded into user products
viii. It has UART baud rate of 115200 bps



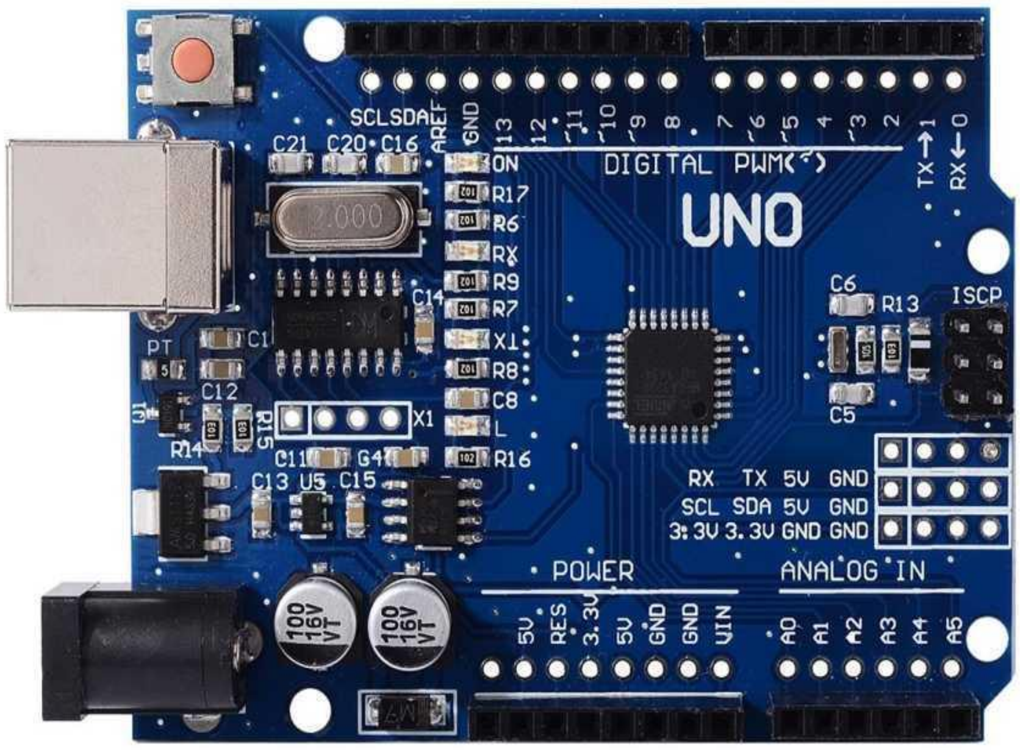
➢ <u>L298N MOTOR DRIVER</u>

This L298N Based Motor Driver Module – 2A is a high power motor driver perfect for driving DC Motors and Stepper Motors. It uses the popular L298 motor driver IC and has the onboard 5V regulator which it can supply to an external circuit. It can control up to 4 DC motors, or 2 DC motors with directional and speed control.
This L298N Based Motor Driver Module – 2A is perfect for robotics and mechatronics projects and perfect for controlling motors from microcontrollers, switches, relays, etc. Perfect for driving DC and Stepper motors for micro mouse, line-following robots, robot arms, etc.



➢ <u>**ARDUINO UNO**</u>

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery. The Arduino Uno has a number of facilities for communicating with a computer, another Arduino Uno board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer.



➢ **BUCK CONVERTER 12 TO 5V**

12V DC input to 5V 2A DC output step down convertor circuit board uses an integrally moulded power inductor and synchronous rectifier control chip, smaller and more efficient. This Module can be used as a DIY mobile power, monitor power supply, power buggies, camera power supply, car power, communications equipment supply, various right size and weight for demanding applications (such as aviation models, etc.)

