Project report on

Surveillance Bot

Submitted by:
LostAndFound io

LOST&
FOUND

Group Members:

1. Jjateen Gundesha BT22ECI002
2. Ayush Ambatkar BT22ECI005
3. Darshan Tate BT22ECI011

A report submitted for the partial fulfilment of the requirements of the course
ECE-206 Sensors and Transducers

Submission Date: 31/10/2023

Mini Project

Under the guidance of:
Dr. Mayank Thacker
Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर
Indian Institute of Information Technology, Nagpur

Table of Contents:

Chapter 1: Introduction

The development of autonomous robotic systems has witnessed significant advancements in recent years, offering innovative solutions across various domains. This report documents the design, development, and implementation of an ESP32-based autonomous bot equipped with a NEO-6M GPS module and an MQ2 gas sensor. This project aims to create a versatile, sensor-equipped bot capable of navigating its surroundings, capturing GPS data for location tracking, and detecting specific gases in the environment.

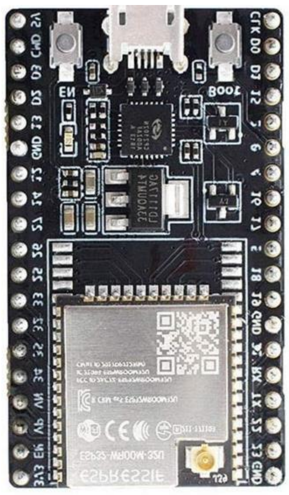
Robotic systems have found applications in diverse fields, from environmental monitoring and agriculture to indoor navigation and industrial automation. The integration of an ESP32 microcontroller, a GPS module, and a gas sensor not only showcases the versatility of the ESP32 platform but also underscores the practicality of such a solution for real-world applications.

In this report, we will delve into the technical aspects of the project, including the hardware components, software development, and practical considerations. Additionally, we will explore the steps involved in building, calibrating, and testing the ESP32-based bot, providing a comprehensive guide for both novice and experienced robotics enthusiasts.

The objectives of this project encompass the following:

1. **Navigation and Positioning:** Implementing accurate location tracking using the NEO-6M GPS module, enabling the bot to determine its position in real-time.
2. **Gas Sensing and Detection:** Integrating the MQ2 gas sensor to identify and monitor specific gases, with a focus on safety and environmental applications.
3. **Motion Control:** Enabling the bot to move efficiently and respond to commands through a motor control system.
4. **Data Logging and Visualization:** Developing a system for capturing and presenting data collected by the sensors for analysis and interpretation.
5. **Safety and Reliability:** Ensuring that the bot operates safely and reliably, especially when dealing with gas sensing, which involves potential hazards.

The combination of these objectives contributes to the creation of a versatile, sensor-equipped bot that can be adapted for a wide range of applications, including environmental monitoring, security, and exploration.



