CHAPTER-1

Introduction

Surveillance is a real time collection and analysis of data that is timely distributes the information to the operator. Surveillance is the task of monitoring the set of conditions. This generally occurs in a military scenario where surveillance war areas, adversary territory. Human surveillance is carried by experienced work forces in close sensitive areas so as to continually monitor for changes. Whereas there is always added risks of losing work force in the time of getting caught by the adversary. With advanced technology in pasted years, there it is possibility to monitor areas of importance remotely by the use of robots instead of human. Apart from the given advantages of not losing any work forces, physical and elegant robots can be used detect subtle elements that are not conspicuous to people. A surveillance robot is a partially automated machine that works as per instructed by operator and move to destination, Streaming video or pictures which can then analyzed by the operator. Surveillance is a crucial task, we cannot put someone life to risk, instead of that we can use this kind of robots which do not need sleep, they don't get hungry, they don't have emotions, they are just stick to their duties and follow the orders. Nothing can be more important than human life. We can use this manpower in other tasks. Here we use an android device to control the robot. And in future instead of security guards we will use this kind of robots.

This robot consists of DC motor, motor drive(L293D), camera module(ESP32) used for controlling of the movement of robot and streaming of live video and GPS, PIR sensor, Arduino Nano, GSM SIM 800C used for intruder monitoring. The robot can be either operated manually. User end communicates with the robot by implementing the concept of Internet of Things. Cameras are generally used for indoor security. In indoor security system multiple cameras are mounted on wall with different angles to track object. These types of systems need a computer or a laptop for monitoring. Nowadays, most of the system uses a mobile robot with a camera for surveillance. The camera mounted on the robot can move to different locations. These types of robots are more flexible than the fixed cameras. With the development in wireless communication and internet, the videos captured by robot can be seen remotely on computer, laptop and android phones.

The Multipurpose Surveillance Robot is an advanced robotic system designed to perform various surveillance tasks with efficiency and precision. It combines cutting-edge technology and versatility to provide a comprehensive solution for monitoring and securing different environments. Equipped with state-of-the-art sensors, cameras, and communication modules, this robot can autonomously navigate through challenging terrains, including indoor and outdoor spaces. Its compact and agile design allows it to maneuver in tight spaces, making it suitable for deployment in various scenarios. The robot's surveillance capabilities include real-time video streaming, audio monitoring, and sensor data collection. Its high-resolution cameras provide clear visuals, enabling remote operators to observe and analyze the surroundings from a safe distance. The audio monitoring feature allows for capturing and analyzing sounds, enhancing situational awareness.

Additionally, the Multipurpose Surveillance Robot supports intelligent algorithms for object detection, facial recognition, and anomaly detection. These features enable the robot to identify specific objects or individuals of interest and alert the operators in real-time. The robot's communication modules facilitate seamless connectivity, enabling remote control and data transmission over wireless networks. This allows operators to control the robot's movements, access its surveillance data, and receive instant notifications, making it an effective tool for both on-site and remote monitoring operations.

CHAPTER-2

Literature Survey

2.1 General Introduction

A literature survey is a fundamental practice, to understand develop the idea. A literature survey not only summarizes the knowledge of the area or field but also gives us an idea of what had to be done. In the following section, we discuss the issues and works related to Multi-Purpose Surveillance Robot.

2.2 Literature Survey

2.2.1 MASS (Military Assistance and Surveillance System):

It was proposed by P. Raja, SwapnilBagwari et al (2018). They used different type of sensor to monitor the soldier such as their location, health conditions, surroundings, sending data to base station, etc. being wearable device it monitors the pulse rate as well as send the respective data to the base station and by using GPS module the location can also be monitored by military base station.

2.2.2 Processing a Spy Robot for a Surveillance System:

It was proposed by T. Veeramanikandasamy et al (2017). The System using Internet Protocol of Arduino UNO an Arduino operating system-based spy robot platform with remote monitoring and control algorithm through Internet of Things (IOT). The information regarding the detection of living objects by PIR sensor is sent to the users through the web server and pi camera capture the moving object which is posted inside the webpage simultaneously.

2.2.3 Mobile autonomous robot, called MARVIN:

It was proposed by Luca plaza, Andria Claudi et al (2019). This paper proposes mobile autonomous robot used in video surveillance applications. The main goal of the robot is to detect human faces in the monitored environment, and to autonomously move to keep a face in exact center of the frame.

.

2.2.4 Design and Development of Surveillance Robot:

It was proposed by K. Krishnaveni, P Gopi Krishna, K.Shivani, B. Ravi teja et al (2019). The multi-functional surveillance robot is designed to deliver an affordable degree of risk saver with out causing human loss, performance and ease, presenting each person with a streamlined user revel in in the war grounds. The multifunctional robotic is aimed in imparting the tracking along with vision, movement, and fireplace with restrained setup.

2.2.5 Robot IoT based Multipurpose Surveillance Robot:

It was proposed by Ningaraju A.M2, Sudarshana Chakravarthy, Suraj Sharma, Harish, Pawan Bharadwaj et al (2022). This paper presents a multipurpose surveillance robot that can be used in military applications for both spying and detecting landmines. We can also use this robot for detecting hazardous gases during rescue operations where humans cannot get inside due to tight spaces.

Summary

The above-discussed literature surveys presented us with a broad perspective on how to tackle what we have aimed and gave us the different possible approaches to follow to build our project i.e., Multi-purpose Surveillance Robot.

CHAPTER-3

Problem Statement and Objectives

3.1 Problem Statement

The currently accessible robotic technologies like locomotive robots and self-guided vehicles like patrol robots, pathfinder robots, industrial carrying robots, etc., make human life comfortable for which they were developed. But the current analytical robots that are available have bulky hardware and controllers mounted onto them, which makes them expensive and hectic to troubleshoot. Hence, to relieve humans from such burden new methods has to be devised and better progression by making the robot light weight and cost-effective.

3.2 Objectives

1. **Video surveillance**: The robot can capture video footage of the intruder providing a clear visual intrusion.
2. **Detection and alerting**: The surveillance robot can be equipped with sensors such as cameras, motion detectors, or other sensors to detect the presence of an intruder. When an intruder is detected, the robot can immediately send an alert to security personnel or law enforcement, providing real-time information about the location.