➢ **DC MOTORS**

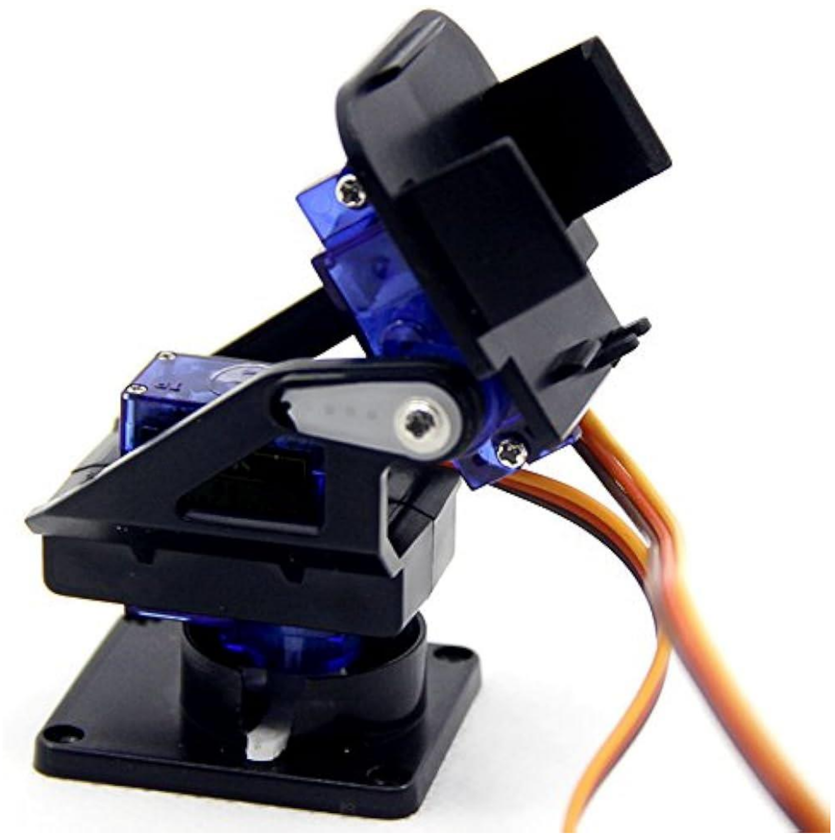
A DC motor is an electrical motor that uses direct current (DC) to produce mechanical force. The most common types rely on magnetic forces produced by currents in the coils. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

➢ SERVO MOTORS SG90

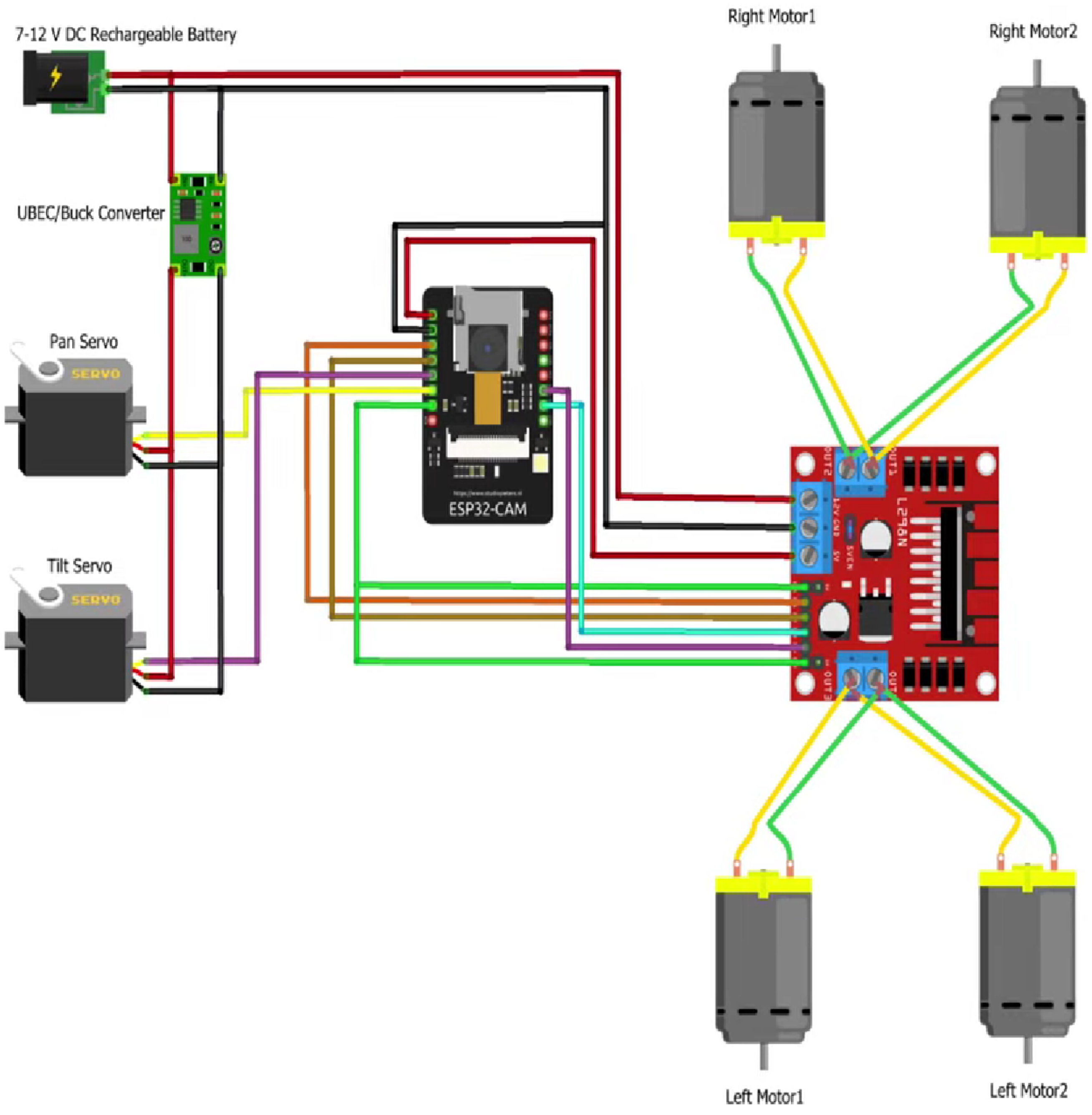
SG90 is a popular micro servo motor commonly used in hobbyist and DIY projects. It is a small, low-cost servo motor that can rotate 180 degrees with a maximum torque of 1.8 kg-cm. It operates at 4.8-6V and has a weight of approximately 9 grams, making it ideal for small-scale robotics and model control applications.