[1] Fig. 1: ESP32 board

[2] Fig. 2: GPS 6M Module

[3] Fig. 3: MQ2 gas sensor

[4] Fig. 4: Hx710B Pressure Sensor

[5] Fig. 5 L298N Motor Driver

[6] Fig. 6 Dual shaft BO Motor

Chapter 2: Code

```
//CamCar.ino

//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 = "Task4";   //Enter SSID Name of your choice

const char* password = "12345678";   //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 ="";

WiFiServer server(81);
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.softAP(ssid, password);
```

```
    IPAddress IP = WiFi.softAPIP();
    Serial.print("AP IP is http://");
    Serial.println(IP);
    server.begin();
//  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();
  WiFiAddr = IP.toString();
//  Serial.println("' to connect");
}

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

}

// Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
//
```

```
// Licensed under the Apache License, Version 2.0 (the "License");

// you may not use this file except in compliance with the License.

// You may obtain a copy of the License at

// app_httpd.cpp

//     http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.

// See the License for the specific language governing permissions and

// limitations under the License.

#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 += "<head>";
page += "    <meta name='viewport' content='width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=0'>";
page += "    <style>";
page += "        body {";
page += "            background-color: #000;";
page += "            color: #fff;";
page += "            text-align: center;";
page += "        }";
page += "";
page += "        .button-container {";
page += "            display: flex;";
page += "            align-items: center;";
page += "            justify-content: center;";
page += "        }";
page += "";
page += "        .arrow-button {";
page += "            width: 100px;";
page += "            height: 100px;";
page += "            background: transparent;";
page += "            border: 2px solid #fff;";
page += "            cursor: pointer;";
page += "            display: inline-block;";
page += "            font-size: 24px;";
page += "            color: #fff;";
page += "            transition: background-color 0.3s, transform 0.3s, box-shadow 0.3s;";
page += "        }";
page += "";
page += "        .arrow-button:hover {";
page += "            background-color: #00FF00;";
page += "            box-shadow: 0 0 10px #00FF00;";
page += "            transform: scale(1.1);";
page += "        }";
page += "";
page += "        .arrow-up {";
page += "            width: 0;";
page += "            height: 0;";
page += "            border-left: 50px solid transparent;";
page += "            border-right: 50px solid transparent;";
page += "            border-bottom: 100px solid white;";
page += "        }";
page += "";
page += "        .arrow-down {";
page += "            width: 0;";
page += "            height: 0;";
page += "            border-left: 50px solid transparent;";
page += "            border-right: 50px solid transparent;";
page += "            border-top: 100px solid white;";
page += "        }";
page += "";
page += "        .arrow-left {";
page += "            width: 0;";
page += "            height: 0;";
page += "            border-top: 50px solid transparent;";
page += "            border-bottom: 50px solid transparent;";
page += "            border-right: 100px solid white;";
page += "        }";
page += "";
page += "        .arrow-right {";
page += "            width: 0;";
page += "            height: 0;";
page += "            border-top: 50px solid transparent;";
page += "            border-bottom: 50px solid transparent;";
page += "            border-left: 100px solid white;";
page += "        }";
page += "";
page += "        .stop-button {";
page += "            background-color: #FF5733;";
page += "            border: 2px solid #fff;";
page += "            text-align: center;";
page += "            display: flex;";
page += "            border-radius: 100px;";
page += "            align-items: center;";
page += "            justify-content: center;";
page += "        }";
page += "";
page += "        .toggle-button {";
page += "            background-color: #FF5733;";
page += "            border: 2px solid #fff;";
page += "            font-size: 48px;";

page += "            cursor: pointer;";

page += "            border-radius: 12px;";

page += "        }";

page += "    </style>";

page += "</head>";

page += "<body>";

page += "<p align=center border-radius: 18px;><IMG SRC='http://" + WiFiAddr +
":81/stream' style='width:300px; transform:rotate(180deg);'></p><br/><br/>";

page += "    <div class='arrow-button arrow-up' onmousedown=\"getsend('back')\"
onmouseup=\"getsend('stop')\" ontouchstart=\"getsend('back')\"
ontouchend=\"getsend('stop')\"></div>";

page += "    <div class='button-container'>";

page += "        <div class='arrow-button arrow-left' onmousedown=\"getsend('left')\"
onmouseup=\"getsend('stop')\" ontouchstart=\"getsend('left')\"
ontouchend=\"getsend('stop')\"></div>";

page += "        <div class='arrow-button stop-button' onmousedown=\"getsend('stop')\"
onmouseup=\"getsend('stop')\">Stop</div>";

page += "        <div class='arrow-button arrow-right' onmousedown=\"getsend('right')\"
onmouseup=\"getsend('stop')\" ontouchstart=\"getsend('right')\"
ontouchend=\"getsend('stop')\"></div>";

page += "    </div>";

page += "    <div class='arrow-button arrow-down' onmousedown=\"getsend('go')\"
onmouseup=\"getsend('stop')\" ontouchstart=\"getsend('go')\"
ontouchend=\"getsend('stop')\"></div>";

page += "    <div class='button-container' style='margin: 8px;'>";

page += "<p align='center'>";

page += "    <button class='toggle-button' onmousedown='getsend(\"ledon\")'><b
background: transparent; border: 2px solid #fff; cursor: pointer; display: inline-block;
font-size: 48px; color: #fff; transition: background-color 0.3s, transform 0.3s, box-
shadow 0.3s;'>&#127774</b></button>";

page += "    <button class='toggle-button' onmousedown='getsend(\"ledoff\")'><b
background: transparent; border: 2px solid #fff; cursor: pointer; display: inline-block;
```

```
font-size: 48px; color: #fff; transition: background-color 0.3s, transform 0.3s, box-
shadow 0.3s;'>&#127769</b></button>";

page += "</p>";

page += "</body>";

page += "<script>";

page += "    function getsend(arg) {";

page += "        var xhttp = new XMLHttpRequest();";

page += "        xhttp.open('GET', arg + '?' + new Date().getTime(), true);";

page += "        xhttp.send();";

page += "    }";

page += "</script>";




    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);

//GPS_Sensors.ino
#include <TinyGPSPlus.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include "HX710B.h"
#include <Servo.h>  // Include the Servo library

// Replace with your network credentials
const char* ssid = "jjj";
const char* password = "12345678";

HX710B pressure_sensor;
long offset = 1855509;

// The TinyGPSPlus object
```

```
TinyGPSPlus gps;

// Define the analog pins for the sensors
const int mq2Pin = 5;  // Change this to the actual pin for the MQ-2 sensor

