Mechatronics Project

(Autonomous Rescue Boat)

Ibrahim Makkawy
Ahmed Yosri
Hanya Abdelkareem
Samira Ahmed
Beshoy Nasem
Hatem Safwat
Omar Khaled
Mostafa Naser

Under the Supervision of:
Dr. Gamal Abdel Nasser

Faculty of Engineering, Assiut University

May 23, 2024

ABSTRACT

The challenge in maritime emergency response involves effectively addressing emergencies and conducting rescue operations at sea. Traditional methods rely heavily on human intervention, leading to issues such as delayed response times, risks to rescuers, limited coverage, resource constraints, energy inefficiency, and situational awareness difficulties.

Our proposed solution involves the development of an autonomous rescue boat equipped with advanced sensing and navigation systems to swiftly detect and respond to heat sources or fires in oceans. By eliminating the need for human intervention in hazardous rescue scenarios, the autonomous vessel enhances overall safety during emergency operations. The autonomous boat will navigate autonomously to specified emergency locations, assess the situation, and execute precise rescue maneuvers. This approach significantly reduces response time, thus increasing the likelihood of survival for individuals in distress while ensuring comprehensive coverage of maritime regions through the deployment of multiple autonomous rescue boats. Furthermore, efficient energy management systems will be integrated to ensure extended operational endurance for prolonged rescue missions, contributing to the overall effectiveness and efficiency of rescue operations at sea.

Table of Contents

Table of Contents

1 INTRODUCTION

Water-based rescue operations pose significant challenges that necessitate advanced technological solutions for improved efficiency, safety, and reliability. Manual intervention in rescue missions can be hindered by environmental factors such as rough waters or limited visibility, leading to delays and increased risks for responders. Moreover, the ability to promptly detect and reach persons or objects in distress is paramount for successful rescue outcomes. In light of these challenges, this project introduces an innovative autonomous rescue system designed to address the complexities inherent in water-based rescue scenarios.

1.1 Problem

The project addresses a multifaceted problem prevalent in water-based rescue operations. Traditional rescue methods often face significant challenges in terms of efficiency, safety, and reliability. Manual intervention in rescue missions can be time-consuming, especially in environments with adverse conditions such as rough waters or limited visibility. Moreover, human involvement introduces inherent risks, particularly in hazardous situations where immediate response is crucial. Existing rescue systems may also lack the capability to accurately detect and reach persons or objects in distress, leading to delays or ineffective rescue attempts. These limitations underscore the urgent need for an advanced autonomous rescue solution capable of overcoming navigational obstacles, detecting distress signals promptly, avoiding collisions, and executing rescue maneuvers with precision and speed.

1.2 Solution

To address the complexities of water-based rescue operations, the project proposes an innovative autonomous rescue boat controlled by an Arduino Mega microcontroller. The system integrates cutting-edge technologies such as GPS navigation, thermal signal detection, obstacle avoidance, and real-time communication capabilities to create a comprehensive autonomous rescue platform. By harnessing the power of dual motors with an L298N motor driver, an ESP32CAM for thermal signal detection, an ESP32 for communication management, an MPU 6050 IMU for stability control, a GPS NEO6M module for precise navigation, a high-capacity LiPo battery for sustained power, and HC-SR04 ultrasonic sensors for obstacle avoidance, the solution offers a robust and efficient means of executing rescue missions in water environments. The custom PCB design further enhances the system's reliability, durability, and maintenance efficiency, making it a viable and effective solution for addressing the challenges faced in water-based rescue operations.

2 MECHANICAL DESIGN AND MOTORS SELECTION

We choose our own design as a pontoon hulls (Pontoon hulls float on top of the water using pontoons or floaters that create lift. It's a type of planning multihull that doesn't lie in the water, so it doesn't displace a lot of water. They don't really handle well. As with any multihull, they aren't agile - they're not great at maneuvering. They also have a very large turning radius. But they are extremely stable: there's no chance you'll capsize this)

Different Hull Types Explained

Hull Shape	Hull Design & Type	Pros & Cons
	**Round-Bottomed Hulls** Displacement hull Sailboats, canoes	Handles well in rough water Tends to roll, can capsize Has maximum hull speed
	**Multihulls** Displacement hull Sailboats, catamarans	Extremely stable & faster Handles well in rough water Large turning radius
	**Flat-Bottomed Hulls** Planing hull Rowboats, skiffs, tug boats	Extremely stable Extremely choppy & wet No good for bluewater
	**V-Shaped Hulls** Planing hull Powerboats	Faster Handles less well in waves Requires more power
	**Pontoon Hulls** Planing hull Pontoon boats	Stable Not agile No good for bluewater

Dimensions of a boat:

Length (L) = 20.57 in (522.478 mm)
Width (W) = 10.94 in (277.876 mm)
Height (H) = 2.38 in (60.452 mm)

The process of selecting motor depends on torque and speed based on:

1. The weight and dimensions of a boat:
   The weight of the boat based on factors such as intended use, design requirements, and material selection. This is by using our own design, which it is based on hull pontoon type and is made of PVC.

2. Establish a power-to-weight ratio:
   Determine a suitable power-to-weight ratio based on the type of boat and its intended use. Typical ratios for small boats range from 1:100 to 1:200.
3. Calculate the power:
   Use the power-to-weight ratio to calculate the required power for the motor. Multiply the weight of the boat by the power-to-weight ratio to obtain the power in watts.
4. Select torque and speed:
   Once the required power is determined, select a motor with torque and speed specifications that can deliver the required power output. Consult motor specifications provided by manufacturers to ensure compatibility with the desired power output.

So, if we are developing a small boat prototype based on the given boat's weight and all dimensions, we can make some calculations to estimate the torque and speed. Keep in mind that these calculations are still approximate and should be verified and refined during the design and testing process.

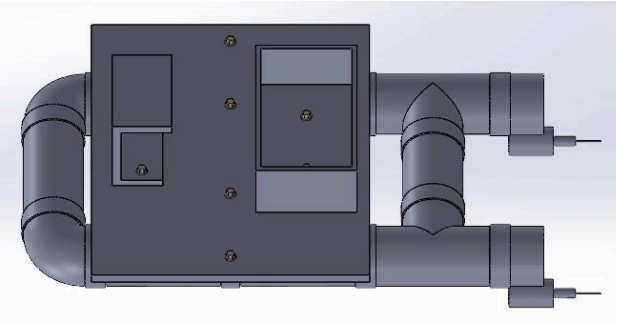
The weight and dimensions of a boat:

- From our own design we can get that all boat's dimensions is:

  Length (L) = 20.57 in (522.478 mm)
  Width (W) = 10.94 in (277.876 mm)
  Height (H) = 2.38 in (60.452 mm)

- And can get the boat's weight and volume as:

  Weight = 7.42 pounds (3.3656 kg)
  Volume = weight / density = 146.67 cubic inches (0.0024 cubic meters)
  (*The density of PVC is around 1400 kg/m³)*



Establish a power-to-weight ratio:

Let's assume a power-to-weight ratio of 1:150. We can calculate the estimated power (P) of the boat as follows:

Power (P) = Weight (W) * Power-to-weight ratio

P = 3.3656 * 150 = 504.84 Watts (Nm/s).

Select torque and speed:

Now we can calculate estimated torque by using the following equation and assume the angular speed equal 2500 rpm:

Torque (T) = Power (P) / Angular Speed (ω)

T = 504.84 / (2500 * 2π/60) = 1.929 Nm (19.29 g-cm)

As we calculated the minimum torque, we select the suitable motor <u>APISQUEEN brush DC motor</u>.

<u>APISQUEEN brush DC motor data sheet:</u>

voltage	7.4V	load：50g-cm
NO	Item	parameters
1	No-load current	⩽1A
2	No-load speed	10000±10%rpm
3	Load current	3.5A
4	Load speed	7500±10%rpm
5	Starting voltage	⩽1V
6	Starting torque	⩾200g-cm
7	Blocking current	⩽5A
8	Terminal resistance	1.5±10%Ω

3 CENTER OF MASS AND CENTER OF GRAVITY

3.1 Center of mass

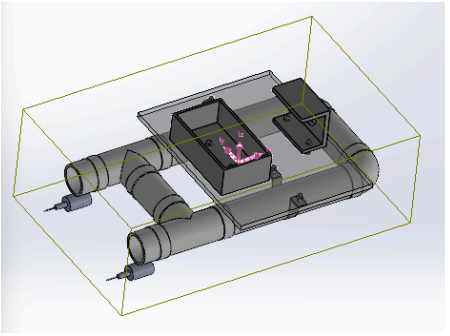
Centre of mass is the point at which the distribution of mass is equal in all directions and does not depend on gravitational field

Center of mass: (inches)

X = 9.14

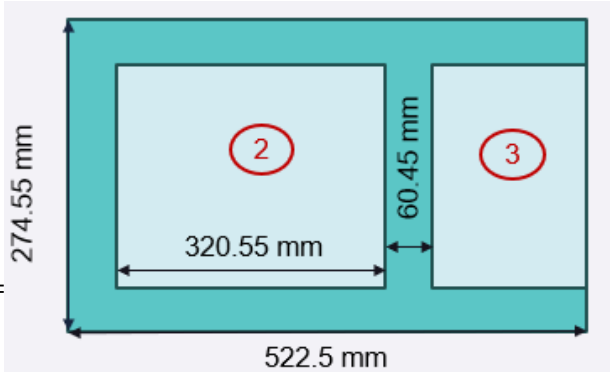
Y = 4.24

Z = 0.88



3.2 Center of gravity

The Centre of gravity is a theoretical point in the body where the body's total weight is thought to be concentrated. It is important to know the center of gravity because it predicts the behavior of a moving body when acted on by gravity. It is also useful in designing static structures such as buildings and bridges.



$$\bar{x} =$$

$$\bar{y} = \frac{\Sigma y_i A_i}{\Sigma A_i}$$

So,

Over all shape 1:

$$A_1 = 522.5 * 274.55 = 143452.375\ mm^2\ x_1 = 261.25\ mm\ y_1 = 137.275\ mm$$

Shape 2:

$$A_2 = 320.55 * 153.65 = 49252.5075\ mm^2\ x_2 = 220.725\ mm\ y_2 = 137.275\ mm$$

Shape 3:

$$A_3 = 81.05 * 153.65 = 12453.3325\ mm^2\ x_3 = 481.975\ mm\ y_3 = 137.275\ mm$$

	A_i	x_i	y_i	$x_i A_i$	$y_i A_i$

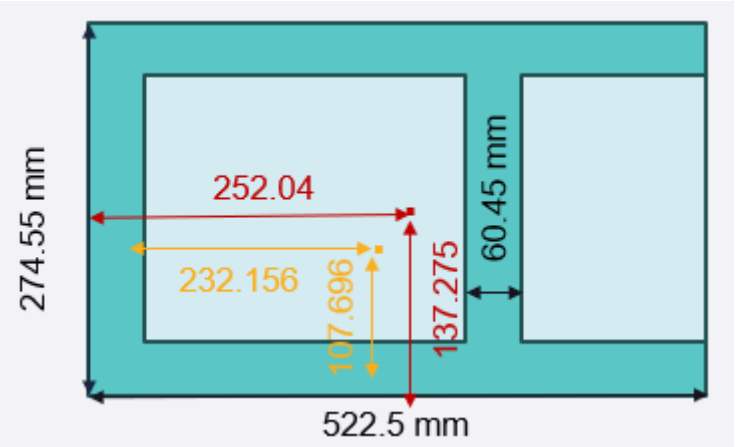
1	14345	261. 2	137. 2	37476932.97	19692424.78
2	− 492	220. 7	137. 2	− 10871259.72	− 6761137.967
3	− 124	481. 9	137. 2	− 6002194.932	− 1709531.219
Σ	81746.535			20603478.32	11221755.59

$$\bar{x} = \frac{\Sigma x_i A_i}{\Sigma A_i} = \frac{20603478.32}{81746.535} = 252.04\ mm$$

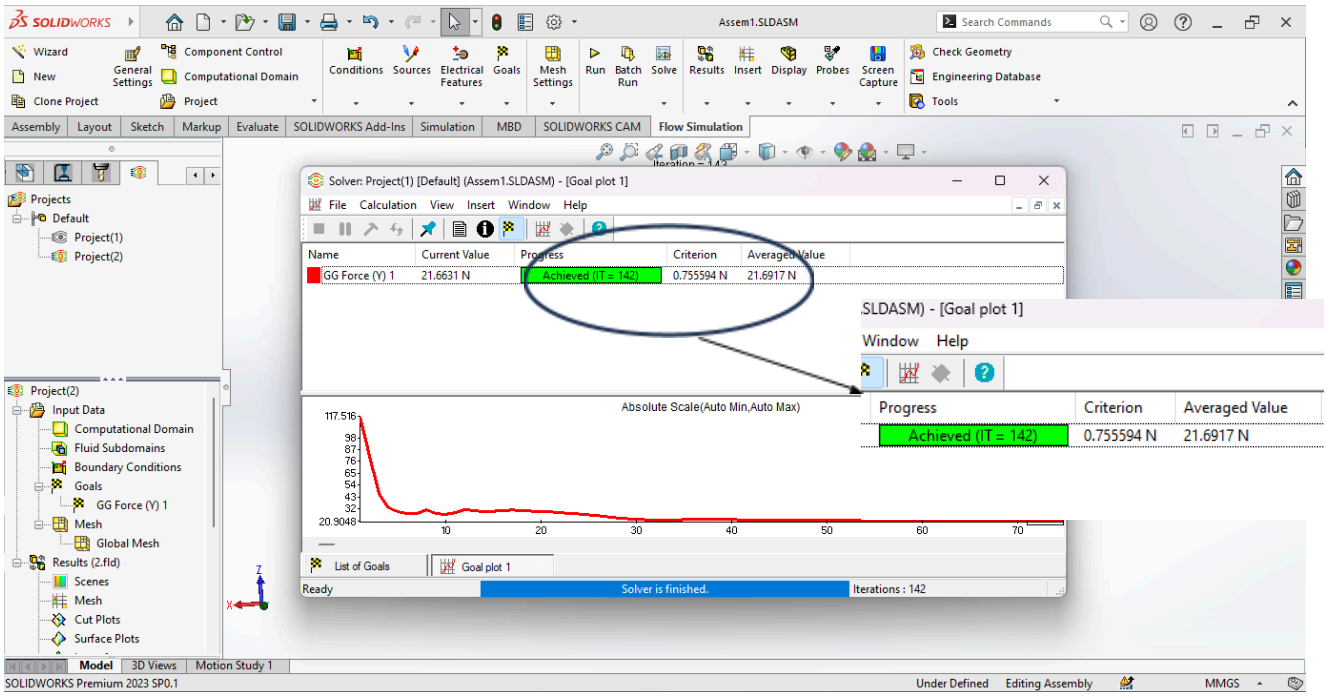
$$\bar{y} = \frac{\Sigma y_i A_i}{\Sigma A_i} = \frac{11221755.59}{81746.535} = 137.275\ mm$$

