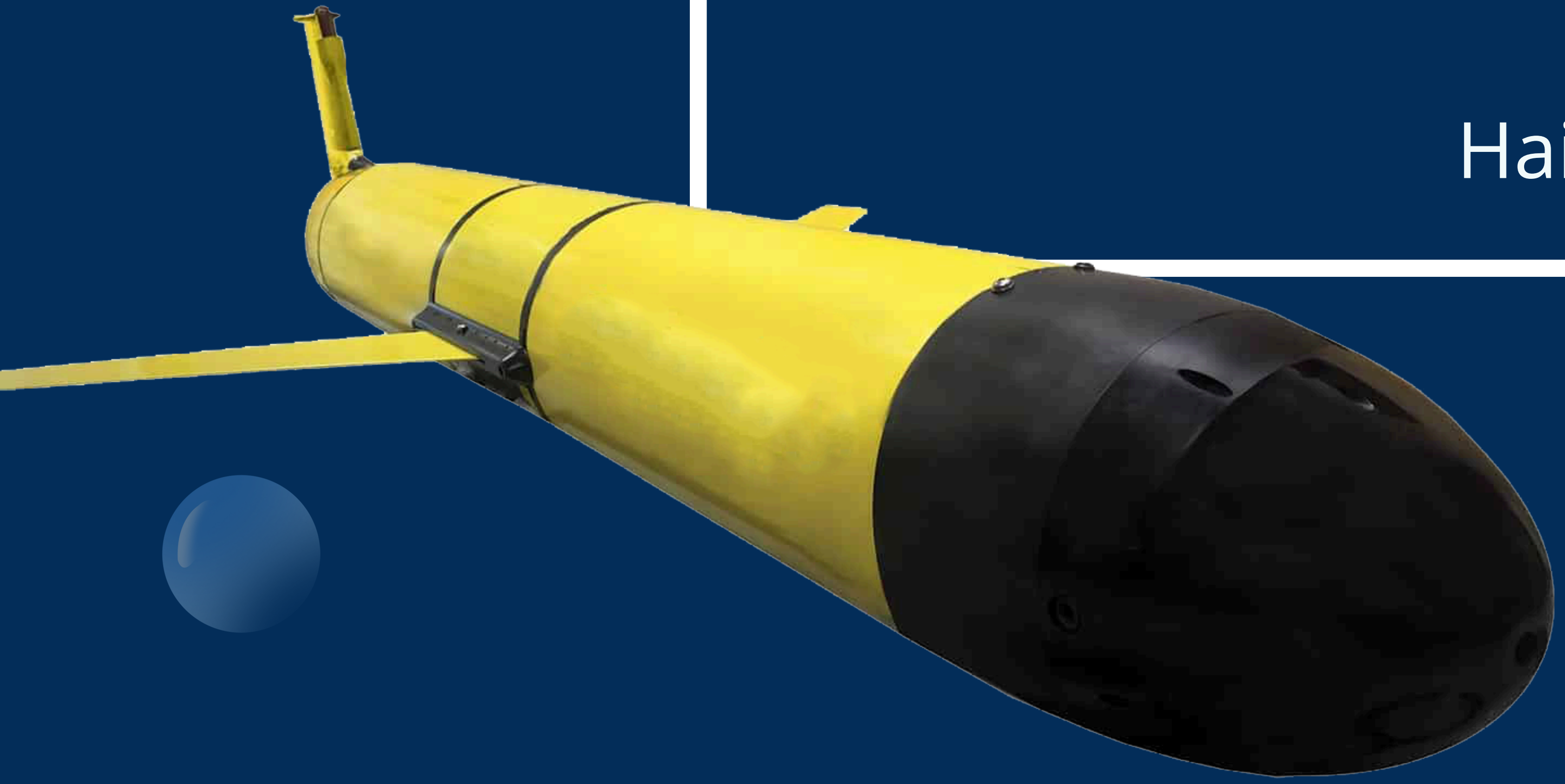
UNDERWATER GLIDER

Zain A. Samdani
Haider Qaizar Hussain



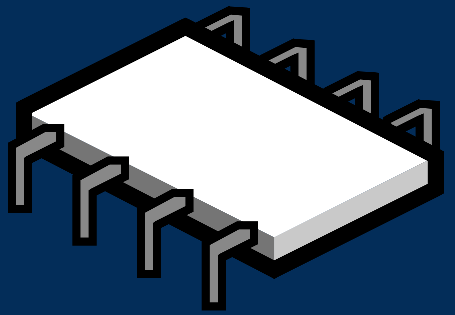
Supervized by,
Francesco Maurelli, Vikram Unnithan, Martin Messmann

Overview

Build a low-cost, open-source underwater glider suitable for educational and research purposes.