// Define the pins for the pressure sensor
const byte MPS_OUT_pin = 18;  // OUT data pin
const byte MPS_SCK_pin = 19;  // clock data pin

// Create instances of the WebServer and the HX711 pressure sensor
WebServer server(80);
Servo servoMotor;  // Create a Servo object to control the servo motor

float pressureValue = 0.0; // Declare pressureValue as a global variable
int servoAngle = 90;  // Initial servo angle

void setup() {
  Serial.begin(9600);
  Serial2.begin(9600);
  pressure_sensor.begin(MPS_OUT_pin, MPS_SCK_pin, 128);
  pressure_sensor.set_offset(offset);
  servoMotor.attach(26);  // Attach the servo to pin 26

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println(".");
  Serial.println("Connected to WiFi");
  Serial.print(WiFi.localIP());
  // Define server routes
  server.on("/", HTTP_GET, handleRoot);
  server.on("/servo", HTTP_GET, handleServo);

  // Start the server
  server.begin();
}

void loop() {
  server.handleClient();

  // Read the analog value from the MQ-2 sensor
  int mq2Value = analogRead(mq2Pin);
  Serial.println("MQ-2 Gas Sensor Reading: " + String(mq2Value));

  // Read the pressure sensor
  // Set the scale
  pressureValue = pressure_sensor.is_ready() ? pressure_sensor.pascal() : 0;
  Serial.println("Pressure Sensor Reading: " + String(pressureValue, 2) + " kPa");

  while (Serial2.available() > 0) {
    if (gps.encode(Serial2.read())) {
      displayInfo();
    }
  }

  if (millis() > 5000 && gps.charsProcessed() < 10) {
    Serial.println(F("No GPS detected: check wiring."));
    while (true);
  }
  Serial.println(servoAngle);
}

void displayInfo() {
  Serial.print(F("Location: "));
  if (gps.location.isValid()) {
    Serial.print("Lat: ");
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print("Lng: ");
    Serial.println(gps.location.lng(), 6);
  } else {
    Serial.println(F("INVALID"));
  }
}

void handleRoot() {
  // Create a simple HTML page with an embedded map, sensor readings, and a servo
motor control slider
  String html = "<html><head>";
  html += "<link rel='stylesheet' href='https://unpkg.com/leaflet@1.7.1/dist/leaflet.css'
/>";

  html += "<style>";

  html += "body { background-color: #242b3a; color: white; font-family: Arial, sans-
serif; text-align: center; margin: 0; padding: 0; }";

  html += ".info { display: inline-block; margin: 0 10px; }";

  html += "h1 { text-align: center; }";

  html += "#map-container { width: 600px; height: 600px; margin: 0 auto; border-
radius: 20px; overflow: hidden; }";

  html += "#map { width: 100%; height: 100%; }";

  html += ".slider-container { margin-top: 20px; }"; // Add some margin for the slider

  html += "</style>";

  html += "</head><body>";

  html += "<h1>Surveillance Bot Data</h1>";

  html += "<div class='info'><p>Latitude: " + String(gps.location.lat(), 6) +
"</p></div>";

  html += "<div class='info'><p>Longitude: " + String(gps.location.lng(), 6) +
"</p></div>";

  html += "<div class='info'><p>MQ-2 Gas Sensor Reading: " +
String(analogRead(mq2Pin)) + "</p></div>";

  html += "<div class='info'><p>Pressure Sensor Reading: " + String(pressureValue, 2) +
" kPa</p></div>";

  html += "<div class='slider-container'>";

  html += "<p>Servo Motor Control:</p>";

  html += "<input type='range' id='servoSlider' min='0' max='180' step='1' value='" +
String(servoAngle) + "' oninput='updateServo(this.value)' />";

  html += "<span id='sliderValue'>" + String(servoAngle) + "</span>";

  html += "</div>";

  html += "<div id='map-container'><div id='map'></div></div>";

  html += "<script src='https://unpkg.com/leaflet@1.7.1/dist/leaflet.js'></script>";

  html += "<script>";
  html += "var map = L.map('map').setView([" + String(gps.location.lat(), 6) + ", " +
String(gps.location.lng(), 6) + "], 13);";

  html += "L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', { attribution:
'© <a href=\"https://www.openstreetmap.org/copyright\">OpenStreetMap</a>
contributors' }).addTo(map);";

  html += "L.marker([" + String(gps.location.lat(), 6) + ", " + String(gps.location.lng(), 6)
+ "]).addTo(map);";

  html += "function updateServo(sliderValue) {
document.getElementById('sliderValue').innerText = sliderValue; var xhr = new
XMLHttpRequest(); xhr.open('GET', '/servo?angle=' + sliderValue, true); xhr.send(); }";

  html += "</script>";

  html += "</body></html>";

  server.send(200, "text/html", html);

}


void handleServo() {

  if (server.args() > 0) {

    String angle = server.arg("angle");

    int newAngle = angle.toInt();


    if (newAngle >= 0 && newAngle <= 180) {

      servoAngle = newAngle;

      servoMotor.write(servoAngle);

      server.send(200, "text/plain", "Servo angle set to " + angle);

    } else {

      server.send(400, "text/plain", "Invalid servo angle");

    }

  } else {

    server.send(400, "text/plain", "Missing servo angle parameter");

  }
```

```
}}



#ObjectDetection.py
import numpy as np
import imutils
import cv2
import requests

prototxt = "MobileNetSSD_deploy.prototxt.txt"
model = "MobileNetSSD_deploy.caffemodel"
confThresh = 0.2

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
	"bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
	"dog", "horse", "motorbike", "person", "pottedplant", "sheep",
	"sofa", "train", "tvmonitor"]

COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

print("Loading model...")
net = cv2.dnn.readNetFromCaffe(prototxt, model)
print("Model Loaded")

# Replace the URL with the URL of your video stream
url = "http://192.168.4.1:81/stream"
```

```
while True:
    response = requests.get(url, stream=True)
    if response.status_code == 200:
        bytes = bytes()
        for chunk in response.iter_content(chunk_size=1024):
            bytes += chunk
            a = bytes.find(b'\xff\xd8')
            b = bytes.find(b'\xff\xd9')
            if a != -1 and b != -1:
                jpg = bytes[a:b + 2]
                bytes = bytes[b + 2:]
                if len(jpg) > 0:
                    nparr = np.frombuffer(jpg, dtype=np.uint8)
                    frame = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
                    frame = cv2.resize(frame, (960, 520))
                    # frame = imutils.resize(frame, width=500)
                    (h, w) = frame.shape[:2]
                    imResize = cv2.resize(frame, (300, 300))
                    blob = cv2.dnn.blobFromImage(imResize, 0.007843, (300, 300), 127.5)

                    net.setInput(blob)
                    detections = net.forward()

                    detShape = detections.shape[2]
                    for i in np.arange(0, detShape):
                        confidence = detections[0, 0, i, 2]
                        if confidence > confThresh:
```

```
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")


            label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY), COLORS[idx], 2)
            if startY - 15 > 15:
                y = startY - 15
            else:
                y = startY + 15
            cv2.putText(frame, label, (startX, y), cv2.FONT_HERSHEY_SIMPLEX,
0.5, COLORS[idx], 2)


        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1)
        if key == 27:
            break


cv2.destroyAllWindows()
```

