TideSync

Abstract,

This project presents a voice-controlled boat powered by a multi-microcontroller architecture (ESP8266, Arduino Uno, and Nano). The boat operates primarily via Wi-Fi and Bluetooth, enabling long-range control through voice commands or a smartphone app. An ultrasonic-based echo locator ensures collision avoidance, while redundant communication modes (Wi-Fi, Bluetooth, and radio) guarantee uninterrupted operation. The system integrates three microcontrollers: the ESP8266 for wireless communication, the Arduino Uno for motor control, and the Arduino Nano for sensor data processing.

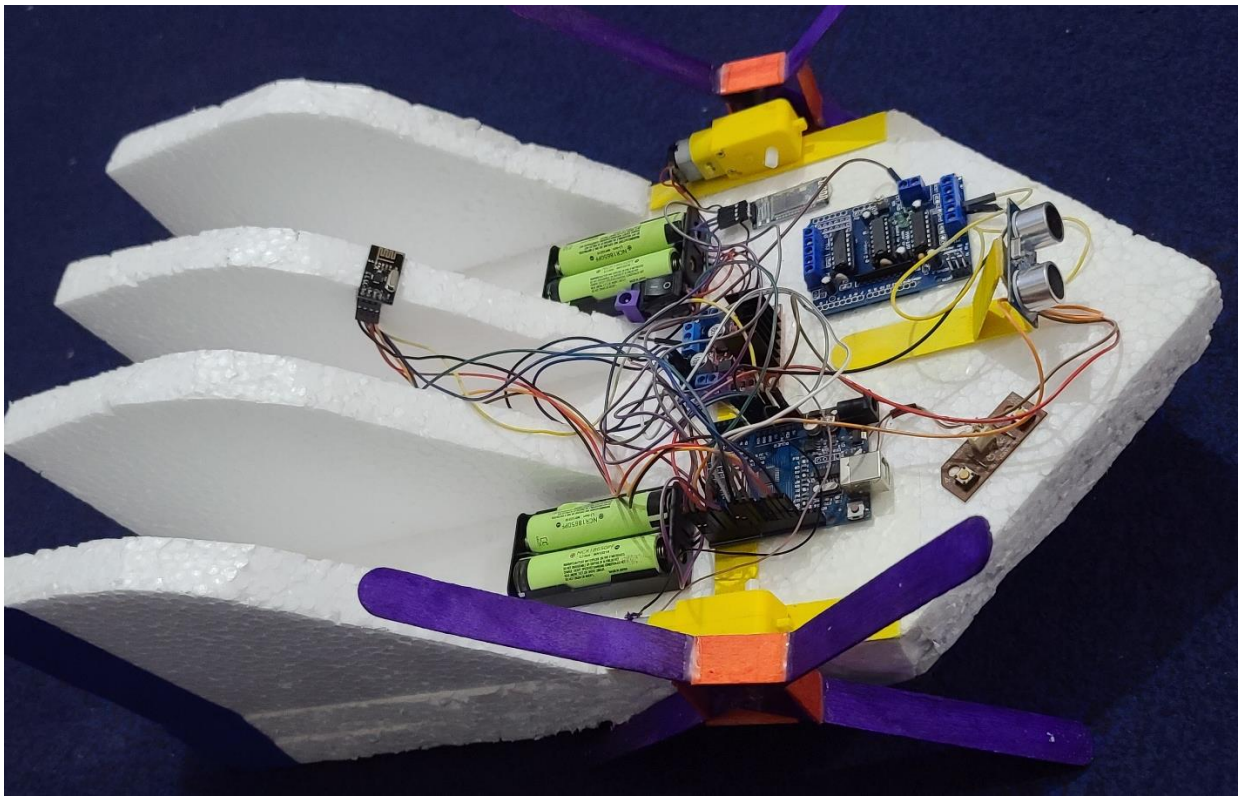


Figure 1. Image of Voice Controlled Boat

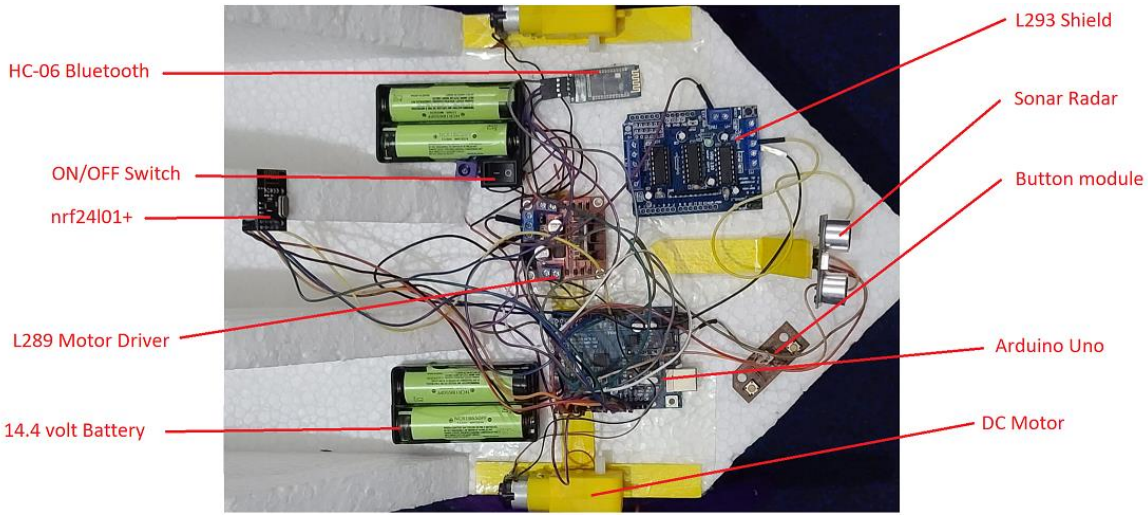
Introduction,

The rapid evolution of embedded systems and wireless communication has unlocked unprecedented possibilities in smart aquatic robotics. This project leverages a multi-microcontroller architecture to create a voice-controlled boat capable of seamless navigation in dynamic water environments. Combining the computational power of an ESP8266 (for Wi-Fi/Bluetooth), an Arduino Uno (motor control), and an Arduino Nano (sensor processing), the boat operates as a cohesive system that prioritizes safety, adaptability, and user convenience.

At its core, the boat responds to voice commands (e.g., "forward," "left") via a pre-trained recognition system, enabling hands-free control. To ensure uninterrupted operation, it dynamically switches between communication modes: Wi-Fi for long-range control, Bluetooth for direct smartphone pairing, and nRF24L01+ radio as a fail-safe when wireless networks are unavailable. An integrated ultrasonic echo locator scans the environment in real time, halting the boat if obstacles are detected within 20 cm, thereby preventing collisions. Designed for versatility, this system addresses challenges in environmental monitoring, search-and-rescue operations, and recreational robotics. Its modular design allows for future expansions, such as water quality sensors or GPS navigation, making it a scalable platform for both hobbyists and researchers.

Circuit Diagram,

Below is the schematic diagram of the Voice Controlled Boat, I've constructed this diagram out of a picture of the circuits due to no designing software being able to support some of my hardware used.



Components List,

Microcontrollers:

ESP8266 (Wi-Fi/Bluetooth interface)

Arduino Uno (Motor control)

Arduino Nano (Ultrasonic sensor)

Sensors:

HC-SR04 Ultrasonic Sensor (Echo locator)

Wireless Modules:

nRF24L01+ (Fallback radio communication)

Motor Control:

L293D Motor Driver Shield

DC Motors (x2)

Power:

LiPo Battery (11.1V)

Miscellaneous:

Jumper Wires, Voltage Regulators, PCB

Advantages,

