A PROJECT REPORT

ON

OFFROAD SURVEILLANCE UGV

under the kind guidance of

Prof. Deepak Kedia

Submitted By-

Arjun Panwar

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GURU JAMBESHWAR UNIVERSITY OF

SCIENCE AND TECHNOLOGY,

HISAR -HARYANA 125001

ACKNOWLEDGEMENT

First of all, I express my deepest sense of gratitude to Almighty for bestowing me with opportunities and instincts for pursuing this career. I am highly grateful to Professor Deepak Kedia, Department of Electronics and Communication Engineering for providing me this opportunity to carry out the present work under his kind guidance. A special thanks for his dynamic co-operation and constant encouragement that helped me in carrying this work in a systematic manner. Without his keen supervision, it would have been impossible to complete the Project. And above all my sincere gratitude and thanks to my parents to whom I owe everything in life. Finally, I am indebted to each and everyone whosoever have contributed to this Project work, supported and helped me to implement it.

ARJUN PANWAR

ABSTRACT

We propose a cost-effective four-wheeled surveillance robot using an ESP 32 Camera module and a smartphone running on the Android Operating System. Surveillance robots typically consist of a video camera, a GPS module, and GSM radios. Android smartphones come with excellent hardware satisfying the above needs. This can be leveraged and used to advantage through APIs (Application Programming Interfaces) provided for the Android operating system. Moreover, the cost for building said robot using a smartphone is mitigated to a great extent. The robot can be controlled remotely from a PC using the internet and a microcontroller-smart phone interface residing on the robot. To capture and archive the real time video from the robot, the ESP 32 Cam module is used. The robot can be controlled based on visual feedback from the same smart phone. Four motors help achieve a zero turning radius.

CONTENTS

CHAPTER 1

1.1 Introduction

A remote-controlled surveillance robot is defined as any robot that is remotely controlled to capture images/video for specific purposes. Mobile robots that are controlled remotely have important rules in area of rescue and military as the human cannot record video safely in critical conditions and environments. These conditions and environments may be buildings where the fire breaks out, Areas with poisonous gases or harmful radiation and the places where there is an exchange of fire such as battlefield. This paper introduces design and implementation a surveillance tanked robot based on Wi-Fi protocol. The movement directions of the robotic tank are controlled by a GUI (Graphical User Interface) designed using visual studio development environment. The robot can transmit real-time video to the intended recipient. Moreover, the cost for building said robot using a smartphone is mitigated to a great extent. The robot can be controlled remotely from a PC using the internet and a microcontroller-smart phone interface residing on the robot. To capture and archive the real time video from the robot, the ESP 32 Cam module is used. The robot can be controlled based on visual feedback from the same smart phone. Four motors help achieve a zero turning radius.

1.2 Objective of Project

The objective is to develop an all-terrain offroad robot which is suited to provide an act of continuous surveillance in hazardous environment and every terrain. The robot is capable to record the real-time streaming in day time and night time as well through wireless camera. Those movements of the robot are controlled manually at the user end using a html-based GUI. This robot reduces human intervention directly in a hazardous place where continuous supervision and security is necessary. Also, chassis of robot is developed using the hollow pipes and designed in such a way that pipes also perform the suspension role, eliminating the need of separate suspension system. This ultimately results in reduce manufacturing cost of Robot.

1.3 Scope

This project of rugged all terrain robot offers a lot of scope for adding newer features. Since all image processing is done remotely, there are no resource constraints apart from the bandwidth of the network. We can program the robot such that it can detect objects and reach them on its own. Thus, we can make it completely autonomous. Also, with addition of GPS navigation and mapping software, the robot will have the capability of finding the best route possible to reach a certain location. There is also the option of adding sound processing to the remote computer, thus giving it greater surveillance capabilities. The possibilities are endless. This robot in its current state provides a platform for further research into improving its capabilities.

CHAPTER 2

2.1 Existing Technology

There are variety of advanced technologies exist in the field of robotics. Specifically for surveillance robots, but the main problem that resides with these technologies is the 'cost'. The cost associated with the design and manufacturing of these surveillance robots is very high consequently cannot be beared by everyone. So, a system is needed which should perform all the functionalities of surveillance robot and should be economical. Moreover, one more issue associated with this technology is the terrain dependency. A normal surveillance robot only performs well on plane surfaces. So, to counter the rough terrains a rigid all terrain robot is needed.