Chapter 3: Working

Working Mechanism

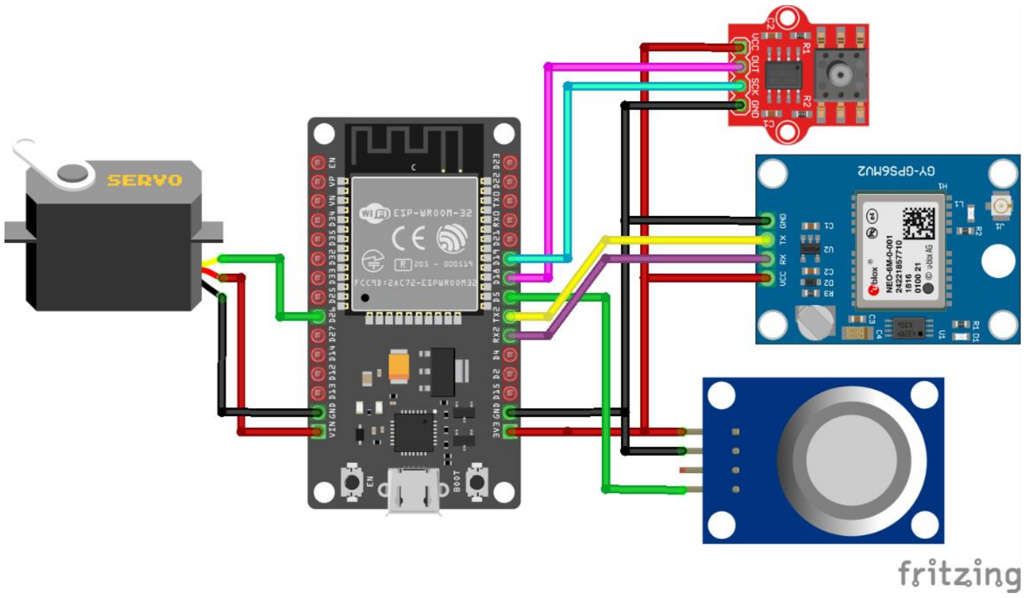
ESP32 CAM:

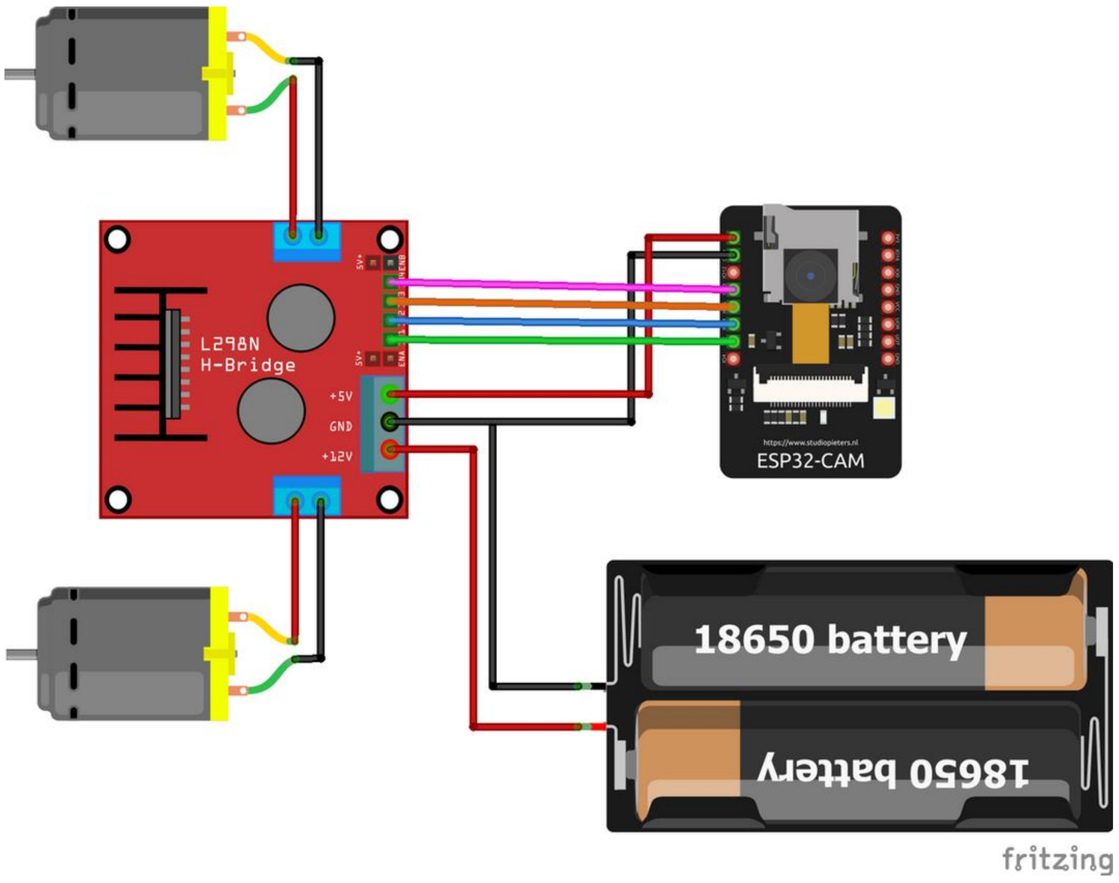
1. **Initialization:**
    - Upon power-up, the ESP32-CAM initializes its components, including the camera module and Wi-Fi connection.
2. **Web Server Setup:**
    - The ESP32-CAM establishes a web server, enabling users to connect remotely through a web browser.
3. **Remote Control:**
    - Users can access a user-friendly web interface hosted on the ESP32-CAM. Through this interface, they can control the robot's movement in real-time using commands such as forward, backward, left, and right.
4. **Object Detection:**
    - Simultaneously, the ESP32-CAM captures live video frames using its camera module.
    - The OpenCV library is employed to analyze these frames, implementing an object detection algorithm.
    - The robot identifies obstacles or objects in its path, enhancing its ability to navigate autonomously.
5. **Integration:**
    - The remote control and object detection functionalities are seamlessly integrated, allowing the robot to respond dynamically to user commands while autonomously avoiding obstacles.
6. **Real-Time Interaction:**
    - The user experiences real-time feedback, observing the robot's movements through the live video stream. Meanwhile, the robot adapts its trajectory based on the detected environment.
7. **Versatility:**
    - This project showcases the versatility of the ESP32-CAM, transforming it into a mobile and intelligent robot capable of remote operation and autonomous decision-making.
8. **User-Friendly Interaction:**
    - The user-friendly web interface simplifies the control process, making it accessible to users with minimal technical expertise.

#ESP32 DEVKIT

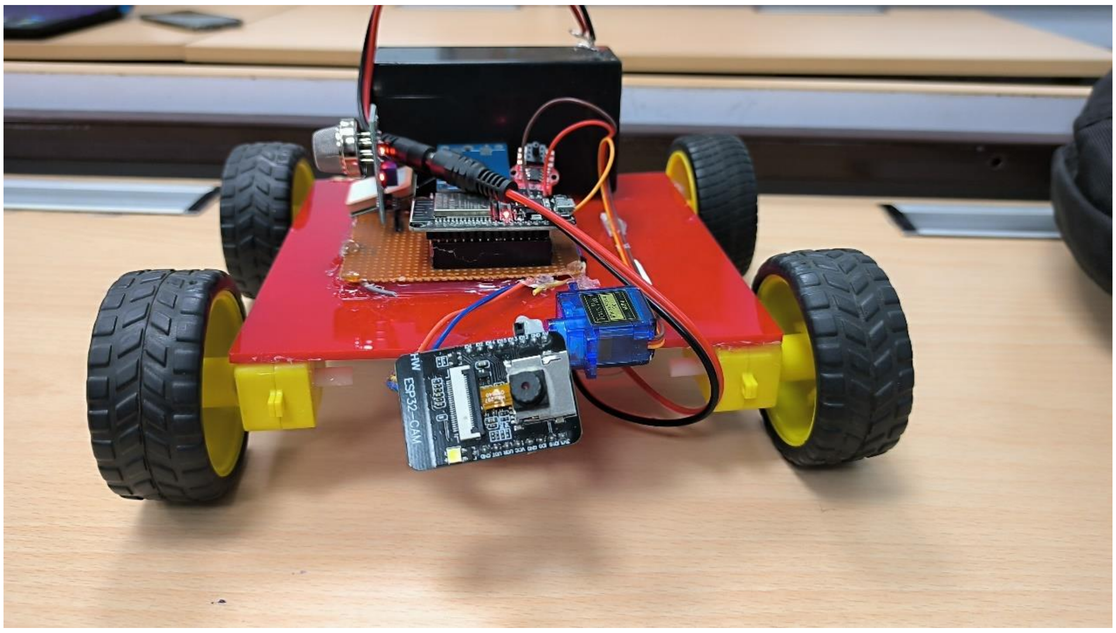
1. **GPS Module:**
   - Reads data from a GPS module using the **TinyGPSPlus** library.
   - Displays latitude and longitude on the HTML page.
2. **MQ-2 Gas Sensor:**
   - Reads analog data from an MQ-2 gas sensor and displays it on the HTML page.
3. **Pressure Sensor (HX710B):**
   - Reads pressure data using the **HX710B** library and displays it on the HTML page.
4. **Servo Motor Control:**
   - Controls a servo motor using the **Servo** library.
   - The servo motor angle can be adjusted using a slider on the HTML page.
5. **Wi-Fi and Web Server:**
   - Connects to a Wi-Fi network using the specified credentials.
   - Creates a web server using the **WebServer** library.
   - Handles requests for the root path ("/") and the "/servo" path.
6. **HTML Page:**
   - Generates an HTML page with the following information:
     - GPS location (latitude and longitude).
     - MQ-2 gas sensor reading.
     - Pressure sensor reading.
     - A slider for controlling the servo motor angle.
     - Displays a map using Leaflet with a marker at the current GPS location.