Components Required

- 3D printed Body and Internal structure
- Main Tubing 3.15 inches
- Stepper motor - to control buoyancy engine
- Power Supply - Lithium-ion cells
- Arduino - Microcontroller



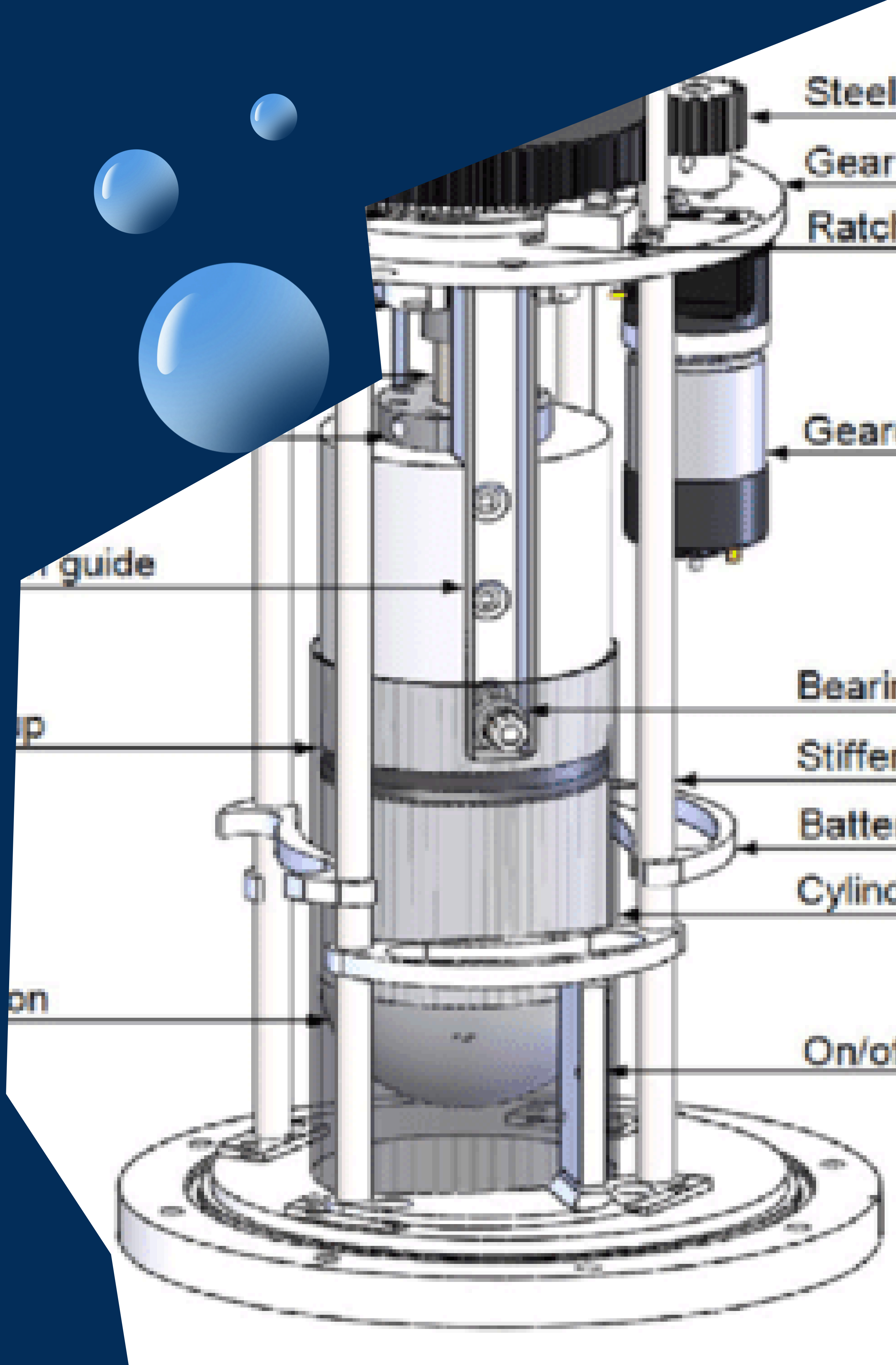
Buoyancy Engine

Allows Underwater gliders to change their buoyancy and, consequently, their depth in the water column

Longer Missions

Conserves Energy

Enables the glider to glide through the water efficiently, conserving energy and extending the duration of underwater missions.



How does it work?

A piston moves within a cylinder inside the glider.

Retracting Piston

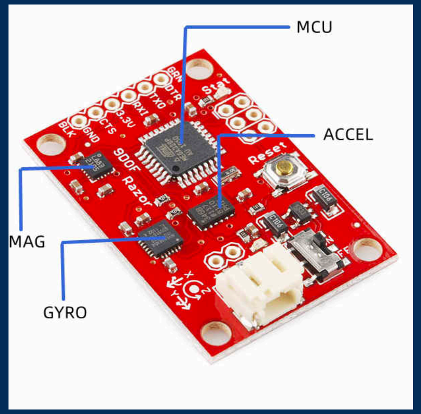
Decreases internal volume, increases density, glider sinks.

Extending Piston

Increases internal volume, decreases density, glider ascends.