Center of mass : $x = 252.04\ mm\ y = 137.275\ mm$

$x = 232.156\ mm\ y = 107.696\ mm$ **Center of gravity :**



4 FLOW SIMULATION (WATER AND AIR)



$$CD = \frac{2*F_d}{A*\rho*v^2} = \frac{2*21.69}{0.01*1000*2^2} = 1.0845$$

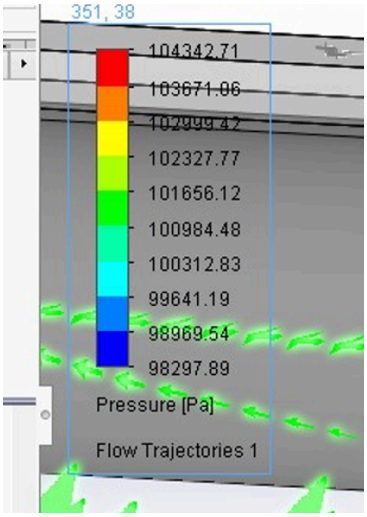
CD = Drag coefficient

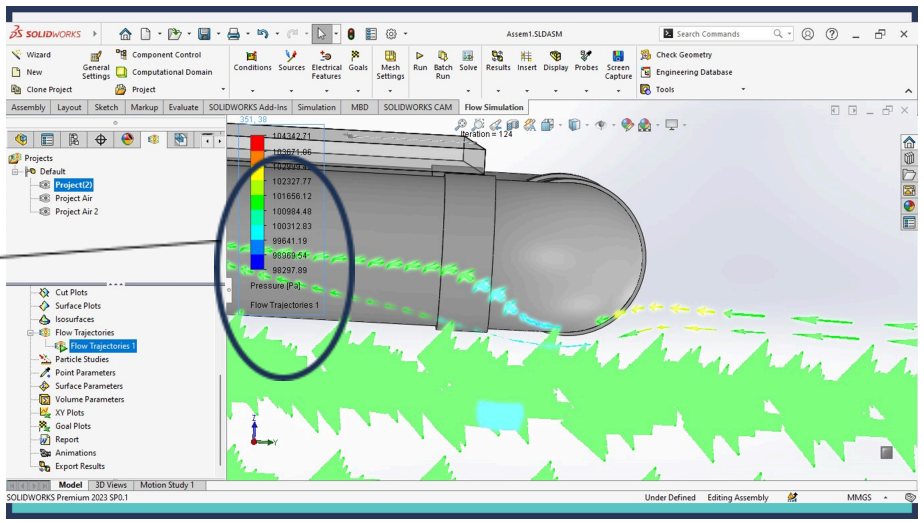
F_d *= Drag force*

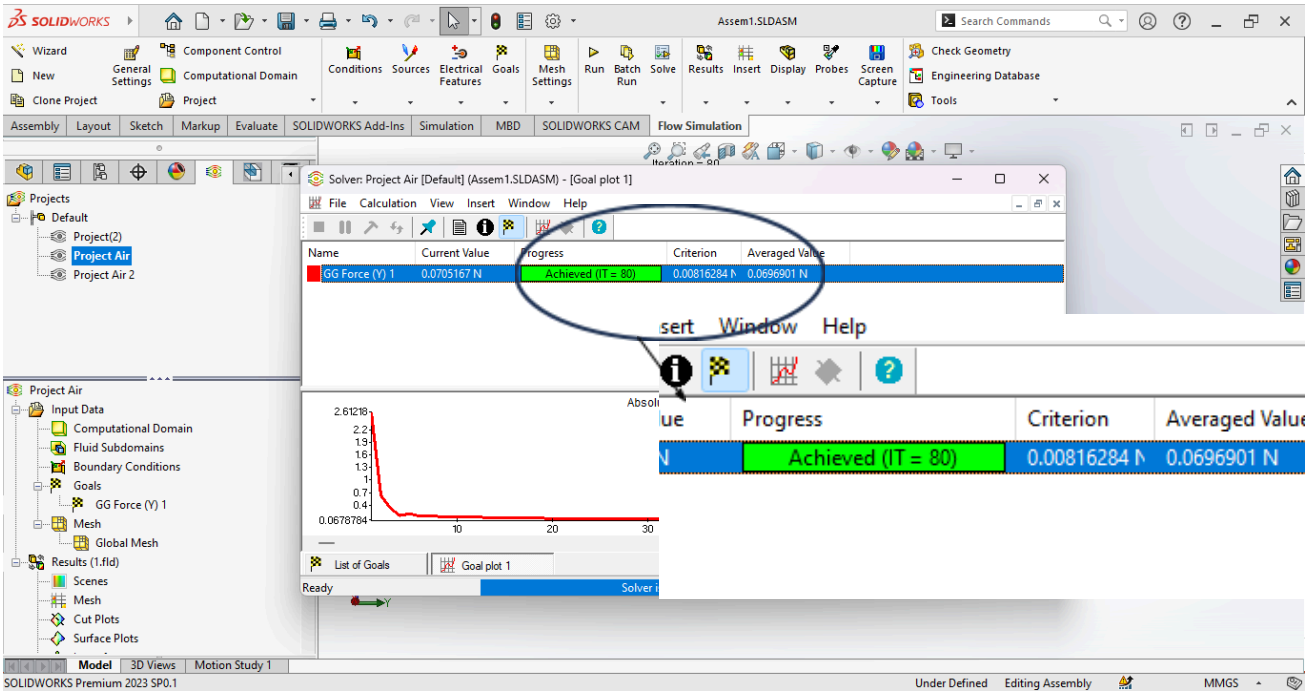
A = Surface area

ρ *= Density*

v *= Motor speed*







$$CD = \frac{2*F_d}{A*\rho*v^2} = \frac{2*0.0696}{0.56*1.225*2^2} = 0.0507$$

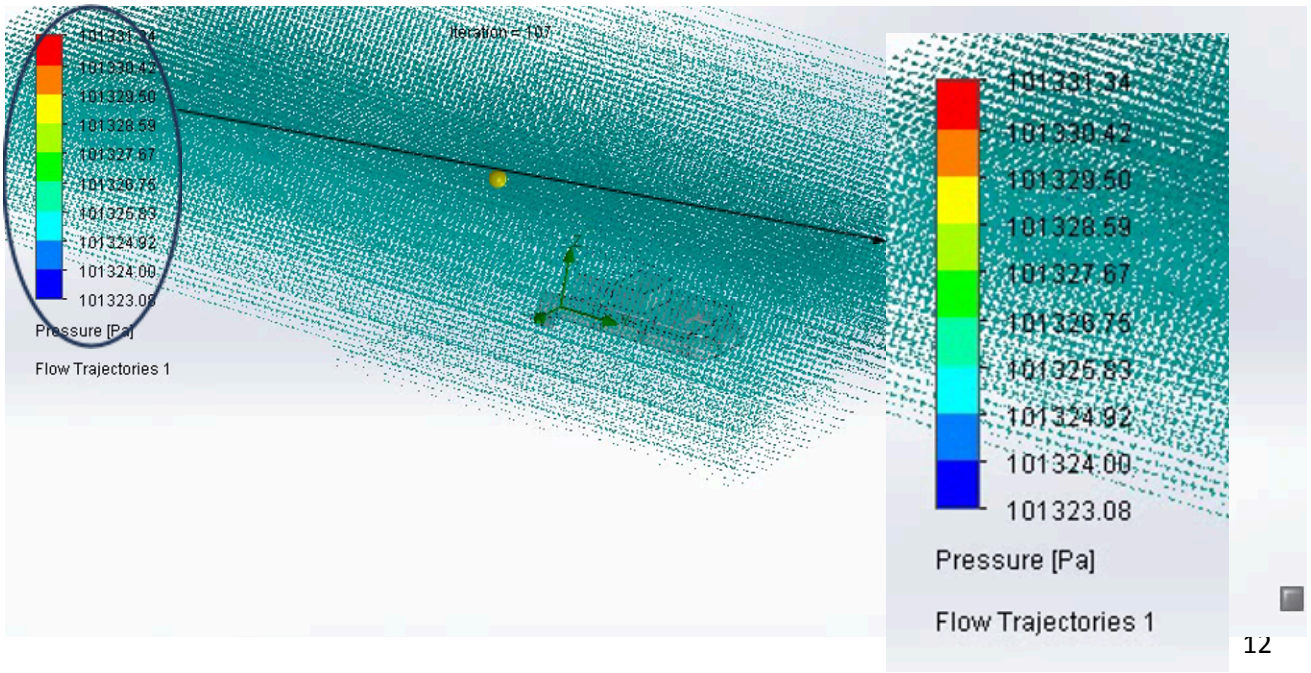
CD = Drag coefficient

F_d = Drag force

A = Surface area

ρ = *Density*

v = Motor speed



5 WIRING AND PCB

The electrical system of the autonomous rescue boat is a critical aspect ensuring reliable operation and seamless integration of components. A custom-designed printed circuit board (PCB) was developed to streamline connections and optimize the system's functionality. The PCB serves as the central hub for interfacing various modules, including the Arduino Mega microcontroller, GPS NEO6M module, ESP32CAM for thermal signal detection, ESP32 for communication, MPU 6050 IMU for stability, L298N motor driver for motor control, and HC-SR04 ultrasonic sensors for obstacle avoidance.

The wiring architecture follows a meticulous design to ensure proper communication and power distribution among components. For instance, the GPS NEO6M module is interfaced with the Arduino Mega using UART communication for real-time location data acquisition. Similarly, the ESP32CAM communicates with the Arduino Mega via UART for thermal signal detection, while the ESP32 manages overall communication tasks. The MPU 6050 IMU connects to the Arduino Mega via the I2C protocol, providing accurate inertial measurement data for balance control.

The motor control system employs the L298N motor driver, with precise wiring connections to the Arduino Mega's digital pins for motor speed and direction control. Additionally, the HC-SR04 ultrasonic sensors are integrated using a combination of digital pins and I2C communication for obstacle detection and avoidance.

The decision to utilize a custom PCB over manual wiring was driven by several advantages, including enhanced reliability, compactness, and ease of maintenance. The PCB design ensures secure and stable connections, reducing the risk of disconnections and short circuits, particularly crucial in dynamic water environments. Furthermore, the compact layout of the PCB optimizes space utilization within the boat, facilitating efficient component arrangement and minimizing clutter.

The electrical system of the autonomous rescue boat is meticulously designed to ensure reliable operation and seamless integration of components. A custom-printed circuit board (PCB) serves as the central hub for interfacing various modules, optimizing functionality and reducing wiring complexity.

5.1 Components Connections

GPS NEO6M Module:

- VCC to 5V.

- GND to GND.

- Tx to Arduino Mega Rx (pin 17).

- Rx to Arduino Mega Tx (pin 16).

ESP32CAM Module:

- VCC to 5V.

- GND to GND.

- Tx to Arduino Mega Rx (pin 19).

- Rx to Arduino Mega Tx (pin 18).

ESP32 Module:

- VCC to 5V.

- GND to GND.

- Tx to Arduino Mega Rx (pin 15).

- Rx to Arduino Mega Tx (pin 14).

MPU 6050 IMU:

- VCC to 5V.

- GND to GND.

- SDA to Arduino Mega SDA.

- SCL to Arduino Mega SCL.

L298N Motor Driver:

- IN1 to Arduino Mega digital pin 6.

- IN2 to Arduino Mega digital pin 5.

- IN3 to Arduino Mega digital pin 4.

- IN4 to Arduino Mega digital pin 3.

- ENA to Arduino Mega PWM pin 7.

- ENB to Arduino Mega PWM pin 2.

- OUT1 to left motor.

- OUT2 to left motor.

- OUT3 to right motor.

- OUT4 to right motor.

- VCC to 5V.

- GND to battery negative terminal and Arduino Mega GND.

- +12V to the 7.4V battery.

HC-SR04 Ultrasonic Sensors:

- VCC to 5V.

- GND to GND.

- Trig to Arduino Mega TX (pin 1).

- Echo to Arduino Mega RX (pin 0).

The connections outlined above ensure proper communication and power distribution among the system's components, facilitating reliable and efficient operation of the autonomous rescue boat. The custom PCB design further enhances reliability, compactness, and ease of maintenance, making it a strategic choice for complex marine applications.

We can see below the PCB schematic (Figure 1), PCB Layout and Footprint (Figure 2), PCB 3D Model (Figure 3), Proteus Wiring Diagram (Figure 4)

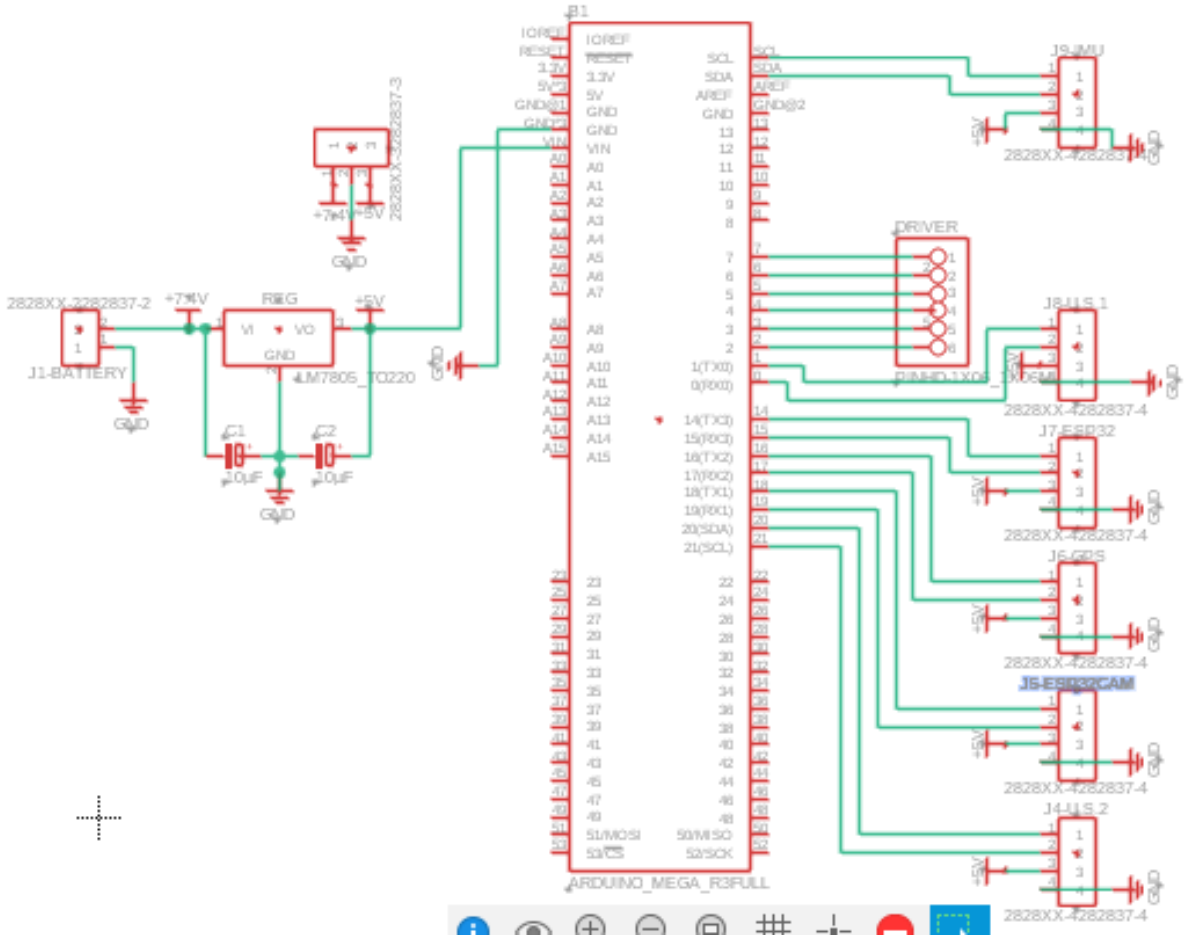


Figure 1: PCB Schematic illustrating the detailed connections and layout of components on the printed circuit board.

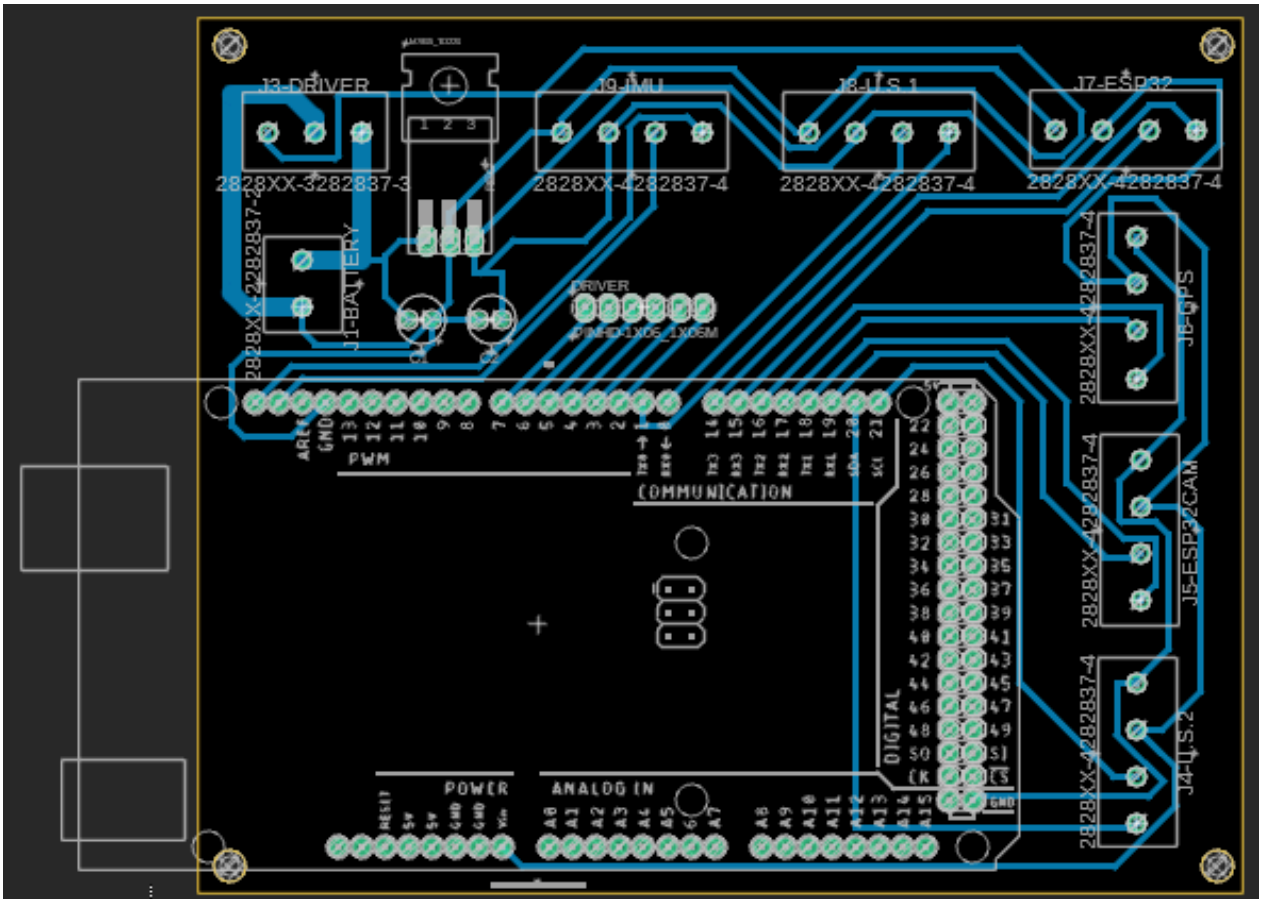


Figure 2: PCB Pin Layout showcasing the organized arrangement of pins and headers for interfacing with external modules.

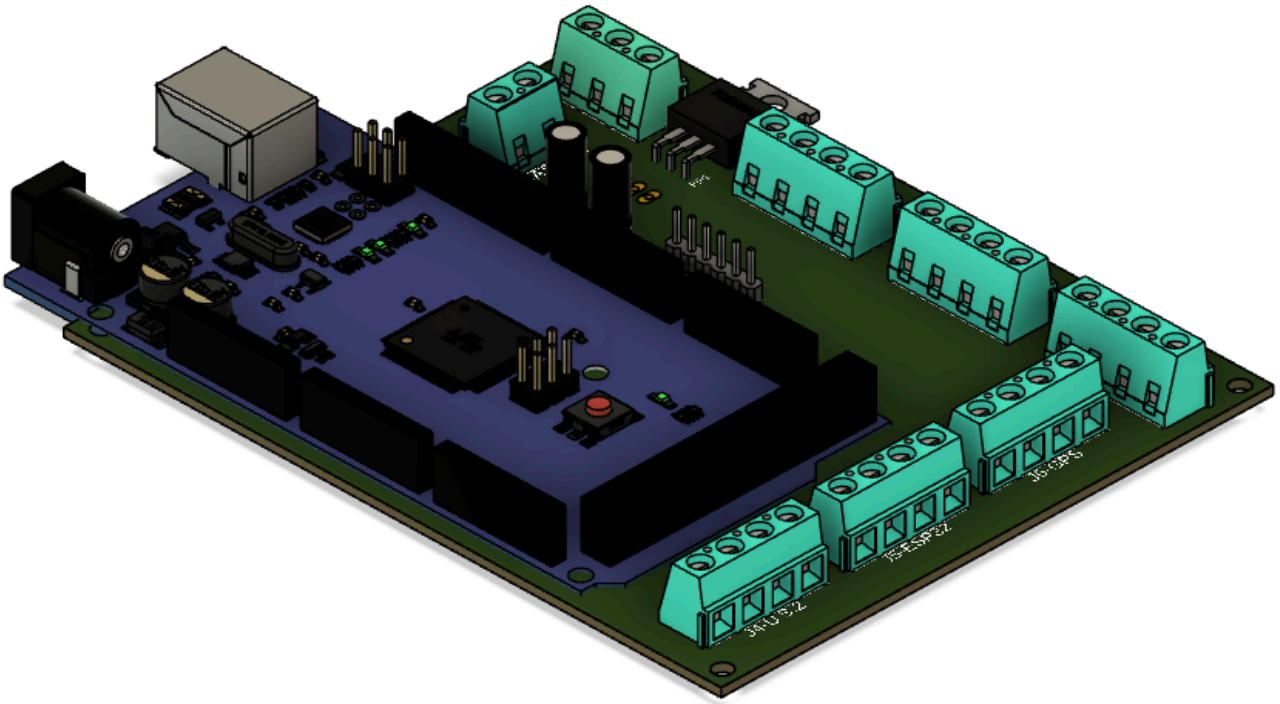


Figure 3: 3D Model of the PCB showcasing its compact and optimized design for integration within the autonomous rescue boat.

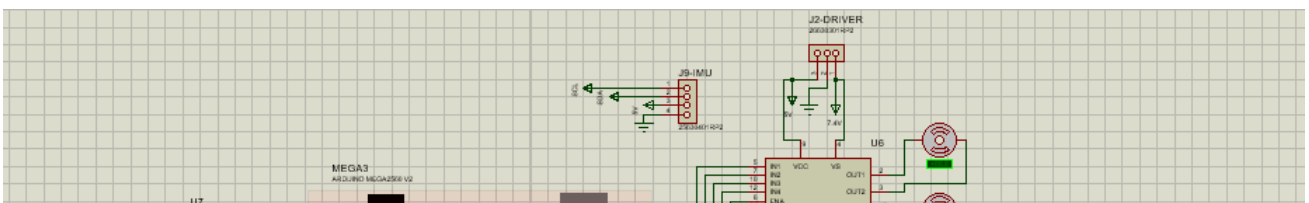


Figure 4: Proteus Wiring Diagram depicting the connections and interactions between components in the system.

In summary, the utilization of a custom-designed PCB not only enhances the reliability and stability of the autonomous rescue boat's electrical system but also simplifies maintenance and optimizes space utilization, making it a strategic choice for efficient and effective rescue operations in challenging water environments.

6 COMMUNICATION PROTOCOLS

6.1 UART Protocol

The Universal Asynchronous Receiver-Transmitter (UART) protocol is used for most components due to its simplicity and effectiveness. UART is a serial communication protocol that allows the microcontroller to communicate with peripherals and other microcontrollers.

Advantages of UART:

- **Ease of Implementation:** UART is straightforward to implement and widely supported by various microcontrollers and devices.
- **Direct Communication:** UART allows for direct point-to-point communication, making it ideal for modules like the GPS NEO6M and ESP32CAM.
- **Minimal Wiring:** Only two wires (Tx and Rx) are needed for communication, simplifying the wiring process.

Implementation in the System:

- **GPS NEO6M Module:** Connected to the Arduino Mega using UART to provide real-time location data.
- **ESP32CAM:** Utilizes UART for transmitting thermal signal detection data to the Arduino Mega.
- **ESP32:** Manages overall communication tasks, sending and receiving data via UART to ensure effective coordination and control.
- **Ultrasonic:** Measure the distance between the system and any obstacles in front of the system and connected to the Arduino Mega UART pins.

Why We Chose UART:

- **Compatibility:** Most of our components, such as the GPS module and ESP32CAM, support UART communication, making it a natural choice.
- **Simplicity:** UART's straightforward setup and reliable performance make it ideal for our application, where robust and simple communication is essential.

6.2 I2C Protocol

The Inter-Integrated Circuit (I2C) protocol is employed for communication between the Arduino Mega and the MPU 6050 IMU. I2C is a multi-master, multi-slave, packet-switched, single-ended, serial communication bus widely used for attaching lower-speed peripheral ICs to processors and microcontrollers.