CHAPTER-4

Methodology

4.1 Block Diagram

The project aims at developing a surveillance robot capable of performing remote operations through wifi and also intruder monitoring and alerting the user with sms. for surveillance operations ESP32-S Wifi module with OV2640 camera module is used, the module hosts its own web server through which the user connects through from his or her phone or laptop, and connects to the desired port, upon connecting the user can visually look for the video from the camera that is being streamed, also the GPIO of the module is connected to the L298N motor driver pop, hence enabling the maneuverability of the robot to the desired place, through the arrow mark controls present in the interface, the other features include varying of motor speed, and lights for visibility can be controlled through a slider in the interface. The below figure 4.1 show the block diagram of surveillance and control of the robot.

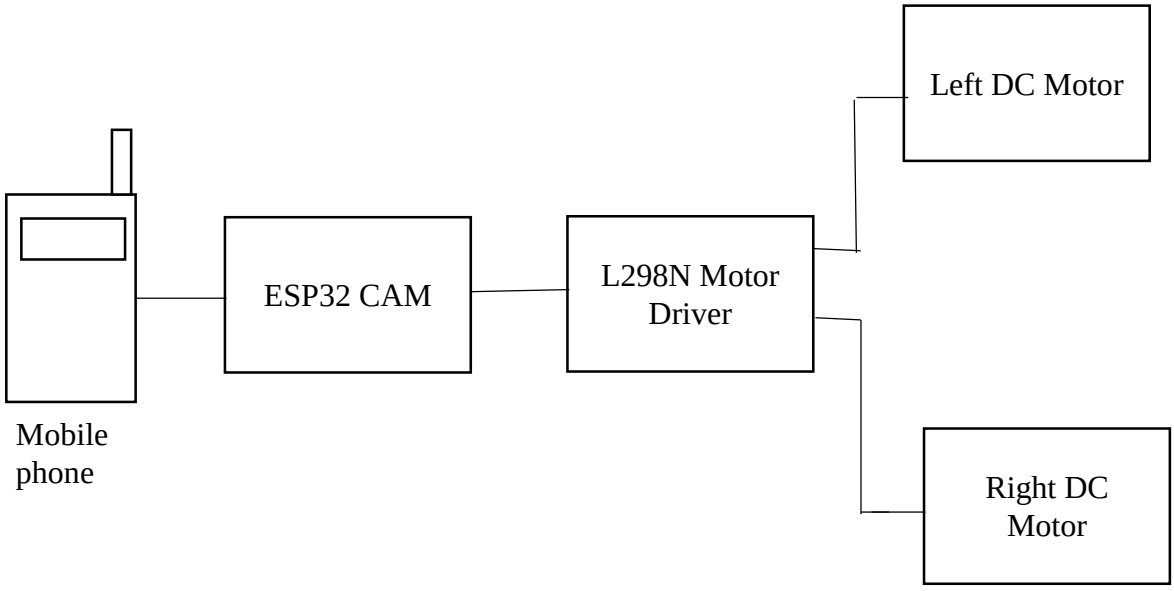

Figure 4.1: Block Diagram of Surveillance and Control

If the user wishes to use the robot for intruder monitoring operations, the robot's intruder mode switch is turned on, upon this a arduino nano microcontroller starts to read the PIR sensors output, if it detects the presence of any moving or living being, a sms alert is send to the programmed number, with the location coordinate read from Ublox Neo 6M GNSS receiver module. The programmed user gets a link on which if clicked is redirected to Google maps, alerting the user to take necessary action. The below figure 4.2 shows the block diagram of Intruder Monitoring.