2.2 Area of improvement

To counter both of the problems associated with the current technology that is 'Cost' and 'Terrain dependency' a system should be developed that should be cheap and have ability to travel effectively on any of given terrain. So, we designed the chassis of robot in such a way that it has inbuild suspensions. That is instead of mounting a separate suspension system we designed the chassis in such a way that it has inbuilt suspension system. Secondly height of this robot is increased so that it can easily overcome obstacles. The increased height and inbuilt suspension system let the robot to travel any type of terrain effectively. The high ground clearance of robot also makes it all terrain vehicle. The chassis design and elimination of suspension system effectively help us to achieve our task of cost cutting

2.3 Proposed System

Our system consists of a remote system(mobile/computer) and a robot. The robot is controlled by a user sitting at the remote system, over the internet. The user controls the robot by sending control signals to the Android smartphone. The smartphone then forwards these signals to the ESP 32, which then with the help of LN 298 Motor driver moves the robot in the required direction. The camera on the Camera module is used to send video feedback to the remote user simultaneously over the internet. This enables the user to navigate the robot remotely.

The system on ground level is divided into two modules-

1) The robot

2) Controller System

The first module consists of the complete robot moving on the ground. And the controller System module consist of the remote user controlling the movement of robot from a graphical user interface. Which can be opened on any web browser using an unique IP address associated with it.

CHAPTER 3

Hardware Required

- The ESP 32 Camera Module
- LM298 Motor Driver
- Camera
- Gear motor (60 RPM each).
- Lithium cell
- Battery holder
- Charging module
- Connecting wires
- Tyers
- Pipes for chassis
- 2103 USB to TTL Module

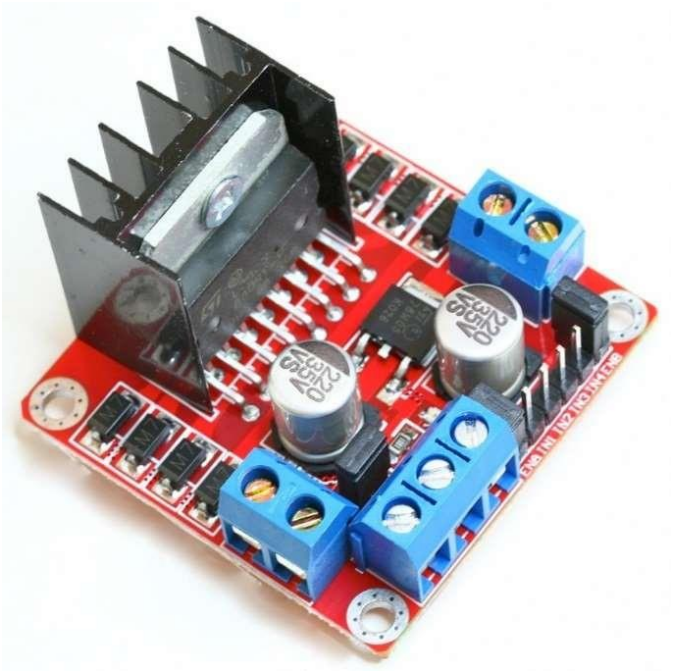
3.1 ESP 32 Camera Module

The ESP32-CAM is a full-featured microcontroller that also has an integrated video camera and microSD card socket. It's inexpensive and easy to use, and is perfect for IoT devices requiring a camera with advanced functions like image tracking and recognition.

The sample software distributed by Espresso includes a sketch that allows you to build a web-based camera with a sophisticated control panel. After you get the hang of programming the device, you'll find that it is very easy to use. Not bad for a board that costs about ten dollars, including the camera

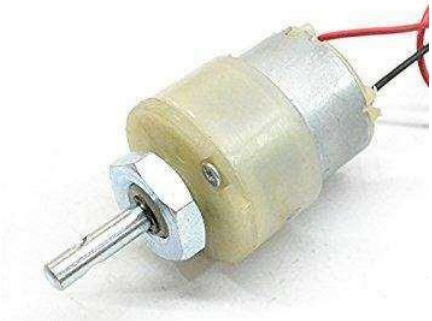
3.2 LM298 Motor Driver

L298N module is a high voltage, high current dual full-bridge motor driver module for controlling DC motor and stepper motor. It can control both the speed and rotation direction of two DC motors. This module consists of an L298 dual-channel H-Bridge motor driver IC. This module uses two techniques for the control speed and rotation direction of the DC motors. These are PWM – For controlling the speed and H-Bridge – For controlling rotation direction. These modules can control two DC motor or one stepper motor at the same time



