Project Report On

Search and Rescue Robot

SUBMITTED BY :

THOTA KOUSHIK SAI

POPURI HEMANTH KUMAR

BHUMANA SWETHA REDDY

CHATHRESH CHIRANJEEVI VASAGIRI

ADITI JOSHI

P.BHAVYA

Contents

1 1.INTRODUCTION **4**

1.1 INTRODUCTION 4

1.2 PURPOSE OF THE PROJECT 5

1.3 EXISTING SYSTEMS 5

1.4 BLOCK DIAGRAM OF EXISTING SYSTEM 6

1.5 LIMITATIONS OF EXISTING SYSTEMS 6

1.6 PROJECT ARCHITECTURE 7

2 LITERATURE REVIEW **8**

2.1 LITERATURE SURVEY 8

3 PROPOSED SYSTEM **9**

3.1 PROPOSED SYSTEM 9

3.2 ADVANTAGES OF PROPOSED SYSTEM 9

3.3 HARDWARE REQUIREMENTS 10

3.3.1 ESP 32 WROVER I.E 10

3.3.2 ESP 32 CAM 11

3.3.3 Motor Driver L298N 11

3.3.4 Side Shaft Motors 12

3.3.5 Pantilt servo 12

3.3.6 XL6009 13

3.3.7 LI-ion battery 13

3.3.8 Temparature sensor(GY-906MLX90614) 14

3.3.9 Wheels 14

3.3.10 Caterpillar Tracks 15
3.3.11 Jumper Wires 15
3.3.12 Breadboard 16
3.3.13 Battery holder 16
3.4 SOFTWARE REQUIREMENTS 17
3.4.1 Ardiuno IDE 17
3.4.2 Fusion 360 17
3.4.3 Cirkit designer 18

4 WORKING OF PROPOSED SYSTEM 19
4.1 BLOCK DIAGRAM 19
4.2 WORKING 19
4.3 Cad model 20
4.4 circuit diagram 20

5 RESULTS 21

6 CONCLUSION AND FUTURE SCOPE 22
6.1 CONCLUSION 22
6.2 FUTURE SCOPE 23

7 REFERENCES 24

8 SOURCE CODE 26

ABSTRACT

Disaster scenarios often present significant challenges in locating and aiding injured individuals. Manual search efforts are time-consuming and expose rescuers to hazardous conditions, delaying critical assistance and putting more lives at risk. To address these issues, we propose an autonomous robot designed to enhance search and rescue operations in disaster-affected areas. This compact robot is equipped with advanced technologies, including a high-resolution camera, image recognition algorithms, and thermal imaging sensors, to swiftly detect and locate injured persons. The integration of these technologies enables the robot to identify victims efficiently, even in complex environments such as collapsed structures or narrow spaces that are difficult or dangerous for human rescuers to access.

Beyond detection, the robot includes a temperature sensor to assess the condition of injured individuals, helping prioritize rescue efforts based on the urgency of medical needs. Additionally, the robot is designed with compartments for carrying essential medical supplies. This feature allows it to provide immediate first aid, such as bandages, pain relief, or hydration, to stabilise victims until human responders can reach them. The robot's compact and rugged design ensures its ability to navigate uneven and debris-filled terrains, maintaining operational efficiency in challenging environments. By automating the search process and delivering preliminary care, the proposed solution reduces the time needed for traditional manual searches, minimizes risks to human rescuers, and enhances overall disaster response efforts. Incorporating this autonomous system into emergency response protocols will save lives, improve the efficiency of rescue missions, and provide a safer alternative for responders. By combining cutting-edge technology with practical design, this solution offers a transformative approach to managing critical situations in disaster-stricken areas.

Chapter 1

1.INTRODUCTION

1.1 INTRODUCTION

Natural disasters and other emergencies often result in widespread destruction, leaving many individuals injured and in urgent need of assistance. In such scenarios, the traditional approach of manually searching for injured individuals is not only time-consuming but also exposes rescuers to significant risks in unstable environments. The delay in locating and aiding victims can have severe consequences, including increased mortality rates and prolonged suffering. To address these challenges, there is a pressing need for an innovative solution that enhances the efficiency and safety of search and rescue operations. Automating the process of detecting and locating injured individuals can significantly reduce the time required to provide critical aid and improve the overall effectiveness of disaster response. In this context, we propose the development of a compact, autonomous robot specifically designed for disaster scenarios. Equipped with cutting-edge technologies such as advanced image recognition, and temperature sensors, the robot will be capable of swiftly identifying victims, even in hazardous and hard-to-reach areas. Additionally, it will carry essential medical supplies to provide immediate assistance, helping stabilize injured individuals until human responders arrive. By combining robust design, advanced technology, and practical functionality, this autonomous robot offers a transformative approach to disaster response. It ensures quicker aid delivery, minimizes risks to rescuers, and saves more lives, making it a critical addition to modern emergency management strategies.

1.2 PURPOSE OF THE PROJECT

To improve search and rescue operations in disaster-affected areas, we proposed developing a compact,semi-autonomous robot tailored for this critical task. This robot will be equipped with a high-resolution camera and various sensors to detect human bodies, utilizing advanced image recognition and thermal imaging technologies. By quickly identifying injured individuals, the robot can significantly reduce the time required for manual searches.Additionally, the robot will feature a temperature sensor to assess the stability of injured persons, enabling rescue teams to prioritize their interventions based on urgency. It will also include compartments for carrying essential medical supplies, allowing it to provide immediate aid, which can be crucial in stabilizing victims before human responders arrive.The robot's compact design will enable it to navigate challenging terrains, such as collapsed structures and narrow spaces, where human rescuers may face significant risks. By automating the search and initial care processes, this solution offers a faster, safer, and more efficient alternative to traditional methods. Ultimately, the deployment of this semi autonomous robot will enhance disaster response efforts, save lives, and improve overall operational effectiveness in critical situations.