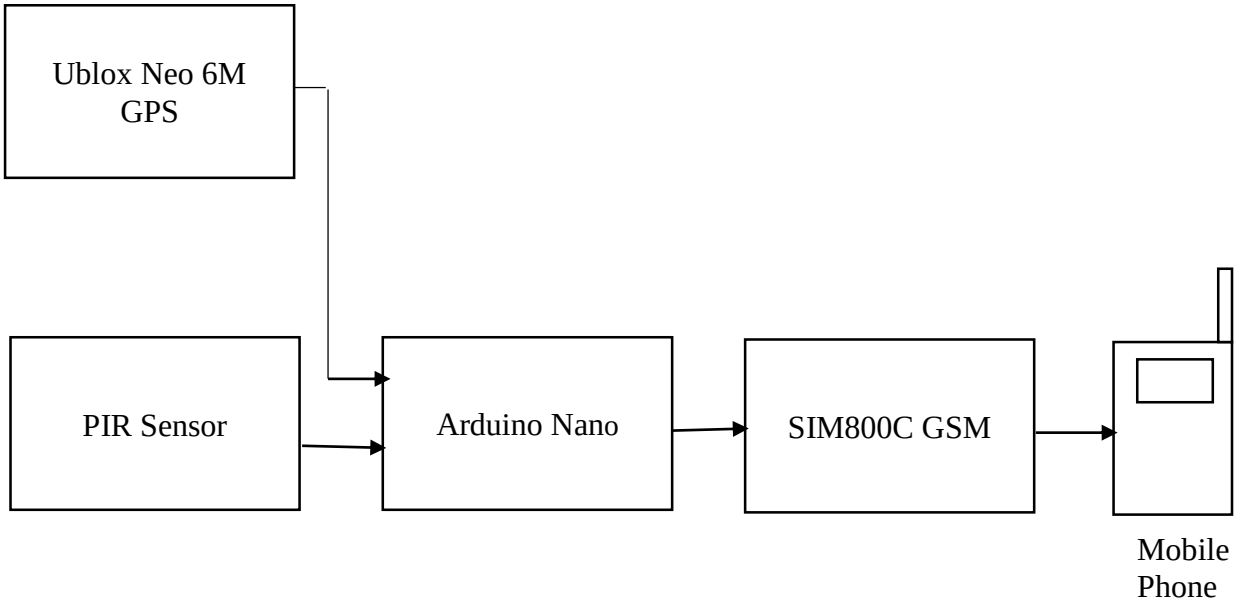

Figure 4.2: Block diagram of Intruder Monitoring

4.2 Flowchart of Intruder Monitoring

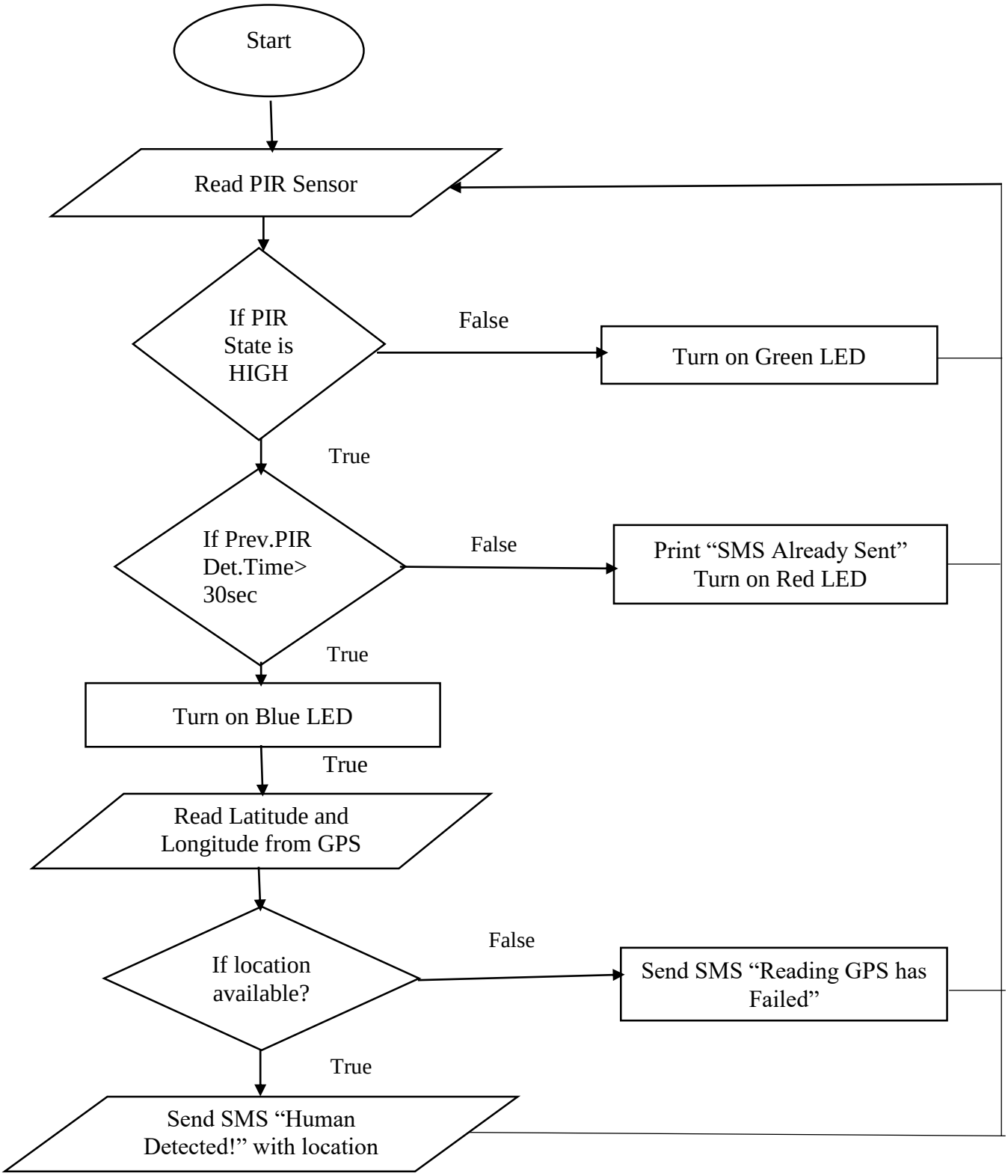

Figure 4.3: Flowchart of Intruder Monitoring

Figure 4.3 represents the flowchart of intruder monitoring. First the system is started, then the PIR sensor is read then it goes to the first condition “If PIR state is HIGH”, if this condition is true then goes to next condition which is “If Prev.PIR.Det time > 30sec”, else green LED is turned ON. If second condition is true, then blue LED is turned ON else “SMS Already Sent “is printed and red LED is turned ON. After the previous condition is true Latitude and Longitude is read from the GPS module the goes to the next condition “If location is available?”, if this condition is true then SMS “Human Detected!” with location link is sent to the user else SMS “Reading GPS has Failed” message is sent.

4.3 System Design and Pin Description

In this project, we developed a surveillance robot with intruder monitoring capabilities using an ESP32-CAM board and an Arduino Nano. The robot is powered by a 3-cell lion battery, which is stepped down to 5V using a DC-DC converter to power the controllers and sensors. The robot is equipped with four motors for movement and a PIR motion sensor to detect human presence. The video is streamed via IP network, and the robot can be remotely controlled using a webpage. The robot also has an RGB LED module that turns on different colors depending on the presence of an intruder and sends SMS alerts using a GSM SIM800S module.

Surveillance and security have become increasingly important issues in today's world, and finding an effective solution can be a challenge. Traditional surveillance methods are often costly and require extensive infrastructure to implement. Additionally, many security systems rely on human intervention, which can be both costly and prone to errors. This is where the surveillance robot with intruder monitoring comes into play. It offers a cost-effective and reliable solution to surveillance and security problems by using cutting-edge technology to detect and deter intruders.

The surveillance robot is a mobile device that can be remotely controlled via a webpage. It is equipped with an ESP32-CAM board, which allows for video streaming through an IP network. Additionally, it features four motors that enable it to move around and navigate its surroundings. The intruder monitoring system is made up of an Arduino Nano, a PIR sensor, and a GSM SIM800S module. The PIR sensor detects human presence, and the GSM module sends alerts to the user via SMS. The system also includes an RGB LED module that changes color based on the status of the PIR sensor.