3.3 Gear Motor

Gearmotors consist of a motor, gear, and bearings in a housing and are commonly used in applications that require a high amount of force (torque) at low speed. DC gearmotors require a battery power source and are used for precise variable-speed applications such as wheelchairs, hospital beds, lifts and medical tables, power seats, and other automotive applications. AC gearmotors operate off an AC power source and are often used in high-power applications, such as lifts, jack, and robotics. Gearmotor accessories like brackets, covers, and extension cables are used to protect, mount, and power gearmotors.

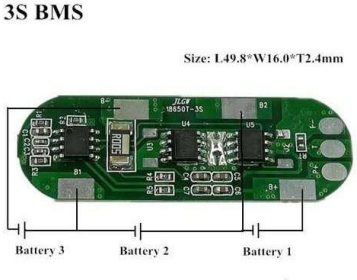


3.4 Lithium Cell

Lithium batteries are now being used increase, such as emergency lighting, where lead acid, cadmium and nickel batteries were used in the past. This article is a general introduction to batteries and the most popular lithium chemistries. It is aimed at readers in the lighting and building services sectors to enable them to make better informed choices when specifying batteries for emergency lighting.

3.5 Charging Module with Adapter

BMS (Battery Management System), as a key integral of battery electric vehicle and hybrid vehicle, is primarily composed of battery electronics (BE) and battery control unit (BCU), with the former responsible for collecting such data about battery as current voltage and temperature and transmitting them to BCU, and the latter BCU accountable for information exchanges with other control units. Adapter is used for charging the lithium cell.



CHAPTER 4

Working

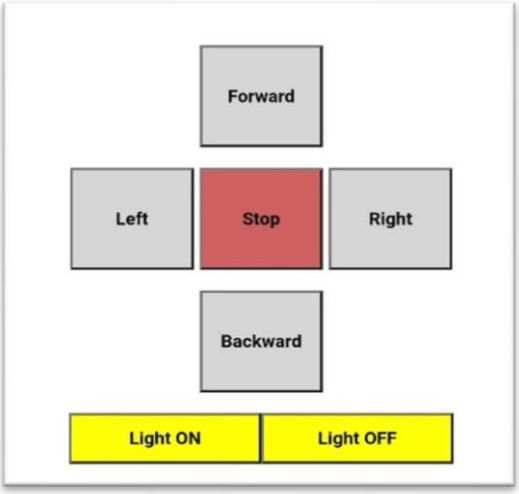
4.1 The working of robot is divided into three parts.

- **First Part**

The remote user who is controlling the robot will get get a Graphical user inteface as soon as he will enter the unique IP address in web browser. The GUI will look like as shown in figure.

It will consist of Four navigation touch buttons namely:

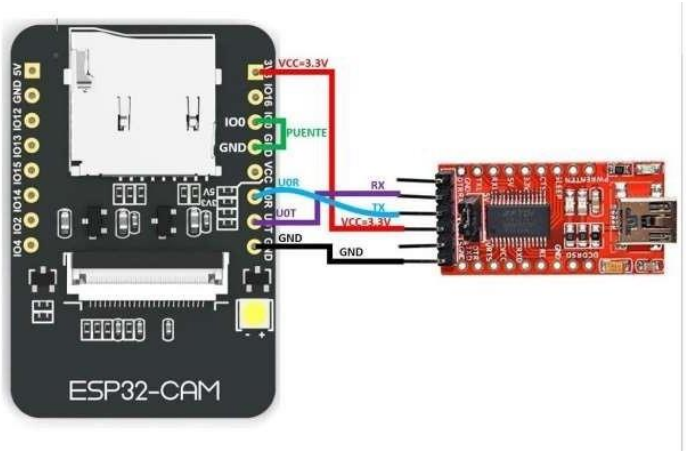
- Forward
- Backward
- Left
- Right
- two buttons for flash light on and off respectively.