Advantages of I2C:

- **Multi-Device Capability:** I2C can connect multiple devices on the same bus, reducing the number of required pins on the microcontroller.
- **Simplicity in Wiring:** Only two wires (SDA and SCL) are needed, regardless of the number of connected devices, which simplifies the design and reduces clutter.

- **Addressability:** Each device on the I2C bus has a unique address, allowing the microcontroller to communicate with multiple devices efficiently.
- **Efficiency:** I2C is efficient for short-distance communication within the system, ensuring quick and reliable data transfer between the IMU and the microcontroller.

Implementation in the System:

- **MPU 6050 IMU:** Connected to the Arduino Mega via I2C, providing critical inertial measurement data for maintaining the stability and balance of the boat.

Why did we choose I2C:

- **Efficiency:** I2C's ability to connect multiple devices with just two wires is ideal for our compact and integrated design.
- **Device Compatibility:** The MPU 6050 IMU is are compatible with I2C, making it the best choice for these components.

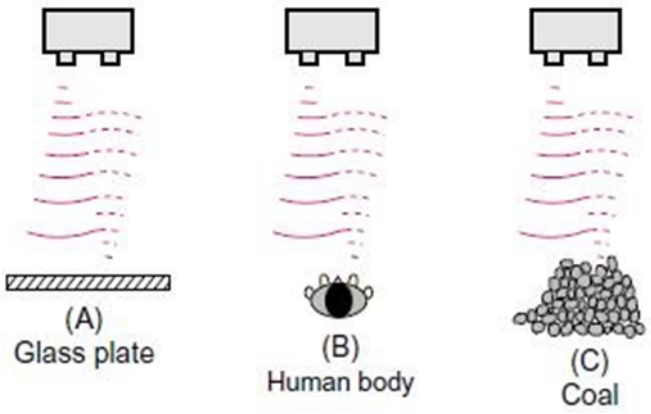
By combining UART and I2C protocols, our system benefits from the strengths of both, ensuring reliable and efficient communication between the Arduino Mega and all connected components. This hybrid approach allows us to leverage the simplicity of UART for straightforward, point-to-point communications and the efficiency of I2C for multi-device connectivity.

7 COMPONENTS:

7.1 Ultrasonic Sensor:

An ultrasonic sensor is an electronic device that measures the distance to an object or detects its presence by using ultrasonic sound waves. It works by emitting sound waves at frequencies higher than the human hearing range (typically above 20 kHz) and then listening for the echoes that bounce back from the object.

The sensor calculates the distance based on the time it takes for the sound waves to travel to the object and return. Ultrasonic sensors are known for their high precision, durability, and ability to perform in various environmental conditions.



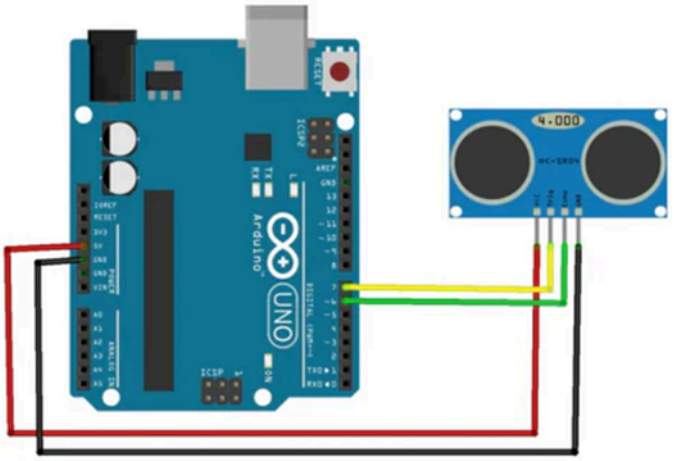
7.1.1 Features:

- Power consumption of 20mA
- Pulse in/out communication.
- Narrow acceptance angle
- Provides exact, non-contact separation estimations within 2cm to 3m.
- The explosion points LED shows estimations in the advancement
- 3-pin header makes it simple to connect utilizing a servo development link

7.1.2 Specifications:

- Power supply: 5V DC
- Quiescent current: <15mA
- Effectual angle: <15°
- Ranging distance: 2cm – 350 cm
- Resolution: 0.3 cm
- Output cycle: 50ms

7.1.3 Connection:



7.1.4 Why do we use Ultrasonic sensors?

1. Non-Contact Measurement

Safe and Hygienic: Ideal for applications where contact with the object is undesirable or hazardous, such as in medical equipment or food processing.

Prevents Wear and Tear: Reduces the risk of damage to the sensor and the object being measured.

2. High Precision and Accuracy

Exact Measurements: Capable of providing precise distance measurements, which is essential in applications like level measurement in tanks or precise object positioning in manufacturing processes.

3. Wide Range of Measurement

Versatility: Can measure distances from a few centimeters to several meters, making them suitable for a wide range of applications, from small-scale robotics to large industrial tanks.

4. Environmental Robustness

Dust and Dirt Resistant: Perform well in dirty, dusty, or wet environments, unlike optical sensors that may be affected by such conditions.

Unaffected by Light Conditions: Operate independently of ambient light, making them reliable in varying lighting conditions, including complete darkness.

5. Quick Response Time

Real-Time Applications: Suitable for dynamic applications requiring rapid measurements, such as in collision avoidance systems in robotics or vehicles.

6. Wide Field of View

Broad Detection Area: Useful for applications where a wide detection zone is needed, such as in automated guided vehicles (AGVs) and industrial automation.

7. Durability and Reliability

Long Lifespan: Built to last in industrial environments, reducing the need for frequent replacements.

Low Maintenance: Robust design minimizes the need for maintenance, providing a reliable solution over time.

8. Ease of Integration

Simple Interfaces: Easy to connect with various microcontrollers, PLCs, and other control systems using standard interfaces like analog voltage, digital pulse, I2C, SPI, or UART.

Flexibility: Can be integrated into existing systems with minimal modifications, making them versatile and adaptable to different setups.

9. Cost-Effective

Affordable Technology: Provides a good balance of performance and cost, making advanced measurement capabilities accessible for a wide range of applications.

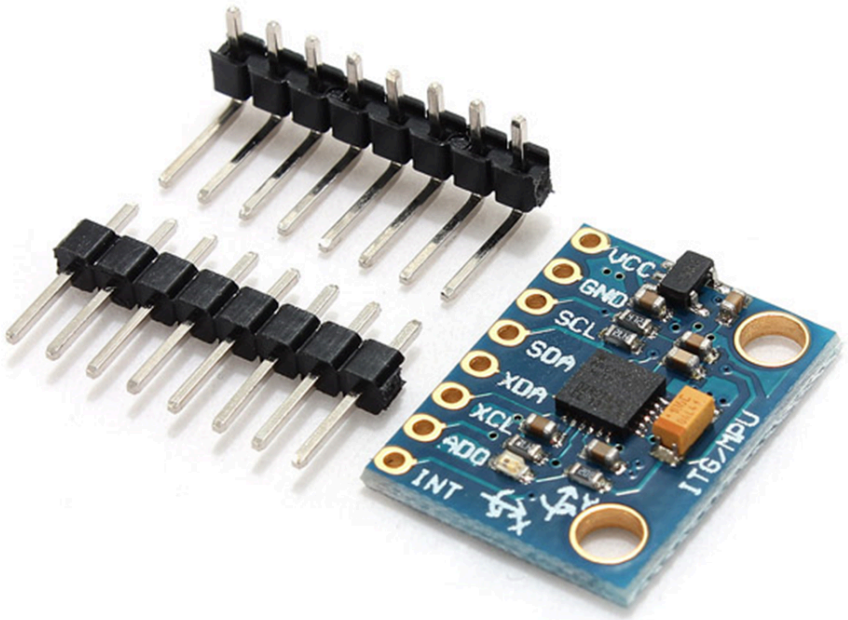
10. Low Power Consumption

Energy Efficiency: Suitable for battery-operated devices and systems where power consumption is a critical factor.

11. Temperature Compensation

Accurate in Varying Conditions: Some models come with built-in temperature compensation to maintain measurement accuracy despite changes in ambient temperature.

7.2 MPU-6050 (IMU):



The MPU-6050 is a widely used Inertial Measurement Unit (IMU) manufactured by TDK Intenseness. It integrates a 3-axis gyroscope and a 3-axis accelerometer into a single chip package.

The type of MPU-6050 is MEMS (Microelectromechanical Systems) IMUs: These are small, lightweight, and cost-effective IMUs commonly used in consumer electronics, drones, and wearable devices. They offer decent performance for many applications at a relatively low cost.

7.2.1 Features:

1.Gyroscope Features

- The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features.
- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video, and GPS synchronization.
- Integrated 16-–bit ADCs enable simultaneous sampling of gyros.

- Enhanced bias and sensitivity temperature stability reduces the need for user calibration.
- Improved low-frequency noise performance.
- Digitally programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5μA
- Factory calibrated sensitivity scale factor
- User self-test

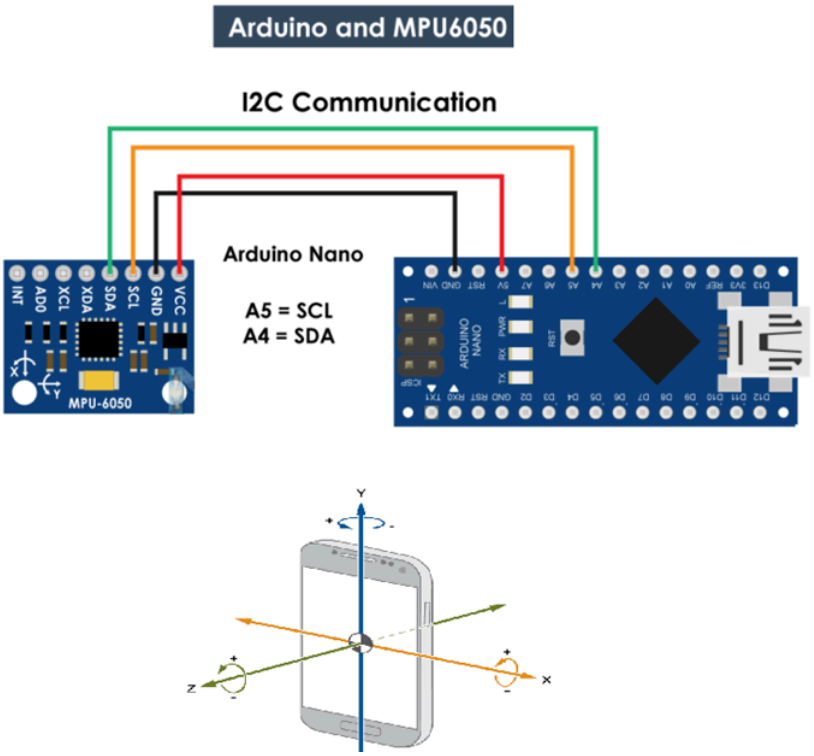
2.Accelerometer Features

- The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:
- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer.
- Accelerometer normal operating current: 500μA
- Low power accelerometer mode current: 10μA at 1.25Hz, 20μA at 5Hz, 60μA at 20Hz, 110μA at 40Hz
- Orientation detection and signaling
- Tap detection.
- User-programmable interrupts
- High-G interrupt
- User self-test

7.2.2 Specifications:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I 2C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024-byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 g shock tolerant
- 400kHz Fast Mode I2C for communicating with all registers.
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only.

7.2.3 Connection:



7.2.4 Why do we use it?

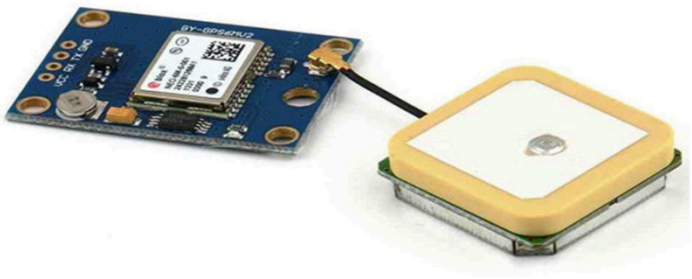
1. Cost-Effective: The MPU-6050 offers a balance between performance and cost, making it an affordable choice for many applications.
2. Compact Size: It comes in a small package, making it suitable for integration into various electronic devices without occupying much space.
3. Low Power Consumption: Designed for low-power operation, the MPU-6050 is suitable for battery-powered applications, helping extend device runtime.
4. Digital Motion Processor (DMP): The built-in DMP offloads sensor fusion calculations from the host microcontroller, simplifying application development and reducing the workload on the main processor.
5. Easy Integration: The MPU-6050 communicates with the host microcontroller via standard I2C or SPI interfaces, facilitating easy integration into different microcontroller platforms.

6. Wide Operating Range: It operates over a wide range of voltages and temperatures, making it suitable for use in diverse environmental conditions. The range of temperature is from -40 to 85.

We used IMU for:

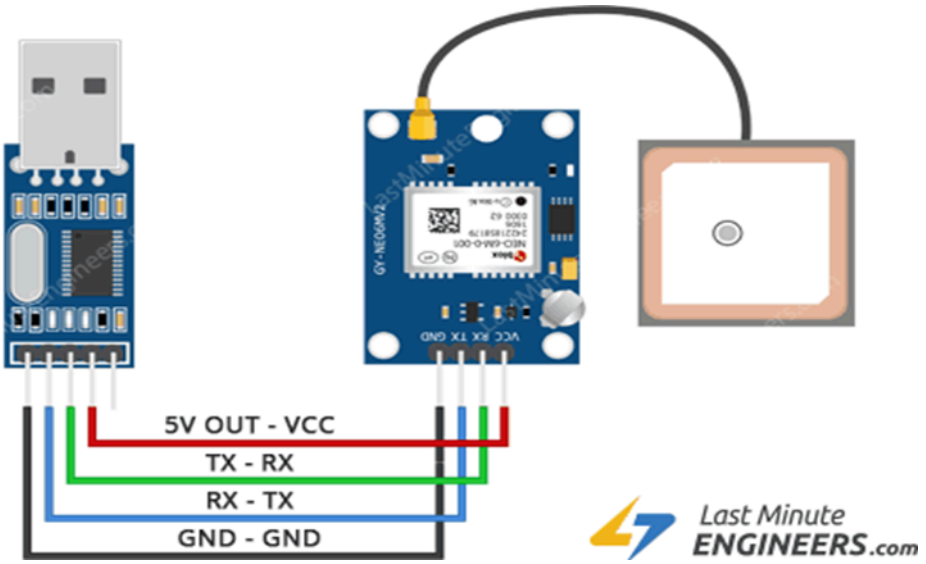
- provide real-time data on the boat's orientation, including its roll, pitch, and yaw angles.
- reduce the rolling motion of the boat caused by waves and wind.
- detecting sudden changes in motion or attitude
- monitor the boat's performance, including its acceleration, deceleration, and turning dynamics.

7.3 GPS-NEO-6 M:



The NEO-6M is a popular GPS module commonly used in various applications such as drones, robotics, navigation systems, and more. It's manufactured by u-blox, a Swiss company specializing in positioning and wireless communication technologies. The NEO-6M module uses the Global Navigation Satellite System (GNSS) to provide accurate positioning data by receiving signals from satellites. It supports multiple satellite systems including GPS, GLONASS, and BeiDou, offering high accuracy and reliability in determining location coordinates, velocity, and time synchronization. The module typically communicates with a microcontroller or a computer through serial communication protocols like UART (Universal Asynchronous Receiver-Transmitter), making it easy to integrate into various projects.

7.3.1 Connection:



7.3.2 Why do we use it?

1. High Accuracy: It provides accurate positioning data with a high level of precision, allowing for reliable navigation and tracking.

2. Low Power Consumption: The module is designed to operate efficiently with low power consumption, making it suitable for battery-powered applications where power efficiency is crucial.

3. Compact Size: The NEO-6M module is compact and lightweight, making it easy to integrate into various electronic devices and systems without taking up much space.

4. Flexible Interface: It communicates with external devices such as microcontrollers or computers via serial communication protocols like UART, which is a common and easy-to-use interface in embedded systems.

5. Cost-effective Solution: The NEO-6M module provides high-performance GPS capabilities at a relatively low cost, making it an affordable option for hobbyists, enthusiasts, and commercial projects alike.

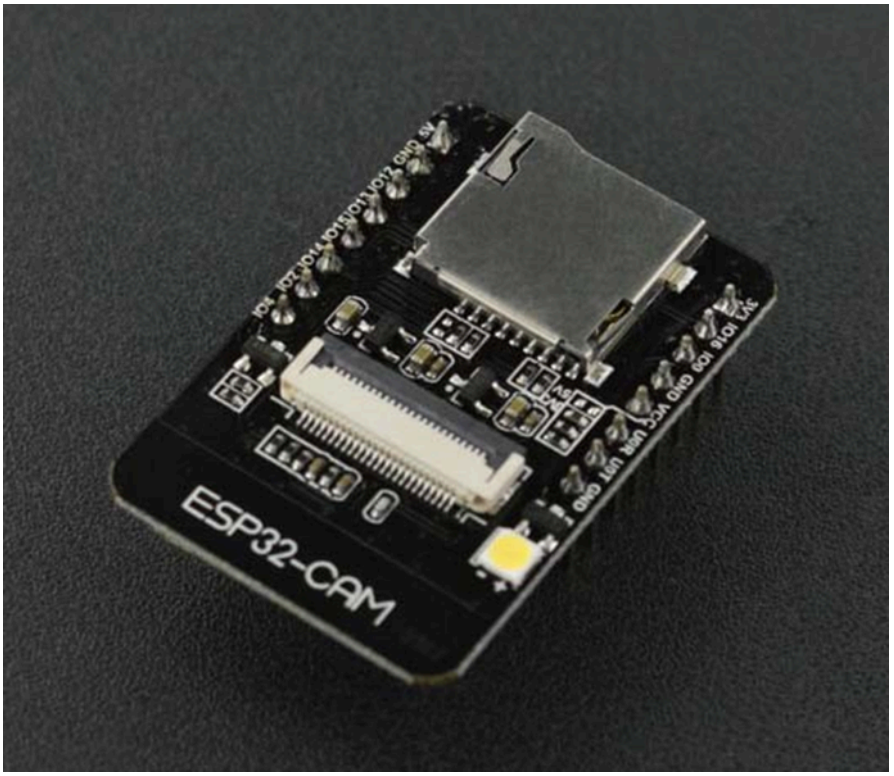
6. Robust Performance: It offers robust performance in various environmental conditions, including urban areas, forests, and open terrain, making it suitable for a wide range of applications requiring reliable GPS navigation and positioning.

7. Wide Operating Temperature Range: The module can operate over a wide temperature range, typically from -40°C to 85°C (-40°F to 185°F), making it suitable for use in harsh environmental conditions.

We used it for:

- Vehicle Tracking: It's used in GPS vehicle tracking systems to monitor the location, speed, and route of vehicles in real-time. This application is prevalent in fleet management, logistics, and transportation industries.
- Emergency Response and Public Safety: Used in emergency response vehicles, search and rescue operations, and public safety applications to track the location of emergency personnel and resources, facilitating rapid response to incidents.

7.4 ESP32-CAM Development Board:



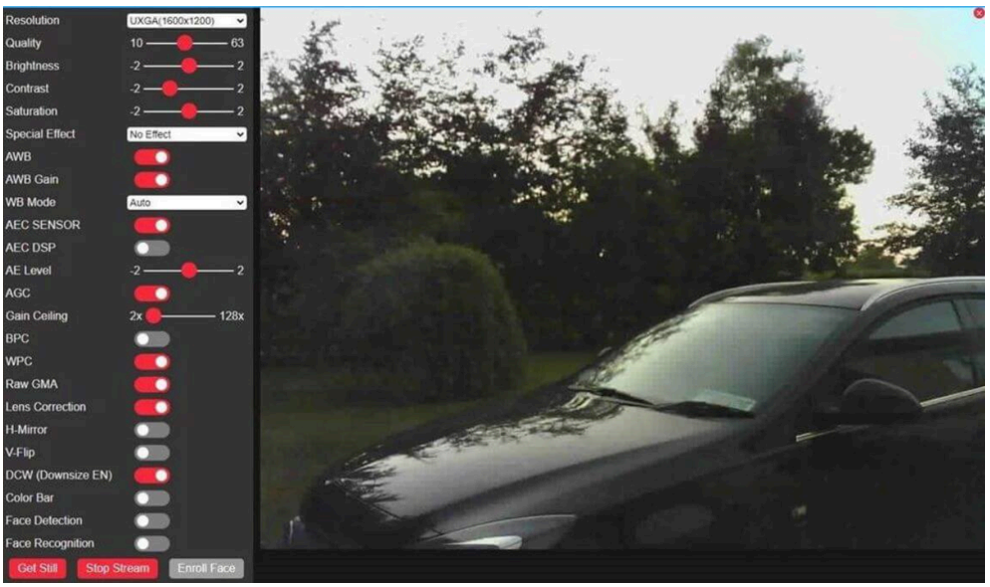
ESP32-CAM is a low-cost ESP32-based development board with onboard camera, small in size. It is an ideal solution for IoT application, prototypes constructions and DIY projects.