With the surveillance robot, users can remotely monitor their surroundings and take action in the event of an intrusion. It is an ideal solution for home security, office surveillance, and other scenarios where remote monitoring is necessary. The system is powered by a 12V Li-ion battery that is stepped down to 5V using a DC-DC converter, making it highly portable and convenient to use. The following figures 4.4, 4.5, 4.6 shows Pin description Power Supply, Surveillance and Control and Intruder Monitoring respectively.

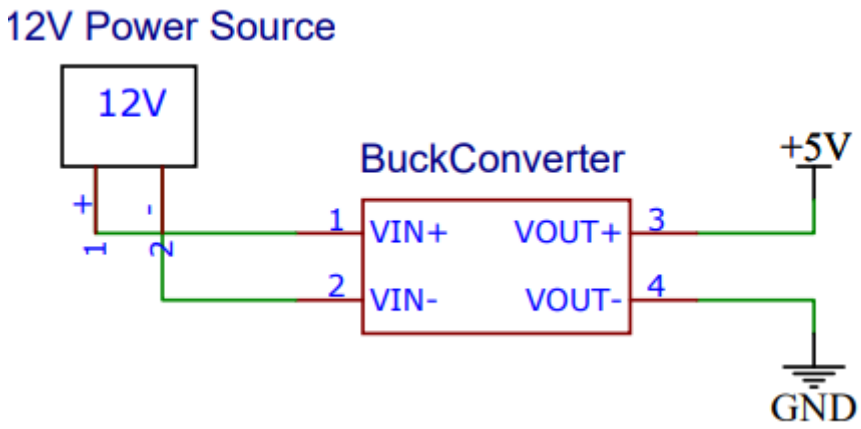

Figure 4.4: Power supply

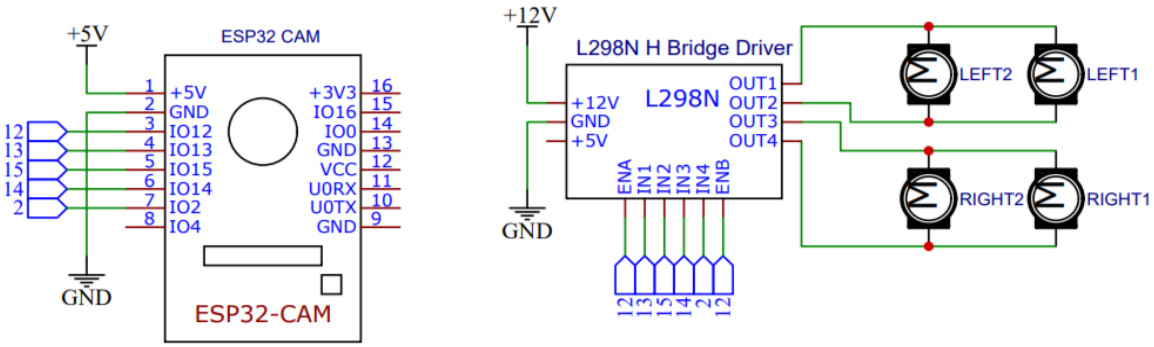

Figure 4.5: ESP32 CAM and L298N Pin descriptions

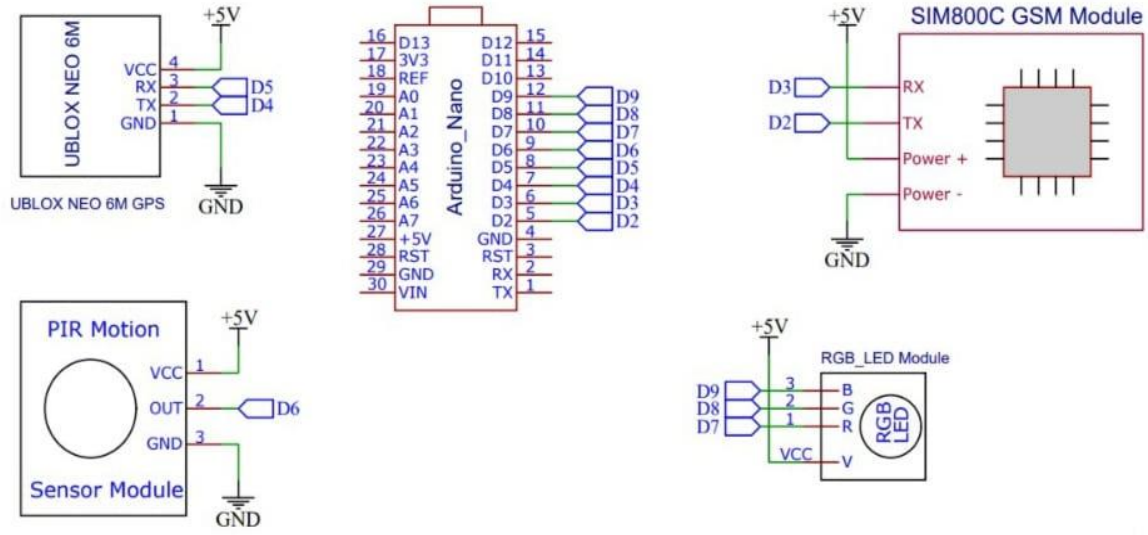

Figure 4.6: PIR Motion Sensor Pin description

4.4 Hardware Requirements

4.4.1 ESP32 camera module

The Camera module also known as compact camera module has been widely used in conferencing, security system and real time monitoring as a video input device. Model of this camera is ESP32. Camera module is used to take picture frame for video. The below figure 4.7 shows the ESP32 camera module.

Figure 4.7: ESP32 camera module

The Specifications include:

- RAM: built-in 520 KB+external 4MPSRAM
- Dimension: 27*40.5*4.5（±0.2）mm/1.06*1.59*0.18”
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE standards

- Wi-Fi: 802.11b/g/n/e/i
- Support Interface: UART, SPI, I2C, PWM
- IO port: 9
- Serial Port Baud-rate: Default 115200 bps
- Image Output Format: JPEG(OV2640 support only), BMP, GRAYSCALE
- Antenna: onboard PCB antenna, gain 2dBi
- Power supply range: 5V

4.4.2 DC Motor

The Motors that operate on 12v DC power supply are used. These are rotatory electrical machine that converts current electrical energy into mechanical energy. The motors used are of 30rpm speed of operation. DC motors are used to control the forward, backward, left and right moment of the robot. The below figure 4.8 shows the DC Motor.