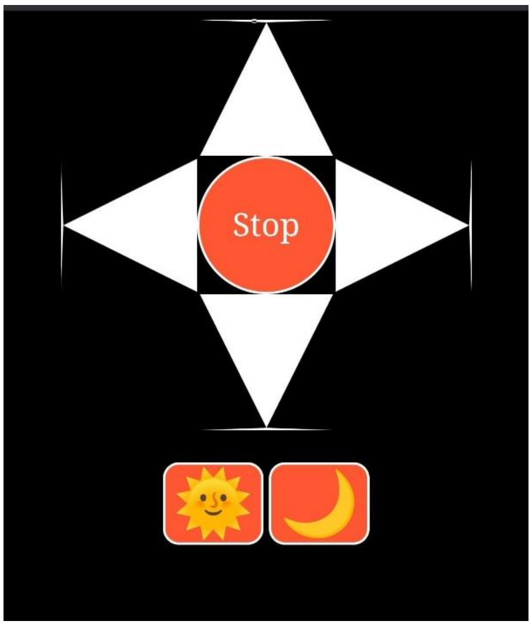
Circuit Diagram:

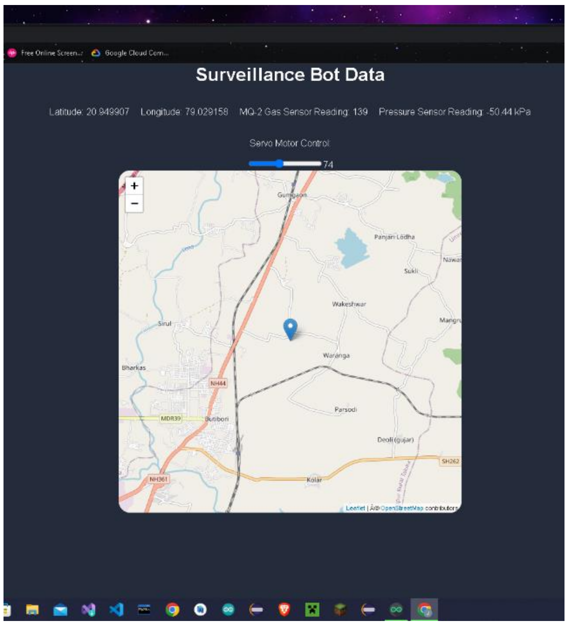




Output Images:





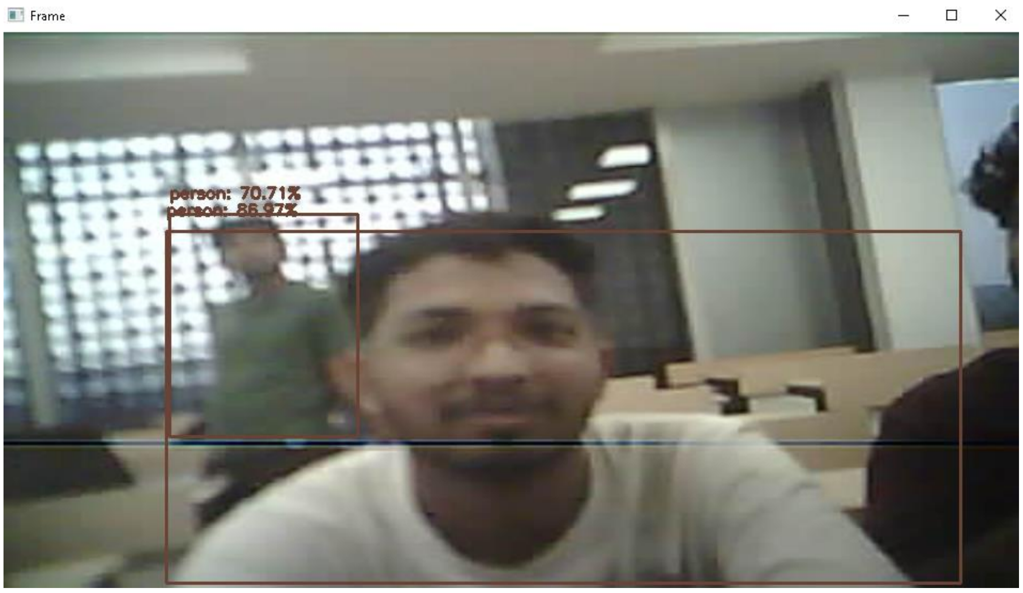

Free Online Screen... Google Cloud Com...

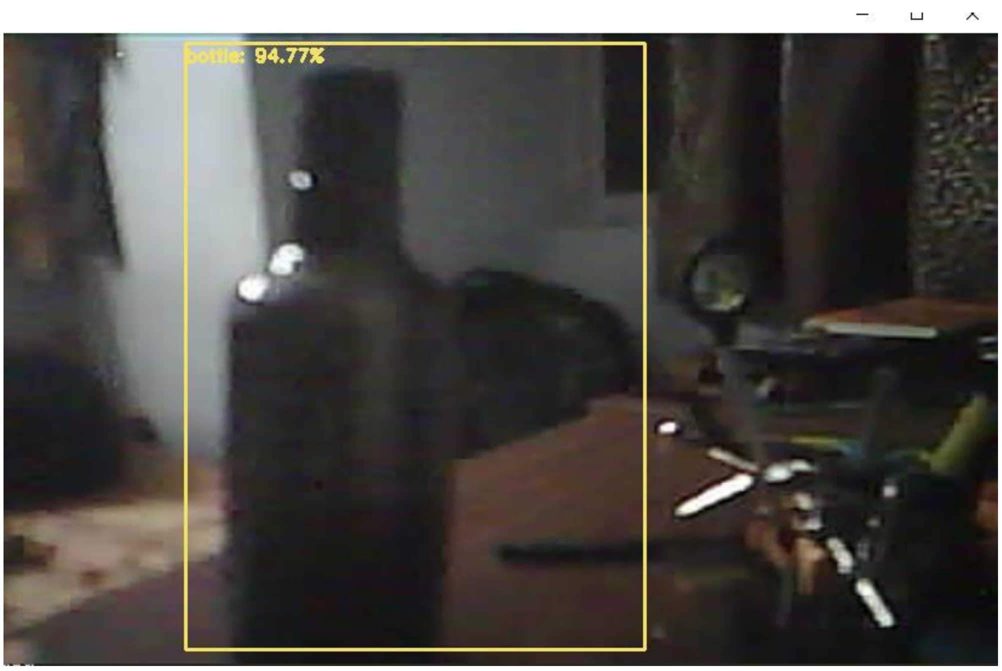
Surveillance Bot Data

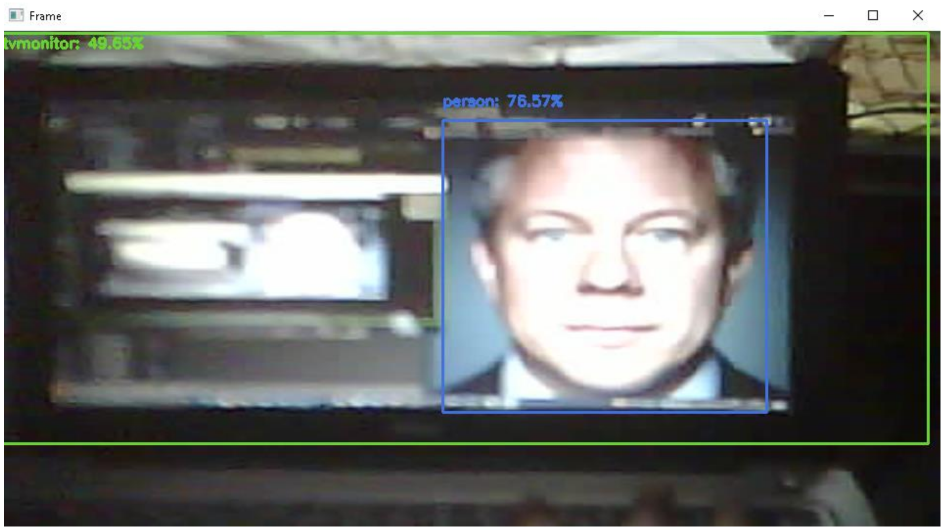
Latitude: 20.949907 Longitude: 79.029158 MQ-2 Gas Sensor Reading: 139 Pressure Sensor Reading: -50.44 kPa