➢ CHASSIS AND WHEELS

➢ **PAN TILT**



Connection diagram



Implementation of the project

- The implementation of the project involves hardware assembly, software development and testing.
- The ESP32 cam will be integrated with motor drivers, power supply and a chassis.
- The software will include video streaming, motor control and security for the same.
- It also supports remote access via web interface or we can also use a dedicated mobile application. Users will be able to control the car, view live video feeds and receive alerts when connected to the same Wi-Fi.

Working of the project

First step is to check the working of the ESP32CAM board. We verify it by using Arduino IDE and as a result we get the “camera ok” message and an URL is generated. After that, we dump this code into the board using a programmer (here Arduino uno instead of an FTDI connector).

By writing a code in HTML, we define the movements of the wheels according to directions (forward, backward, left, right). Then we dump this code and integrate all the hardware part and provide power supply for the same.

After integrating all the components on the chassis and wheels by interconnecting them according to our purpose, we fit the ESP board onto the pan tilt to get rotation for our surveillance purpose.

After programming the board, motors and providing power supply for the required components, we test it.

For power supply, we have used 12V batteries for the motors and used Arduino UNO to power the CAM board by using a lightweight power bank of specifications:-
output=5V, 10000mah.

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     = "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(initCameraWebSock
et, 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(initCarInputWebSoc
ket, 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").addEve
ntListener("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());
```