Figure 4.8: DC Motor

The Specifications include:

- Shaft length: 7 mm.
- Shaft Diameter: 5.5 mm.
- Size: 55 x 48 x 23 mm.
- Operating Voltage: 3 to 12V.
- Current (without loading): 40-180mA.
- RPM: 100 rpm.

4.4.3 L298N Motor Driver

This dual bidirectional motor driver is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to control two motors of up

easily and independently to 2A each in both directions. It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. The below figure 4.9 shows the L298N Motor Driver.

Figure 4.9: L298N Motor Driver

The Specifications include:

- Model: L298N 2A
- Operating Voltage: 5V ~ 35V DC
- Peak Current: 2A
- Continuous Current: 0-36mA
- Dimension (L x W x H in mm): 44 x 44 x 28

4.4.4 GPS (Ublox Neo 6M)

The u-blox NEO-6M is a popular GPS receiver module that uses the Global Navigation Satellite System (GNSS) to determine its location. Here are some key features of the NEO-6M : High accuracy, Low power consumption, Serial interface etc. The below figure 4.10 shows the GPS (Ublox Neo 6M).

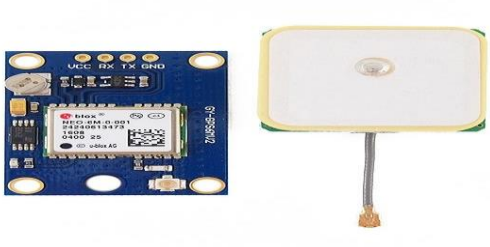

Figure 4.10: GPS (Ublox Neo 6M)

The Specifications include:

- Receiver type: 50-channel GPS/GLONASS/QZSS/SBAS receiver
- Frequency: L1 1575.42 MHz
- Tracking sensitivity: -162dBm
- Position accuracy: 2.5 meters CEP
- Velocity accuracy: 0.1 meters/second
- Time accuracy: 30 nanoseconds RMS
- Power supply: 3.3V to 5V DC input
- Power consumption: 50mA typical (GPS only)

4.4.5 Arduino Nano

Arduino Nano is a small and affordable development board based on the ATmega328 microcontroller. It is similar to the larger Arduino boards in functionality, but with a smaller size and lower cost. It is breadboard friendly and has 14 digital I/O pins and 6 analog input pins. It can be programmed using the Arduino IDE and can be powered and programmed via a USB connection. With its compact size, the Arduino Nano is a great choice for projects where space and cost are important factors, such as robotics, wearable electronics, and small-scale automation. The below figure 4.11 shows the Arduino Nano.

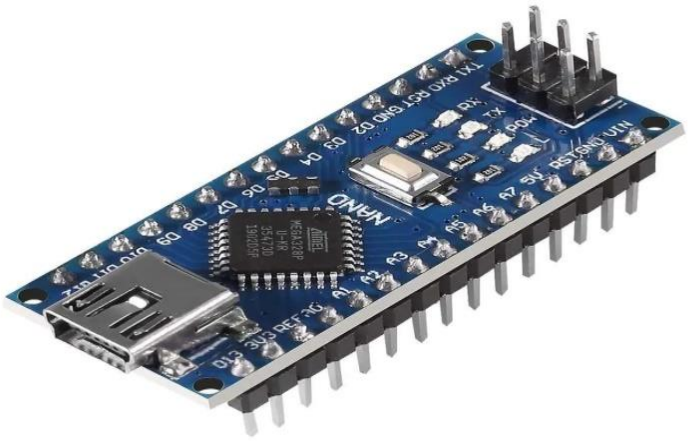

Figure 4.11: Arduino Nano

The specifications include:

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Flash Memory: 32 KB (ATmega328P) of which 2 KB used by boot loader
- SRAM: 2 KB (ATmega328P)
- EEPROM: 1 KB (ATmega328P)
- Clock Speed: 16 MHz

4.4.6 PIR Sensor

PIR (Passive Infrared) sensors are electronic devices that detect the presence of people or animals within a certain range by sensing their body heat or infrared radiation. They are commonly used in motion detection systems for security, lighting, and energy conservation applications. The below figure 4.12 shows the PIR Sensor.

Figure 4.12: PIR Sensor

The specifications include:

- Voltage: 4.8 V – 20 V
- Current (idle): <50 µA
- Logic output: 3.3 V / 0 V
- Sensing range: <120 °, within 7 m
- Temperature: – 15 ~ +70 °C
- Lens diameter: 23 mm

4.4.7 GSM SIM800C

GSM 800C (Global System for Mobile Communications 800C) is a mobile communication standard that operates on the 800 MHz frequency band. It provides voice and data services, including short message service (SMS) and circuit-switched data (CSD), at data rates of up to 9.6 kbps. One of the key features of GSM 800C is its support for international roaming, which allows users to use their mobile phones while traveling abroad. The below figure 4.13 shows the GSM SIM800C.

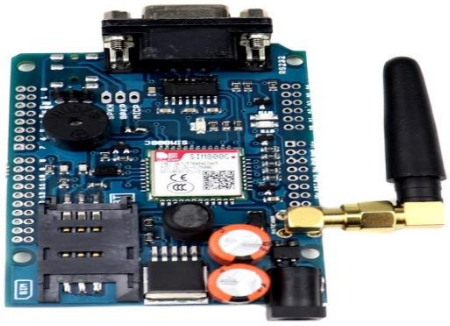

Figure 4.13: GSM SIM800C

The specifications include:

- Quad-band 850/900/1800/1900MHz
- Dimensions: 17.6*15.7*2.3mm
- Control via AT commands: (3GPP TS 27.007, 27.005 and SIM Com enhanced AT Commands)
- Supply voltage range 3.4 ~ 4.4V
- Operation temperature:-40℃ ~85℃

4.5 Software Requirements

4.5.1 Arduino IDE

The Arduino Integrated Development Environment (IDE) is a software application used to write, upload, and debug code for Arduino microcontroller boards. The Arduino IDE is a powerful and easy-to-use tool for programming Arduino boards and creating interactive projects. It has a large and supportive community, with many tutorials and examples available online. The below figure 4.14 shows the Arduino IDE Software.