The system features **voice-activated navigation**, allowing users to control the boat hands-free using simple, natural language commands such as “turn right” or “stop.” This intuitive interface enhances both accessibility and user experience, especially in dynamic environments or for users with physical limitations.

A **multi-microcontroller architecture** distributes key functions across three dedicated boards: the ESP8266 handles wireless communication, the Arduino Uno manages motor control, and the Nano is responsible for sensor data processing. This parallelized approach eliminates processing bottlenecks, ensuring efficient real-time response and overall system stability.

To ensure continuous connectivity, the design incorporates **redundant communication modes**. The system can automatically switch between Wi-Fi, Bluetooth, and radio signals based on availability and signal strength, making it reliable even in remote or signal-poor aquatic environments.

Real-time collision prevention is achieved through an ultrasonic echo locator that continuously scans for nearby obstacles. When a potential collision is detected, the system immediately triggers a stop command, preventing accidents and making the boat safer for cluttered or unpredictable environments.

The platform's **scalability** is a major strength, enabled by its modular design. This allows for the easy addition of new sensors—such as pH or turbidity sensors for water quality analysis—or advanced features like GPS-based autonomous navigation, expanding its application potential in both research and environmental monitoring.

Cost-effective prototyping is made possible by leveraging low-cost, readily available components such as the ESP8266 and Arduino boards. This approach lowers the barrier to entry, making advanced aquatic robotics more accessible to hobbyists, educators, and researchers.

With significant **educational and research value**, the system serves as a practical example of embedded systems design, wireless communication protocols, and sensor integration. It is a valuable teaching tool for students and enthusiasts seeking hands-on experience with robotics.

The architecture is also designed for **energy efficiency**. By delegating tasks to specialized microcontrollers, the system reduces redundant power consumption, enabling longer field operations and improved battery life.

Support for **Over-the-Air (OTA) updates** via the ESP8266 allows firmware to be updated wirelessly. This feature enables remote debugging, performance tuning, and the rollout of new features without requiring physical access to the boat, increasing flexibility and convenience.

Finally, **cross-platform control** is ensured through compatibility with both Android and iOS applications. Whether connected via Bluetooth or Wi-Fi, users can easily control the system from a wide range of devices, making it accessible to virtually anyone with a smartphone or tablet.