Servo Motor Control

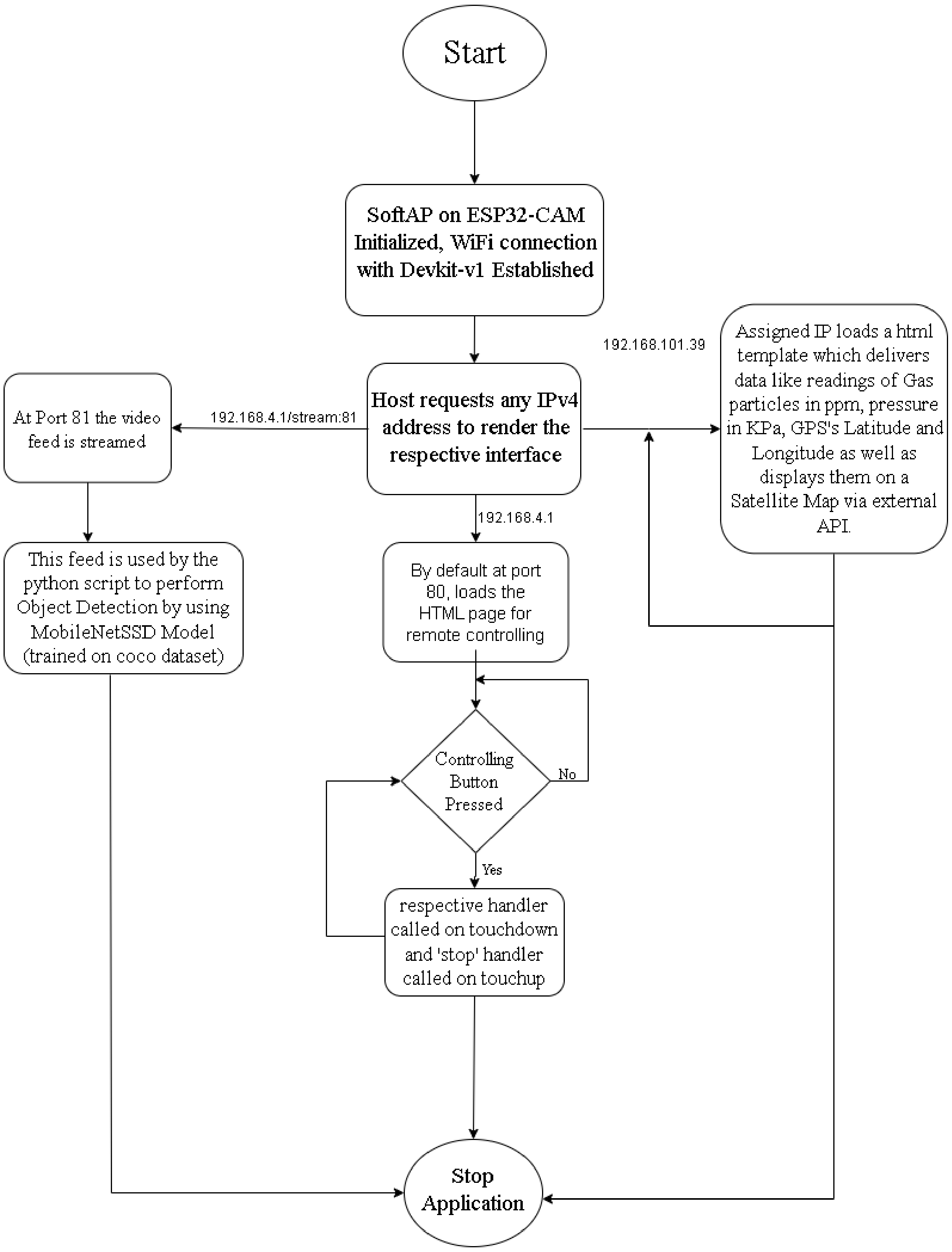
74


Frame
person: 70.71%
person: 86.97%


bottle: 94.77%


Frame
tvmonitor: 49.65%
person: 76.57%

Flowchart:


Start
SoftAP on ESP32-CAM Initialized, WiFi connection with Devkit-v1 Established
Host requests any IPv4 address to render the respective interface
192.168.4.1/stream:81
At Port 81 the video feed is streamed
This feed is used by the python script to perform Object Detection by using MobileNetSSD Model (trained on coco dataset)
192.168.101.39
Assigned IP loads a html template which delivers data like readings of Gas particles in ppm, pressure in KPa, GPS's Latitude and Longitude as well as displays them on a Satellite Map via external API.
192.168.4.1
By default at port 80, loads the HTML page for remote controlling
Controlling Button Pressed
No
Yes
respective handler called on touchdown and 'stop' handler called on touchup
Stop Application

References:

[1] https://tinyurl.com/efnfrwm4

[2] https://tinyurl.com/imagui24

[3] https://robu.in/wp-content/uploads/2020/02/G3-462x462.jpg

[4] https://tinyurl.com/y997b3aw

[5] https://www.youtube.com/watch?v=R-CZLimCcW8

[6] https://esp32.com/viewtopic.php?t=7294

[7] https://tinyurl.com/yfbxxm4w

[8] https://tinyurl.com/y2je9we4

[9] https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf

[10] https://tinyurl.com/Pressure26

[11] https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/

[12] https://esp32io.com/tutorials/esp32-gps