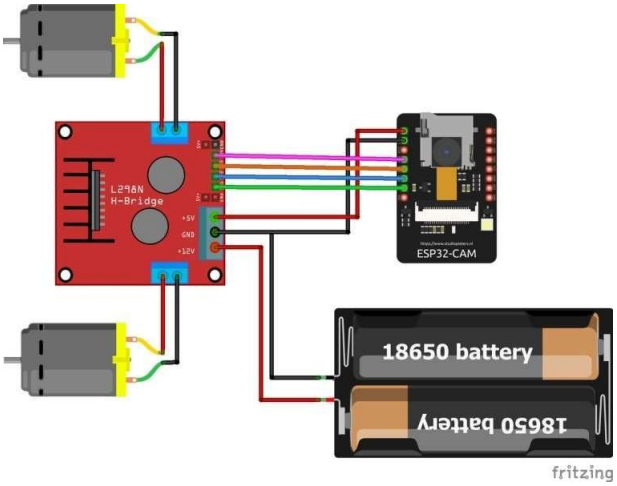


Beside this the GUI will consist of the window showing the live video streamed by robot.At the user end the user will provide input through GUI which will be given to robot through the channel established by wi fi.

- **Second Part**

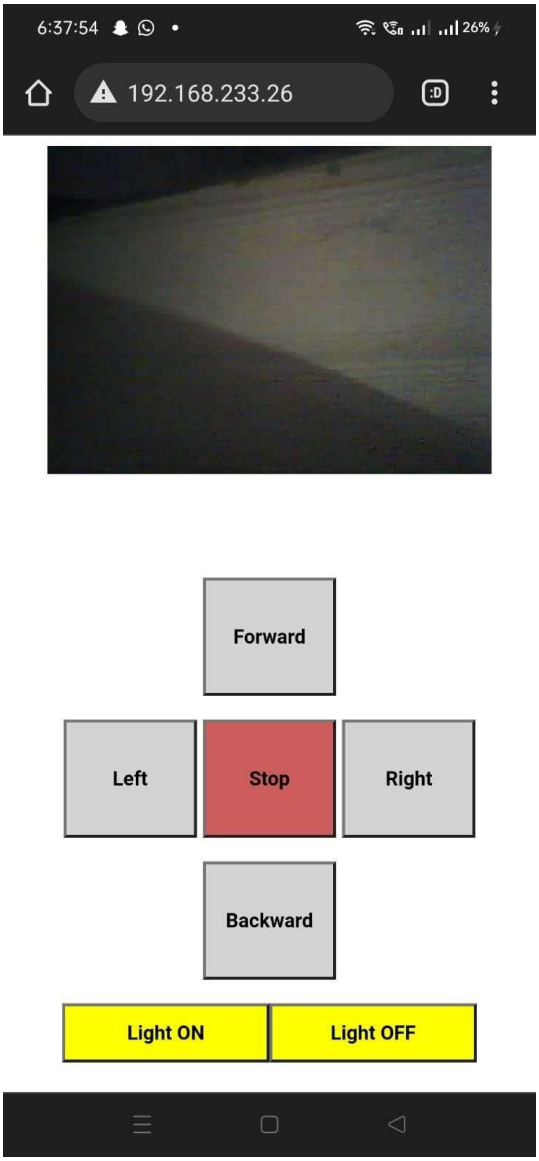
The input provided by user will be recieved by the ESP 32 cam module on the robot. At the robot level the whole system is powered by silicon cells. It will process the input and accordingly function the LM 298 motor driver which will further provide signals to the motors accordingly. The motors will function (forward, backward, turn) according to the signal received.





- **Third Part**

The third part consist of the providing output at the user level in the form of video. The camera mounted on ESP 32 module will record the real time video and with the help of established wi fi channel the live video will be streamed to the authorised user. GUI will consist of seprate window which will show the video streamed by the robot.



