PROJECT REPORT: UNDERWATER DRONE FOR ENHANCED MARITIME SECURITY

TEAMMATES

22BCE9057 - KANNAN KARTHIKA ABIMANYU

22BEC7112 - SNEHIDH SHAJI

22BCE8021 - SHAWN STANLEY

22BEC7303 - KASTURI B V S V SUHASH

22BEC7044 - PURUSHOTHAMAN M

22BCE7417 - THOMAS VARGHESE GEORGE

UNDER THE GUIDANCE OF

Prof. K Srinivas

VIT-AP UNIVERSITY

Index

1INTRODUCTION

Maritime security remains a pressing global concern, with threats ranging from illegal activities to piracy and terrorism posing significant challenges for effective monitoring and surveillance across vast oceanic expanses. In response, this project aims to develop an underwater drone system capable of conducting extended maritime patrols, gathering intelligence, and rapidly responding to potential threats.

Integrating state-of-the-art sensors, communication systems, and autonomous navigation capabilities, this innovative platform promises to revolutionise maritime security operations. The drone's versatility lies in its ability to operate covertly, evading detection while collecting invaluable data on maritime activities through advanced payloads such as high-resolution cameras.

The successful deployment of this underwater maritime security drone has the potential to significantly enhance national and international efforts in combating maritime threats, safeguarding trade routes, and protecting marine ecosystems. By leveraging cutting-edge technologies and fostering collaboration, this project represents a significant step forward in ensuring the safety and security of the world's oceans.

2. BACKGROUND

The maritime domain is vital for global trade and prosperity, yet ensuring security and maintaining situational awareness over vast oceanic expanses poses significant challenges. Traditional naval assets face limitations in operational endurance, coverage area, and covert surveillance capabilities. The proliferation of illegal maritime activities, coupled with potential maritime terrorism threats, underscores the need for enhanced security measures.

Existing surveillance methods, including surface vessels, aircraft, and satellite imagery, each have strengths and weaknesses. Surface vessels offer direct observation but limited range, aircraft provide broader coverage with limited endurance, and satellite imagery lacks real-time data and can be hindered by atmospheric conditions.

The development of an underwater maritime security drone addresses these limitations by offering a stealthy, persistent, and versatile platform. Leveraging sensor technologies, and navigation systems, this innovative solution promises to revolutionise maritime security operations.

Operating covertly beneath the surface, the drone can evade detection while gathering critical intelligence through advanced sensors like high-resolution cameras, and specialised tracking equipment. Its navigation capabilities enable efficient patrol routes, adaptive mission planning, and rapid threat response with reduced human supervision.

Integrated with existing command-and-control infrastructure, the underwater drone system will provide real-time data to security forces, enabling informed decision-making and coordinated response efforts, particularly in remote or contested waters.

3. PROBLEM DEFINITION:

Security and surveillance operations in the maritime domain are difficult because it is very large, deep, and has different environmental conditions. Often, these methods of patrolling and monitoring cost a lot of money, take time or are not effective.These limitations underscore the need for innovative solutions that can augment existing capabilities and improve the overall effectiveness of maritime security efforts.

The project aims to address these challenges by developing and deploying autonomous underwater drones equipped with advanced sensors and communication systems. These drones will be designed to operate efficiently in various underwater conditions, including shallow and deep waters, and adverse weather conditions. They will be capable of conducting long-duration missions, collecting real-time data, and transmitting information.

4. OBJECTIVES

The primary objective of this engineering project is to design, develop, and document an underwater drone capable of performing various tasks in aquatic environments. This underwater drone aims to serve multiple purposes including scientific research, environmental monitoring, inspection, exploration, and educational outreach. The documentation of this project will comprehensively cover the design process, technical specifications, operational capabilities, and potential applications of the underwater drone. In response to the growing need for effective maritime security measures, this project endeavours to create an innovative solution in the form of an Underwater ROV (Remotely Operated Vehicle). Leveraging the capabilities of the ESP32 CAM module, waterproof motors, and L293D H-bridge motor drivers, we aim to develop a versatile platform capable of conducting underwater surveillance and inspection tasks with precision and reliability.

5. Procedure

5.1 Components Used

a. Arduino Uno:

Arduino, a versatile microcontroller platform, empowers innovators to bring their electronic creations to life with ease and creativity. Its open-source nature fosters a vibrant community dedicated to exploration and invention. In this project, Arduino is used to control the movement of the ROV.