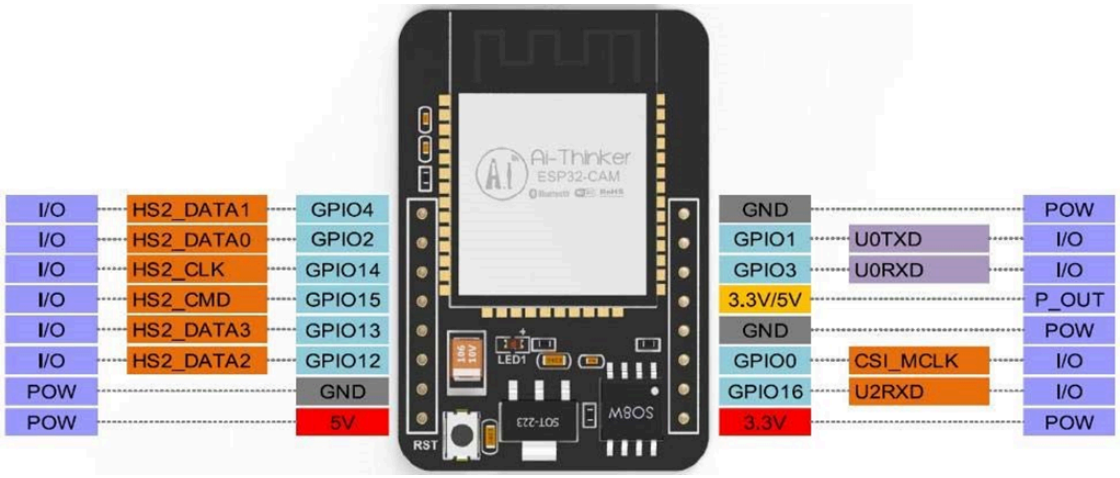
The board integrates WiFi, traditional Bluetooth and low power BLE , with 2 highperformance 32-bit LX6 CPUs. It adopts 7-stage pipeline architecture, on-chip sensor, Hall sensor, temperature sensor and so on, and its main frequency adjustment ranges from 80MHz to 240MHz.

Fully compliant with WiFi 802.11b/g/n/e/i and Bluetooth 4.2 standards, it can be used as a master mode to build an independent network controller, or as a slave to other host

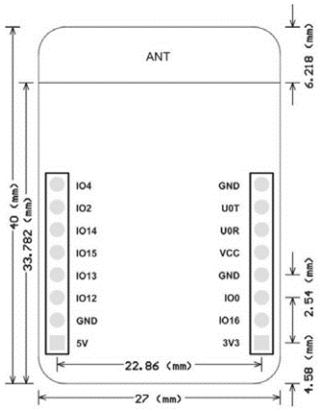
MCUs to add networking capabilities to existing devices

ESP32-CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is an ideal solution for IoT applications.





Schematic Diagram



Dimension Diagram

7.4.1 Features:

- Up to 160MHz clock speed Summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, Built-in Flash lamp.
- Support image Wi-Fi upload.
- Support TF card.
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode,
- Support Smart Config/AirKiss technology.
- Support for serial port local and remote firmware upgrades (FOTA)

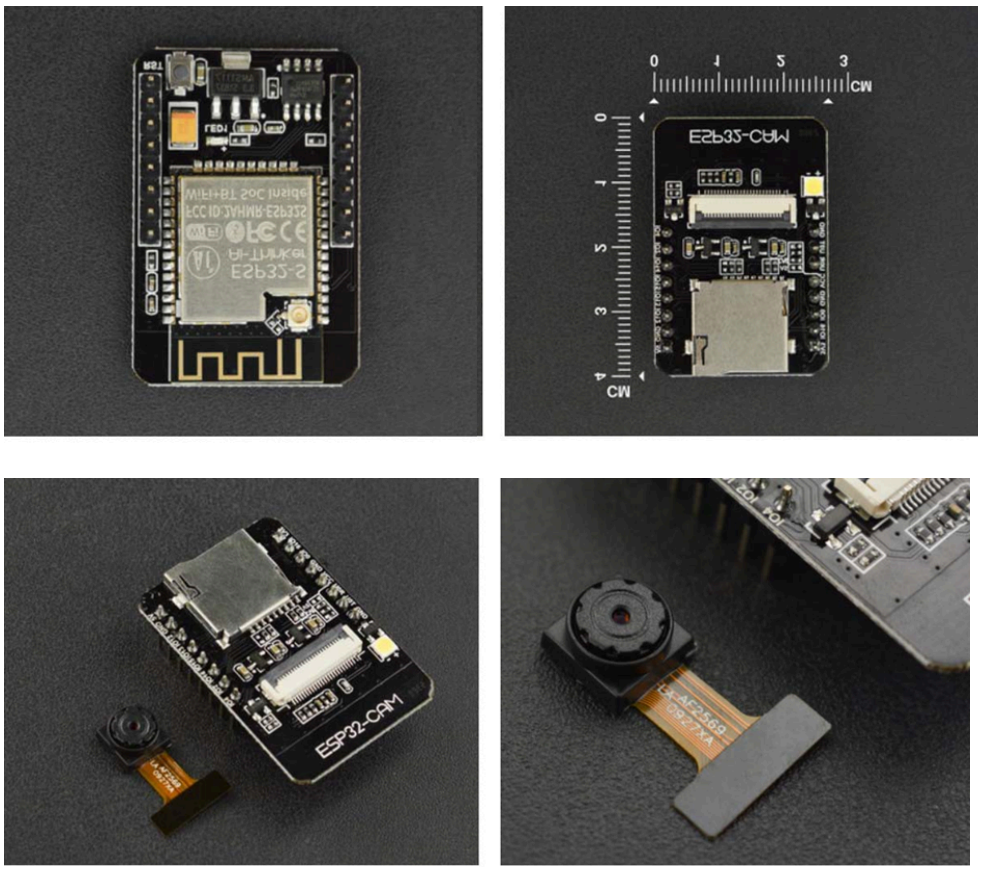
7.4.2 Specifications:

- SPI Flash: default 32Mbit
- RAM: built-in 520 KB+external 4MPSRAM
- Dimension: 27*40.5*4.5（±0.2）mm/1.06*1.59*0.18”
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE standards
- Wi-Fi: 802.11b/g/n/e/I
- Support Interface: UART, SPI, I2C, PWM
- Support TF card: maximum support 4G.
- IO port: 9
- Serial Port Baud-rate: Default 115200 bps

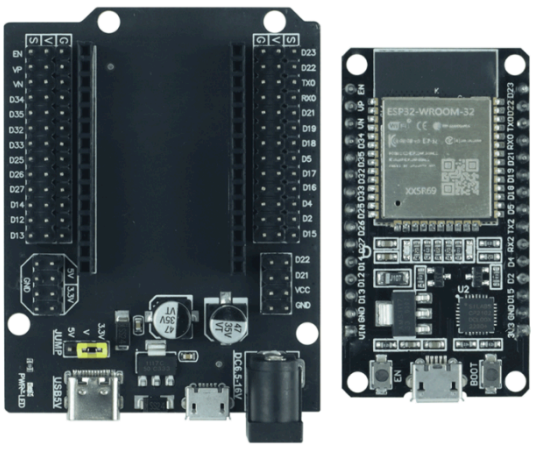
- Image Output Format: JPEG (OV2640 support only), BMP, GRAYSCALE
- Spectrum Range: 2412 ~2484MHz
- Antenna: onboard PCB antenna, gain 2dBi
- Transmit Power: 802.11b: 17±2 dBm (@11Mbps), 802.11g: 14±2 dBm (@54Mbps); 802.11n: 13±2 dBm (@MCS7)
- Receiving Sensitivity: CCK, 1 Mbps: -90dBm.
  - CCK, 11 Mbps: -85dBm.
  - 6 Mbps (1/2 BPSK): -88dBm.
  - 54 Mbps (3/4 64-QAM): -70dBm.
  - MCS7 (65 Mbps, 72.2 Mbps): -67dBm
- Power consumption: Turn off the flash: 180mA@5V.
  - Turn on the flash and adjust the brightness to the maximum: 310mA@5V.
  - Deep sleep: the lowest power consumption can reach 6mA@5V.
  - Modern sleep: up to 20mA@5V
  - Light-sleep: up to 6.7mA@5V
- Security: WPA/WPA2/WPA2-Enterprise/WPS
- Power supply range: 5V
- Operating temperature: -20 °C ~ 85 °C
- Storage environnement : -40 °C ~ 90 °C, < 90%RH
- Weight: 10g

7.4.3 Shipping list:

- ESP32-CAM Development Board x1



7.5 Node MCU ESP32:

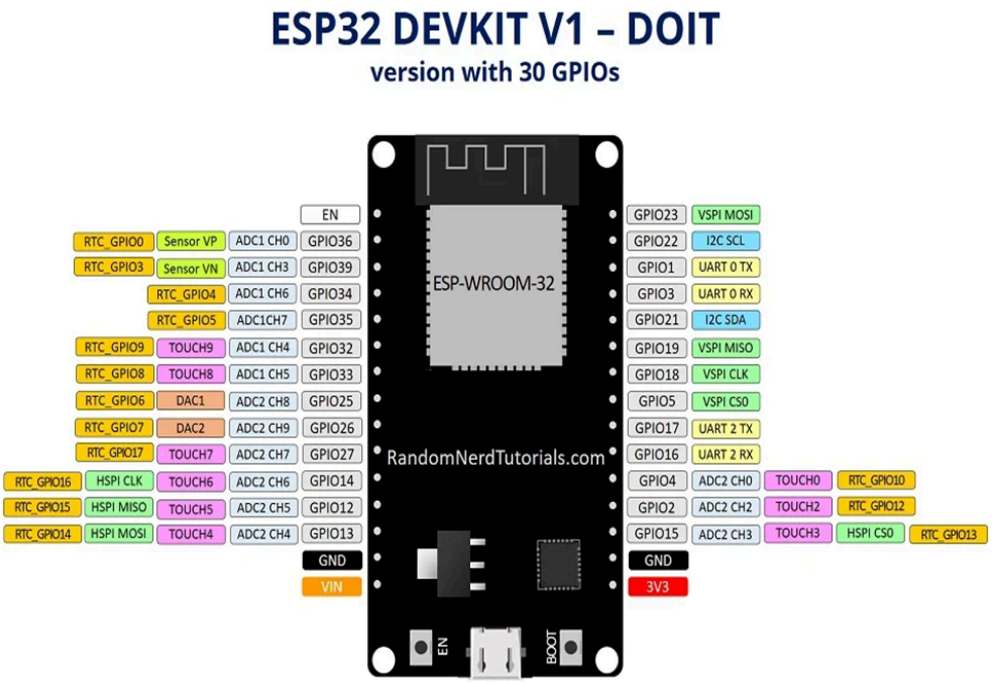


NodeMCU is famous for the ESP8266E module with the LUA programming language. Now, this is a more powerful NodeMCU with ESP32 on it! ESP32 is the big brother of ESP8266. It comes with a dual-core 32-bit processor, built-in WiFi, and Bluetooth, more SRAM and Flash memory, more GPIO, more ADC, and many other peripherals. NodeMCU ESP32 is an ESP-WROOM-32 module in breadboard friendly form factor, you can develop your project by using this compact microcontroller on a breadboard.

7.5.1 Features:

- NodeMCU based on ESP-WROOM-32 module.
- 30 GPIO Version

- ESP32 is a dual-core 32-bit processor with built-in 2.4 GHz Wi-Fi and Bluetooth.
- 4MByte flash memory
- 520KByte SRAM.
- 2.2 to 3.6V Operating voltage range.
- In breadboard-friendly breakout.
- USB micro B for power and Serial communication used to load the program and serial debugging too.



7.5.2 Additional features:

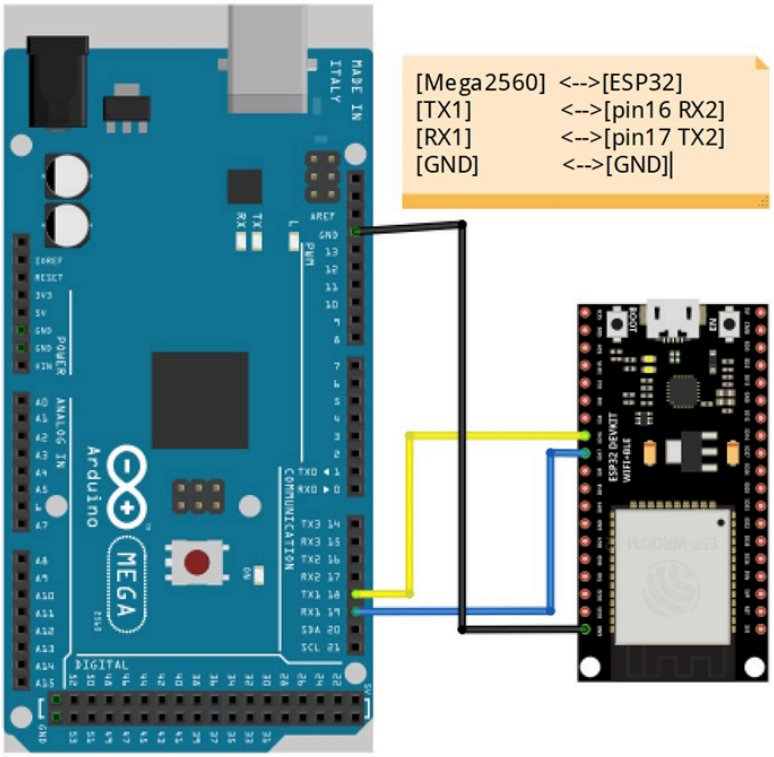
NodeMCU ESP32

Microcontroller Development Board

Technical Specifications

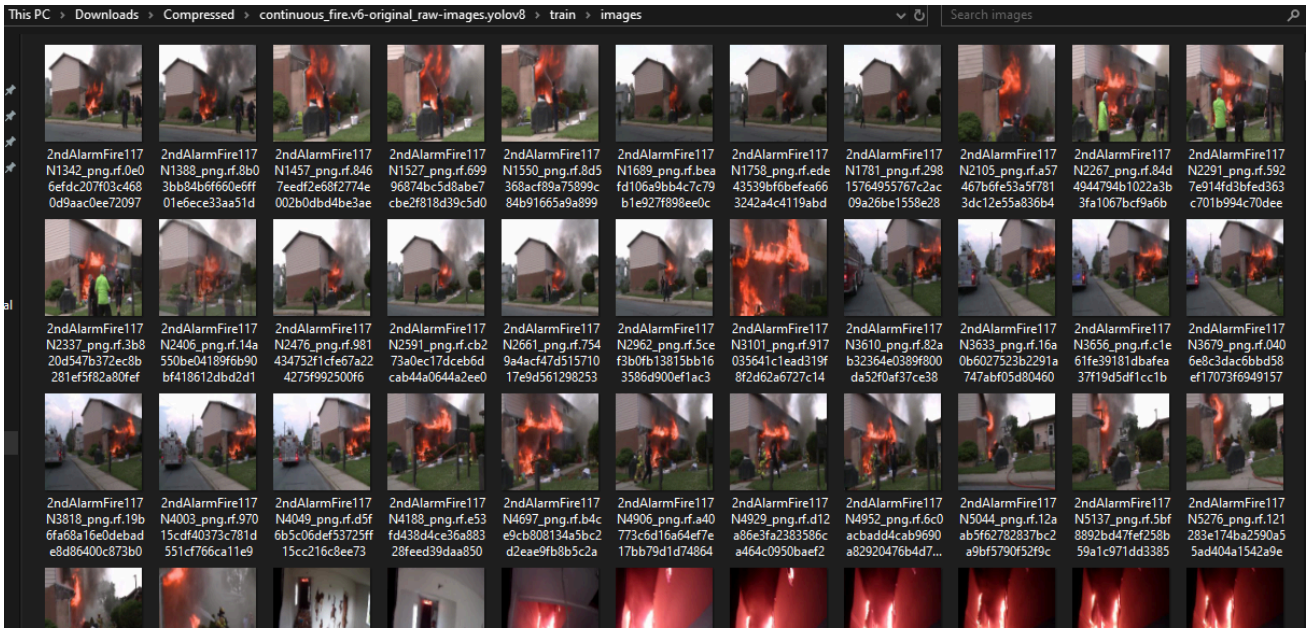
Model	NodeMCU ESP32
Article No.	SBC-NodeMCU-ESP32
Type	ESP32
Processor	Tensilica LX6 Dual-Core
Clock Frequency	240 MHz
SRAM	512 kB
Memory	4 MB
Wireless Standard	802.11 b/g/n
Frequency	2.4 GHz
Bluetooth	Classic / LE
Data Interfaces	UART / I2C / SPI / DAC / ADC
Operating Voltage	3,3V (operable via 5V-microUSB)
Operating Temperature	-40°C - 125°C
Dimensions (W x D x H)	48 x 26 x 11.5 mm
Scope Of Delivery	NodeMCU ESP32
EAN	4250236816104