1.3 EXISTING SYSTEMS

Search and rescue robots thus contribute positively toward the efficiency and safety of operations in dangerous spatial environments.Ground robots, like Boston Dynamics' Spot or ANYbotics' ANYmal, can gain access to rough terrains, including rubble and stairs, since they are equipped with application-specific sensors such as cameras, LiDARS, and GPS.Aerial robots like the DJI Matrice 300 RTK and the Aeryon SkyRanger R70 can provide aerial reconnaissance, including thermal imaging and real-time information, for search operations over larger areas. For underwater rescue, Remotely Operated Vehicles (ROVs), like the BlueROV2, carry cameras and sonar to ascertain whereabouts of victims or recover objects.Other hybrid robots, like those in the RoboCup Rescue competition, respect some features from both aerial and ground robots, allowing them to operate in several environments.Swarm robotics involves many robots working collaboratively and autonomously,increasing search coverage for large-scale operations.Additionally,there are autonomous vehicles like Toyota's Robocar deployed to transport supplies and victims from one point to another.

1.4 BLOCK DIAGRAM OF EXISTING SYSTEM

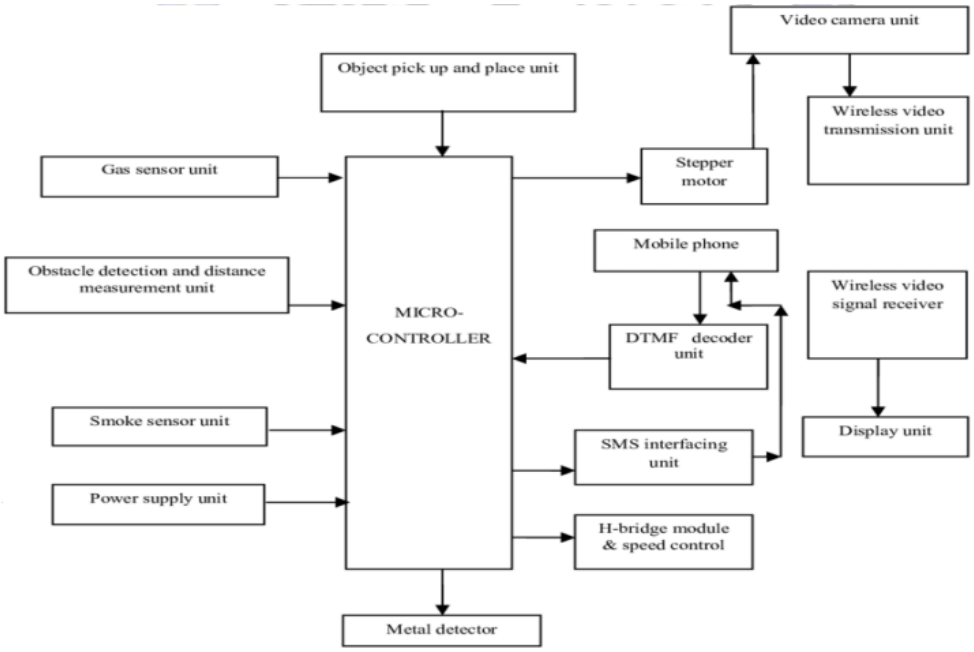

Figure 1.1: BLOCK DIAGRAM OF EXISTING SYSTEM

1.5 LIMITATIONS OF EXISTING SYSTEMS

The available literature on search and rescue robots presents several limitations. The first pertains to their inability to adapt to different environments. Ground robots often face difficulties when traversing soft or unstable surfaces, drones face challenges in harsh weather conditions, while underwater robots are limited to specific aquatic environments. The battery is also a major impairing factor that hits operational efficiency, especially for those equipped with heavy sensors or in harsh working conditions, requiring frequent recharging. Thirdly, other communication problems may affect their levels of performance, especially in areas where there have been collapses, such as in the buildings or in areas where the GPS data transmission systems are down so that real-time sharing of information is not possible. The other factors that must be considered include costs and availability since, in developed nations, such robots are pretty costly and become unavailable for all rescue teams, particularly in areas of limited resource allocation.In addition, operation complexity limits the utilization of a robot some require skilled training to be used effectively.

1.6 PROJECT ARCHITECTURE

The semi-Autonomous Search and Rescue Robot is designed to efficiently operate in disaster zones with a compact and stable frame, allowing it to navigate rugged terrains. It uses tracks or wheels with adaptive suspension, and optional extendable limbs for climbing over debris. The robot has compartments to store medical supplies for victim stabilization.Equipped with a high-resolution camera for 180-degree vision, esp 32 cam for detecting trapped victims, and proximity sensors for obstacle avoidance, the robot ensures safe navigation.

A temperature sensor helps prioritize victims based on their health condition.Onboard, the robot has a powerful processor for running image recognition and sensor data fusion. It communicates wirelessly through Wi-Fi, Bluetooth, or neo6mv2, and is powered by rechargeable batteries. Semi-Autonomous navigation is managed with pathfinding algorithms, enabling safe and efficient movement.The robot uses health monitoring interface for urgent response.motors control movement, while a compartment delivers medical supplies.This design ensures timely, effective rescue operations while minimizing risks to human rescuers.