Fig1: Arduino Uno

b. Airtight case:

We used an airtight container for our underwater drone project, ensuring all electronic components were securely housed and protected from water ingress, thus enabling reliable operation in submerged environments.

Fig2: Air tight case

c. Joystick controllers:

Joystick controllers are used to change the direction of the ROV. In this project, two joysticks are used for the left and right movement of the ROV.

Fig3: Joystick Controller

d. ESP32 S3 cam module:

The ESP32-S3 CAM module integrates advanced imaging capabilities with the power of the ESP32-S3 microcontroller, making it a crucial component in underwater ROV projects, enabling real-time video streaming and high-resolution image capture for remote exploration and inspection tasks.

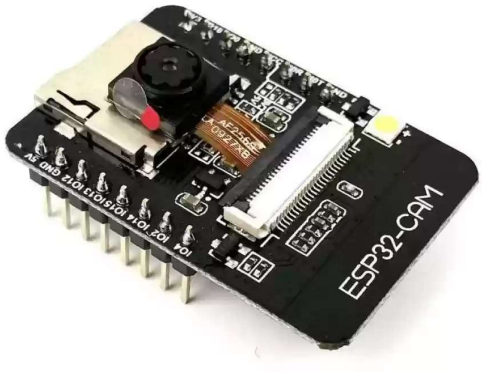


Fig4: ESP32 cam module

e. L293D H-bridge IC:

The L-293D motor IC acts as the bridge between the joysticks and the motors.

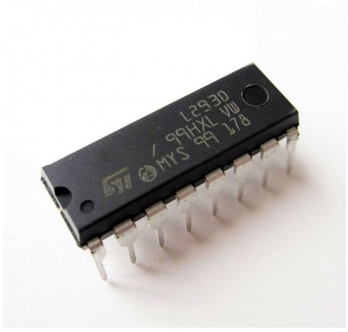


Fig5: L293D Motor driver IC

f. Breadboards:

A breadboard is a fundamental tool for prototyping electronic circuits, offering a convenient platform to experiment with component connections without soldering. Its grid layout and spring-loaded connectors enable quick and flexible circuit assembly, making it essential for both beginners and experienced electronics enthusiasts.

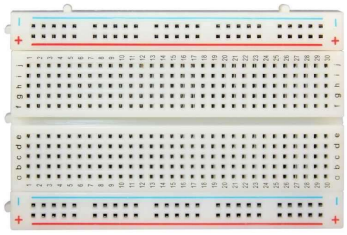


Fig6: Breadboard

g. Waterproof DC motors:

Waterproof DC motors are vital components of underwater ROVs, providing reliable propulsion in aquatic environments while resisting water ingress and corrosion. Their robust design enables precise control of movement, essential for tasks such as underwater surveillance and inspection.

Fig7: Motors

h. 1000 mah battery:

A 1000mAh battery powers underwater ROVs, enabling extended operational durations for missions like exploration and surveillance. Its compact size ensures efficient integration within the limited space of the ROV while delivering sufficient energy for propulsion and electronics.

Fig8: Battery

i. Wire connectors:

Wire connectors are the essential links that ensure seamless electrical conductivity and secure connections in various electronic devices and systems.

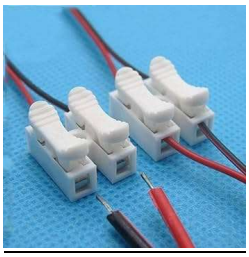


Fig9: Wire connectors

j. TFT Display:

A TFT display integrated into an underwater ROV offers real-time visual feedback of the surrounding underwater environment, enhancing operator control and situational awareness. When paired with an ESP32 CAM module, live video transmission from the underwater ROV becomes feasible, enabling remote monitoring and navigation, crucial for tasks like inspection and surveillance in aquatic environments.

Fig10: TFT Display

k. Jumper wires:

Jumper wires are essential connectors that bridge components on a breadboard or within other electronic circuits, enabling flexibility and experimentation without soldering.