7.5.3 Connection:

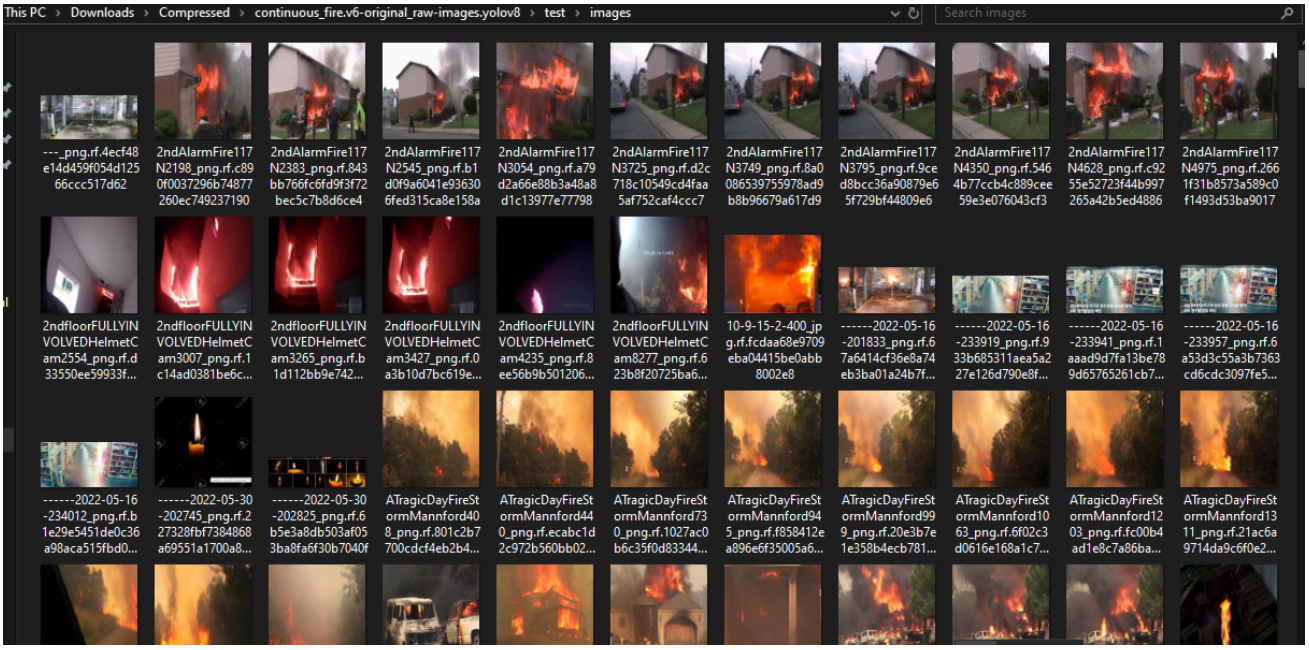


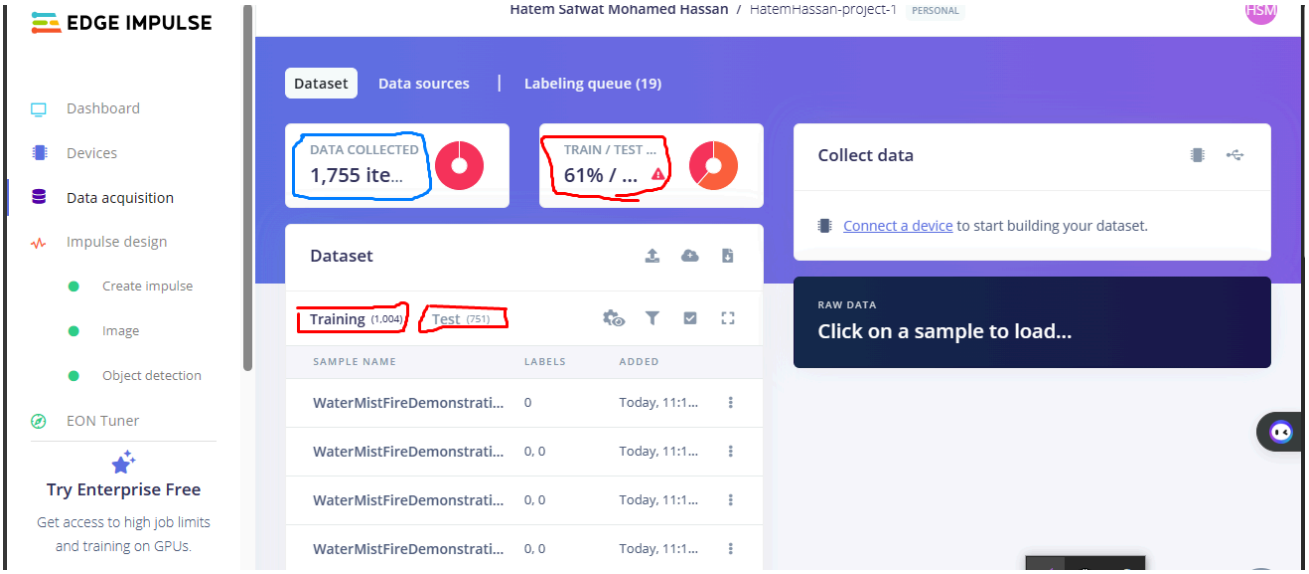
7.5.4 Training Procedures

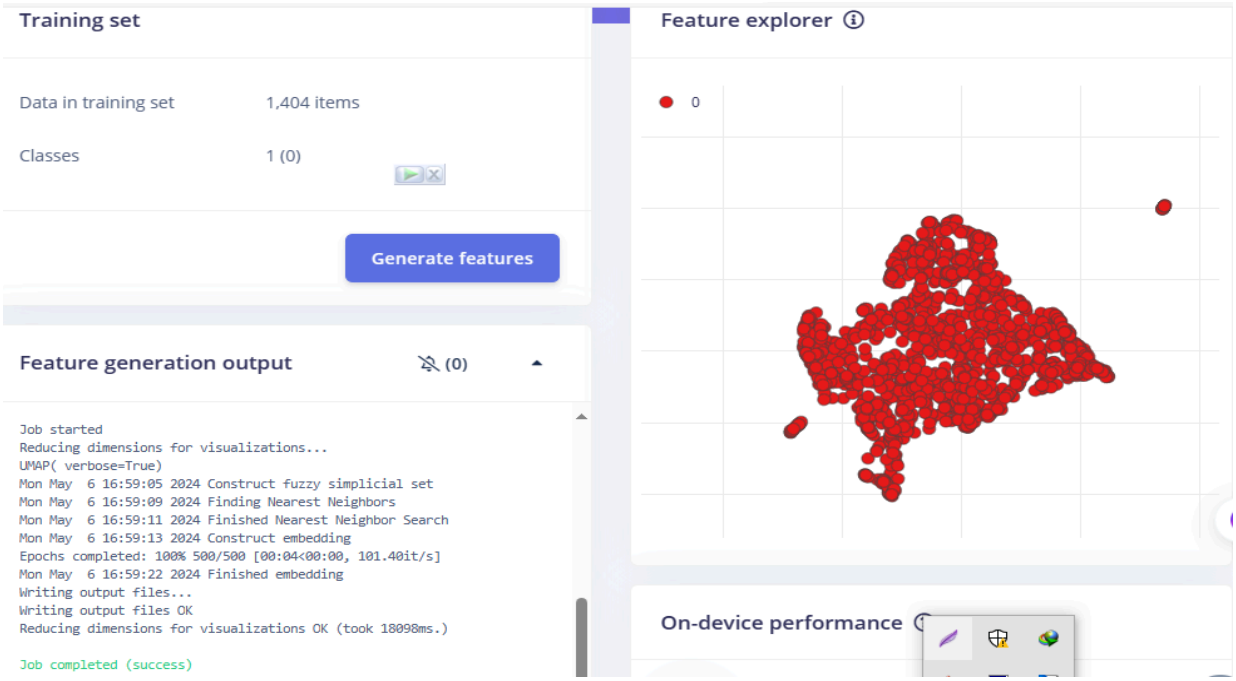
During the training process for the ESP32 Camera, firstly, we got hundreds of fire images from the internet and hand-captured, then we put all these samples on the EDGE IMPULSE website to start training the camera using the provided samples, and finally we got a recognition accuracy of 80% fire



detection.


This PC › Downloads › Compressed › continuous_fire.v6-original_raw-images.yolov8 › test › images
Search images


EDGE IMPULSE
Hatem Satwat Mohamed Hassan / HatemHassan-project-1
PERSONAL
Dashboard
Devices
Data acquisition
Impulse design
Create impulse
Image
Object detection
EON Tuner
Try Enterprise Free
Get access to high job limits and training on GPUs.
Dataset
Data sources
Labeling queue (19)
DATA COLLECTED
1,755 ite...
TRAIN / TEST ...
61% / ...
Collect data
Connect a device to start building your dataset.
Dataset
Training (1,004)
Test (751)
SAMPLE NAME
LABELS
ADDED
WaterMistFireDemonstrati... 0 Today, 11:1...
WaterMistFireDemonstrati... 0, 0 Today, 11:1...
WaterMistFireDemonstrati... 0, 0 Today, 11:1...
WaterMistFireDemonstrati... 0, 0 Today, 11:1...
RAW DATA
Click on a sample to load...


Training set
Data in training set 1,404 items
Classes 1 (0)
Generate features
Feature generation output
(0)
Job started
Reducing dimensions for visualizations...
UMAP(verbose=True)
Mon May 6 16:59:05 2024 Construct fuzzy simplicial set
Mon May 6 16:59:09 2024 Finding Nearest Neighbors
Mon May 6 16:59:11 2024 Finished Nearest Neighbor Search
Mon May 6 16:59:13 2024 Construct embedding
Epochs completed: 100% 500/500 [00:04<00:00, 101.40it/s]
Mon May 6 16:59:22 2024 Finished embedding
Writing output files...
Writing output files OK
Reducing dimensions for visualizations OK (took 18098ms.)
Job completed (success)
Feature explorer
0
On-device performance

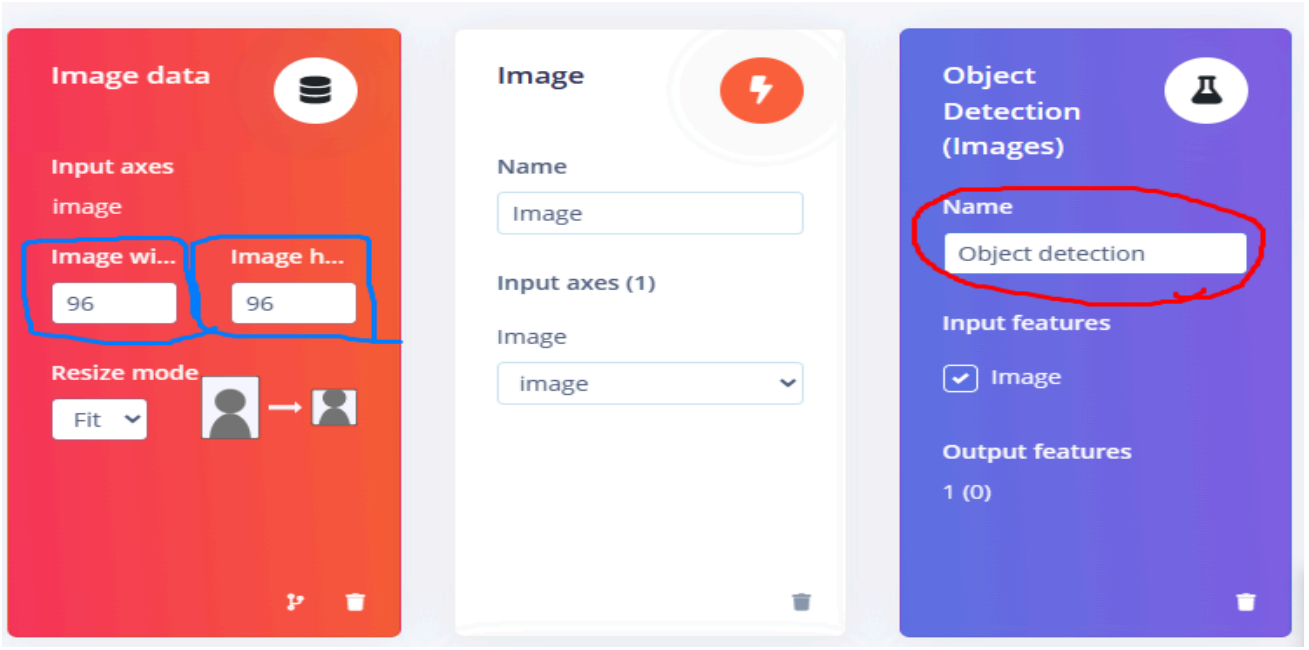
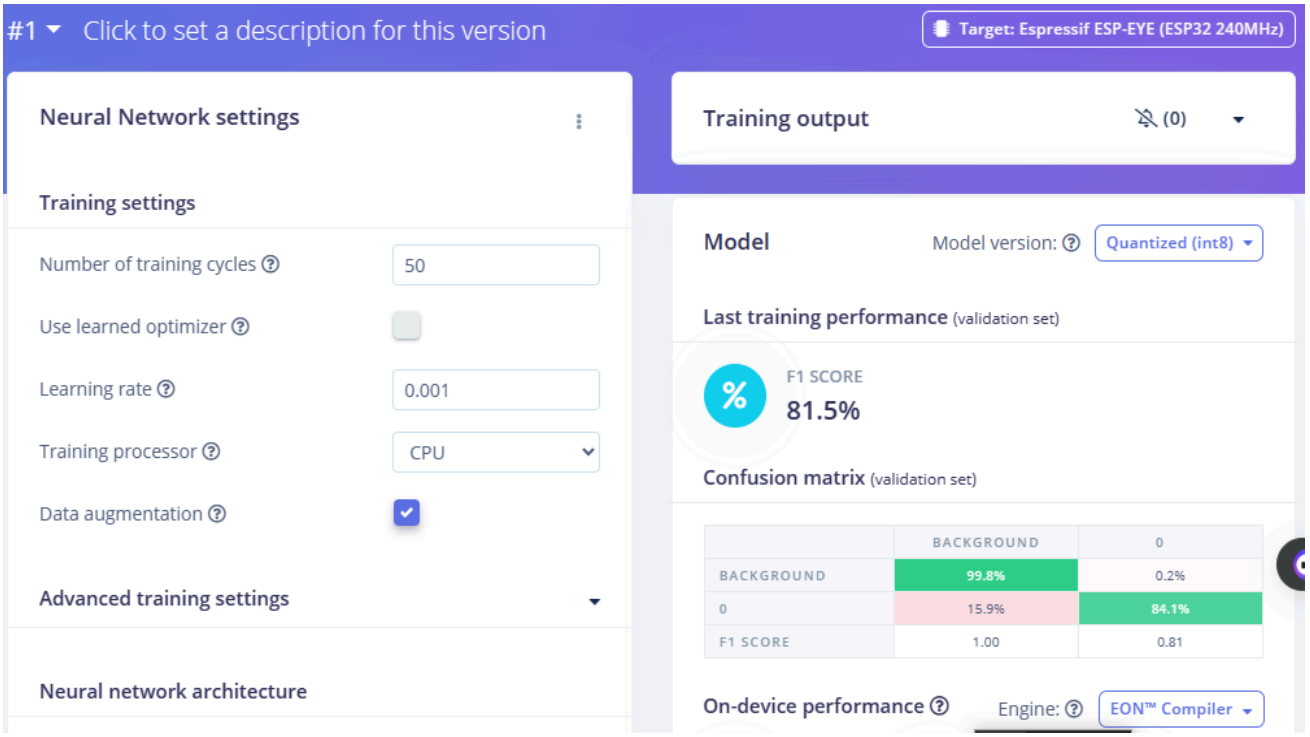
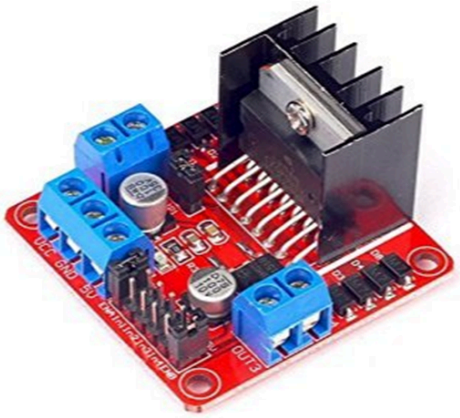


Image data
Input axes
image
Image wi...
96
Image h...
96
Resize mode
Fit
Image
Name
Image
Input axes (1)
Image
image
Object Detection (Images)
Name
Object detection
Input features
Image
Output features
1 (0)


#1
Click to set a description for this version
Target: Espressif ESP-EYE (ESP32 240MHz)
Neural Network settings
Training settings
Number of training cycles
50
Use learned optimizer
Learning rate
0.001
Training processor
CPU
Data augmentation
Advanced training settings
Neural network architecture
Training output
(0)
Model
Model version:
Quantized (int8)
Last training performance (validation set)
F1 SCORE
81.5%
Confusion matrix (validation set)
BACKGROUND
0
BACKGROUND
99.8%
0.2%
0
15.9%
84.1%
F1 SCORE
1.00
0.81
On-device performance
Engine:
EON™ Compiler

7.6 L298N H-Bridge:

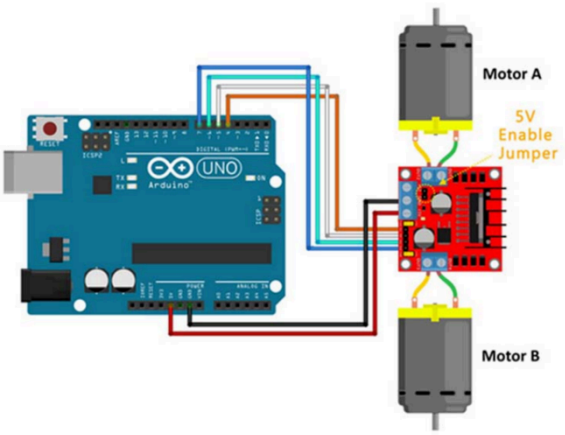


This dual bidirectional motor driver is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions. It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes.

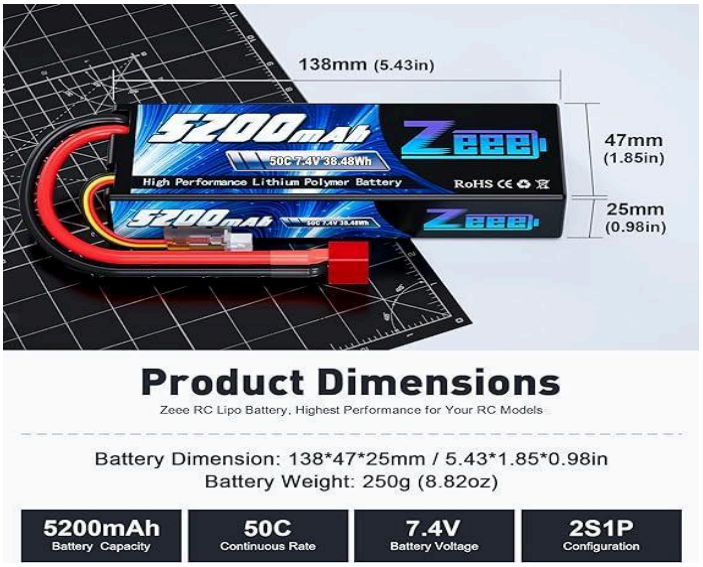
7.6.1 Features:

- Input Voltage: 3.2V~40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver
- Power Supply: DC 5 V - 35 V
- Peak current: 2 Amp
- Operating current range: 0 ~ 36mA
- Control signal input voltage range:
- Low: -0.3V ≤ Vin ≤ 1.5V.
- High: 2.3V ≤ Vin ≤ Vss.
- Enable signal input voltage range:
  - Low: -0.3 ≤ Vin ≤ 1.5V (control signal is invalid).
  - High: 2.3V ≤ Vin ≤ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- Storage temperature: -25 ℃ ~ +130 ℃.
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino)
- Size: 3.4cm x 4.3cm x 2.7cm

7.6.2 Connection:



7.7 Battery:



We use a Lithium Polymer (LiPo) Battery which is preferred for its high energy density and lighter weight. The battery provides the necessary electrical power to all the system components, including the microcontroller (Arduino), motors, sensors (ultrasonic sensors), and any other electronic devices on board.

The battery's capacity determines how long the boat can operate before recharging. The placement and weight of the battery can also impact the boat’s balance and stability in water. Proper battery placement is important for maintaining an optimal center of gravity and ensuring smooth navigation.

7.7.1 Power Budget:

Power Supply ()			
Component	**Voltages**	**Currents**	**Power Consumed**
Arduino Mega	6 V	40 mA	0.24 W
ESP32	3.3 V	12 mA	0.0423 W
1 Ultrasonic Sensor	5 V	15 mA	0.075 W
GPS Module	2 V	10 mA	0.02 W
ESP32Camera	3.6 V	500 mA	1.8 W
Motor Driver	4.6 V	20 mA	0.092 W
2 Motors	7.4 V	4000 mA	29.6 W
TOTAL	7.4 V	4577 mA	33.86 W

Battery Life (hours) = Battery Capacity (mAh) / Total Power Consumption (mA)

Battery Life = 5200 mAh/4577 mA ≈ 1.136 hours = 68.17 minutes (on a single charge)

So, our battery gives us a total working time of 68.17 minutes without the charger and on once charge.

Then we calculate the maximum discharge current that we can take out from the battery follows:

$$Maximum\ Discharge\ Current\ =\ Capacity\ \times\ C\ Rating$$

$$Maximum\ Discharge\ Current\ =\ 5.2Ah\ \times\ 50\ =\ 260A$$

So, with a 50C rating and a capacity of 5200mAh (5.2Ah), our battery can be discharged at a maximum current of 260 amps.

8 Dynamic Positioning model and control

Three-DOF Dynamic Model: The availability of a sufficiently accurate USV model enabling effective control design is imperative for both control methodology design and simulation study purposes. This requires a prior investigation of both a precise mathematical USV model with reasonable system parameters. Generally, the study of a standard USV dynamic model can be divided into two parts: kinematics, which treats only geometrical aspects of motion, and kinetics, which is the analysis of the forces causing the motion.

