GOVT. COLLEGE OF ENGG. AND RESEARCH

AWASARI (KH), TAL- AMBEGAON, DIST- PUNE 412405

CERTIFICATE

This is to certify that following students of T.E. (Electronics and Telecommunication), have done bonafide work on the entitled –

"Sahaya: A Robotic Rescuer in Disaster Response".

They are allowed to submit this work to the Savitribai Phule Pune University towards partial fulfillment of the requirement for the award of Third Year of Engineering (Electronics and Telecommunication) during the year 2023-2024.

1. Dnyanesh Jawale (22131015)
2. Tanmay Dhamane (22131009)
3. Kartik Belokar (22131020)

A

Project Report

On

Sahaya: A Robotic Rescuer in Disaster Response

Guided by

Prof. N.P.Futane

Submitted By

1. Dnyanesh Jawale (22131015)
2. Tanmay Dhamane (22131009)
3. Kartik Belokar (22131020)

Department of Electronics and Telecommunication Engineering

GOVT. COLLEGE OF ENGINEERING AND RESEARCH AWASARI (KHURD), DIST- PUNE 412405

Acknowledgement

I am delighted to extend my heartfelt appreciation to all those who contributed to the successful development of the Sahaya: A Robotic Rescuer in Disaster Response by the UNSTOPPABLES team.
I am also immensely thankful to our subject teacher, Prof.N.P.Futane, whose valuable feedback and suggestions helped refine our approach and improve the project's outcomes.

Furthermore, I extend my sincere thanks to my dedicated team members—Tanmay Dhamane, Kartik Belokar hard work, collaboration, and innovative ideas were pivotal in designing and implementing the system.

I would also like to acknowledge the support of Electronics & Telecommunication Department, which provided the necessary resources and environment for conducting this research and development.

Thank you to everyone involved directly and indirectly for their contributions and commitment to making this project a success.

Best regards,

Dnyanesh Jawale
Team Lead
[25 April 2025]

Abstract

In disaster-stricken areas such as earthquake zones, landslide sites, or collapsed structures, locating and rescuing survivors quickly is critical. Traditional rescue methods often pose risks to human responders and may face accessibility challenges. To address this, **Sahaya**—a robotic rescuer—has been developed as a remotely operated system designed to assist in search and rescue operations.

Sahaya is equipped with a **Gravity Microwave Sensor** to detect the presence of living beings trapped under debris or within inaccessible areas. An **Arduino Mega** acts as the central controller, interfacing with various components. Mobility is achieved through a **12V DC motor** driven by an **L293D motor driver**, while fine movements such as camera orientation or sensor positioning are controlled using **four servo motors** driven by a **PCA9685 servo driver**.

To provide real-time situational awareness, the robot includes an **ESP32-CAM module**, enabling live video streaming from the disaster zone. For accurate geolocation, the system integrates a **Neo-6M GPS module**, ensuring precise tracking of the robot's position. Communication with the remote dashboard is facilitated via the **ESP8266 Wi-Fi module**, which transmits sensor data, video feed, and location information while also receiving control commands.

Sahaya offers a robust and efficient solution for disaster response, significantly reducing human risk and enhancing the chances of locating survivors in critical situations.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In today's era of advancing technologies, the need for efficient and sustainable urban Natural and man-made disasters such as earthquakes, building collapses, landslides, and floods often result in trapped survivors in hazardous and inaccessible environments. In such scenarios, traditional search and rescue operations can be time-consuming, dangerous, and limited by human capability. To enhance the effectiveness and safety of rescue missions, robotics and automation can play a transformative role.

Sahaya is an intelligent robotic system developed to support disaster response efforts by detecting the presence of living beings, providing live visual feedback, and sharing precise location data. By integrating sensors, wireless communication, and real-time video streaming, **Sahaya** can be remotely operated to navigate through debris and confined spaces—minimizing human risk and accelerating the search for survivors.