Chapter 2

LITERATURE REVIEW

2.1 LITERATURE SURVEY

In the past, related research has taken place within the area of multi-agent systems such as UAV-UGV cooperation [1,2,3] and multi-robots[4,5] like multi-UAVs [6.7] and multi-UGVs[8.9.10]. UAVs and UGVs have several advantages and disadvantages over each other in terms of movement, weight carrying capacity, and perception abilities and past work [2,3] focus on combining UAVs and UGVs' adv advantages together to create a multi-robot system that will be implemented to try to a specific task as in [1,2,7]. Research in the multi-robot system has been done because it has many advantages and applications in many fields like the military, surveillance, etc. UAVs and UGVs can work simultaneously used for exploring unknown areas When multiple robots are present and to be controlled simultaneously, the system requires careful attention on how each robot behaves so as to avoid any collision or any unexpected scene [11]. But past work tends to use one sort of mobile robots, either UAV [6,7,9] or UGV [3,10] to fulfill the task. In this paper, it consists of UGV with a UAV that explores and maps the unknown indoor environment using motion capture to get the situation of the UAV and a camera mounted on the UAV to take an aerial view.There has been an identical work of indoor environment mapping (8) using UGVs and a sick laser. In our work, the UAV provides a clear aerial view of the unknown environment and therefore the main system installed on UGV creates a Lidar map cloud.

Chapter 3

PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

To improve search and rescue operations in disaster-affected areas, we proposed developing a compact, semi autonomous robot tailored for this critical task. This robot will be equipped with a high-resolution camera and various sensors to detect human bodies, utilizing advanced image recognition and thermal imaging technologies. By quickly identifying injured individuals, the robot can significantly reduce the time required for manual searches.

Additionally, the robot will feature a temperature sensor to assess the stability of injured persons, enabling rescue teams to prioritize their interventions based on urgency. It will also include compartments for carrying essential medical supplies, allowing it to provide immediate aid, which can be crucial in stabilizing victims before human responders arrive.

The robot's compact design will enable it to navigate challenging terrains, such as collapsed structures and narrow spaces, where human rescuers may face significant risks. By automating the search and initial care processes, this solution offers a faster, safer, and more efficient alternative to traditional methods. Ultimately, the deployment of this autonomous robot will enhance disaster response efforts, save lives, and improve overall operational effectiveness in critical situations.

3.2 ADVANTAGES OF PROPOSED SYSTEM

The proposed compact, autonomous robot offers several significant advantages for search and rescue operations in disaster-affected areas. Its high-resolution camera and advanced sensors, including

thermal imaging, allow it to quickly detect injured individuals, drastically reducing the time required for manual searches and increasing the chances of saving lives. The robot's temperature sensor provides real-time data on the stability of victims, enabling rescue teams to prioritize those in the most critical condition. Additionally, its medical supply compartments ensure that immediate care can be provided, stabilizing victims before human responders arrive, which is essential in life-threatening situations.The compact design of the robot allows it to navigate difficult terrains, such as collapsed structures and narrow spaces, where human rescuers might be unable to reach safely. By automating the initial search and medical assessment process, the robot offers a faster, safer, and more efficient solution compared to traditional methods. This reduces the risks to human rescuers, ensuring that they can focus on higher-level tasks once the robot identifies and stabilizes victims. Ultimately, deploying this autonomous robot will enhance disaster response efforts, streamline operations, save lives, and provide invaluable support in critical and dangerous situations.

3.3 HARDWARE REQUIREMENTS

3.3.1 ESP 32 WROVER I.E

The ESP32-WROVER is a powerful and versatile microcontroller module featuring the ESP32 chip. It offers built-in Wi-Fi and Bluetooth connectivity, making it ideal for IoT applications. The module includes 16MB of PSRAM and a flash memory size of 4MB, enabling it to handle demanding tasks. It supports a variety of input/output interfaces, including GPIO, SPI, I2C, UART, and PWM. The ESP32-WROVER has multiple ADC and DAC pins, making it suitable for sensor and analog applications. Key advantages include low power consumption, high processing power, and flexibility in wireless communication.

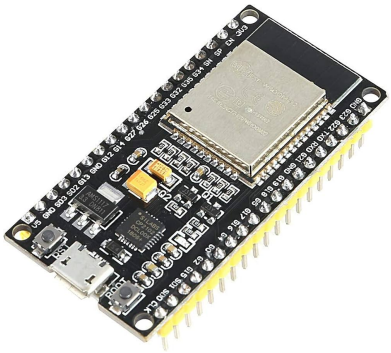

Figure 3.1: Esp 32

3.3.2 ESP 32 CAM

The ESP32-CAM is a low-cost development board with a built-in ESP32 chip, featuring Wi-Fi and Bluetooth capabilities. It integrates a 2MP OV2640 camera for image and video processing, making it ideal for IoT, surveillance, and remote monitoring applications. The board offers GPIO pins, UART, SPI, I2C, PWM, and ADC for versatile connectivity and control. Key advantages include low power consumption, ease of use with Arduino IDE, and ability to stream video or capture images directly. It's compact, affordable, and ideal for camera-based projects in remote or wireless applications.

Figure 3.2: Esp 32 Cam

3.3.3 Motor Driver L298N

The L298N Motor Driver is a dual H-bridge motor driver that allows control of two DC motors or a stepper motor. It operates with voltage ranges from 4.5V to 46V and can deliver up to 2A of continuous current per channel, making it suitable for controlling motors in robotics and automation. The key pins include IN1, IN2, IN3, IN4 for motor control, ENA, ENB for enabling channels, and VCC, GND for power supply. Advantages include high efficiency, overcurrent protection, thermal shutdown, and the ability to control motors in both directions.

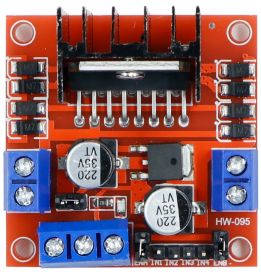

Figure 3.3: Motor driver

3.3.4 Side Shaft Motors

A side shaft motor is a type of engine where the crankshaft extends out from the side of the motor, making it ideal for applications requiring lateral power transmission. These motors are commonly used in equipment like go-karts, lawnmowers, generators, and small machinery. Side shaft motors are known for their durability, compact design, and efficient power delivery. They often feature air cooling, high torque output, and compatibility with various gear systems, making them versatile for both industrial and recreational uses.