4.2 Code For programing

//ESP32 Camera Surveillance Car

```
#include "esp_camera.h"
#include <WiFi.h>
//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//            or another board which has PSRAM enabled
//
// Adafruit ESP32 Feather
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_M5STACK_PSRAM
#define CAMERA_MODEL_AI_THINKER
const char* ssid = "PBX"; //Enter SSID WIFI Name
const char* password = "singh8456"; //Enter WIFI Password
#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_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
#else
#error "Camera model not selected"
#endif
```

// GPIO Setting

```
extern int gpLb = 2; // Left 1
extern int gpLf = 14; // Left 2
extern int gpRb = 15; // Right 1
extern int gpRf = 13; // Right 2
extern int gpLed = 4; // Light
extern String WiFiAddr ="";
void startCameraServer();
void setup() {
Serial.begin(115200);
Serial.setDebugOutput(true);
Serial.println();
pinMode(gpLb, OUTPUT); //Left Backward
pinMode(gpLf, OUTPUT); //Left Forward
pinMode(gpRb, OUTPUT); //Right Forward
pinMode(gpRf, OUTPUT); //Right Backward
pinMode(gpLed, OUTPUT); //Light

//initialize
digitalWrite(gpLb, LOW);
digitalWrite(gpLf, LOW);
digitalWrite(gpRb, LOW);
digitalWrite(gpRf, LOW);
digitalWrite(gpLed, LOW);
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;

//init with high specs to pre-allocate larger buffers
if(psramFound()){
config.frame_size = FRAMESIZE_UXGA;
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;
}

//drop down frame size for higher initial frame rate
sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s,  FRAMESIZE_CIF);

WiFi.begin(ssid, password);

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());
WiFiAddr = WiFi.localIP().toString();
Serial.println("' to connect");
}

void loop() {
// put your main code here, to run repeatedly:

}
```