Fig11: Jumper wires

6. CONNECTIONS:

a. H-bridge (pins numbered counter-clockwise from the top left)
   i. Pin 1 to Arduino pin 11
   ii. Pin 2 to Arduino pin 12
   iii. Pin 3 to the right motor negative wire
   iv. Pin 4 to ground
   v. Pin 5 to ground
   vi. Pin 6 to the right motor positive wire
   vii. Pin 7 to Arduino pin 9
   viii. Pin 8 to 7.4 V from the battery
   ix. Pin 9 to Arduino pin 10
   x. Pin 10 to Arduino pin 8
   xi. Pin 11 to left motor positive wire
   xii. Pin 12 to ground
   xiii. Pin 13 to ground
   xiv. Pin 14 to left motor negative wire
   xv. Pin 15 to Arduino pin 7
   xvi. Pin 16 to 5 V from Arduino

b. Left joystick
    i. L/R+ to 5 V from Arduino
    ii. L/R to Arduino analog pin A0
    iii. GND to ground
c. Right joystick
    i. L/R+ to 5 V from Arduino
    ii. L/R to Arduino analog pin A1
    iii. GND to ground

d. Camera
    i. 5v to +ve of the battery.
    ii. GND to -ve of the battery.

7. Circuit Diagram

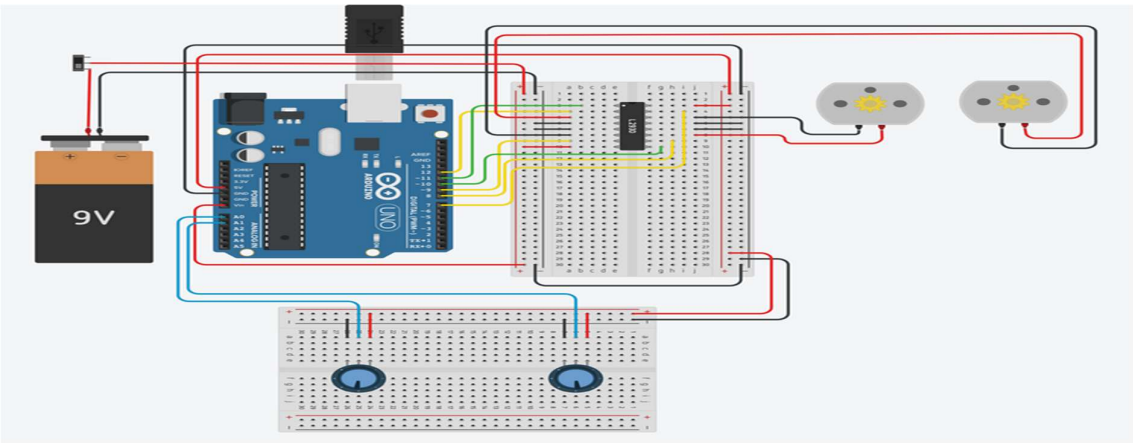
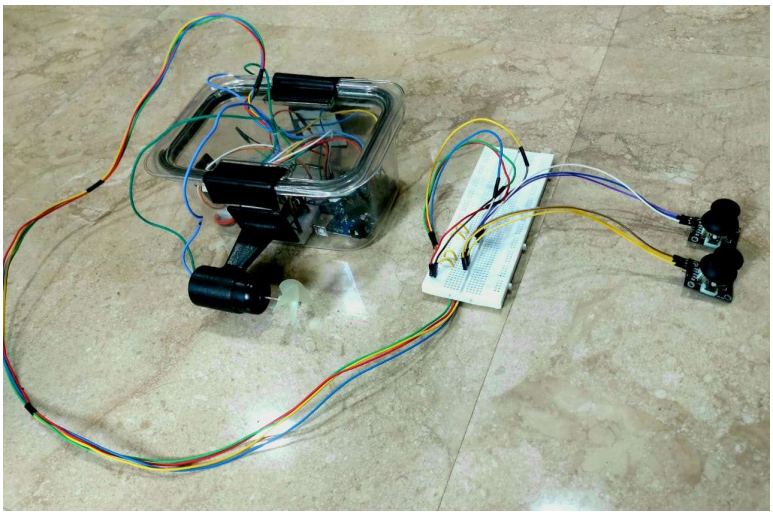
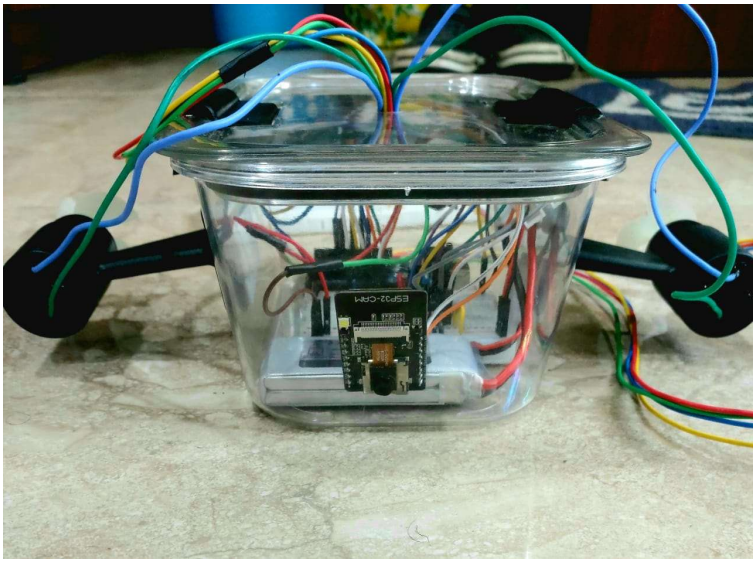