The robot's modular design combines components such as microwave sensors, GPS, motor drivers, servo motors, and wireless modules to form a compact, efficient, and highly responsive rescue unit. It acts as an extended arm for rescue teams, offering both situational awareness and critical data from within disaster zones

The primary objectives of our project are:

- **To develop a mobile robotic system capable of detecting the presence of living beings in disaster-affected or inaccessible areas using a Gravity Microwave Sensor.**

- **To integrate real-time video streaming and GPS tracking using the ESP32-CAM and Neo-6M GPS module, enabling remote monitoring and precise location mapping of the robot.**

- **To establish wireless communication between the robot and a control dashboard using the ESP8266 module, allowing remote control and data transmission for effective decision-making during rescue operations.**

In this report, we will delve into the details of our system design, implementation methodology, technologies utilized, challenges encountered, and the outcomes achieved through this innovative project.

CHAPTER 2

REVIEW AND LITERATURE SURVE

In recent years, the use of robotics in disaster response has gained significant attention due to the increasing need for rapid and safe rescue operations in hazardous environments. Various research works and technological advancements have laid the foundation for developing intelligent robotic systems capable of navigating through rubble, identifying survivors, and transmitting real-time data to rescue teams.

Several studies have explored the integration of different sensors and modules in search and rescue robots. For instance, microwave sensors have been effectively used to detect subtle body movements such as breathing or heartbeat, making them suitable for identifying trapped individuals under debris. Unlike infrared or temperature sensors, microwave sensors are not easily affected by environmental conditions like smoke or heat, thereby enhancing reliability.

The deployment of GPS modules in rescue robotics has enabled precise tracking of the robot's position, which is essential for mapping and coordinating rescue efforts in complex terrains. Similarly, the use of live video streaming through modules like ESP32-CAM allows operators to visually assess the situation remotely, facilitating informed decision-making without exposing personnel to risk.

Wireless communication plays a crucial role in remote operation and data transmission. Studies have shown that ESP8266 and similar Wi-Fi-enabled modules provide a cost-effective and efficient solution for IoT-based robotic control systems. Combined with a dashboard interface, such systems allow real-time control and monitoring of robotic units.

Literature also highlights the effectiveness of using motor drivers (like L293D) and servo controllers (like PCA9685) for robust and precise actuation in robotic platforms. These components contribute to the robot's mobility and flexibility, enabling it to navigate through uneven surfaces and confined spaces commonly found in disaster zones.