- **App.httpd.cpp**

```
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

extern int gpLb;
extern int gpLf;
extern int gpRb;
extern int gpRf;
extern int gpLed;
extern String WiFiAddr;

void WheelAct(int nLf, int nLb, int nRf, int nRb);

typedef struct {
    size_t size; //number of values used for filtering
    size_t index; //current value index
    size_t count; //value count
    int sum;
    int * values; //array to be filled with values
} ra_filter_t;

typedef struct {
```

```
    httpd_req_t *req;
    size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321"
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";

static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t sample_size){
  memset(filter, 0, sizeof(ra_filter_t));

  filter->values = (int *)malloc(sample_size * sizeof(int));
  if(!filter->values){
    return NULL;
  }
  memset(filter->values, 0, sample_size * sizeof(int));

  filter->size = sample_size;
  return filter;
}
```

```
static int ra_filter_run(ra_filter_t * filter, int value){
    if(!filter->values){
        return value;
    }
    filter->sum -= filter->values[filter->index];
    filter->values[filter->index] = value;
    filter->sum += filter->values[filter->index];
    filter->index++;
    filter->index = filter->index % filter->size;
    if (filter->count < filter->size) {
        filter->count++;
    }
    return filter->sum / filter->count;
}

static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){
    jpg_chunking_t *j = (jpg_chunking_t *)arg;
    if(!index){
        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
        return 0;
    }
    j->len += len;
    return len;
}
```

```
static esp_err_t capture_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    int64_t fr_start = esp_timer_get_time();

    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.printf("Camera capture failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");

    size_t fb_len = 0;
    if(fb->format == PIXFORMAT_JPEG){
        fb_len = fb->len;
        res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
    } else {
        jpg_chunking_t jchunk = {req, 0};
        res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk)?ESP_OK:ESP_FAIL;
        httpd_resp_send_chunk(req, NULL, 0);
        fb_len = jchunk.len;
    }
    esp_camera_fb_return(fb);
```

```
    int64_t fr_end = esp_timer_get_time();
    Serial.printf("JPG: %uB %ums", (uint32_t)(fb_len), (uint32_t)((fr_end - fr_start)/1000));
    return res;
}

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];

    static int64_t last_frame = 0;
    if(!last_frame) {
        last_frame = esp_timer_get_time();
    }

    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.printf("Camera capture failed");
      res = ESP_FAIL;
    } else {
      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.printf("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;
    }
    int64_t fr_end = esp_timer_get_time();

    int64_t frame_time = fr_end - last_frame;
    last_frame = fr_end;
    frame_time /=1000;
    uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
    Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps)"
        ,(uint32_t)(_jpg_buf_len),
        (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
        avg_frame_time, 1000.0 / avg_frame_time
    );
}

last_frame = 0;
```

```
    return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
    char*  buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[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, "var", variable, sizeof(variable)) == ESP_OK
&&
                httpd_query_key_value(buf, "val", value, sizeof(value)) == 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;
    }

    int val = atoi(value);
    sensor_t * s = esp_camera_sensor_get();
    int res = 0;

    if(!strcmp(variable, "framesize")) {
        if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s,
(framesize_t)val);
    }
    else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
    else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
    else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
    else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
    else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s,
(gainceiling_t)val);
    else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
    else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
    else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
```

```
    else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
    else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
    else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
    else if(!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
    else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
    else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
    else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
    else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
    else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
    else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
    else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
    else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
    else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s, val);
    else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
    else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
    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);
}
```

```
static esp_err_t status_handler(httpd_req_t *req){
    static char json_response[1024];

    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';

    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
    p+=sprintf(p, "\"contrast\":%d,", s->status.contrast);
    p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
    p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
    p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
    p+=sprintf(p, "\"awb\":%u,", s->status.awb);
    p+=sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
    p+=sprintf(p, "\"aec\":%u,", s->status.aec);
    p+=sprintf(p, "\"aec2\":%u,", s->status.aec2);
    p+=sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
    p+=sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
    p+=sprintf(p, "\"agc\":%u,", s->status.agc);
    p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
    p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
    p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);
    p+=sprintf(p, "\"wpc\":%u,", s->status.wpc);
    p+=sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
```

```
    p+=sprintf(p, "\"lenc\":%u,", s->status.lenc);
    p+=sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
    p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
    p+=sprintf(p, "\"colorbar\":%u", s->status.colorbar);
    *p++ = '}';
    *p++ = 0;
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, json_response, strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t *req){