Figure 3.4: Side Shaft Motors

3.3.5 Pantilt servo

A pan-tilt servo is a mechanism that allows precise control of the horizontal (pan) and vertical (tilt) movements of an object, typically used in cameras or sensors. It typically consists of two servo motors connected at a right angle, allowing 2D movement control. The common pins include PWM (Pulse Width Modulation) for controlling the position of the servos, VCC for power, and GND for grounding. Advantages of a pan-tilt servo include precise angular control, compact size, ease of integration in robotic systems, and the ability to enable a wide range of motion.

Figure 3.5: Pantilt Servo

3.3.6 XL6009

The XL6009 is a DC-DC boost converter used to step up low input voltage to a higher output voltage. It supports input voltages ranging from 3V to 32V and can provide output voltages from 5V to 35V with a current up to 4A. Key pins include Vin (input voltage), Vout (output voltage), GND (ground), EN (enable pin), and FB (feedback for adjusting output voltage). Advantages of the XL6009 include high efficiency (up to 94

Figure 3.6: XL6009

3.3.7 LI-ion battery

A Li-ion (Lithium-ion) battery is a rechargeable battery commonly used in portable electronics, electric vehicles, and energy storage systems due to its high energy density and long cycle life. The battery typically consists of positive (cathode) and negative (anode) electrodes, a separator, and an electrolyte. It has three primary pins: positive (+), negative (-), and balance pin (for battery management).

Figure 3.7: LI-ion battery

3.3.8 Temparature sensor(GY-906MLX90614)

The GY-NEO6MV2 is a GPS module based on the u-blox NEO-6M chip, designed for obtaining position, velocity, and time data. It communicates via UART (TX/RX) or I2C interface, making it compatible with most microcontrollers like Arduino. The module has 4 main pins: VCC (power), GND (ground), TX (transmit), and RX (receive). Advantages include high accuracy (up to 2.5 meters), low power consumption, easy integration, and fast GPS signal acquisition.It's ideal for applications like navigation systems, drones, and tracking devices.

Figure 3.8: Temparature sensor(GY-906MLX90614)

3.3.9 Wheels

Wheels are circular components designed to rotate around an axle, enabling motion and reducing friction between a vehicle or object and the surface it moves on. They are fundamental to transportation and mechanical systems, found in everything from cars and bicycles to machinery and robots.

Figure 3.9: Wheels

3.3.10 Caterpillar Tracks

Caterpillar tracks, also known as tank tracks, are continuous treads used on vehicles like tanks, bulldozers, and robots to enhance traction and mobility. They distribute the vehicle's weight over a large surface area, reducing ground pressure and preventing it from sinking on soft or uneven terrains. Made from durable materials like metal or reinforced rubber, caterpillar tracks provide excellent stability and are ideal for navigating challenging environments, such as mud, sand, or snow. Their robust design makes them essential for heavy-duty applications and off-road operations.

Figure 3.10: Caterpillar Tracks

3.3.11 Jumper Wires

Jumper wires are insulated wires with connectors at each end, used to create quick and temporary connections in electronic circuits without soldering. They are commonly employed in prototyping and testing circuits on breadboards or connecting components like sensors, modules, and microcontrollers. Available in three types—male-to-male, male-to-female, and female-to-female—they allow versatile connections between male and female headers. Typically PVC-coated for insulation, jumper wires come in various lengths and colors to simplify circuit organization.

Figure 3.11: Jumper Wires

3.3.12 Breadboard

A breadboard, solderless breadboard, or protoboard is a construction base used to build semi-permanent prototypes of electronic circuits. Unlike a perfboard or stripboard, breadboards do not require soldering or destruction of tracks and are hence reusable. For this reason, breadboards are also popular with students and in technological education.A breadboard is a board used to connect electronic components, such as wires, resistors, capacitors, and coils, to conduct various experiments and projects.

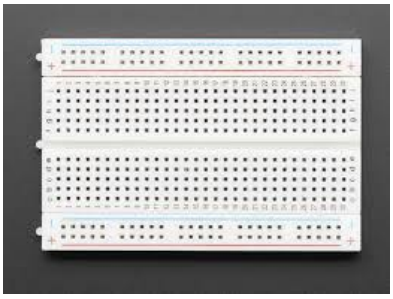

Figure 3.12: Breadboard

3.3.13 Battery holder

A battery holder is a device used to securely hold and connect batteries in electronic circuits. It provides a convenient and safe way to insert and remove batteries while ensuring proper electrical contact. Battery holders come in various sizes to accommodate different types of batteries (e.g., AA, AAA, CR2032). Advantages include easy battery replacement, improved safety, and organized power supply, making them ideal for portable electronics and DIY projects.

Figure 3.13: Battery holder

3.4 SOFTWARE REQUIREMENTS

3.4.1 Ardiuno IDE

The Arduino IDE (Integrated Development Environment) is a user-friendly software platform used for writing, compiling, and uploading code to Arduino microcontrollers. It supports primarily C and C++ programming languages and provides a simple interface to interact with Arduino boards through USB connections. The IDE features a code editor with syntax highlighting and error checking, a vast library collection for easy component integration, and a Serial Monitor for real-time debugging and data communication.

Figure 3.14: Ardiuno IDE

3.4.2 Fusion 360

Fusion 360 is a versatile 3D CAD, CAM, and CAE software by Autodesk, designed for product design, engineering, and manufacturing. It combines powerful 3D modeling tools, including parametric and freeform design, with simulation capabilities for stress and motion analysis. The software also integrates CAM features like 2D and 3D milling and supports cloud-based collaboration for real-time teamwork.