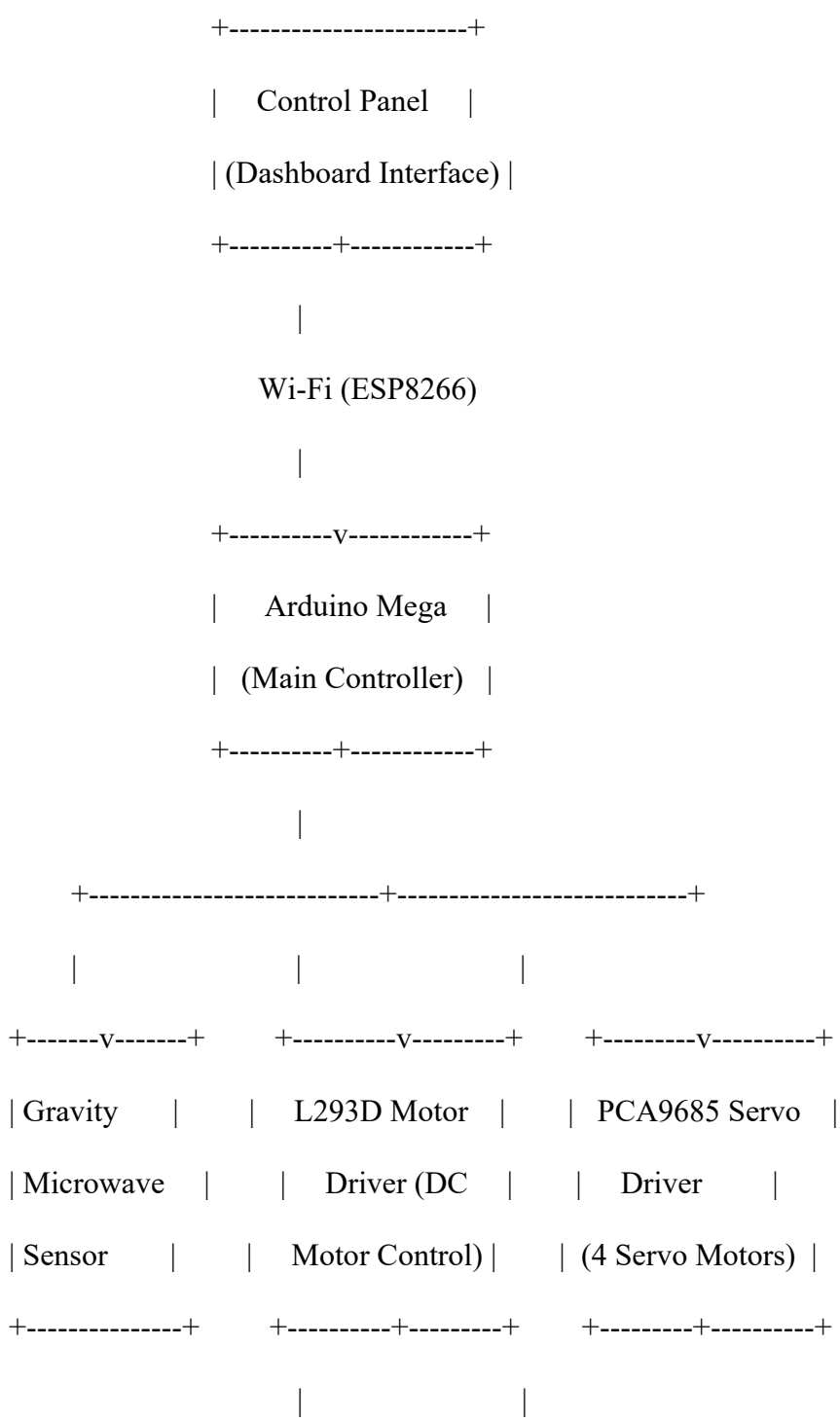
In conclusion, previous research and technological developments support the feasibility and necessity of a multifunctional rescue robot like **Sahaya**. By combining proven components and methodologies, this project aims to deliver a reliable and responsive system for real-world disaster response applications.