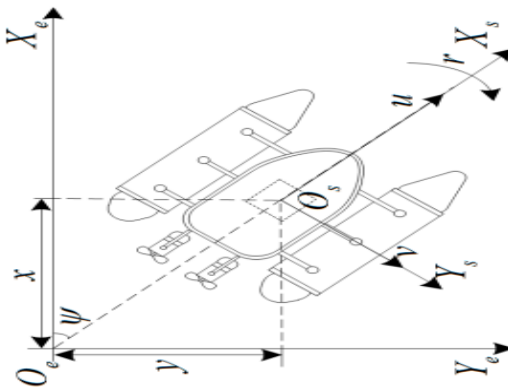
Within the field of kinematics, an ocean vessel moves in six DOFs, which are defined by: surge, sway, heave, roll, pitch, and yaw. However, the six-DOF model is complicated. Because some DOFs in the USV model have inherent stability, the model can be simplified. The buoyancy of the USV stabilizes the regular motion so that heave can be ignored. Because of the sufficient longitudinal and lateral metacentric height of the USV, the motion of roll and pitch can also be ignored. Therefore, the six-DOF model can be simplified to a three-DOF model to describe the planar motions of USV in surge, sway and yaw.

To determine the equations of motion, two reference coordinate systems are considered: the inertial or fixed to earth frame OXY e e e that may be taken to coincide with the USV fixed coordinates in some initial condition and the body-fixed frame OXY s s s. Since the motion of the earth hardly affects the USV (different from air vehicles), the earth-fixed frame OXY e e e can be considered to be inertial. The body axes O X s s and OYs s coincide with the principal axes of inertia and are usually defined as follows: O X s s is the longitudinal axis (directed from aft to fore); OYs s is the transverse axis (directed to starboard.

8.1 Kinetic:

In addition to the kinematic model, it is also necessary to analyze the force of the USV to set up the kinetic model. When designing the control system of the USV, there are clear advantages using the vectorial model instead of the component forms of the Taylor-series expansions in the Abkowitz model The main reasons are that system properties such as symmetry, skew-symmetry and positiveness of matrices can be incorporated into the stability

:n by



$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\upsilon}$$

$$\dot{\boldsymbol{\eta}} = [\ \dot{x} \quad \dot{y} \quad \dot{\psi}\ ]^T$$

$$\boldsymbol{\eta} = [\ x \quad y \quad \psi\ ]^T$$

$$\boldsymbol{\upsilon} = [\ u \quad v \quad r\ ]^T$$

$$\mathbf{J}(\boldsymbol{\eta}) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M\dot{\upsilon}+C(\upsilon)\upsilon+D(\upsilon)\upsilon=\tau+\tau_E}$$

$$\mathbf{M}=\begin{bmatrix} m-X_{\dot{u}} & 0 & -my_g \\ 0 & m-Y_{\dot{v}} & mx_g-Y_{\dot{r}} \\ -my_g & mx_g-N_{\dot{v}} & I_z-N_{\dot{r}} \end{bmatrix}$$

$$\mathbf{C}(\upsilon)=\begin{bmatrix} 0 & 0 & -m(x_g r+v)+Y_{\dot{v}}v+\frac{Y_{\dot{r}}+N_{\dot{v}}}{2}r \\ 0 & 0 & (m-X_{\dot{u}})u \\ m(x_g r+v)-Y_{\dot{v}}v-\frac{Y_{\dot{r}}+N_{\dot{v}}}{2}r & -(m-X_{\dot{u}})u & 0 \end{bmatrix}$$

$$\mathbf{D}(\upsilon)=\mathbf{D+D_n}(\upsilon)=-\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix}-\begin{bmatrix} X_{u|u|}|u| & 0 & 0 \\ 0 & Y_{v|v|}|v|+Y_{v|r|}|r| & Y_{r|v|}|v|+Y_{r|r|}|r| \\ 0 & N_{v|v|}|v|+N_{v|r|}|r| & N_{r|v|}|v|+N_{r|r|}|r| \end{bmatrix}$$

and an added mass matrix, which describes the hydrodynamic effects generated by the motion of the vehicle on the water surface, and it is denoted as

enter of

enter of gravity and assumes horizontal symmetry; Iz denotes the moment of inertia about the Zs -axis. The notation of SNAME for hydrodynamic derivatives is used in this expression. For instance, the hydrodynamic added mass force Y along the y-axis due to the acceleration .

where C(υ) is the Coriolis and centripetal matrix, D(υ) is the drag matrix, including the linear drag term (D) and the nonlinear drag term (Dn(υ)), τ is the vector of forces and moments generated by the propulsion system, and τE = h τuE τvE τrE iT is the vector of forces and moment caused by the disturbance is no force generated the rudder.

where i and j are unit vectors in the Xe and Ye directions. dP is the transverse distance from the centerline of the USV to the centerline of each thruster, which equals 0.26 m in the paper. lp is the longitudinal distance from the thruster to the center of gravity. Because the direction of the torque by the port and starboard thrusters is both in the Zs -axis, to simplify the calculation, the scalars of the torque are taken in the model. τr is the magnitude of Mr .

$$\tau=\begin{bmatrix} \tau_u & 0 & \tau_r \end{bmatrix}^T \qquad \mathbf{M_r}=\mathbf{X_{P1}}\times(l_p\mathbf{i}+d_p\mathbf{j})+\mathbf{X_{P2}}\times(-l_p\mathbf{i}-d_p\mathbf{j})$$

$$\tau_r=(X_{P1}-X_{P2})\cdot\sqrt{{l_p}^2+{d_p}^2}\frac{d_p}{\sqrt{{l_p}^2+{d_p}^2}}=(X_{P1}-X_{P2})\cdot d_P$$

$$\tau_u=X_{P1}+X_{P2}. \qquad \boldsymbol{\tau}=\begin{bmatrix} X_{P1}+X_{P2} & 0 & (X_{P1}-X_{P2})\cdot d_P \end{bmatrix}^T.$$

8.2 Propeller Thrust Model:

the thrust generated by the rudderless double thrusters can be decomposed into two forces: the longitudinal force to keep the USV moving forward and the steering moment to change the heading of the USV. In USVs with a combination of thrusters and rudders, the steering moment is the effect of the rudders. However, in t $X_P = (1-t_P)\rho D^4 K_T(J_0)n|n|$ a rudder. The differential thrust generated by double thr ılifies the mechanical structure of the USV, and makes the steering of the USV more flexible. Besides, it greatly improves the reliability and energy conversion rate of the USV. The propeller thrust model can be expressed as

8.3 Components Connections

where ρ is water density, D is the diameter of the propeller, and tP represents the coefficient of thrust reduction. When the propellers are working behind the USV, the water flow velocity at the stern is increased due to the suction effect, resulting in an increase in the hull pressure resistance and frictional resistance. The thrust portion used to offset this increased resistance ΔT is the thrust deduction tP = (XP − ΔT)/XP. The advance ratio J0 is given by

$$J_0 = V(1-\omega)/(nD)$$

- tP is the thrust deduction factor, which accounts for the reduction in effective thrust due to the presence of the hull and other appendages near the propeller.
- $\rho\rho$ is the density of the fluid (e.g., water).
- DD is the diameter of the propeller.
- $KTKT$ is the thrust coefficient of the propeller.
- $J0J0$ is the advance coefficient, defined as $J0=VnDJ0=nDV$, where VV is the fluid velocity and nn is the propeller speed in revolutions per unit time.
- nn is the propeller speed.
- $|n||n|$ is the absolute value of the propeller speed.

where i and j are unit vectors in the Xe and Ye directions. dP is the transverse distance from the centerline of the USV to the centerline of each thruster. lp is the longitudinal distance from the thruster to the center of gravity. Because the direction of the torque by the port and starboard thrusters is both in the Zs -axis, to simplify the calculation, the scalars of the torque are taken in the model. τr is the magnitude of Mr .

It can be seen that the propulsion of the USV is provided entirely by the resultant force generated by the port and starboard thrusters and that the rotational moment is related to the differential thrust and the distance of two thrusters. This cancels the function of the rudder to make

the calculation simpler. Modeling analysis of the propeller thrust is detailed in the next part. In order to simplify controller design, we make the following assumptions of the kinetic model: When the USV is slow (the maximum velocity of the USV is 1m/s.), the effect of nonlinear drag term (Dn(υ)) can be ignored. Because the effect of off-diagonal terms on the USV's dynamics is smaller than the diagonal terms, the off-diagonal terms of M and D can be ignored. Assuming the coincident center of added mass and gravity N. v can be replaced by Y. r . A combination of approximate fore-aft symmetry and light draft suggests that the sway force arising from yaw rotation and the yaw moment induced by the acceleration in the sway direction are much smaller than the inertial and added mass terms. Therefore, in the Coriolis and centripetal matrix C(υ), N. v = Y. r = 0. Assuming the USV sails in a calm environment, the environmental disturbances τE can be neglected.

Based on these assumptions, the kinetic model can be simplified to:

$$\begin{cases} \dot{x} = u\cos\psi - v\sin\psi \\ \dot{y} = u\sin\psi + v\cos\psi \\ \dot{\psi} = r \\ (m - X_{\dot{u}})\dot{u} - (m - Y_{\dot{v}})vr + X_u u = X_{P1} + X_{P2} \\ (m - Y_{\dot{v}})\dot{v} + (m - X_{\dot{u}})ur + Y_v v = 0 \\ (I_z - N_{\dot{r}})\dot{r} - ((m - X_{\dot{u}}) - (m - Y_{\dot{v}}))uv + N_r r = (X_{P1} - X_{P2}) \cdot d_p \end{cases}$$

9 Motion Steps

In straight line motion v = r = 0, V = u, and the three-DOF dynamic model can be simplified to a one-DOF model

In straight line motion v = r = 0, V = u, and the three-DOF dynamic model can be simplified to a one-DOF model . According to the propeller thrust model, the one-DOF model is given by:

$$(m - X_{\dot{u}})\dot{u} - (m - Y_{\dot{v}})vr + X_u u = X_{P1} + X_{P2}$$

When applying the same motor commands to the port and starboard thrusters, the thrusts of them are the same. Because the performance of port and starboard thrusters is the same, in straight line motion, the thrust.

$$X_{P1} = X_{P2}, X_{P1} + X_{P2} = 2X_P$$

Then, Equation (can be further expressed as:

$$\dot{u} = \frac{X_u}{(m - X_{\dot{u}})}u + \frac{2X_P}{(m - X_{\dot{u}})}$$

When the USV is in the steady state with constant velocity u_0,

$$0 = \frac{X_u}{(m - X_{\dot{u}})}u_0 + \frac{2X_{P0}}{(m - X_{\dot{u}})}$$

substituting perturbation values $u = u_0 + \Delta u$, $X_{P1} = X_{P0} + \Delta X_P$ into Equation

$$\Delta\dot{u} = \frac{X_u}{(m - X_{\dot{u}})}(u_0 + \Delta u) + \frac{2(X_{P0} + \Delta X_P)}{(m - X_{\dot{u}})}$$

According to Equation (23), in the initial limit,

$$\dot{u}(0) = 2\frac{\Delta X_p}{(m - X_{\dot{u}})} = 2\frac{c\Delta u\Delta n + d|\Delta n|\Delta n}{(m - X_{\dot{u}})}$$

When the USV is stationary, i.e., $u_0 = 0$, (25) can be simplified to

$$\dot{u}(0) = \widetilde{c}un + \widetilde{d}|n|n$$

go gain C,d,Xu it is needed to repeat

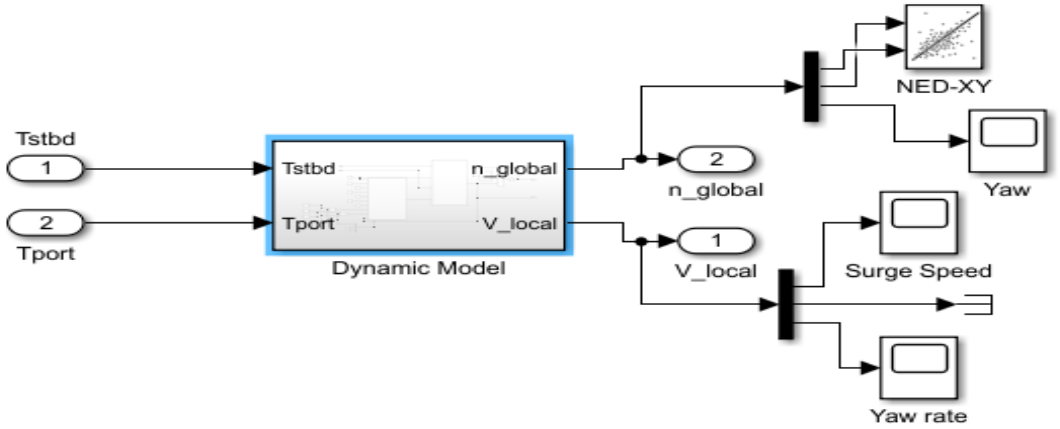
the straight line experiments many

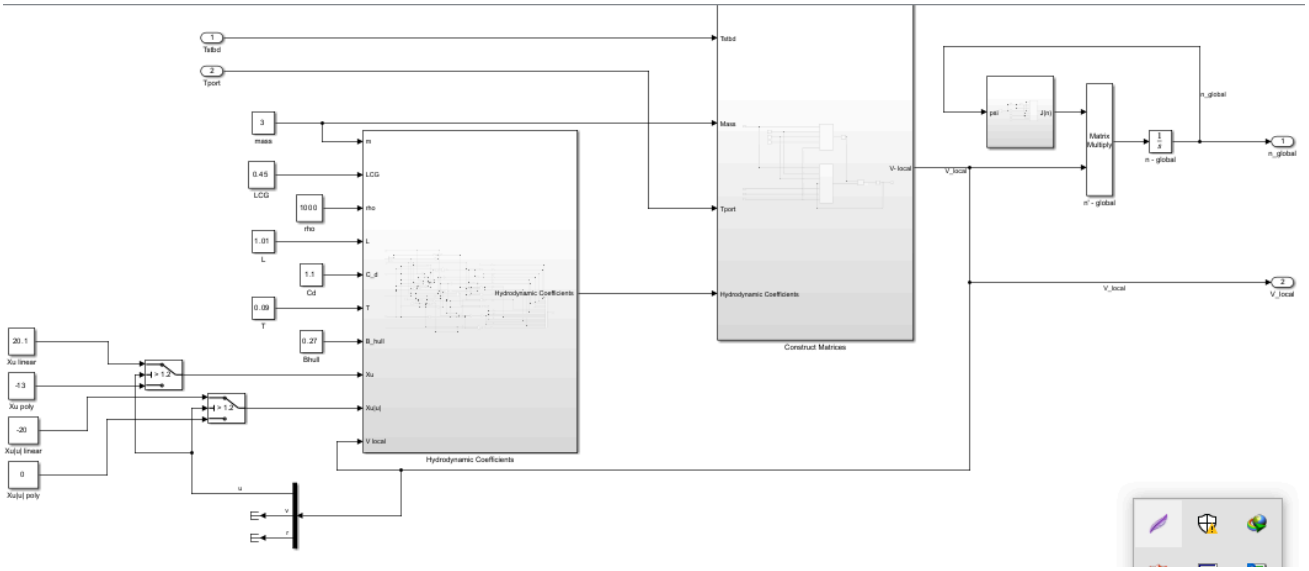
times till

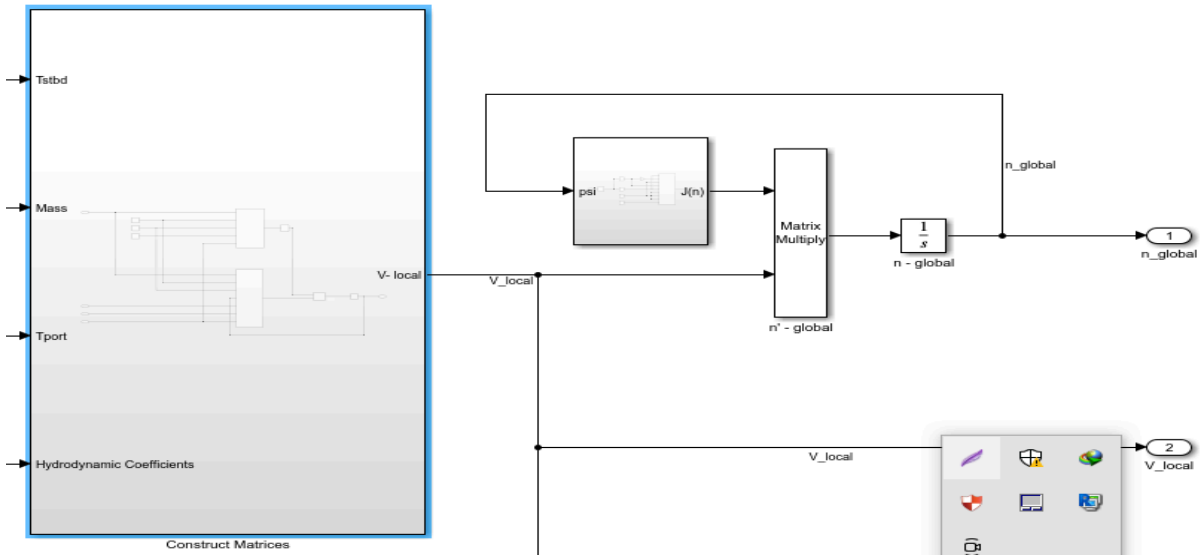
getting the means of these values.

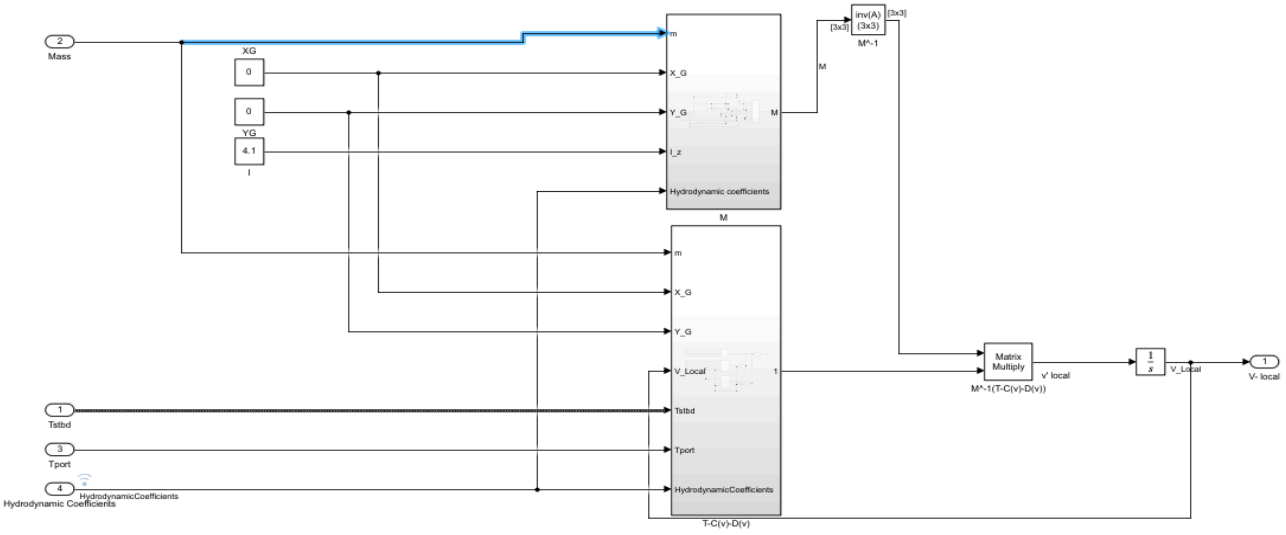
To gain Nr,Yv, we have to move the usv in a circular motion