Figure 3.15: Fusion 360

3.4.3 Cirkit designer

Cirkit Designer is an intuitive software tool designed for creating, simulating, and visualizing electronic circuits. It combines schematic design, PCB layout, and simulation features into a single interface, making it ideal for hobbyists, students, and professionals. With real-time circuit simulation, users can test designs before physical implementation, saving time and resources. Its user-friendly interface supports drag-and-drop functionality, allowing quick and efficient circuit prototyping. Cirkit Designer often integrates with 3D visualization tools to preview layouts and optimize space. It is compatible with various component libraries, providing flexibility for diverse projects, from simple circuits to complex electronic systems.

Figure 3.16: Caption

Chapter 4

WORKING OF PROPOSED SYSTEM

4.1 BLOCK DIAGRAM

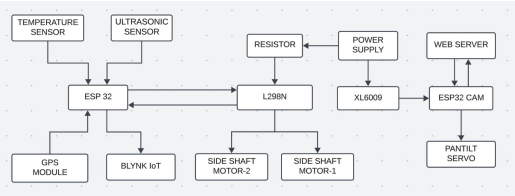

Figure 4.1: BLOCK DIAGRAM

4.2 WORKING

To enhance search and rescue operations in disaster-affected areas, we propose the development of an semi-autonomous robot designed to address critical challenges in such scenarios. This robot will leverage a high-resolution camera and advanced sensors to detect human bodies using image recognition .By rapidly identifying injured individuals, the robot can significantly reduce the time required for traditional manual searches.The robot will also feature a temperature sensor to monitor the condition of injured individuals, allowing rescue teams to prioritize interventions based on urgency. Furthermore, it will be equipped with compartments to carry essential medical supplies, enabling it to provide immediate aid and stabilize victims before human responders arrive.Its compact design will enable seamless navigation through challenging terrains, such as collapsed structures and narrow spaces, minimizing risks to human rescuers. By semi-automating the search and initial care processes, the robot offers a faster, safer, and more efficient alternative to traditional rescue methods.