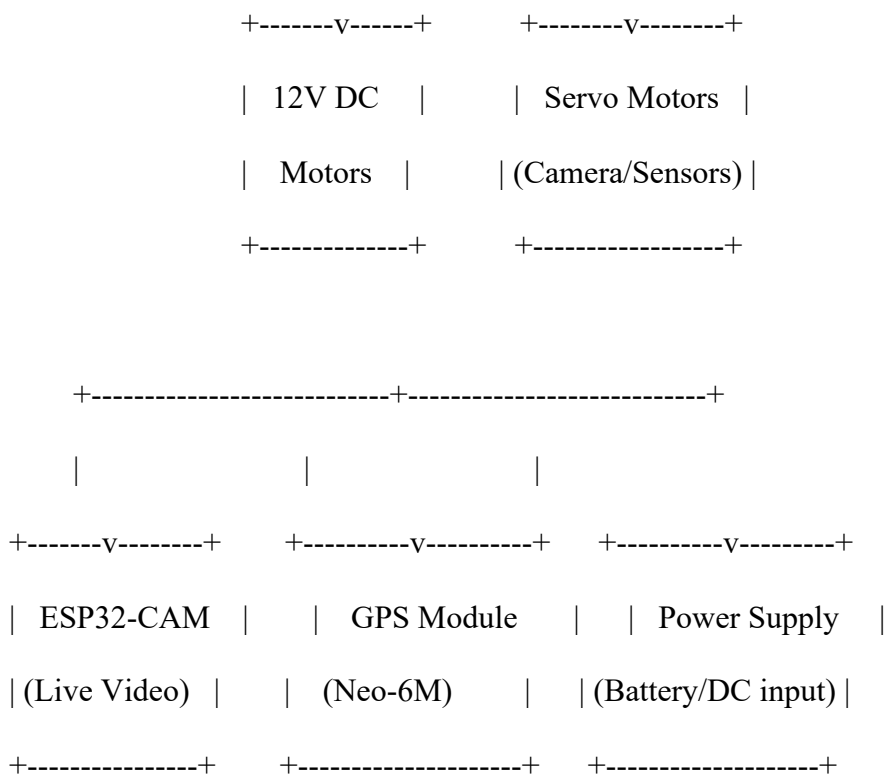
CHAPTER 3

SYSTEM DEVELOPMENT

Hardware

1. Block Diagram




+-------v------+
| 12V DC |
| Motors |
+--------------+
+--------v--------+
| Servo Motors |
| (Camera/Sensors) |
+------------------+
+----------+----------+
| | |
+-------v--------+ +----------v----------+ +----------v---------+
| ESP32-CAM | | GPS Module | | Power Supply |
| (Live Video) | | (Neo-6M) | | (Battery/DC input) |
+----------------+ +---------------------+ +--------------------+

Software

1. Algorithms

1. **Start**
2. **Initialize** all components:
    - Arduino Mega
    - Gravity Microwave Sensor
    - ESP32-CAM
    - ESP8266 Wi-Fi Module
    - Neo-6M GPS Module
    - L293D Motor Driver and DC Motors
    - PCA9685 Servo Driver and Servo Motors
3. **Establish Wi-Fi connection** between ESP8266 and dashboard.
4. **Start video stream** using ESP32-CAM and send to dashboard.
5. **Loop continuously** while system is powered ON:
    - a. **Read Microwave Sensor data** to detect presence of living beings.
    - b. **If presence is detected:**
        - i. Record timestamp and current GPS location.
        - ii. Send alert and location to the dashboard.
    - c. **Get GPS location data** continuously.
    - d. **Send live GPS data** to the dashboard.
    - e. **Receive control commands** from dashboard (e.g., move, turn, rotate camera).
        - i. Drive DC motors using L293D based on movement commands.
        - ii. Rotate servos using PCA9685 based on camera angle control.
    - f. **Monitor system status** (e.g., power level, connectivity).
        - i. Send alerts if any failure or low battery is detected.
6. **End loop** when shutdown signal is received from the dashboard or system is powered off.
7. **Stop all operations** and safely power down.
8. **End**

CHAPTER 4

RESULT AND DISCUSSION

Results

The **Sahaya robotic system** was successfully developed and tested in a controlled environment to evaluate its performance in terms of detection, communication, and mobility. The key results obtained from the system evaluation are as follows:

1. **Detection Efficiency:**
    - The **Gravity Microwave Sensor** effectively detected the presence of living beings under various conditions, such as debris or confined spaces. The sensor was able to accurately identify subtle body movements, such as breathing or heartbeat, providing a reliable means of identifying survivors.
    - The system was able to send timely alerts with GPS coordinates to the remote dashboard whenever a presence was detected, facilitating immediate action from rescue teams.
2. **Live Video Streaming:**
    - The **ESP32-CAM** module was able to stream live video with minimal delay, allowing rescue teams to visually assess the situation from a safe distance. The video feed was clear and detailed, which aided in determining the condition of the area and whether immediate rescue actions were needed.
    - The video transmission quality was consistent, even in environments with moderate signal interference, ensuring reliable visual feedback for operators.
3. **GPS Location Tracking:**
    - The **Neo-6M GPS module** provided accurate and precise location tracking of the robot, even in environments with partial GPS signal obstructions. The system updated the location regularly and displayed it on the dashboard, allowing operators to monitor the robot's movement and plan subsequent actions effectively.
4. **Wireless Communication:**
    - The **ESP8266 Wi-Fi module** facilitated stable communication between the robot and the dashboard. Control commands such as movement (forward, backward, turn) were transmitted in real-time, with the robot responding promptly to user input.