Code,

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 8); // CE, CSN
const byte address[6] = "00001";

#define IN1 4
#define IN2 5
#define IN3 6
#define IN4 7
const  int trigPin = 3;      // TRIG pin connected to pin D6 of Arduino Nano
const int  echoPin = 2;      // ECHO pin connected to pin D7 of Arduino Nano
int i,j,k  = 0;
float duration,distance;
String voice;

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
```

```
  pinMode(IN4, OUTPUT);

  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);


  radio.begin();
  Serial.begin(9600);
  radio.openReadingPipe(1, address);
  radio.startListening();
}
void loop() {
  while (Serial.available() > 0) {
    delay(10);
    char c=Serial.read();
    if(c=='#')
    {
     break;
    }
    voice+=c;
  }

    //Serial.print(value);
    char position[10];
    radio.read(&position, sizeof(position));
    Serial.println(position);
    //dist();
    if (voice == "forward") {
      forward();
      delay(5000);
    } else if (voice == "backward") {
      backward();
      delay(5000);
    } else if (voice == "left") {
      left();
      delay(2000);
    } else if (voice == "right") {
      right();
      delay(2000);
    } else if (position == "Forward") {
      forward();
      delay(5000);
    } else if (position == "Backward") {
      backward();
      delay(5000);
```

```
    } else if (position == "Left") {
      left();
      delay(2000);
    } else if (position == "Right")  {
      right();
      delay(2000);
    }
     else {
      stop();
    }
}
/*
void loop() {
  // Read joystick position
  char position[10];
  radio.read(&position, sizeof(position));

  // If joystick position is not neutral, use it to control the boat
  if (strcmp(position, "Neutral") != 0) {
    controlBoat(position);
  }
  // Otherwise, check for voice commands
  else {
    while (Serial.available() > 0) {
      delay(10);
      char c = Serial.read();
      if (c == '#') {
        break;
      }
      voice += c;
    }
    controlBoat(voice.c_str());
  }
}
*/
void controlBoat(const char* command) {
  dist(); // Check for obstacles before moving
  if (distance >= 20) { // Only move if no obstacle is detected
    if (strcmp(command, "forward") == 0 || strcmp(command, "Forward") == 0) {
      forward();
      delay(5000);
    } else if (strcmp(command, "backward") == 0 || strcmp(command, "Backward") ==
0) {
      backward();
      delay(5000);
```

```
    } else if (strcmp(command, "left") == 0 || strcmp(command, "Left") == 0) {
      left();
      delay(2000);
    } else if (strcmp(command, "right") == 0 || strcmp(command, "Right") == 0) {
      right();
      delay(2000);
    } else {
      stop();
    }
  }
}

void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void left() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void right() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
 void dist()
     {
```

```
       digitalWrite(trigPin,LOW);          //  to send a pulse via the TRIG pin
of HC-SR04
       delayMicroseconds(2);
       digitalWrite(trigPin,HIGH);
       delayMicroseconds(10);
       digitalWrite(trigPin,LOW);

       duration = pulseIn(echoPin,HIGH);   // read the duration of the pulse

       distance = (duration*0.0343)/2;     // measure the distance in cms.
Speed  of sound is 340 m/s or 0.0343 cm/us
       if(distance<20)                     //  if distance is < 20 cms , STOP
robot
       {
        stop();
       }
        delay(1000);
     }
```

```
# ESP8266 code:
#include <SoftwareSerial.h>
#include <VoiceRecognitionV2.h>
#include <nRF24L01.h>
#include <RF24.h>
// Voice Recognition
VR myVR(2, 3); // RX, TX pins
#define forwardCommand 0
#define backwardCommand 1
#define leftCommand 2
#define rightCommand 3
// nRF24L01 Radio
RF24 radio(4, 5); // CE, CSN
const byte address[6] = "00001";
// Motor Control Pins (Arduino Uno)
SoftwareSerial unoSerial(6, 7); // RX, TX to Uno

void setup() {
 Serial.begin(9600);
 myVR.begin(9600);
 unoSerial.begin(9600);
 // Initialize nRF24L01
 radio.begin();
```

```
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MAX);
  // Load Voice Commands
  myVR.load((uint8_t)forwardCommand);
  myVR.load((uint8_t)backwardCommand);
  myVR.load((uint8_t)leftCommand);
  myVR.load((uint8_t)rightCommand);
}
void loop() {
  // Check for voice commands
  int command = myVR.recognize();
  if (command >= 0) {
    sendCommand(command);
  }

  // Check for Bluetooth/Wi-Fi app commands
  if (Serial.available()) {
    String appCommand = Serial.readString();
    sendCommand(appCommand);
  }
}
void sendCommand(int cmd) {
  switch (cmd) {
    case forwardCommand:
      radio.write("F", 1); // Fallback radio
      unoSerial.write("F"); // Send to Arduino Uno
      break;
    case backwardCommand:
      radio.write("B", 1);
      unoSerial.write("B");
      break;
    case leftCommand:
      radio.write("L", 1);
      unoSerial.write("L");
      break;
    case rightCommand:
      radio.write("R", 1);
      unoSerial.write("R");
      break;
  }
}
```