4.3 Cad model

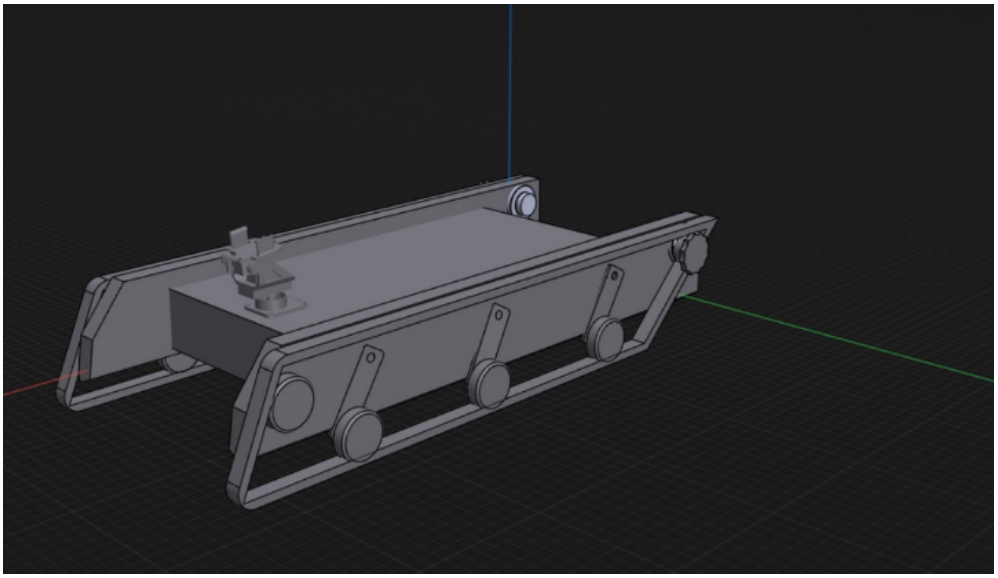

Figure 4.2: Cad model

4.4 circuit diagram

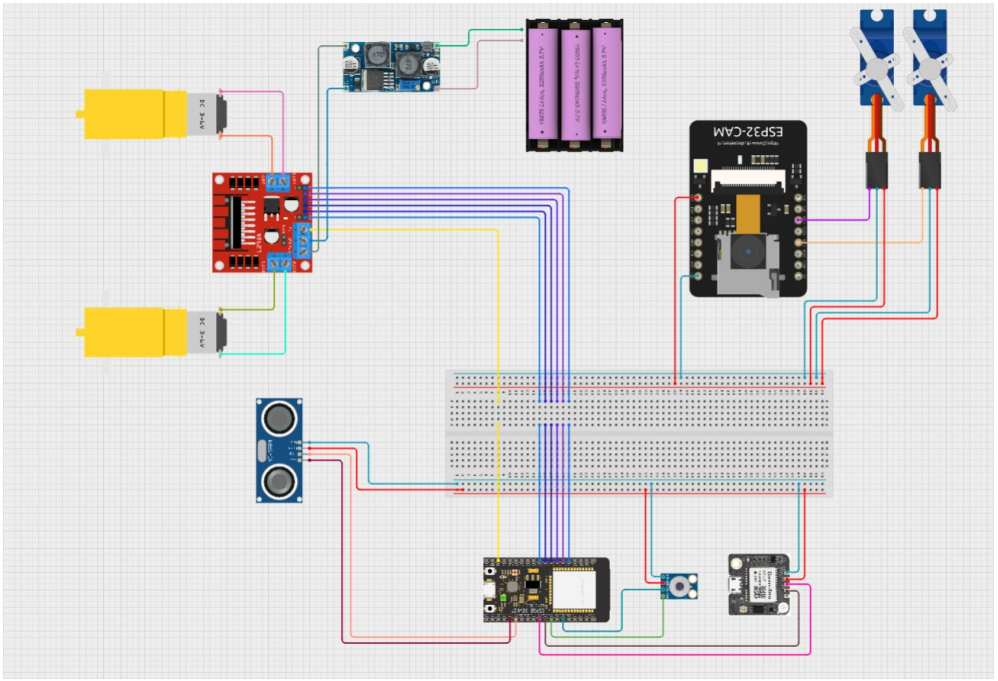

Figure 4.3: Cad model

Chapter 5

RESULTS

The semi-Autonomous Search and Rescue Robot is designed to efficiently operate in disaster zones with a compact and stable frame, allowing it to navigate rugged terrains. It uses tracks or wheels with adaptive suspension, and optional extendable limbs for climbing over debris. The robot has compartments to store medical supplies for victim stabilization.

Equipped with a high-resolution camera for 180-degree vision, esp 32 cam for detecting trapped victims, and proximity sensors for obstacle avoidance, the robot ensures safe navigation. A temperature sensor helps prioritize victims based on their health condition.

Onboard, the robot has a powerful processor for running image recognition and sensor data fusion. It communicates wirelessly through Wi-Fi, Bluetooth, or neo6mv2, and is powered by rechargeable batteries. Semi-Autonomous navigation is managed with pathfinding algorithms, enabling safe and efficient movement.The robot uses health monitoring interface for urgent response.motors control movement, while a compartment delivers medical supplies.This design ensures timely, effective rescue operations while minimizing risks to human rescuers.

Chapter 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

In conclusion, the proposed compact, semi-autonomous robot represents a groundbreaking advancement in disaster response and emergency management. By leveraging state-of-the-art technologies such as advanced image recognition, thermal imaging, and temperature sensors, the robot addresses critical challenges associated with traditional search and rescue operations. Its ability to swiftly identify and locate injured individuals, even in hazardous and inaccessible areas, significantly reduces response time and enhances the efficiency of rescue efforts.Equipped with essential medical supplies, the robot ensures immediate aid to stabilize victims, thereby improving survival rates until human responders can intervene. Its robust yet compact design enables seamless navigation through challenging terrains, such as collapsed structures and confined spaces, minimizing risks to human rescuers. By automating the search and initial care processes, this innovative solution not only accelerates rescue operations but also prioritizes safety and effectiveness in critical scenarios.This proposed system underscores the importance of integrating technology into disaster management strategies to save lives and mitigate the impact of emergencies. The development and deployment of such autonomous robots mark a transformative shift in how rescue operations are conducted, fostering a faster, safer, and more reliable response to natural disasters and other emergencies. As a vital tool in modern emergency management, the robot exemplifies how innovative engineering can address urgent humanitarian needs, paving the way for a more resilient and responsive disaster relief framework.

6.2 FUTURE SCOPE

1. Advanced AI and Machine Learning Integration Incorporate AI algorithms to improve the robot's decision-making capabilities, such as prioritizing victims based on severity of injuries, identifying potential hazards, and optimizing navigation paths in dynamic environments.

2. Enhanced Mobility and Terrain Adaptability Equip the robot with advanced locomotion systems, such as articulated legs or wheels with adaptive suspension, to traverse more challenging terrains like debris, waterlogged areas, or steep inclines.

3 Real-Time Communication and Coordination Integrate communication modules to relay real-time data, including victim locations, environmental hazards, and situational updates, to rescue teams. The robot could also work in swarms, coordinating with multiple units to cover larger areas effectively.

Chapter 7

REFERENCES

1.Y. Liu, G. Nejat and B. Doroodgar, ”Learning based semi-autonomous control for robots in urban search and rescue”, 2012 IEEE International Symposium on Safety Security and Rescue Robotics (SSRR), pp. 1-6, 2012.

2. R. Murphy, ”Trial by fire [rescue robots]”, IEEE Robotics Automation Magazine, vol.11, no. 3, pp. 50-61, 2004.

3. E. Guizzo, Can japan send in robots to fix troubled nuclear reactors?, Jun 2021, [online] Available: https://spectrum.ieee.org/japan-robots-to-fix-troubled-nuclear-reactors.

4. J. Lobo, P. Lucas, J. Dias and A. T. De Almeida, ”Inertial navigation system for mobile land vehicles”, 1995 Proceedings of the IEEE International Symposium on Industrial Electronics, vol. 2, pp. 843-848, 1995.

5. S. Mehmood, S. Ahmed, A. S. Kristensen and D. Ahsan, ”Multi criteria decision analysis (mcda) of unmanned aerial vehicles (uavs) as a part of standard response to emergencies”, Proc. 4th Int. Conf. Green Comput. Eng. Technol., pp. 31, 2018.

6. V. Radhakrishna, P.V. Kumar, V. Janaki and N. Rajasekhar, ”Estimating Prevalence Bounds of Temporal Association Patterns to Discover Temporally Similar Patterns” in Recent Advances in Soft Computing. ICSC-MENDEL 2016. Advances in Intelligent Systems and Computing, Cham:Springer, vol. 576, 2017.

7. A. Cheruvu, V. Radhakrishna and N. Rajasekhar, ”Using normal distribution to retrieve temporal associations by Euclidean distance”, 2017 International Conference on Engineering MIS (ICEMIS), pp. 1-3, 2017.

8. Awasthi Ankita, Kuldeep K. Saxena and Vanya Arun, ”Sustainability and survivability in manufacturing sector” in Modern Manufacturing Processes, Woodhead Publishing, pp. 205-219, 2020.