Fig12: Circuit explaining the connections between Arduino, motors and joystick.

8 **Results:**





9. Codes:

9.1 For joystick and motor control:

```
/*
Code to control an ROV with left and right motors
using "tank" steering with two joysticks
*/

// define pins
const int joyLpin = A0;     // left joystick
const int joyRpin = A1;     // right joystick
const int motorLfwd = 7;   // left motor forward pin
const int motorLbck = 8;    // left motor backward pin
```

```
const int motorLen = 10;   // left motor enable pin
const int motorRfwd = 9;    // right motor fwd pin
const int motorRbck = 12;    // right motor backward pin
const int motorRen = 11;   // right motor enable pin

// variables
int joyL;                  // left joystick reading (0-1023 from ADC)
int joyR;                  // right joystick reading (0-1023 from ADC)
int joyLneutral;           // left joystick neutral position
int joyRneutral;           // right joystick neutral
const int deadzone = 20;    // joystick "dead zone" to prevent drift
int motorLspeed;           // left motor speed (0-255 for PWM)
int motorRspeed;           // right motor speed (0-255 for PWM)

void setup() { // code that only runs once
  // set motor control pins as outputs
  pinMode(motorLfwd,OUTPUT);
  pinMode(motorRfwd,OUTPUT);
  pinMode(motorLbck,OUTPUT);
  pinMode(motorRbck,OUTPUT);
  pinMode(motorLen,OUTPUT);
  pinMode(motorRen,OUTPUT);
  // calibrate joysticks
  joyLneutral = analogRead(joyLpin);
  joyRneutral = analogRead(joyRpin);

  Serial.begin(9600);

}

void loop() {  // code that loops forever
  // read joysticks
  joyL = analogRead(joyLpin);
  joyR = analogRead(joyRpin);

  Serial.print("Left: ");
  Serial.print(joyL);
  // set left motor direction and speed
  if((joyL-joyLneutral) < -deadzone){ // joystick pushed forward
    digitalWrite(motorLfwd,HIGH);
    digitalWrite(motorLbck,LOW);
    motorLspeed = map(joyL,joyLneutral,0,0,255);
    Serial.print(" fwd ");
  }
  else if((joyL-joyLneutral) > deadzone){
    digitalWrite(motorLfwd,LOW);
    digitalWrite(motorLbck,HIGH);
    motorLspeed = map(joyL,joyLneutral,1023,0,255);
    Serial.print(" back ");
  }
  else{
    digitalWrite(motorLfwd,LOW);
```

```
  digitalWrite(motorLbck,LOW);
  Serial.print(" stop ");
  motorLspeed = 0;
 }
 Serial.print(motorLspeed);

 Serial.print(" Right: ");
 Serial.print(joyR);

 // set right motor direction and speed
 if((joyR-joyRneutral) < -deadzone){ // joystick pushed forward
  digitalWrite(motorRfwd,HIGH);
  digitalWrite(motorRbck,LOW);
  motorRspeed = map(joyR,joyRneutral,0,0,255);
  Serial.print(" fwd ");
 }
 else if((joyR-joyRneutral) > deadzone){
  digitalWrite(motorRfwd,LOW);
  digitalWrite(motorRbck,HIGH);
  motorRspeed = map(joyR,joyRneutral,1023,0,255);
  Serial.print(" back ");
 }
 else{
  digitalWrite(motorRfwd,LOW);
  digitalWrite(motorRbck,LOW);
  Serial.print(" stop ");
  motorRspeed = 0;
 }
 Serial.println(motorRspeed);
 analogWrite(motorLen,motorLspeed);
 analogWrite(motorRen,motorRspeed);

}
```