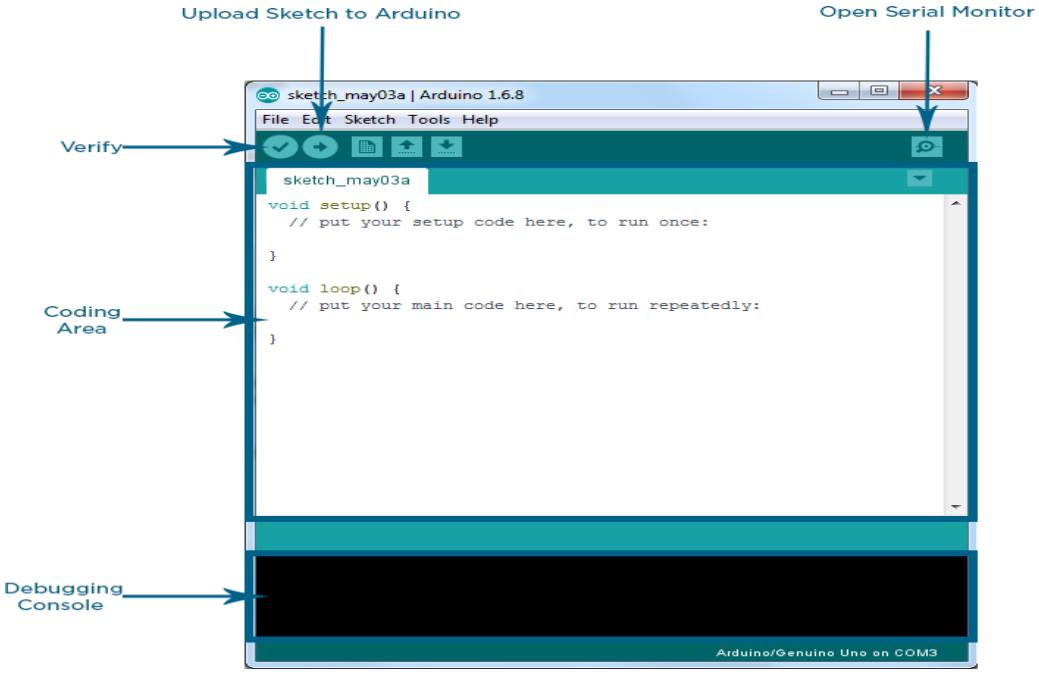

Figure 4.14: Arduino IDE Software

CHAPTER-5

Results and Discussion

5.1 Preliminary tests

This section reports all the preliminary trials which were run in order to evaluate the final performance of the robotic system. Initially, the testing was performed on single components, then tests were performed on the overall system. Each session of tests was made of 10 trials

Test 01

The camera's servo motor has been connected to the ESP32 board in the system's current iteration. Nevertheless, the Arduino Uno board was first linked to this motor. The L298N motor driver and the four DC motors were then taken care of by this latter board. With this arrangement, the system was relatively sluggish to react, and after 10 trials, we consistently found that it was slow (4/10), if not extremely slow (6/10). The system's performance altered as soon as the servo motor was unplugged from the Arduino board and linked to the ESP32 (10/10).

Test 02

In these tests, the ESP32 wireless connection via the webserver was used to regulate the position of the DC motors via the L298N motor driver. This test's primary goal was to determine whether the L298N motor driver was responding as instructed. We tested the DC motor's direction of rotation, for instance, when the "LEFT" button on the website was pressed. Over all of the trials, communication and motor behavior were seen to be reliable and consistent (10/10).

Test 03

We evaluated the performance of PIR sensor whether it works well in order to detect human intruder and allow the robot to SMS alert. On 10 trials, 9 times (over 10) the system was detecting the human intruder. The failure was due to PIR sensor was positioned in wrong side of the vehicle. Then we corrected the position of the sensor, placed it in a proper position so that it senses the intruder temperature and detects the human intruder and sends a SMS alert to the user.

5.2 Results

This Multipurpose Surveillance Robot is designed to perform various surveillance tasks in different environments. Some potential features and applications of such a robot could include:

Surveillance and monitoring: The robot can be equipped with cameras, sensors, and other surveillance devices to capture and transmit real-time video footage or images from the target area. It can provide enhanced situational awareness, especially in areas that are difficult to access for humans.

Security and threat detection: The robot may be programmed to detect and identify potential security threats, such as unauthorized access, suspicious objects, or unusual activities. It can use technologies like facial recognition, object recognition, and motion detection to enhance security measures.

Remote operation and control: The robot can be remotely operated by a human operator or controlled autonomously using advanced algorithms. This allows for flexible deployment and reduces the need for human presence in potentially dangerous or remote locations. The project model of the surveillance robot is shown in the Figure 5.1