9. Bisht, Pankaj Singh and Ankita Awasthi, ”Analysis of E-glass fiber wheel rim by using ANSYS”, Recent Advances in Mechanical Engineering: Select Proceedings of ITME 2019, pp. 79-91, 2021.

10.Zibulewsky, J. Defining disaster: The emergency department perspective. In Baylor University Medical Center; Taylor Francis: New York, NY, USA, 2001; Volume 14, pp. 144–149.

Chapter 8

SOURCE CODE

```
    #define BLYNK_TEMPLATE_ID "TMPL3ToopsM3Z"
#define BLYNK_TEMPLATE_NAME "rescuebot"
#define BLYNK_AUTH_TOKEN "ogTQcidRFwkUC_tEbhHvkHdH3OEtWKga"
#include <BlynkSimpleEsp32.h>
const int motor1A = 21; // IN1
const int motor1B = 19; // IN2
const int enA = 26; // ENA
const int motor2A = 22; // IN3
const int motor2B = 23; // IN4
const int enB = 32; //ENB
void forward();
void backward();
void left();
void right();
void stopMotorsDrive();
#include <Adafruit_MLX90614.h>
#include <TinyGPS++.h>
#include <BlynkSimpleEsp32.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
TinyGPSPlus gps;
```

```
#define echoPin 14
#define trigPin 12
#define GPS_BAUDRATE 9600
long duration, distance;
void setup() {
 Serial.begin(115200);
 Blynk.begin(BLYNK_AUTH_TOKEN, "Karthika","Karthika@2020");
 pinMode(motor1A, OUTPUT);
 pinMode(motor1B, OUTPUT);
 pinMode(motor2A, OUTPUT);
 pinMode(motor2B, OUTPUT);
 pinMode(enA, OUTPUT);
 pinMode(enB, OUTPUT);
 Serial2.begin(GPS_BAUDRATE);
 mlx.begin();
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 Serial.println("ESP32 - MLX90614 & GPS with Blynk IoT");
}
 BLYNK_WRITE(V1) { //Forward Button
 int value = param.asInt();
 if (value == 1) {
 Serial.println("Switch on V0 is ON");
 forward();
 } else {
 Serial.println("Switch on V0 is OFF");
 stopMotorsDrive();
 }
}
BLYNK_WRITE(V2) { //Forward Button
```

```
 int value = param.asInt();
 if (value == 1) {
 Serial.println("Switch on V1 is ON");
 backward();
 } else {
 Serial.println("Switch on V1 is OFF");
 stopMotorsDrive();
 }
}
BLYNK_WRITE(V4) { //Forward Button
 int value = param.asInt();
 if (value == 1) {
 Serial.println("Switch on V2 is ON");
 right();
 } else {
 Serial.println("Switch on V2 is OFF");
 stopMotorsDrive();
 }
}
BLYNK_WRITE(V3) { //Forward Button
 int value = param.asInt();
 if (value == 1) {
 Serial.println("Switch on V3 is ON");
 left();
 } else {
 Serial.println("Switch on V3 is OFF");
 stopMotorsDrive();
 }
}
void loop() {
```

```
Blynk.run();
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);
 distance = duration / 58.2;
 Blynk.virtualWrite(V7, distance);
 float ambientTemp = mlx.readAmbientTempC();
 float objectTemp = mlx.readObjectTempC();
 Blynk.virtualWrite(V5, ambientTemp);
 Blynk.virtualWrite(V0, objectTemp);
 if (Serial2.available() > 0) {
   if (gps.encode(Serial2.read())) {
     if (gps.location.isValid()) {
       Blynk.virtualWrite(V8, gps.location.lat());
       Blynk.virtualWrite(V9, gps.location.lng());
     }
     if (gps.altitude.isValid()) {
       Blynk.virtualWrite(V10, gps.altitude.meters());
     }
     if (gps.speed.isValid()) {
       Blynk.virtualWrite(V11, gps.speed.kmph());
     }
     if (gps.date.isValid() && gps.time.isValid()) {
       String gpsDateTime = String(gps.date.year()) + "-" +
                            String(gps.date.month()) + "-" +
                            String(gps.date.day()) + " " +
                            String(gps.time.hour()) + ":" +
```

```
                              String(gps.time.minute()) + ":" +
                              String(gps.time.second());
        Blynk.virtualWrite(V12, gpsDateTime);
      }
    }
  }
  if (millis() > 5000 && gps.charsProcessed() < 10) {
    Blynk.virtualWrite(V9, "No GPS data received: check wiring");
  }
  delay(100);
}
 void forward() {
 analogWrite(enA, 255);
 analogWrite(enB, 255);
 digitalWrite(motor1A, HIGH);
 digitalWrite(motor1B, LOW);
 digitalWrite(motor2A, HIGH);
 digitalWrite(motor2B, LOW);
}
void backward() {
 analogWrite(enA, 255);
 analogWrite(enB, 255);
 digitalWrite(motor1A, LOW);
 digitalWrite(motor1B, HIGH);
 digitalWrite(motor2A, LOW);
 digitalWrite(motor2B, HIGH);
}
void left() {
 analogWrite(enA, 255);
 digitalWrite(motor1A, HIGH);
```

```
 digitalWrite(motor1B, LOW);
 analogWrite(enB, 255);
 digitalWrite(motor2A, LOW);
 digitalWrite(motor2B, HIGH);
}
void right() {
 analogWrite(enA, 255);
 digitalWrite(motor1A, LOW);
 digitalWrite(motor1B, HIGH);
 analogWrite(enB, 255);
 digitalWrite(motor2A, HIGH);
 digitalWrite(motor2B, LOW);
}
void stopMotorsDrive() {
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 digitalWrite(motor1A, LOW);
 digitalWrite(motor1B, LOW);
 digitalWrite(motor2A, LOW);
 digitalWrite(motor2B,LOW);
}
/*********
  Rui Santos
  Complete instructions https://RandomNerdTutorials.com/esp32-cam-projectsebook/

  Permission is hereby granted, free of charge, to any person obtaining a copy of th
  associated documentation files.
  The above copyright notice and this permission notice shall be included in

  all copies or substantial portions of the Software.
```

```
*********/

#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"             // disable brownout problems
#include "soc/rtc_cntl_reg.h"    // disable brownout problems
#include "esp_http_server.h"
#include <ESP32Servo.h>

// Replace with your network credentials
const char* ssid = "Karthika";
const char* password = "Karthika@2020";