```
}
```

Challenges faced

The main challenge that we faced was on addressing the power problem. By using the batteries alone, we couldn't power both CAM board and motors simultaneously.

There wasn't sufficient current to work all of the components altogether so we had to explore different solutions for the same.

We finally ended up using a lightweight power bank of 10000mah which gives output of 5V and connected it to Arduino UNO to help power the CAM board and used batteries for the motors.

A more permanent solution would be to use a Lipo battery.

Applications

The surveillance car project has diverse applications including;

- Home security,
- Industrial monitoring and
- Educational purposes.

Its adaptability and reliability makes it a valuable asset for various security and surveillance needs.

The use of ESP32 cam board in this project offers enhanced monitoring and cost effectiveness making it an ideal solution for diverse surveillance applications.

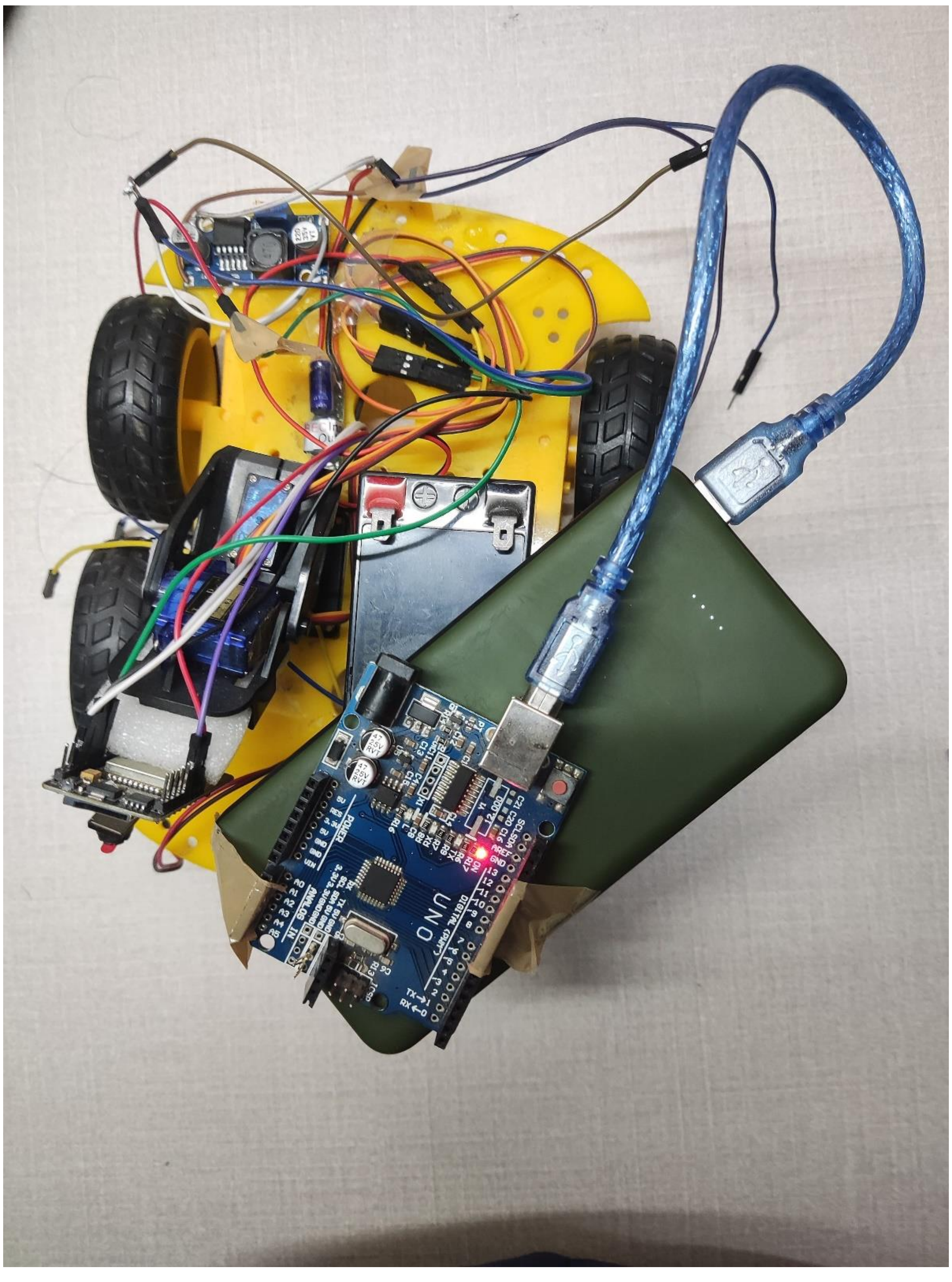
Future enhancements

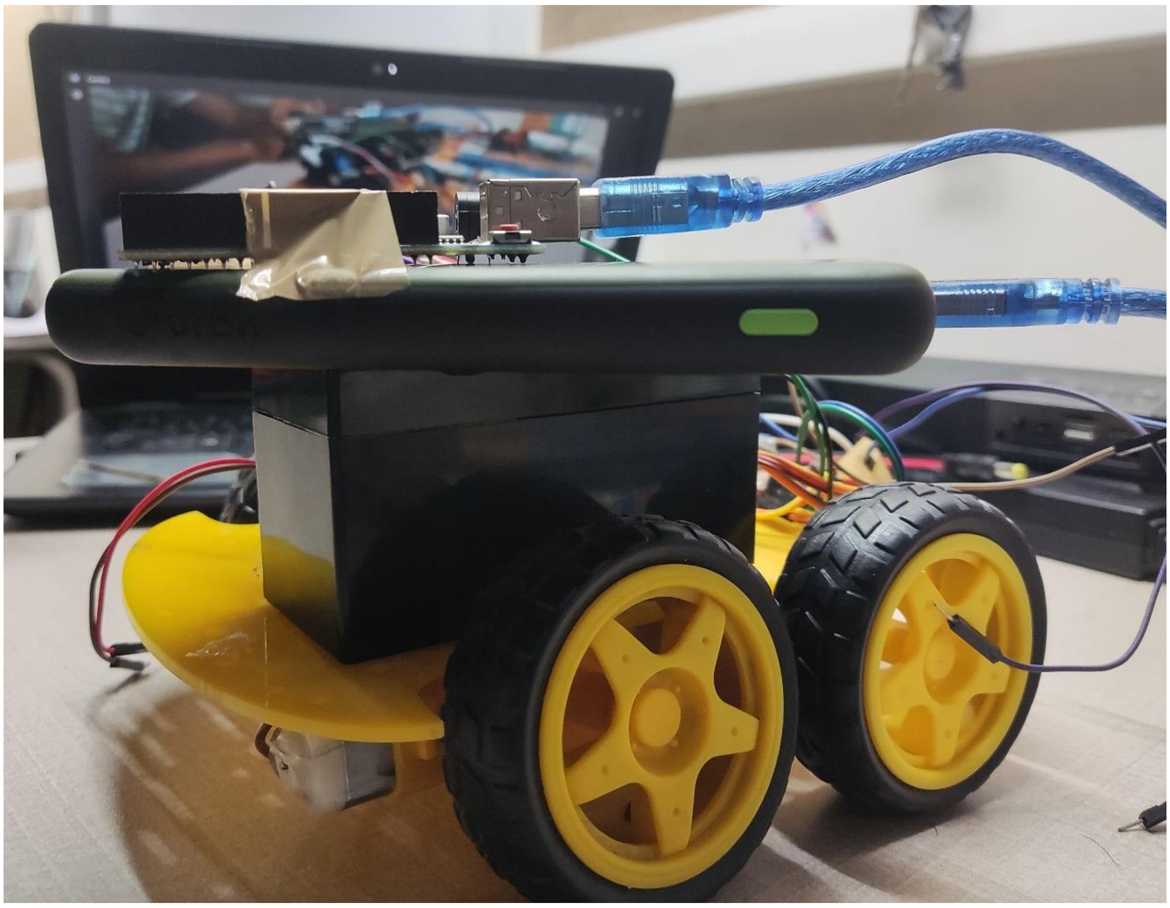
- Future advancements for this project include *integrating AI based object recognition, detection and monitoring* or *colour detection and monitoring,* and *enhancing the autonomous navigation capabilities* and *expanding the surveillance range.*

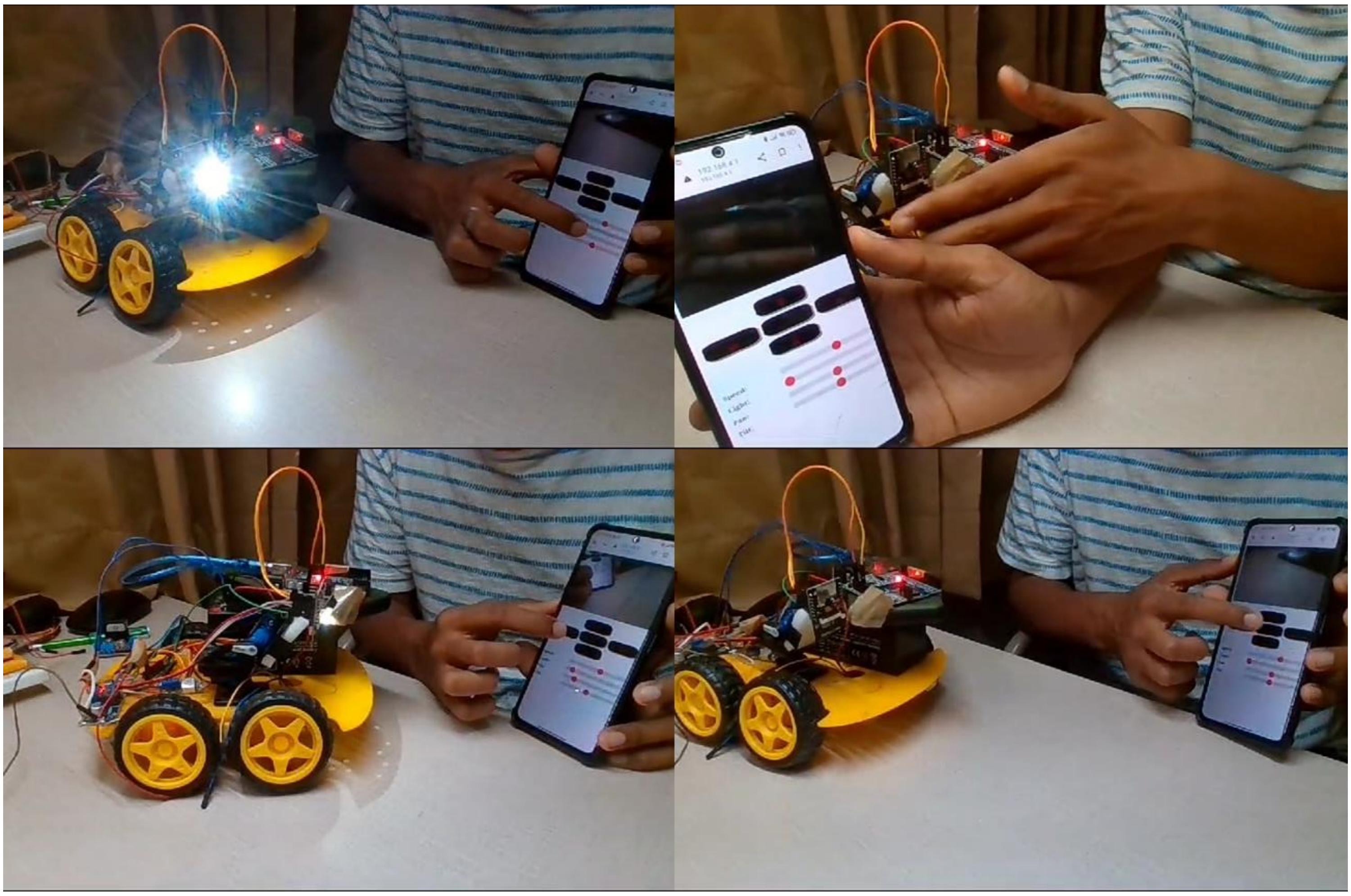
  These enhancements will further elevate the security measures and operational efficiency of the system.

Conclusion and result

- In conclusion this surveillance car project presents a comprehensive solution for enhancing security measures. Its robust features, secure connectivity and potential for future advancements makes it a promising choice for effective surveillance applications.






Speed:
Light:
Pan:
Tilt:

References

- https://www.youtube.com/watch?v=tyY7AN132Xs
- https://www.youtube.com/watch?v=Rm_R0MAmHHA
- https://randomnerdtutorials.com/esp32-cam-pan-and-tilt-2-axis/
- https://www.youtube.com/watch?v=HfQ7lhhgDOk
- https://www.instructables.com/ESP32-CAM-Surveillance-Car-1/
- https://docs.arduino.cc/hardware/uno-rev3#:~:text=Arduino%20UNO%20is%20a%20microcontroller,header%20and%20a%20reset%20button.
- https://en.wikipedia.org/wiki/Arduino_Uno
- https://www.espressif.com/en/products/socs/esp32#:~:text=ESP32%20is%20highly%2Dintegrated%20with,Hybrid%20Wi%2DFi%20%26%20Bluetooth%20Chip
- https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/#:~:text=The%20L298N%20is%20a%20dual,and%20explain%20how%20it%20works.
- https://randomnerdtutorials.com/esp32-cam-car-robot-web-server/

thank you