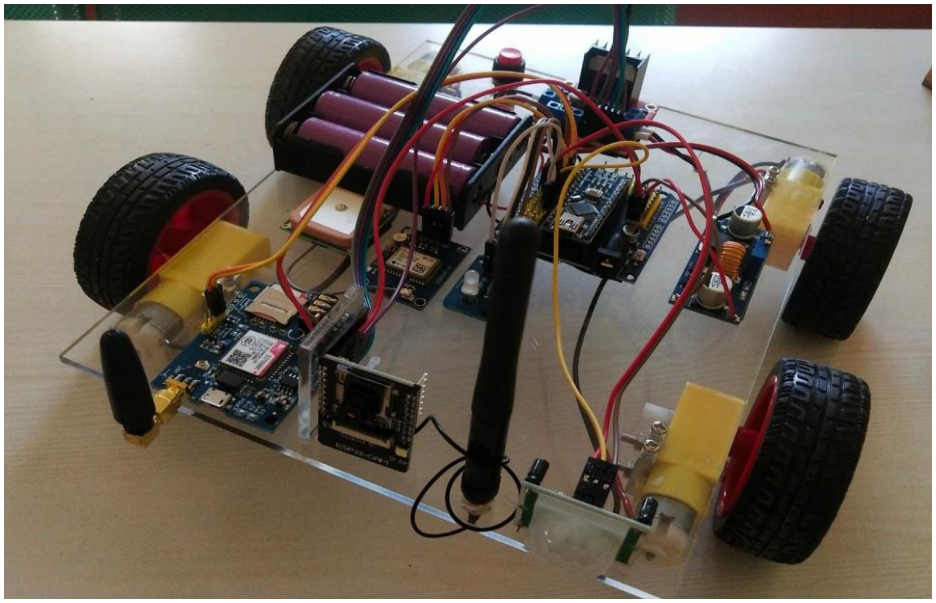

Figure 5.1: Project Model

Surveillance robots can be used for a wide range of applications, such as monitoring and securing a location, inspecting and maintaining equipment, collecting data, and providing situational awareness. This robot is equipped with sensors and cameras could be used to monitor a construction site for safety hazards and provide real-time footage to operators.

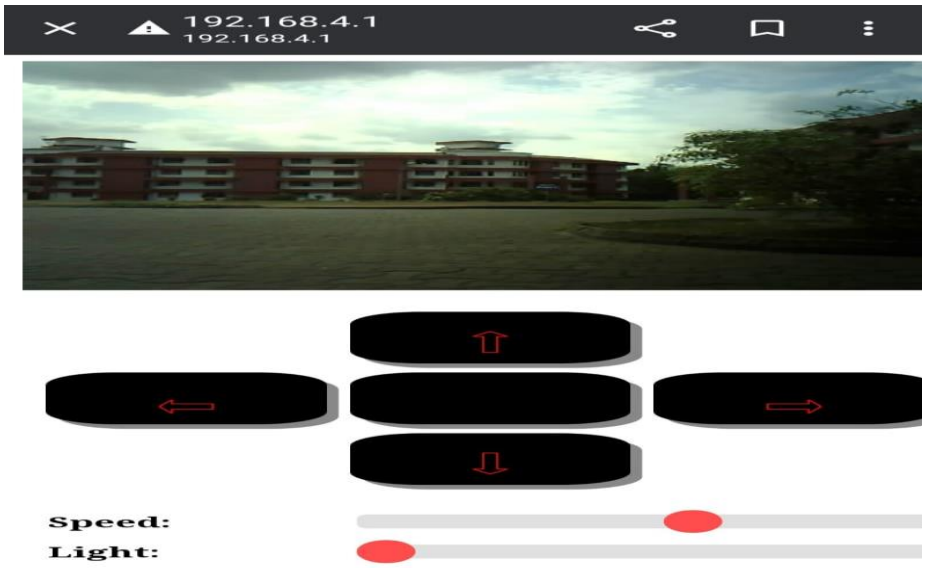

Figure 5.2: Video Streaming and control icons

The above Figure 5.2 shows the live streaming of the video and control buttons for the robot movement.

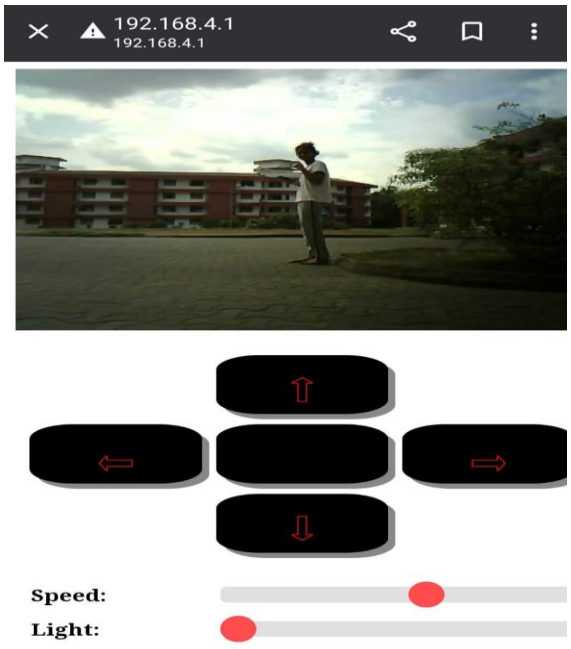

Figure 5.3: Human Intruder

Figure 5.4: Message Alert

The above Figure 5.3 shows the image of human intruder and Figure 5.4 shows the message alert of human detection in the user mobile phone.

When an intruder is spotted, the surveillance robot can immediately send an alert to security personnel or law enforcement, providing real-time details about the location and type of intrusion. The robot is equipped with sensors such as cameras, motion detectors, or other sensors. Once an intruder is detected, the robot can track and monitor their movements. This can help to provide situational awareness for security personnel and assist in coordinating a response.

Conclusion

Surveillance robots have become increasingly popular in recent years due to their ability to monitor and surveil areas that may be difficult or dangerous for humans to access. These robots are equipped with cameras, sensors, and other monitoring tools that allow them to gather information about their surroundings and transmit it back to their operators. The use of surveillance robots has both advantages and disadvantages. On the one hand, they can help improve security and safety by providing real-time information about potential threats or dangers. They can also be used in situations where it may be difficult or dangerous for humans to enter, such as in hazardous environments or during natural disasters.

References

[1]. Zhang, G. Song, G Qias T. Meng and H. Sun, "An indoor security system with a jumping robot as the arveillance perminta," in IEEE Tran aution im Cone Electronics, vol. 57, no 4, pp. 1774 1781 November 2018].

[2]. S. Hameed, M. Hamza Khan, N. Jafri, A. Azfar Khan, and M. BilalTaak, “Military Spying Robot,” in 2278-3075, May 2019, vol. 8, no.7C2, Accessed: Jul. 16, 2020.

[3]. Ch. Kulkarni, S. Grama, P. Gubbi, Ch. Krishna and J. Antony, "Surveillance Robot Using Arduino Microcontroller. Android APIs and the Internet, IEEE International Conference on Systems Informatics, Modeling and Simulation.

[4]. Gaurav S Bagul, Vikram C Udawant, Kalpana V Kapade and Jayesh M Zope,” IOT Based Surveillance Robot” International Journal of Trend in Research and Development, 2019 March.