#define PART_BOUNDARY "123456789000000000000987654321"

#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM_B
//#define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #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      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
```

```
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
  #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

#elif defined(CAMERA_MODEL_M5STACK_PSRAM_B)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     22
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
```

```
#else
  #error "Camera model not selected"
#endif

#define SERVO_1      14
#define SERVO_2      15

#define SERVO_STEP   5

Servo servoN1;
Servo servoN2;
Servo servo1;
Servo servo2;

int servo1Pos = 0;
int servo2Pos = 0;

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="

PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-
Length: %u\r\n\r\n";

httpd_handle_t camera_httpd = NULL;
httpd_handle_t stream_httpd = NULL;

static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<html>
  <head>
```

```
<title>ESP32-CAM Robot</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
  body { font-family: Arial; text-align: center; margin:0px auto; padding-top: 3
  table { margin-left: auto; margin-right: auto; }
  td { padding: 8 px; }
  .button {
    background-color: #2f4468;
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
  }
  img {  width: auto ;
    max-width: 100% ;
    height: auto ;
  }
</style>
```

```
  </head>
  <body>
    <h1>ESP32-CAM Pan and Tilt</h1>
    <img src="" id="photo" >
    <table>
      <tr><td colspan="3" align="center"><button class="button" onmousedown="toggleC

      <tr><td align="center"><button class="button" onmousedown="toggleCheckbox('lef

      </td><td align="center"><button class="button" onmousedown="toggleCheckbox('ri

      <tr><td colspan="3" align="center"><button class="button" onmousedown="toggleC
     ontouchstart="toggleCheckbox('down');">Down</button></td></tr>
     </table>
   <script>
   function toggleCheckbox(x) {
     var xhr = new XMLHttpRequest();
     xhr.open("GET", "/action?go=" + x, true);
     xhr.send();
   }
   window.onload = document.getElementById("photo").src = window.location.href.slice
  </script>
  </body>
</html>
)rawliteral";

static esp_err_t index_handler(httpd_req_t *req){
  httpd_resp_set_type(req, "text/html");
  return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}
```

```
static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
```

```
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}
```

```
static esp_err_t cmd_handler(httpd_req_t *req){
  char*  buf;
  size_t buf_len;
  char variable[32] = {0,};

  buf_len = httpd_req_get_url_query_len(req) + 1;
  if (buf_len > 1) {
    buf = (char*)malloc(buf_len);
    if(!buf){
      httpd_resp_send_500(req);
      return ESP_FAIL;
    }
    if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
      if (httpd_query_key_value(buf, "go", variable, sizeof(variable)) == ESP_OK) {
      } else {
        free(buf);
        httpd_resp_send_404(req);
        return ESP_FAIL;
      }
    } else {
      free(buf);
      httpd_resp_send_404(req);
      return ESP_FAIL;
    }
    free(buf);
  } else {
    httpd_resp_send_404(req);
    return ESP_FAIL;
  }
```

```
sensor_t * s = esp_camera_sensor_get();
//flip the camera vertically
//s->set_vflip(s, 1);          // 0 = disable , 1 = enable
// mirror effect
//s->set_hmirror(s, 1);          // 0 = disable , 1 = enable

int res = 0;

if(!strcmp(variable, "up")) {
  if(servo1Pos <= 170) {
    servo1Pos += 10;
    servo1.write(servo1Pos);
  }
  Serial.println(servo1Pos);
  Serial.println("Up");
}
else if(!strcmp(variable, "left")) {
  if(servo2Pos <= 170) {
    servo2Pos += 10;
    servo2.write(servo2Pos);
  }
  Serial.println(servo2Pos);
  Serial.println("Left");
}
else if(!strcmp(variable, "right")) {
  if(servo2Pos >= 10) {
    servo2Pos -= 10;
    servo2.write(servo2Pos);
  }
  Serial.println(servo2Pos);
```

```
    Serial.println("Right");
  }
  else if(!strcmp(variable, "down")) {
    if(servo1Pos >= 10) {
      servo1Pos -= 10;
      servo1.write(servo1Pos);
    }
    Serial.println(servo1Pos);
    Serial.println("Down");
  }
  else {
    res = -1;
  }

  if(res){
    return httpd_resp_send_500(req);
  }

  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
```

```
  };

  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 1;
  config.ctrl_port += 1;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
  servo1.setPeriodHertz(50);    // standard 50 hz servo
  servo2.setPeriodHertz(50);    // standard 50 hz servo
  servoN1.attach(2, 1000, 2000);
```

```
servoN2.attach(13, 1000, 2000);

servo1.attach(SERVO_1, 1000, 2000);
servo2.attach(SERVO_2, 1000, 2000);

servo1.write(servo1Pos);
servo2.write(servo2Pos);

Serial.begin(115200);
Serial.setDebugOutput(false);

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;

if(psramFound()){
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  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;
}
// Wi-Fi connection
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

Serial.print("Camera Stream Ready! Go to: http://");
Serial.println(WiFi.localIP());
```

```
  // Start streaming web server
  startCameraServer();
}

void loop() {

}
```