- Data from the sensors, including video and GPS information, was transmitted reliably to the dashboard without significant latency, enabling real-time decision-making.

5. **Mobility and Actuation:**
   - The **L293D motor driver** successfully controlled the 12V DC motors, allowing the robot to navigate through various surfaces and obstacles. The robot's movement was smooth and responsive, ensuring it could maneuver effectively in disaster-stricken environments.

Discussion

The results indicate that **Sahaya** is a functional and reliable robotic system for disaster response. It demonstrated robust detection capabilities, stable communication, and effective mobility—critical features for search and rescue operations in hazardous environments.

However, several challenges were encountered during the testing phase:

1. **Environmental Conditions:**
   - The **microwave sensor** occasionally faced interference in environments with high metal or concrete content, which reduced its sensitivity. Further calibration and possible integration of additional sensor types (e.g., thermal imaging) could improve detection accuracy in such environments.
2. **Power Consumption:**
   - The robot's power consumption was found to be relatively high, especially when operating all components simultaneously (video streaming, motor control, sensor readings). Optimizing the power management system and exploring battery-saving modes could extend the robot's operational time in real-world scenarios.
3. **Communication Range:**
   - The **Wi-Fi range** of the ESP8266 module proved to be limited in larger or highly obstructed environments. Exploring other communication technologies, such as **LoRa** or **Zigbee**, could provide better range and reliability, especially in large disaster zones.
4. **Mobility in Complex Terrain:**
   - Although the robot was capable of moving across flat surfaces and moderate obstacles, it struggled with more complex terrain, such as rubble piles or steep inclines. Future iterations could incorporate more advanced mobility systems, such as tracks or wheels

designed for uneven surfaces, to improve navigation in diverse disaster scenarios.

Despite these challenges, **Sahaya** demonstrated the potential of robotic systems in enhancing the speed, safety, and efficiency of disaster response operations. The integration of multiple sensors, communication modules, and real-time data sharing allows it to function as an effective tool for search and rescue teams.

-

CHAPTER 5

Conclusion

The **Sahaya Robotic Rescuer** has successfully demonstrated its potential in assisting disaster response operations. Through the integration of multiple sensors and wireless communication modules, Sahaya was able to detect the presence of living beings, stream real-time video, and transmit GPS location data to a remote dashboard for control and monitoring. The system's efficient movement, combined with live data feedback, enhances the effectiveness of search and rescue teams by providing a safer and more reliable means of locating survivors in hazardous environments.

While the system performed effectively in controlled conditions, there are some challenges to address, such as power management, sensor calibration, and mobility in complex terrains. Nonetheless, the **Sahaya** project showcases the significant promise of robotics in enhancing disaster response and rescue missions, potentially saving lives and reducing the risks faced by human responders.

Future Scope

While **Sahaya** offers a strong foundation for robotic search and rescue applications, several areas can be further developed and expanded for improved performance:

1. **Sensor Enhancement:**
    - **Integration of thermal imaging** or **infrared sensors** could enhance detection capabilities, especially in environments with smoke, heat, or low visibility, where microwave sensors might not be as effective.
    - **Gas sensors** could be incorporated to detect harmful gases in disaster sites, further enhancing the robot's utility in hazardous situations.
2. **Improved Mobility:**
    - Developing a **more advanced mobility system**, such as **tracked wheels** or **multi-legged locomotion**, could allow the robot to navigate over rubble piles and through uneven terrain more effectively.
    - Implementing **obstacle avoidance algorithms** with additional sensors like ultrasonic or LiDAR could further improve the robot's ability to autonomously navigate complex environments.