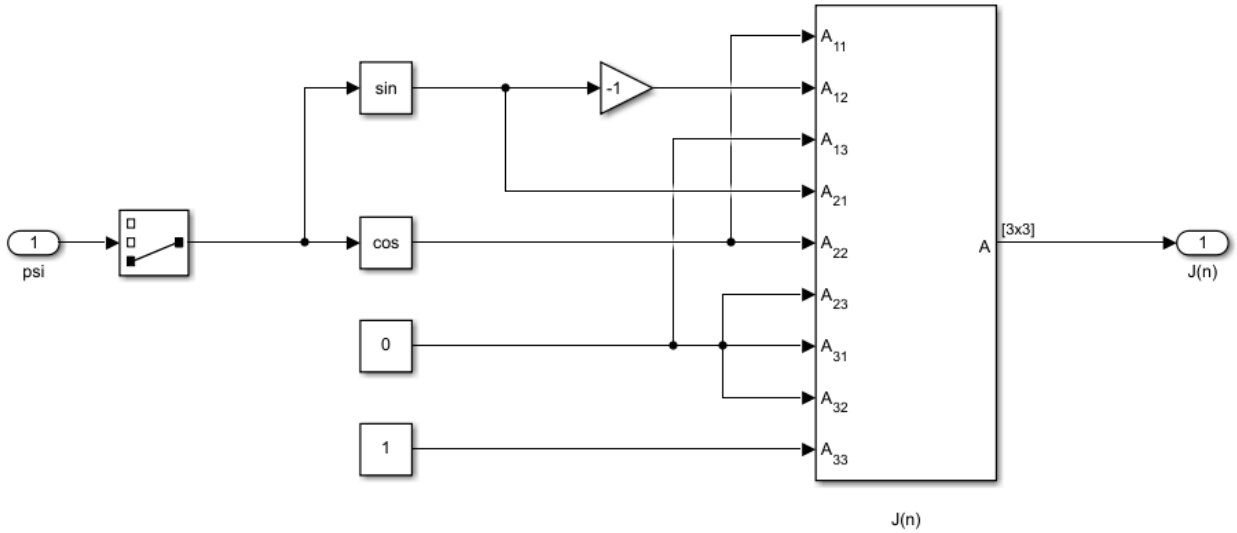
10 MATLAB MODEL OF USV

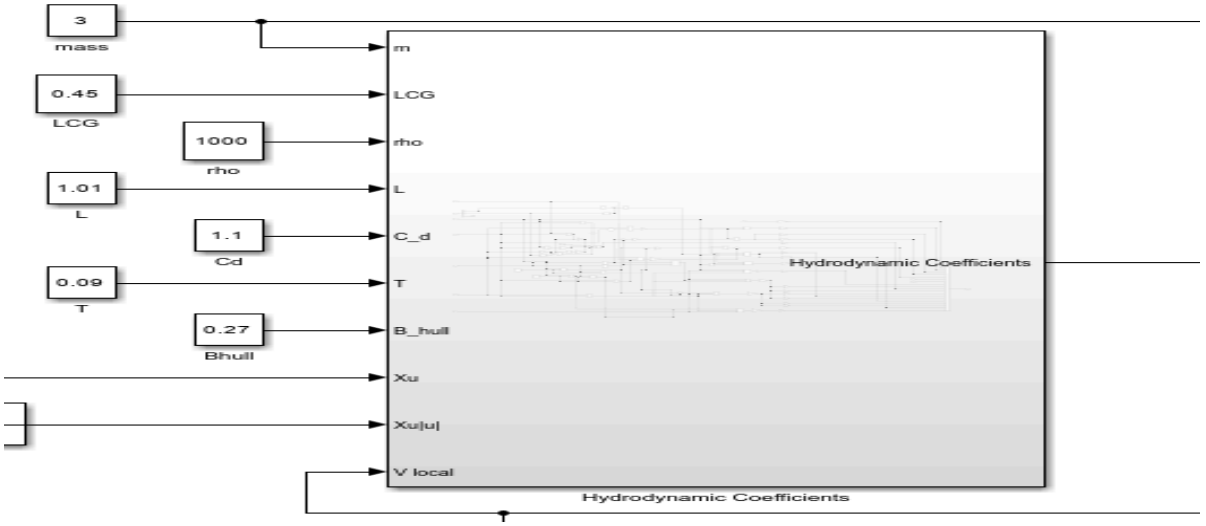





Tstbd
Mass
Tport
Hydrodynamic Coefficients
V- local
Construct Matrices
psi
J(n)
Matrix Multiply
n' - global
n - global
n_global
V_local
1
n_global
2
V_local

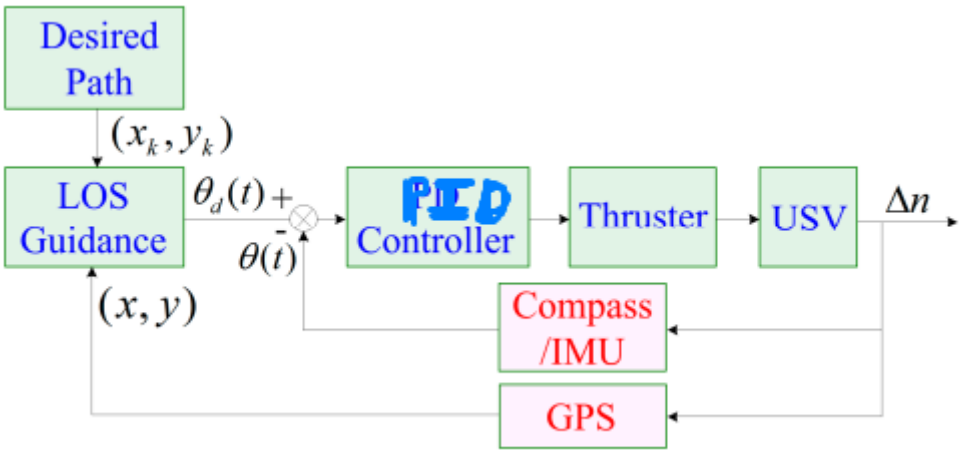

2
Mass
XG
0
0
YG
4.1
I
m
X_G
Y_G
I_z
Hydrodynamic coefficients
M
inv(A)
M^-1
V_Local
Matrix Multiply
M^-1(T-C(v)-D(v))
v' local
V- local
1
Tstbd
3
Tport
4
HydrodynamicCoefficients
Hydrodynamic Coefficients
T-C(v)-D(v)

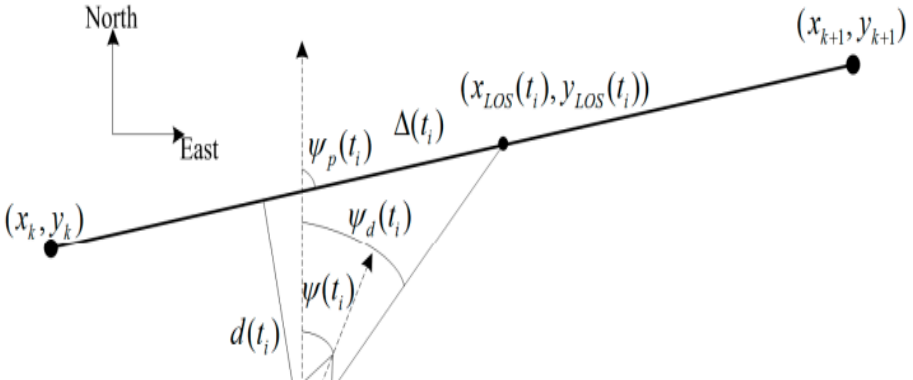

1
psi
sin
cos
0
1
-1
A11
A12
A13
A21
A22
A23
A31
A32
A33
A
[3x3]
1
J(n)
J(n)


3
mass
0.45
LCG
1000
rho
1.01
L
1.1
Cd
0.09
T
0.27
Bhull
m
LCG
rho
L
C_d
T
B_hull
Xu
Xu|u|
V local
Hydrodynamic Coefficients
Hydrodynamic Coefficients

11 PID+LOS Control Algorithm:

For the USV motion control system, the design of the path-following controller is an important part. Due to the simple structure and functional stability of PID control, it is currently used more in autonomous navigation of the ship. Based on the established USV model, the USV motion control system through PID +LOS guidance is designed. The advantages of the LOS algorithm lie in that it is light in computation and easy for implementation. The main principle of the LOS guidance algorithm is to imitate the behavior of a helmsman, which steers the vehicle towards a lookahead distance ahead of the projection point of the vehicle along the path. The USV assumes a tracking target on the tracking path and then sails along the line of the USV and the hypothetical tracking target. As the USV approaches the desired path, the heading deviation is slowly reduced, and the desired path can be accurately tracked. (xk , yk) and (xk+1 , yk+1) are two adjacent desired waypoints, and the line connecting them is the desired path. (x, y) is the actual position of USV measured by GPS. ψ(ti) is the actual heading measured by inertial measurement units (IMU) or compass. (xLOS(ti), yLOS(ti)) is the hypothetical tracking target. The line connecting (xLOS(ti), yLOS(ti)) and the USV is the "line of sight". The angle between the "line of sight" and the north direction is the desired heading of the USV, ψd (ti), given by ψd (ti) = ψp(ti) + ψr(ti) (36) where ψp(ti) is the path-tangential angle. ψr(ti) is the velocity-path relative angle, ψr(ti) = arctan(−d(ti)/Δ(ti)) It ensures that the velocity is directed toward a point on the path that is located a lookahead distance Δ(ti) > 0 ahead of the direct projection of (xk , yk) on to the. The lookahead distance Δ(ti) is dependent on the cross-track error. This results in lower values for Δ(ti) (and thus a more agile and aggressive response) when the USV is far from the desired path, and greater Δ(ti) values when the USV is closer to the path and less abrupt behavior is needed so as to avoid oscillating around the path. The selection of Δ(ti) refers to.





GPS NEO6M Module:

- VCC to 5V.

- GND to GND.

- Tx to Arduino Mega Rx (pin 17).

- Rx to Arduino Mega Tx (pin 16).

The root mean square error (RMSE) of the distances between the desired waypoints and the actual positions of the USV is adopted to evaluate the performance of the controller, given by RMSE = vuut Pn i=1 ((xi − xdi) 2 + (yi − ydi) 2) n (42) where (xdi, ydi) is the position of the i desired waypoint. n is the number of the desired points. (xi , yi) is the i actual position of the USV

$$d_{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} ((x_i - x_{di})^2 + (y_i - y_{di})^2)}{n}}$$

12 CONTROL AND ALGORITHM:

First we include the necessary libraries for the USV systems: GPS,IMU,PID,Motors,Kalman filter…

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <SimpleKalmanFilter.h>
#include <math.h>
#include <PID_v1.h>
```

1- Then we initialize the the constants for your systems of Different components:

```
//Motors
// right 3,4 enB
#define ENB 2
#define IN3 4
#define IN4 3
//left 1,2 enA
#define ENA 7
#define IN1 6
#define IN2 5


//components connection pins
define scl 21
define Sda 20
#define RxGps 17
#define TxGps 16
#define RxUS2 21
#define TxUS2 20
#define RxUS1 0
#define TxUS1 1
#define Rxcam 19
#define Txcam 18
#define Rxesp32 15
#define Txesp32 14

// The serial connection to different dvices
SoftwareSerial GpsSerial(RxGps, TxGps);
SoftwareSerial USC1erial(RxUS1, TxUS1);
SoftwareSerial USC2erial(RxUS2, TxUS2);
SoftwareSerial EspSerial(Rxesp32, Txesp32);
SoftwareSerial CamSerial(Rxcam, Txcam);

//GPSValues
```

```
float originLon = 31.17490267;
float originLat=  27.18799450 ;
float earthRadius = 6.3781*pow(10,6);

//KalamFilter factors
const float e_mea = 0.1;//Measurement Uncertainty
const float e_est =  0.4;//Estimation Uncertainty
const float q = 0.01;//Process Noise

//Kalman filter accelerationx,y,z,rotation x,rotation y,longititude,latitude to get rid of noise
SimpleKalmanFilter kalmanFilterX(e_mea, e_est, q);
SimpleKalmanFilter kalmanFilterY(e_mea, e_est, q);
SimpleKalmanFilter kalmanFilterZ(e_mea, e_est, q);
SimpleKalmanFilter kalmanFilterAngleX(e_mea, e_est, q);
SimpleKalmanFilter kalmanFilterAngleY(e_mea, e_est,q);
SimpleKalmanFilter kalmanFilterAngleZ(e_mea, e_est, q);
SimpleKalmanFilter kalmanFilterlng(e_mea, e_est,q);
SimpleKalmanFilter kalmanFilterlat(e_mea, e_est, q);

// received values from the IMU
unsigned long gpsTimestamp = 0;
float accelX, accelY, accelZ;
float gyroX, gyroY, gyroZ;
float accAngleX, accAngleY;
float gyroAngleX, gyroAngleY;
float angleX, angleY;

// Variables to store GPS data
double gpsLat, gpsLng;
float lngKalman,latKalman,Xkalman,Ykalman,XRotationkalman,YRotationkalman;

//ultraSonic
int duration;
int distance;
int maxCurrent = 255;

//PID  setpoints,input,output,factors, MakOutput=255=maxPWM
double Setpoint, Input, Output;
double kp = 2, ki = 2, kd = 0.2;
PID myPID(&Input, &Output, &Setpoint, kp, ki, kd, DIRECT);


int IRSensor = 34; // connect IR sensor module to Arduino pin 26
int LED = 13; // connect LED to Arduino pin 13

//Encountrering Object distance to avoid
float distance1 =0;
float distance2 =0;
```

3 –

We Detect if there was an signal coming from the comminucation:

We receive primarly the GPS Longtitude,Latitude,Lpeed,Time number of of satellites

```
while (GpsSerial.available() > 0){
  gps.encode(GpsSerial.read());
  if (gps.location.isUpdated()){
    // Longitude and Latitude in degrees (double)
    gpsLng = gps.location.lng();
    gpsLat = gps.location.lat();
    gpsTimestamp = millis();


     Speed in meters pr second (double)
    Serial.print("Speed in m/s = ");
     Serial.println(gps.speed.mps());

     Number of satellites in use (u32)
    Serial.print("Number os satellites in use = ");
     Serial.println(gps.satellites.value());
   }
 }
```

We need to converting GPS coordinates (latitude and longitude) to XY coordinates involves transforming geographic coordinates into a planar coordinate system.

To convert from GPS degree coordinates to XY coordinates: there are many methods to do it like UTM (Universal Transverse Mercator), Mercator projection, but we would need to use The equirectangular projection.

equirectangular projection (also known as the plate carrée projection) is a simple map projection where the horizontal and vertical coordinates are directly proportional to longitude and latitude, respectively. This projection is straightforward to implement on resource-constrained devices like Arduino, but it's only suitable for small areas due to distortions over large distances.

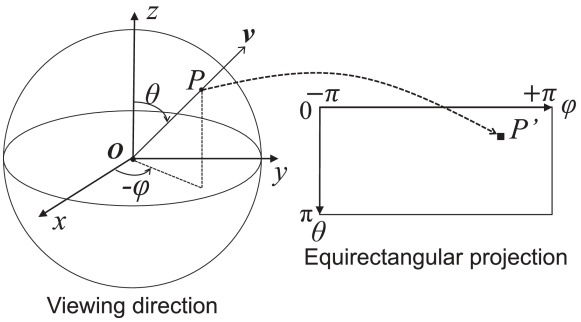
- **OriginLat** and **OriginLon**: The reference point for the projection. These can be set to the center of your area of interest.
- **EarthRadius**: The radius of the Earth in meters.
- **Longitude:** Longitude of the USV
- **Latitude:** Latitude of the USV

- equirectangular projection formula converts latitude and longitude to X, Y coordinates:

x=(longitude−originLon)×(π180)×earthRadius×cos(latitude×(π180))x=(longitude−originLon)×(180π)×earthRadius×cos(latitude×(180π))

y=(latitude−originLat)×(π180)×earthRadius$y$=(latitude−originLat)×(180π)×earthRadius

- These formulas convert degrees to radians and account for the Earth's curvature.



We create two functions to convert the received GPS coordinates to XY coordinates, so easy to manipulate and convcieve the the 3D world around us.

```
int convertGPSToX(float longitude,float latitude){
  float x = (longitude - originLon) * (PI / 180) * earthRadius * cos(latitude * PI / 180);
   return x;
}
int convertGPSToY(float longitude,float latitude){
   float y = (latitude - originLat) * (PI / 180) * earthRadius;
   return y;
}
```

4 –we initialize the motors to reach the the half of motor maxCurrent:

```
//MotorsInizialization
void motorInt(){
 for(int i=0;i<maxCurrent/2;i++){
      analogWrite(ENA,i);
      analogWrite(ENB, i);
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      delay(20);
  }
}
```

5- We initialize the the reference point, so all other coordinates by in comparison to this point

```
2-  //GPSValues
3-  float originLon = 31.17490267;
4-  float originLat=  27.18799450 ;
```

```
5-  float earthRadius = 6.3781*pow(10,6);
```

*desired point to go to :

```
// Global Variables
float OriginalTarggetedLng = 31.17505422;
float OriginalTarggetedLat = 27.18816046;
```

*the current position of the vechile:

```
float currentPointX;
float currentPointY;
```

*we convert all these axes to the XY Geometry coordinates:

```
// Convert GPS coordinates to X and Y points
  TargettedPointX = convertGPSToX(PrimeTarggetedLng, PrimerTarggetedLat);
  TargettedPointY = convertGPSToY(PrimeTarggetedLng, PrimerTarggetedLat);
  currentPointX = convertGPSToX(lngKalman, latKalman);
  currentPointY = convertGPSToY(lngKalman, latKalman);
```

12.1IMU

It is used to detect the Acceleration(X,y,Z,rX,rY,rZ) of the USV:

```
sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  //IMU linear acceleration
  accelX = a.acceleration.x, accelY=a.acceleration.y, accelZ=a.acceleration.z;
  Serial.println(a.acceleration.x);
  Serial.println(a.acceleration.y);
  Serial.println(a.acceleration.z);

  //IMU rotational acceleration in Radians
  Serial.print(g.gyro.x);
  Serial.print(g.gyro.y);
  Serial.print(g.gyro.z);
```

We need to change the values received from Radian to degree to fit your system model dynamic.

```
//IMU rotational acceleration in Degrees
  accAngleX = atan(g.gyro.y / sqrt(pow(g.gyro.x, 2) + pow(g.gyro.z, 2))) * 180 / PI;
  accAngleY = atan(-1 * g.gyro.x / sqrt(pow(g.gyro.y, 2) + pow(g.gyro.z, 2))) * 180 / PI;
  float dt = (millis() - gpsTimestamp) / 1000.0; // time difference in seconds
```

Updates every some millis of time of the rotation of usv change of motion happen to accumulate the sum of vusv deviation of the heading angle.