    httpd_resp_set_type(req, "text/html");
    String page = "";
     page += "<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0, maximum-scale=1.0, user-scalable=0\">\n";
 page += "<script>var xhttp = new XMLHttpRequest();</script>";
 page += "<script>function getsend(arg) { xhttp.open('GET', arg +'?' + new
Date().getTime(), true); xhttp.send() } </script>";
 //page += "<p align=center><IMG SRC='http://" + WiFiAddr + ":81/stream'
style='width:280px;'></p><br/><br/>";
 page += "<p align=center><IMG SRC='http://" + WiFiAddr + ":81/stream'
style='width:300px; transform:rotate(180deg);'></p><br/><br/>";

 page += "<p align=center> <button style=background-
color:lightgrey;width:90px;height:80px onmousedown=getsend('go')
onmouseup=getsend('stop') ontouchstart=getsend('go') ontouchend=getsend('stop')
><b>Forward</b></button> </p>";
```

```
 page += "<p align=center>";
 page += "<button style=background-color:lightgrey;width:90px;height:80px;
onmousedown=getsend('left') onmouseup=getsend('stop') ontouchstart=getsend('left')
ontouchend=getsend('stop')><b>Left</b></button> ";
 page += "<button style=background-color:indianred;width:90px;height:80px
onmousedown=getsend('stop')
onmouseup=getsend('stop')><b>Stop</b></button> ";
 page += "<button style=background-color:lightgrey;width:90px;height:80px
onmousedown=getsend('right') onmouseup=getsend('stop')
ontouchstart=getsend('right') ontouchend=getsend('stop')><b>Right</b></button>";
 page += "</p>";

 page += "<p align=center><button style=background-
color:lightgrey;width:90px;height:80px onmousedown=getsend('back')
onmouseup=getsend('stop') ontouchstart=getsend('back') ontouchend=getsend('stop')
><b>Backward</b></button></p>";

 page += "<p align=center>";
 page += "<button style=background-color:yellow;width:140px;height:40px
onmousedown=getsend('ledon')><b>Light ON</b></button>";
 page += "<button style=background-color:yellow;width:140px;height:40px
onmousedown=getsend('ledoff')><b>Light OFF</b></button>";
 page += "</p>";

   return httpd_resp_send(req, &page[0], strlen(&page[0]));
}

static esp_err_t go_handler(httpd_req_t *req){
   WheelAct(HIGH, LOW, HIGH, LOW);
```

```
    Serial.println("Go");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}
static esp_err_t back_handler(httpd_req_t *req){
    WheelAct(LOW, HIGH, LOW, HIGH);
    Serial.println("Back");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

static esp_err_t left_handler(httpd_req_t *req){
    WheelAct(HIGH, LOW, LOW, HIGH);
    Serial.println("Left");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}
static esp_err_t right_handler(httpd_req_t *req){
    WheelAct(LOW, HIGH, HIGH, LOW);
    Serial.println("Right");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

static esp_err_t stop_handler(httpd_req_t *req){
    WheelAct(LOW, LOW, LOW, LOW);
```

```
    Serial.println("Stop");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

static esp_err_t ledon_handler(httpd_req_t *req){
    digitalWrite(gpLed, HIGH);
    Serial.println("LED ON");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}
static esp_err_t ledoff_handler(httpd_req_t *req){
    digitalWrite(gpLed, LOW);
    Serial.println("LED OFF");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t go_uri = {
        .uri       = "/go",
        .method    = HTTP_GET,
        .handler = go_handler,
        .user_ctx = NULL
```

```
};

httpd_uri_t back_uri = {
    .uri       = "/back",
    .method    = HTTP_GET,
    .handler = back_handler,
    .user_ctx = NULL
};

httpd_uri_t stop_uri = {
    .uri       = "/stop",
    .method    =HTTP_GET,
    .handler  = stop_handler,
    .user_ctx = NULL
};

httpd_uri_t left_uri = {
    .uri       = "/left",
    .method    = HTTP_GET,
    .handler = left_handler,
    .user_ctx = NULL
};

httpd_uri_t right_uri = {
    .uri       = "/right",
    .method    = HTTP_GET,
```

```
    .handler = right_handler,
    .user_ctx = NULL
};

httpd_uri_t ledon_uri = {
    .uri       = "/ledon",
    .method    = HTTP_GET,
    .handler = ledon_handler,
    .user_ctx = NULL
};

httpd_uri_t ledoff_uri = {
    .uri       = "/ledoff",
    .method    = HTTP_GET,
    .handler = ledoff_handler,
    .user_ctx = NULL
};

httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler = index_handler,
    .user_ctx = NULL
};

httpd_uri_t status_uri = {
```

```
    .uri       = "/status",
    .method    = HTTP_GET,
    .handler = status_handler,
    .user_ctx = NULL
};

httpd_uri_t cmd_uri = {
    .uri       = "/control",
    .method    =HTTP_GET,
    .handler  = cmd_handler,
    .user_ctx = NULL
};

httpd_uri_t capture_uri = {
    .uri       = "/capture",
    .method    = HTTP_GET,
    .handler = capture_handler,
    .user_ctx = NULL
};

httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler = stream_handler,
    .user_ctx = NULL
};
```

```
    ra_filter_init(&ra_filter, 20);
    Serial.printf("Starting web server on port: '%d'", config.server_port);
    if (httpd_start(&camera_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(camera_httpd, &index_uri);
        httpd_register_uri_handler(camera_httpd, &go_uri);
        httpd_register_uri_handler(camera_httpd, &back_uri);
        httpd_register_uri_handler(camera_httpd, &stop_uri);
        httpd_register_uri_handler(camera_httpd, &left_uri);
        httpd_register_uri_handler(camera_httpd, &right_uri);
        httpd_register_uri_handler(camera_httpd, &ledon_uri);
        httpd_register_uri_handler(camera_httpd, &ledoff_uri);
    }
    config.server_port += 1;
    config.ctrl_port += 1;
    Serial.printf("Starting stream server on port: '%d'", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
}
void WheelAct(int nLf, int nLb, int nRf, int nRb)
{
 digitalWrite(gpLf, nLf);
 digitalWrite(gpLb, nLb);
 digitalWrite(gpRf, nRf);
 digitalWrite(gpRb, nRb);
}
```

CHAPTER 5

Conclusion

This project of rugged all terrain robot offers a lot of scope for adding newer features. We can program the robot such that it can detect objects and reach them on its own. Thus, we can make it completely autonomous. Also, with addition of GPS navigation and mapping software, the robot will have the capability of finding the best route possible to reach a certain location. There is also the option of adding sound processing to the remote computer, thus giving it greater surveillance capabilities. The possibilities are endless. This robot in its current state provides a platform for further research into improving its capabilities. No doubt this system has certain limits associated with it such as battery range. Still, we have managed to achieve our goal of development of low cost UGV robot.

By increasing the budget, a metal detecting system can also be added to this robot and it can effectively work as a mining robot. The metal detector sensor will detect the metal under the surface of earth and will provide alert at user end.