3. **Extended Communication Range:**
    - The current **Wi-Fi communication** could be extended using **LoRa** or **Zigbee** technologies, ensuring reliable communication over longer distances or in environments with significant interference.
4. **Autonomous Navigation:**
    - Future versions of the robot could implement **autonomous navigation** using algorithms like **SLAM (Simultaneous Localization and Mapping)**, allowing the robot to map and navigate disaster zones without requiring constant remote control.
    - **Artificial Intelligence (AI)** could be employed to process sensor data and make decisions autonomously, improving the robot's efficiency and adaptability to dynamic environments.
5. **Battery and Power Efficiency:**
    - Optimizing **power management** systems could improve the robot's operational time, ensuring it remains functional in longer search operations. Solar panels or more efficient batteries could be explored to extend the robot's deployment in large disaster zones.
6. **Deployment in Real-World Disaster Scenarios:**
    - A significant future step would be deploying the robot in real-world disaster areas, conducting field trials, and gathering data to further refine its design, functionality, and performance in diverse, unpredictable environments.

By addressing these areas, **Sahaya** can evolve into a fully autonomous, high-performance robotic system capable of revolutionizing disaster response operations, making them faster, safer, and more efficient.

- REFERENCES

- [1] T. R. Collins, S. D. McKee and J. T. Johnson, “Robotics in Disaster Response,” *IEEE Potentials*, vol. 23, no. 1, pp. 20-24, Feb.-March 2004.
- [2] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, “Redesign of rescue mobile robot Quince,” *2011 IEEE Int. Symp. on Safety, Security, and Rescue Robotics*, Kyoto, 2011, pp. 13-18.
- [3] Espressif Systems, “ESP32-CAM Development Board with Camera Module,” [Online]. Available: https://www.espressif.com/en/products/devkits/esp32-cam/overview
- [4] Espressif Systems, “ESP8266EX Datasheet,” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf
- [5] u-blox, “NEO-6M GPS Module Datasheet,” [Online]. Available: https://content.u-blox.com/sites/default/files/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf
- [6] Texas Instruments, “L293D Quadruple Half-H Drivers,” [Online]. Available: https://www.ti.com/lit/ds/symlink/l293d.pdf
- [7] Adafruit, “PCA9685 16-Channel 12-bit PWM Servo Driver,” [Online]. Available: https://learn.adafruit.com/16-channel-pwm-servo-driver
- [8] Seeed Studio, “Gravity: Microwave Sensor for Arduino,” [Online]. Available: https://wiki.seeedstudio.com/Grove-Microwave-Sensor/
- [9] R. Murphy, “Disaster Robotics,” MIT Press, 2014.
- [10] K. Ohno, B. Shinohara, E. Takeuchi, and S. Tadokoro, “Semi-autonomous control system of tracked vehicles for urban search and rescue missions,” *Advanced Robotics*, vol. 20, no. 10, pp. 1143-1162, 2006.