9.2 For camera:

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "esp_http_server.h"
//Replace with your network credentials
const char* ssid = "project1";
const char* password = "12345678";

#define PART_BOUNDARY "123456789000000000000987654321"
```

```
// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and M5STACK
WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM

// Not tested with this model
//#define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27

  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

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

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

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

```
  #define SIOC_GPIO_NUM     23

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

#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27

  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
#else
  #error "Camera model not selected"
#endif

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

httpd_handle_t stream_httpd = NULL;

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

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

```
    return res;
  }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

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

```
  config.server_port = 80;

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

  //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

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

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  if(psramFound()){

    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
```

```
  }

  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

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

  // Start streaming web server
  startCameraServer();
}

void loop() {
  delay(1);
```

10. References

10.1 Research papers:

1. SLAM for Underwater Vehicles: A Review of the State of the Art: https://www.mdpi.com/2072-4292/11/23/2827
2. Underwater Robots for Mine Countermeasures: A Review of Recent Advancements: https://link.springer.com/10.1007%2F978-3-540-30301-5_44
3. Underwater Drones for Harbor Security and Inspection: https://www.mdpi.com/journal/drones/special_issues/2T203BI6PJ
4. Development of a Low-Cost Underwater Drone for Coral Reef Monitoring: https://www.researchgate.net/profile/Pawel-Terefenko-2/publication/357929172_Application_of_Unmanned_Aerial_Vehicles_and_Image_P

rocessing_Techniques_in_Monitoring_Underwater_Coastal_Protection_Measures/links/61e7cbb29a753545e2df373c/Application-of-Unmanned-Aerial-Vehicles-and-Image-Processing-Techniques-in-Monitoring-Underwater-Coastal-Protection-Measures.pdf

5. Problem Identification for Underwater Remotely Operated Vehicle (ROV): A Case Study: https://www.sciencedirect.com/science/article/pii/S1877705812026112
6. Structural Design of Underwater Drone using Brushless DC Motor: https://www.ijraset.com/research-paper/underwater-drone-using-brushless-dc-motorn

10.2 Books:

- Autonomous underwater vehicles: modelling, control design, and simulation
- Autonomous vehicles in support of naval operations
- Autonomous underwater vehicles: technology and applications

11. CONCLUSION

In summary, the development of our underwater drone marks a significant stride in engineering prowess, offering a multifaceted solution for underwater exploration and data acquisition. By seamlessly integrating state-of-the-art technology, we have created a sophisticated tool capable of delivering high-resolution imagery and critical data insights from beneath the water's surface.

This project underscores the convergence of cutting-edge engineering with real-world applications, addressing a spectrum of needs across industries such as marine research, environmental monitoring, and infrastructure assessment. The real-time capabilities of our drone elevate efficiency, precision, and safety in underwater operations.

Looking forward, our drone stands poised for further optimization and customization to cater to diverse operational demands. As we remain steadfast in our pursuit of engineering excellence, this endeavour exemplifies our unwavering commitment to technological advancement and its positive impact on society and the environment.

12. FUTURE MODIFICATIONS:

1 Ultrasonic sensors can be added for some important tasks like obstacle avoidance,distance measurement etc.
2. IMU(Inertial Measurement Unit) can also be added for accurate moments. It combines data from accelerometers, gyroscopes, and sometimes magnetometers to provide information about the drone's orientation, acceleration, and rotation. It helps in stabilizing the drone, maintaining its heading, and controlling its movements.
3.Sensors like pressure sensors for measuring the depth and altitude changes. Temperature sensors can also be added to calculate the change in water temperature which is very crucial for marine life.

4. Upgrading propulsion systems, such as adding more powerful thrusters or incorporating advanced propulsion technologies like water jets or propeller pods, can enhance maneuverability, speed, and endurance.

ACKNOWLEDGMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is the result of a studied blend of both research and knowledge.

We express our earnest gratitude to our internal guide, Dr.K Srinivas, our project guide, for his constant support, encouragement, and guidance. We are grateful for his cooperation and his valuable suggestions.

Finally, we express our gratitude to all other members who are involved either directly or indirectly in the completion of this project.