[5]. S. Jagadesh, R. Karthikeyan, R. Manoj Prabhakaran, R. Ramesh and C. Shanmugam. "Surveillance Robot Using Microcontroller ESP8266". IJAREEIE Vol. 8, Issue 3, March 2022.

APPENDIX-A

#Program for ESP32 CAM

```
#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     = "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: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")'>⇧</td>

<td></td>

</tr>

<tr>

<td class="button" ontouchstart='sendButtonInput("MoveCar","3")' ontouchend='sendButtonInput("MoveCar","0")'>⇦</td>

<td class="button"></td>

<td class="button" ontouchstart='sendButtonInput("MoveCar","4")' ontouchend='sendButtonInput("MoveCar","0")'>⇨</td>

</tr>

<tr>

<td></td>

<td class="button" ontouchstart='sendButtonInput("MoveCar","2")' ontouchend='sendButtonInput("MoveCar","0")'>⇩</td>

<td></td>

</tr>

<tr/><tr/>

<tr>

<td style="text-align:left">Speed:</td>

<td colspan=2>

<input type="range" min="0" max="255" value="150" class="slider" id="Speed" oninput='sendButtonInput("Speed",value)'>

</td>

</tr>

<tr>

<td style="text-align:left">Light:</td>

<td colspan=2>

<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 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);
      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());
}
```

APPENDIX-B

#Program for Intruder Monitoring

```
// Libraries:
#include <SoftwareSerial.h>
/* Built-In Library */
#include<TinyGPS++.h>
// Pin Numbers: GPS
#define GPS_RxPin 4    // Connect to TX of GPS Module
#define GPS_TxPin 5    // Connect to RX of GPS Module
// Pin Numbers: Sim Module
#define Sim_Rx_Pin 2  //Connect to Tx of SIM Module
#define Sim_Tx_Pin 3  //Connect to Rx of SIM Module
// Pin Numbers: PIR
#define PIR_Pin 6  // (Input) Connect to the push-button.
// Pin Numbers: RGB LED
#define RLED_Pin 7
#define GLED_Pin 8
#define BLED_Pin 9
// Configuration: Mobile
const String MobileNumber1 = "9731601048";   //Replace xxxxxxxxxx with 10 digit
mobile number
// Configuration: GPS
unsigned long GPSReadWaitTime = 15000;
// Software Serial Port Objects:
SoftwareSerial SerialSIM(Sim_Rx_Pin, Sim_Tx_Pin);
SoftwareSerial SerialGPS(GPS_RxPin, GPS_TxPin);
// Objects:
TinyGPSPlus gps;
// Variables:
float gpsLat, gpsLng;
unsigned long prevPIRDetTime;
String msgContent;
void setup() {
  // Pin Mode Configuration:
```

```
 pinMode(PIR_Pin, INPUT);
pinMode(RLED_Pin, OUTPUT);
 pinMode(GLED_Pin, OUTPUT);
 pinMode(BLED_Pin, OUTPUT);
 digitalWrite(RLED_Pin, HIGH);
 digitalWrite(GLED_Pin, HIGH);
 digitalWrite(BLED_Pin, HIGH);
 /* Establish serial communication with the PC */
 Serial.begin(9600);
/* Establish serial communication with the GPS Module */
 SerialGPS.begin(9600);
/* Establish serial communication with the SIM Module */
 SerialSIM.begin(9600);
 delay(500);
msgContent = "Human Detection Robot Is Ready!";
 SendSMS(MobileNumber1, msgContent);
}
void loop() {
 if (digitalRead(PIR_Pin)) {
   if (millis() - prevPIRDetTime > 30000) {
     Serial.println("Human Detected Sending SMS! BLUE LED ON");
     digitalWrite(RLED_Pin, HIGH);
     digitalWrite(GLED_Pin, HIGH);
     digitalWrite(BLED_Pin, LOW);
     SendLocation("Human Detected!");
     prevPIRDetTime = millis();
   } else {
     Serial.println("Human Detected SMS Already Sent! RED LED ON");
     digitalWrite(RLED_Pin, LOW);
     digitalWrite(GLED_Pin, HIGH);
     digitalWrite(BLED_Pin, HIGH);
   }
 } else {
   Serial.println("GREEN LED ON");
   digitalWrite(RLED_Pin, HIGH);
```

```
    digitalWrite(GLED_Pin, LOW);
    digitalWrite(BLED_Pin, HIGH);
  }
}
bool GetGPS_LatLng() {
Serial.println("Reading GPS...  ");
SerialGPS.listen();
while (!SerialGPS.isListening());
bool isLocFound = false;
  unsigned long millisB4Entry = millis();
  while (!isLocFound && millis() - millisB4Entry < GPSReadWaitTime) {
    while (SerialGPS.available()) {
      if (gps.encode(SerialGPS.read())) {
        if (gps.location.isValid()) {
          gpsLat = gps.location.lat();
          gpsLng = gps.location.lng();
          isLocFound = true;
          break;
        }
      }
    }
  }
  return isLocFound;
}
void SendLocation(String str) {
  if (GetGPS_LatLng()) {
    Serial.println("Location Found");
    msgContent = str + "\nLink to Current Location http://maps.google.com/?q=" +
String(gpsLat, 6) + "," + String(gpsLng, 6);
    SendSMS(MobileNumber1, msgContent);
  } else {
    Serial.println("Location Not Found");
    msgContent = str + "\nReading GPS has Failed.";
    SendSMS(MobileNumber1, msgContent);
  }
```

```
}

void SendSMS(String mobileNumber, String message) {
  Serial.println("Sending SMS...  ");
SerialSIM.listen();
  while (!SerialSIM.isListening());
 // Set the GSM Module in Text Mode:
  SerialSIM.print("AT+CMGF=1\r");
  delay(500);
  String cmdSetMobileNumber = "AT+CMGS=\"+91" + mobileNumber + "\"\r";
  SerialSIM.print(cmdSetMobileNumber);
  delay(500);
  SerialSIM.print(message);
  SerialSIM.write(26);              // ASCII code of CTRL+Z
  delay(500);
Serial.println("SMS Sent.");
}
```