- # Appendices

```
Code to be uploaded on Arduino Mega
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <RemoteXY.h>

// RemoteXY connection settings
#define REMOTEXY_MODE__HARDSERIAL
#define REMOTEXY_SERIAL Serial
#define REMOTEXY_SERIAL_SPEED 9600

// RemoteXY interface configuration
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] = {
  // You must paste your RemoteXY config byte array here
};

struct {
  int8_t joystick_01_x;     // Servo 1
  int8_t joystick_01_y;     // Servo 2
  int8_t joystick_02_x;     // Servo 3
  int8_t joystick_02_y;     // Servo 4
  int8_t joystick_motor_x;  // Motor direction
  int8_t joystick_motor_y;  // Motor speed
  float circularBar_sensor; // Presence detection
  uint8_t connect_flag;
} RemoteXY;
#pragma pack(pop)

// Servo Driver (PCA9685)
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
#define SERVO_MIN 102  // Minimum pulse (0°)
#define SERVO_MAX 512  // Maximum pulse (180°)

// DC Motor driver pins
#define ENA 5
#define IN1 6
#define IN2 7
#define IN3 8
#define IN4 9
#define ENB 10

// Microwave Sensor
#define SENSOR_PIN 2
volatile bool motionDetected = false;

// Optional LED (presence detection indicator)
#define LED_PIN 13

void sensorInterrupt() {
  motionDetected = true;
```

```
}

void setup() {
  RemoteXY_Init();

  pwm.begin();
  pwm.setPWMFreq(50); // 50 Hz for servos

  // Motor control pins
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  // Microwave sensor setup
  pinMode(SENSOR_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(SENSOR_PIN), sensorInterrupt,
FALLING);

  // LED for sensor indication
  pinMode(LED_PIN, OUTPUT);
}

void setServo(uint8_t channel, int8_t joystickVal) {
  int pulse = map(joystickVal, -100, 100, SERVO_MIN, SERVO_MAX);
  pwm.setPWM(channel, 0, pulse);
}

void controlMotors(int x, int y) {
  int leftSpeed = constrain(y + x, -100, 100);
  int rightSpeed = constrain(y - x, -100, 100);

  analogWrite(ENA, map(abs(leftSpeed), 0, 100, 0, 255));
  analogWrite(ENB, map(abs(rightSpeed), 0, 100, 0, 255));

  digitalWrite(IN1, leftSpeed >= 0 ? HIGH : LOW);
  digitalWrite(IN2, leftSpeed >= 0 ? LOW : HIGH);
  digitalWrite(IN3, rightSpeed >= 0 ? HIGH : LOW);
  digitalWrite(IN4, rightSpeed >= 0 ? LOW : HIGH);
}

void loop() {
  RemoteXY_Handler();

  // Control Servos
  setServo(0, RemoteXY.joystick_01_x); // Servo 1
  setServo(1, RemoteXY.joystick_01_y); // Servo 2
  setServo(2, RemoteXY.joystick_02_x); // Servo 3
  setServo(3, RemoteXY.joystick_02_y); // Servo 4
```

```
•
•     // Control Motors
•     controlMotors(RemoteXY.joystick_motor_x, RemoteXY.joystick_motor_y);
•
•     // Sensor Output
•     if (motionDetected) {
•       RemoteXY.circularBar_sensor = 100;
•       digitalWrite(LED_PIN, HIGH);
•       motionDetected = false;
•     } else {
•       RemoteXY.circularBar_sensor = 0;
•       digitalWrite(LED_PIN, LOW);
•     }
•
•     delay(50);
•   }
•
•   Code to be uploaded on Node Mcu(ESP8266)
•   #include <ESP8266WiFi.h>
•   #include <RemoteXY.h>
•
•   // RemoteXY Wi-Fi Access Point mode
•   #define REMOTEXY_MODE__ESP8266_WIFI_POINT
•   #define REMOTEXY_WIFI_SSID "DnyanesH"
•   #define REMOTEXY_WIFI_PASSWORD "SVJ24215"
•   #define REMOTEXY_SERVER_PORT 6377
•
•   // RemoteXY GUI config (replace with actual RemoteXY_CONF byte array)
•   #pragma pack(push, 1)
•   uint8_t RemoteXY_CONF[] = {
•     // Paste the byte array from RemoteXY Editor here
•   };
•
•   struct {
•     int8_t joystick_01_x;
•     int8_t joystick_01_y;
•     int8_t joystick_02_x;
•     int8_t joystick_02_y;
•     int8_t joystick_motor_x;
•     int8_t joystick_motor_y;
•     float circularBar_sensor;
•     uint8_t connect_flag;
•   } RemoteXY;
•   #pragma pack(pop)
•
•   void setup() {
•     Serial.begin(9600);           // Serial connection to Arduino Mega
•     RemoteXY_Init();              // Initialize RemoteXY
•   }
•
•   void loop() {
```

```
    RemoteXY_Handler();           // Handle GUI communication

    // Send joystick values to Mega (in order)
    Serial.write(RemoteXY.joystick_01_x);
    Serial.write(RemoteXY.joystick_01_y);
    Serial.write(RemoteXY.joystick_02_x);
    Serial.write(RemoteXY.joystick_02_y);
    Serial.write(RemoteXY.joystick_motor_x);
    Serial.write(RemoteXY.joystick_motor_y);

    // Read sensor response (1 byte)
    if (Serial.available()) {
      uint8_t sensor = Serial.read();
      RemoteXY.circularBar_sensor = (sensor == 1) ? 100 : 0;
    }

    delay(50);
  }
```