```
// Update angle using gyro acclerations
 gyroAngleX += gyroX * dt;
 gyroAngleY += gyroY * dt;
```

12.2Kalman Filter and Sensor Fusion

The Kalman filter is an algorithm that provides estimates of some unknown variables given the measurements observed over time. It's widely used in navigation, control systems, and signal processing due to its efficiency in handling noisy data and its ability to fuse multiple sensor inputs. Here's a detailed explanation:

Core Concepts of the Kalman Filter

1. **State Vector**:
    - Represents the system's variables of interest. For example, in navigation, this might include position, velocity, and orientation.
    - Denoted as $xkxk$ at time step kk.
2. **State Transition Model**:
    - Describes how the state evolves over time.
    - Typically linear and represented as:

$xk=Fkxk−1+Bkuk+wkxk=Fkxk−1+Bkuk+wk$

where Fk**F**k is the state transition matrix, Bk**B**k is the control input matrix, uk**u**k is the control vector, and wk**w**k is the process noise (assumed to be Gaussian with zero mean and covariance Qk**Q**k).

3. **Measurement Model**:
    - Relates the state vector to the measurement vector.
    - Typically linear and represented as:

$zk=Hkxk+vkzk=Hkxk+vk$

where Hk**H**k is the observation matrix and vk**v**k is the measurement noise (assumed to be Gaussian with zero mean and covariance Rk**R**k).

4. **Process Noise and Measurement Noise**:
    - wk**w**k: Process noise covariance Qk**Q**k.
    - vk**v**k: Measurement noise covariance Rk**R**k.

Kalman Filter Algorithm Steps

1. Prediction Step

- **State Prediction**:

  $xk|k-1=Fkxk-1|k-1+Bkuk\mathbf{x}k|k-1=\mathbf{F}k\mathbf{x}k-1|k-1+\mathbf{B}k\mathbf{u}k$

- **Error Covariance Prediction**:

  $Pk|k-1=FkPk-1|k-1FkT+Qk\mathbf{P}k|k-1=\mathbf{F}k\mathbf{P}k-1|k-1\mathbf{F}kT+\mathbf{Q}k$

2. Update Step

- **Measurement Residual** (Innovation):

  $yk=zk-Hkxk|k-1\mathbf{y}k=\mathbf{z}k-\mathbf{H}k\mathbf{x}k|k-1$

- **Innovation Covariance**:

  $Sk=HkPk|k-1HkT+Rk\mathbf{S}k=\mathbf{H}k\mathbf{P}k|k-1\mathbf{H}kT+\mathbf{R}k$

- **Kalman Gain**:

  $Kk=Pk|k-1HkTSk-1\mathbf{K}k=\mathbf{P}k|k-1\mathbf{H}kT\mathbf{S}k-1$

- **State Update**:

  $xk|k=xk|k-1+Kkyk\mathbf{x}k|k=\mathbf{x}k|k-1+\mathbf{K}k\mathbf{y}k$

- **Error Covariance Update**:

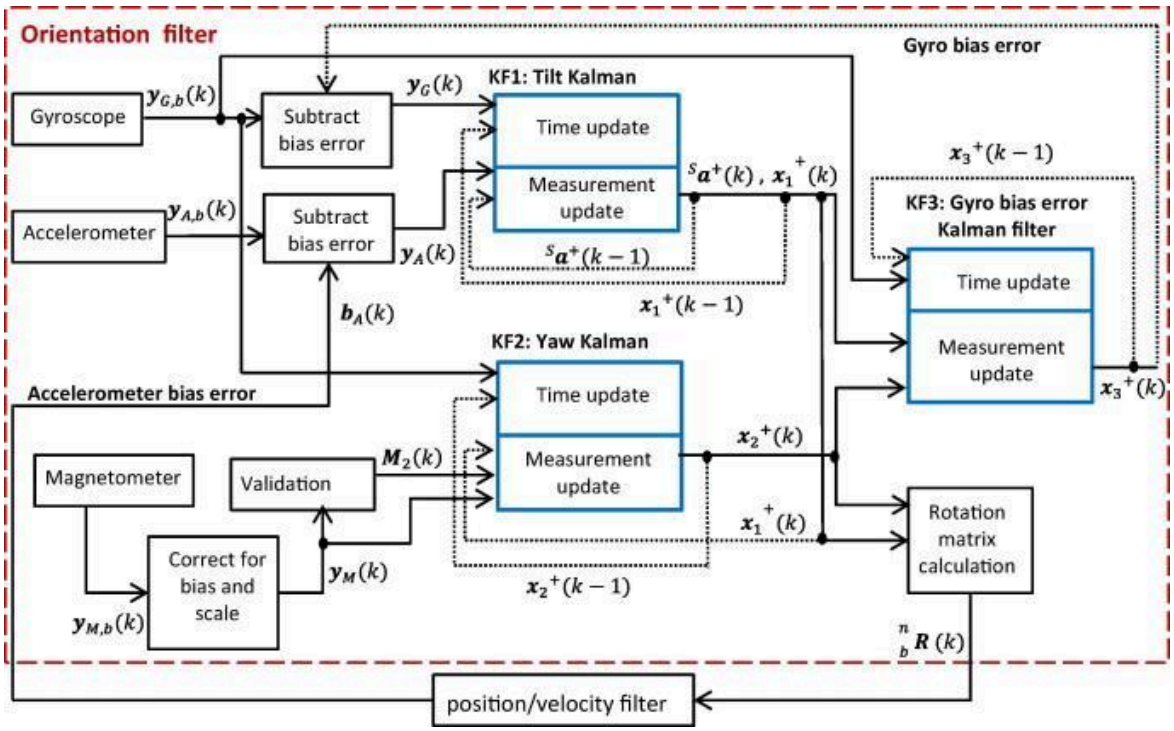
  $Pk|k=(I-KkHk)Pk|k-1\mathbf{P}k|k=(\mathbf{I}-\mathbf{K}k\mathbf{H}k)\mathbf{P}k|k-1$

Intuition Behind the Kalman Filter

- **Prediction Step**: Based on the previous state and known control inputs, predict the current state. This step also estimates the uncertainty in the prediction.
- **Update Step**: Adjust the predicted state using new measurements. The Kalman Gain determines the weight given to the new measurements relative to the prediction, based on their respective uncertainties.

Key Benefits

1. **Optimality**: For linear systems with Gaussian noise, the Kalman filter provides the optimal estimate.
2. **Real-Time Processing**: It processes data as it comes in, making it suitable for real-time applications.
3. **Sensor Fusion**: Effectively combines information from multiple sensors to produce a more accurate estimate than any single sensor alone.



We can find these functions and operations being computed by the(simpleKalmanFilter)Library in Arduino:

We initialize the prime factors of Kalman factor of Measurement,Estimation,Noise.

```
//KalamFilter factors
const float e_mea = 0.2;//Measurement Uncertainty
const float e_est =  0.4;//Estimation Uncertainty
const float q = 0.01;//Process Noise
```

we update and filter the Angle of rotation and Deviation, and fuse the The Gps values alongside the imu acceleration values to gain the possible filtered smooth values

```
// Correct the drift with accelerometer angle
angleX = 0.98 * gyroAngleX + 0.02 * accAngleX;
angleY = 0.98 * gyroAngleY + 0.02 * accAngleY;

// Apply Kalman Filter to the angles
float Xkalman = kalmanFilterX.updateEstimate(accelX);
float Ykalman = kalmanFilterY.updateEstimate(accelY);

// Apply Kalman Filter to the angles
float XRotationkalman = kalmanFilterAngleX.updateEstimate(angleX);
float YRotationkalman = kalmanFilterAngleY.updateEstimate(angleY);

 // Fuse GPS and IMU data with Kalman filter
kalmanFilterIng.setProcessNoise(Xkalman);
```

```
  kalmanFilterlat.setProcessNoise(Xkalman);

   // Retrieve fused position estimates
   float lngKalman = kalmanFilterlng.updateEstimate(gpsLng);
   float latKalman = kalmanFilterlat.updateEstimate(gpsLat);
   // updated data: lngKalman,latKalman,Xkalman,Ykalman,XRotationkalman,YRotationkalman
   Serial.print("longtitude: ");Serial.println(lngKalman);
   Serial.print("Latitude: ");Serial.println(latKalman);
   Serial.print("X Acceleration: ");Serial.println(Xkalman);
   Serial.print("Y Acceleration: ");Serial.println(Ykalman);
   Serial.print("X rotation: ");Serial.println(XRotationkalman);
   Serial.print("Y rotation: ");Serial.println(YRotationkalman);
   Serial.print("Output ");Serial.println(Output);
```

12.3Motion

Initilization of USV motors gradually to half of power because of avoiding instant full load:

```
//MotorsInizialization
void motorInt(){
 for(int i=0;i<maxCurrent/2;i++){
      analogWrite(ENA,i);
      analogWrite(ENB, i);
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      delay(20);
  }
}
```

13 PID Controlled Guidance Algorithm:

Inistializating Kp,Ki,Kd

- **Increaseing Kp** to improve the response speed, but watch for increased oscillations.
- **Increasing Ki** to eliminate steady-state error, but be cautious of potential instability.
- **Increaseing Kd** to reduce overshoot and oscillations, but avoid amplifying noise.

To gain Kp,ki,kd, we follow the best strategy to follow Automatic Tuning (Ziegler-Nichols Method):

- Increase Kp until you observe a stable oscillation.
- Note the critical gain ($KuKu$) and oscillation period ($TuTu$).
- Calculate PID parameters:

Kp=0.6. Ku

Ki=2·Kp/Tu

Kd=Kp·Tu/8

Instialization of KP,Ki,Kd:

```
double kp = 2, ki = 2, kd = 0.2;
```

we calculate Heading and targeting angle:

```
double calculateLOSHeading(double currLat, double currLng, double targLat, double targLng) {
  double deltaY = targLat - currLat;
  double deltaX = targLng - currLng;
  double angle = atan2(deltaX, deltaY);
  return degrees(angle);
}
```

```
gpsImu(); // Assume this function updates lngKalman and latKalman
 float TargettedPointX;
 float TargettedPointY;

 // Convert GPS coordinates to X and Y points
 TargettedPointX = convertGPSToX(PrimeTarggetedLng, PrimerTarggetedLat);
 TargettedPointY = convertGPSToY(PrimeTarggetedLng, PrimerTarggetedLat);
 currentPointX = convertGPSToX(lngKalman, latKalman);
 currentPointY = convertGPSToY(lngKalman, latKalman);
```

if the vehicle faces an obstacle, it rotates 1.5m meter horizinatally and vertically, and when It reaches this point it rotates towards the desired point:

```
bool obstacleDetection(){

  digitalWrite(TxUS1, LOW);
  delayMicroseconds(2);

  digitalWrite(TxUS1,HIGH);
  delayMicroseconds(10);
  digitalWrite(TxUS1, LOW);

  duration = pulseIn(RxUS1, HIGH);
  distance = duration * 0.0344 / 2;
  distance1 = distance;
 if(distance<20){
  return true;
 }
 else
  return false;
}
```

```
if (obstacleDetection()) {
  obsTarggetedLng = currentPointX - 1.5;
  obsTarggetedLat = currentPointY + 1.5;
  cond = true;
 }

 if (cond) {
  PrimeTarggetedLng = obsTarggetedLng;
  PrimerTarggetedLat = obsTarggetedLat;
   if (abs(PrimeTarggetedLng - TargettedPointX) < 0.3 && abs(PrimerTarggetedLat - TargettedPointY) <
0.3) {
    cond = false;
  }
 } else {
  PrimeTarggetedLng = OriginalTarggetedLng;
  PrimerTarggetedLat = OriginalTarggetedLat;
 }
```

Transfer points to XY coordinates:

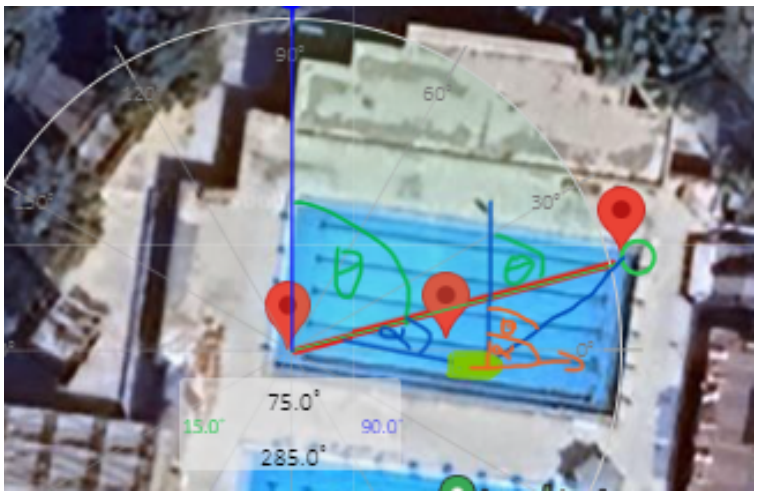
```
/ Recalculate target and current points
 TargettedPointX = convertGPSToX(PrimeTarggetedLng, PrimerTarggetedLat);
 TargettedPointY = convertGPSToY(PrimeTarggetedLng, PrimerTarggetedLat);
 currentPointX = convertGPSToX(lngKalman, latKalman);
 currentPointY = convertGPSToY(lngKalman, latKalman);
```

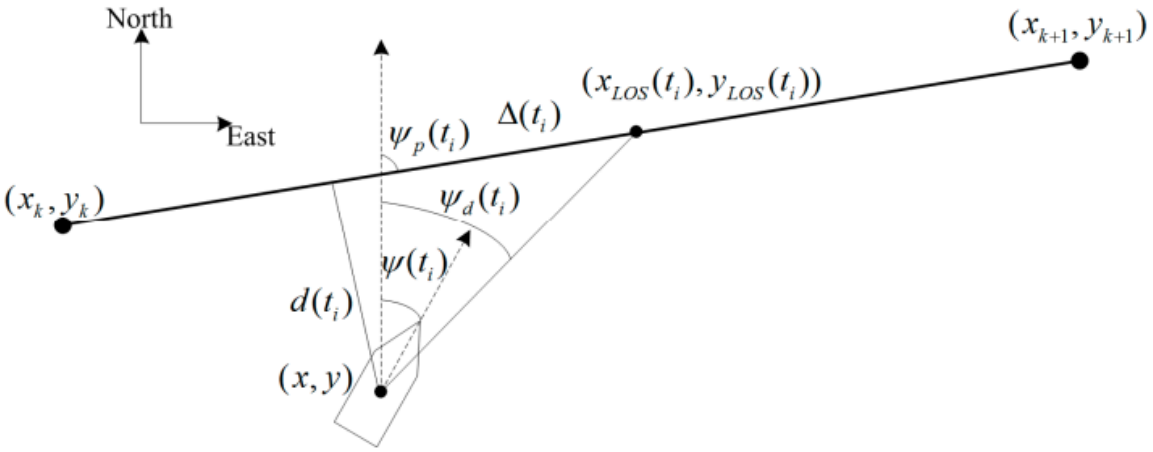
```
        // Calculate target heading
            float targetHeading = calculateLOSHeading(TargettedPointY, TargettedPointX, currentPointY, currentPointX);
          // Update PID controller
          float currentHeading = XRotationkalman;
          Input = currentHeading;
          Setpoint = targetHeading;
          myPID.Compute();
```

The motors change its direction according to delta= heading degree-desired degree based on pid control of motors speed

```
        void setMotorSpeed(double output) {
          Serial.println(output);
            if (output > 0) {
              // Turn right
              //analogWrite(ENB, maxCurrent - output);
              analogWrite(ENB, maxCurrent - output);
              analogWrite(ENA, maxCurrent);
              digitalWrite(IN1, HIGH);
              digitalWrite(IN2, LOW);
              digitalWrite(IN3, HIGH);
              digitalWrite(IN4, LOW);
            } else {
              // Turn left
              analogWrite(ENB, maxCurrent);
              analogWrite(ENA, maxCurrent + output);
              //analogWrite(ENA, maxCurrent + output);
              digitalWrite(IN1, HIGH);
              digitalWrite(IN2, LOW);
              digitalWrite(IN3, HIGH);
              digitalWrite(IN4, LOW);
            }
            delay(10);
        }
```

Sketch of motion planning Algorithm controlling motors thrusters speeds:





When it reaches the desired position it detects if there was a fire to rescue or not.

```
if(firDetection()){
   Serial.println("fire");
 }
```

14 Sustainability and Carbon footprint

As we care for our planet, we were careful that our project does not harm our planet, we tried to make an ecofriendly design from a recycled plastic that helped us save a sufficient amount of CO2 emission as follows:

Total Carbon Footprint= Weight x CO2e per kg

PVC Pipes = 2.5 kg CO2e/kg=2.5 kg CO2e

PMMA Plastic Plate = 50 g CO2e/kg=4.5 kg CO2e

LDPE Plastic Box = 50 g CO2e/kg=2.0 kg CO2e

Total Carbon Footprint = 2.5x2.5+4.5x0.05+2x0.05=6.575 kg CO2e

Therefore, using a recycled plastic, enabled us to save total of 6.575 kg CO2e and preventing them from increasing the greenhouse gas emissions and participate in saving our Earth.

15 CONCLUSION

In conclusion, the development and implementation of the autonomous rescue boat system demonstrate a significant advancement in water-based rescue operations, integrating a range of cutting-edge technologies to achieve precise navigation, obstacle avoidance, and thermal signal detection. The system, controlled by an Arduino Mega microcontroller, successfully integrates GPS, IMU, motors, motor driver, ultrasonic sensor, and communication modules on a custom-designed PCB, ensuring reliable operation and ease of maintenance and troubleshooting. However, testing revealed performance variability due to differing and dynamic conditions encountered by the sensors. GPS accuracy fluctuated in dense environments, and thermal signal detection by the ESP32CAM was sometimes affected by environmental factors. Despite these challenges, the system overall functioned as intended but separately, navigating to predefined locations, detecting thermal signals, and returning to safe zones. The custom PCB design contributed to the system's robustness, reducing wiring complexities and enhancing stability. Future iterations could focus on improving sensor robustness and incorporating advanced data filtering techniques to mitigate environmental effects. Thus, the autonomous rescue boat system shows significant potential for reliable and efficient rescue missions, with ongoing refinement needed to address sensor-related challenges.

16 REFERENCES

- Chunyue Li , Jiajia Jiang,, Fajie Duan, Wei Liu , Xianquan Wang, Lingran Bu,Zhongbo Sun and Guoliang Yang.(2 May 2019), Modeling and Experimental Testing of an Unmanned Surface Vehicle with Rudderless Double Thrusters.
- Carlos Gutiérrez, Bashir Yacub Enrique Sierra, Shamir Sánchez and Juan Contreras.(January 30th, 2022), Speed and Heading Control System for an Unmanned Surface Vehicle - USV.
- Lifei Song, Chuanyi Xu, Le Hao, Jianxi Yao, and Rong Guo.(24 October 2022), Research on PID Parameter Tuning and Optimization Based on SAC-Auto for USV Path Following,jornal of marine and science.
- Sławomir ROMANIUK, Zdzisław GOSIEWSKI.(17 july 2017), KALMAN FILTER REALIZATION FOR ORIENTATION AND POSITION ESTIMATION ON DEDICATED PROCESSOR.
- Alejandro Gonzalez-Garcia and Herman Castañeda.(15 june 2021),Guidance and Control Based on Adaptive Sliding.
- Fossen, T. I. (2011). Handbook of Marine Craft Hydrodynamics and Motion Control. John Wiley & Sons. - Comprehensive resource on marine vehicle dynamics and control.
- Breivik, M., & Fossen, T. I. (2008). "Guidance laws for autonomous underwater vehicles: A survey." IFAC Conference ₹on Manoeuvring and Control of Marine Craft (MCMC). - Discusses various guidance laws for marine vehicles including USVs.
- Pettersen, K. Y., & Lefeber, E. (2001). "Way-point tracking control of ships." IFAC Proceedings Volumes, 34(4), 127-132. - Focuses on waypoint tracking for marine vessels using control techniques.
- Fossen, T. I., & Breivik, M. (2006). "Line-of-sight path following of underactuated marine craft." IFAC Conference on Control Applications in Marine Systems. - Key paper on LOS guidance principles for marine vehicles.
- Breivik, M., & Fossen, T. I. (2009). "Adaptive line-of-sight guidance for marine craft." IFAC Proceedings Volumes, 42(1), 1585-1590. - Adaptive LOS guidance strategies.
- Åström, K. J., & Hägglund, T. (2006). Advanced PID Control. ISA - Instrumentation, Systems, and Automation Society. - Detailed exploration of PID control theory and applications.
- Sørensen, A. J. (2011). "A survey of dynamic positioning control systems." Annual Reviews in Control, 35(1), 123-136. - Discusses PID control in the context of dynamic positioning systems, relevant to USVs.
- Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems." Journal of Basic Engineering, 82(1), 35-45. - The original paper introducing the Kalman filter.
- Grewal, M. S., & Andrews, A. P. (2008). Kalman Filtering: Theory and Practice Using MATLAB. John Wiley & Sons. - Practical guide on Kalman filtering with applications in navigation.
- Farrell, J. A., & Barth, M. (1999). The Global Positioning System and Inertial Navigation. McGraw-Hill. - Comprehensive resource on integrating GPS and IMU using Kalman filtering.
- Gade, K. (2016). "The seven ways to find heading." Journal of Navigation, 69(5), 955-970. - Discusses methods for determining heading, relevant for integrating IMU data.
- Lapierre, L., Soetanto, D., & Pascoal, A. M. (2003). "Nonlinear path following with applications to the control of autonomous underwater vehicles." Proceedings of the 42nd IEEE Conference on Decision and Control. - Discusses path following and desired position selection.
- Melsa, J. L., & Schultz, D. L. (2014). "Selected applied digital control techniques for unmanned surface vehicles." IFAC Proceedings Volumes, 47(3), 10991-10996. - Application of digital control techniques for USVs.
- Martinsen, K., Breivik, M., & Fossen, T. I. (2013). "USV trajectory tracking using nonlinear model predictive control." Proceedings of the IEEE/RSJ International Conference on Intelligent

Robots and Systems (IROS), 2652-2658. - Combines path following, control, and estimation techniques.

- Perez, T., & Fossen, T. I. (2009). "Kinematic models for path following and trajectory tracking of marine craft." IEEE Journal of Oceanic Engineering, 34(2), 333-345. - Discusses kinematic models integrating guidance and control.
- Edoardo I. Sarda, Huajin Qu, Ivan R. Bertaska and Karl D. von Ellenrieder. (2019),Station-keeping control of an unmanned surface vehicle exposed to current and wind disturbances.
- Wenbin Huang, Hao Xu, Jiaxiang Feng.(2021), Dynamic modelling and controlling Unmanned Surface Vehicle.