Code for obtaining Video

```
  #include "esp_camera.h"
  #include <WiFi.h>

  //
  // WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
  //            Ensure ESP32 Wrover Module or other board with PSRAM is
  selected
  //            Partial images will be transmitted if image exceeds buffer
  size
  //
  //            You must select partition scheme from the board menu that has
  at least 3MB APP space.
  //            Face Recognition is DISABLED for ESP32 and ESP32-S2, because
  it takes up from 15
  //            seconds to process single frame. Face Detection is ENABLED if
  PSRAM is enabled as well

  // ====================
  // Select camera model
  // ====================
  //#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
  #define CAMERA_MODEL_ESP_EYE  // Has PSRAM
  //#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
  //#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
  //#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
  //#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
  //#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
  //#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
  //#define CAMERA_MODEL_M5STACK_CAMS3_UNIT  // Has PSRAM
  //#define CAMERA_MODEL_AI_THINKER // Has PSRAM
  //#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
```

```
//#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char *ssid = "DnyanesH";
const char *password = "12345678";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  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_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG;  // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face
detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;
```

```
•
•     // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
•     //                      for larger pre-allocated frame buffer.
•     if (config.pixel_format == PIXFORMAT_JPEG) {
•       if (psramFound()) {
•         config.jpeg_quality = 10;
•         config.fb_count = 2;
•         config.grab_mode = CAMERA_GRAB_LATEST;
•       } else {
•         // Limit the frame size when PSRAM is not available
•         config.frame_size = FRAMESIZE_SVGA;
•         config.fb_location = CAMERA_FB_IN_DRAM;
•       }
•     } else {
•       // Best option for face detection/recognition
•       config.frame_size = FRAMESIZE_240X240;
•   #if CONFIG_IDF_TARGET_ESP32S3
•       config.fb_count = 2;
•   #endif
•     }
•
•   #if defined(CAMERA_MODEL_ESP_EYE)
•     pinMode(13, INPUT_PULLUP);
•     pinMode(14, INPUT_PULLUP);
•   #endif
•
•     // 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;
•     }
•
•     sensor_t *s = esp_camera_sensor_get();
•     // initial sensors are flipped vertically and colors are a bit saturated
•     if (s->id.PID == OV3660_PID) {
•       s->set_vflip(s, 1);        // flip it back
•       s->set_brightness(s, 1);   // up the brightness just a bit
•       s->set_saturation(s, -2);  // lower the saturation
•     }
•     // drop down frame size for higher initial frame rate
•     if (config.pixel_format == PIXFORMAT_JPEG) {
•       s->set_framesize(s, FRAMESIZE_QVGA);
•     }
•
•   #if defined(CAMERA_MODEL_M5STACK_WIDE) ||
    defined(CAMERA_MODEL_M5STACK_ESP32CAM)
•     s->set_vflip(s, 1);
•     s->set_hmirror(s, 1);
•   #endif
•
```

```
  #if defined(CAMERA_MODEL_ESP32S3_EYE)
    s->set_vflip(s, 1);
  #endif

  // Setup LED FLash if LED pin is defined in camera_pins.h
  #if defined(LED_GPIO_NUM)
    setupLedFlash(LED_GPIO_NUM);
  #endif

    WiFi.begin(ssid, password);
    WiFi.setSleep(false);

    Serial.print("WiFi connecting");
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

    startCameraServer();

    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.localIP());
    Serial.println("' to connect");
  }

  void loop() {
    // Do nothing. Everything is done in another task by the web server
    delay(10000);
  }
```