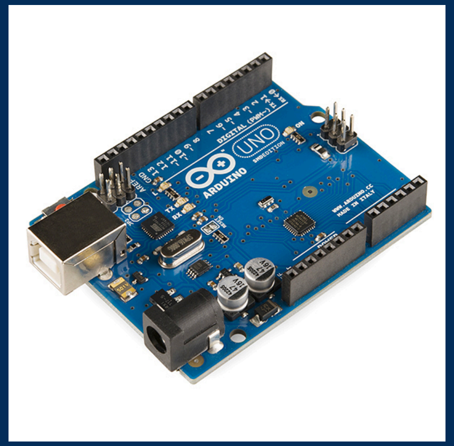
Control System

The control system of the underwater glider revolves around an Arduino-based microcontroller, which manages operations and data collection.



IMU (Inertial Measurement Unit)

- Accelerometer: Measures tilt and pitch
- Gyroscope: Tracks rotational movements

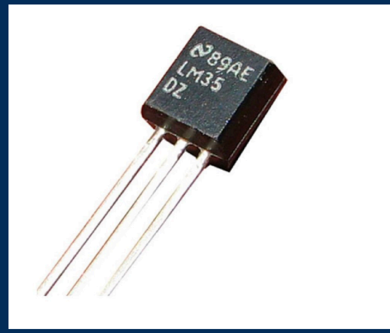


Arduino

- Receives data from the IMU
- Processes sensor data to determine the glider's orientation and trajectory

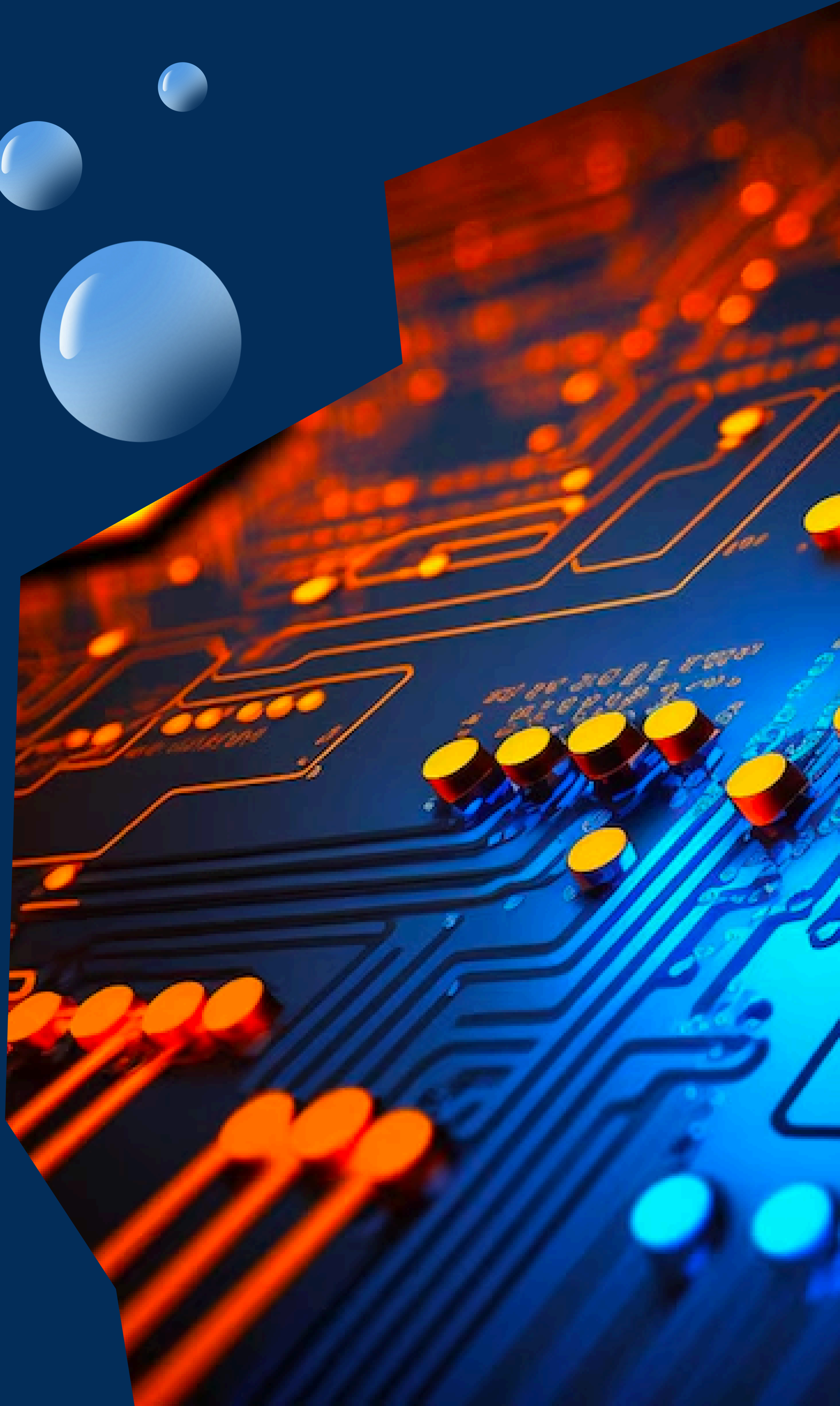
Actuators

- Buoyancy Engine: Adjusts depth by changing buoyancy.
- Motor and Fins: Control movement and direction

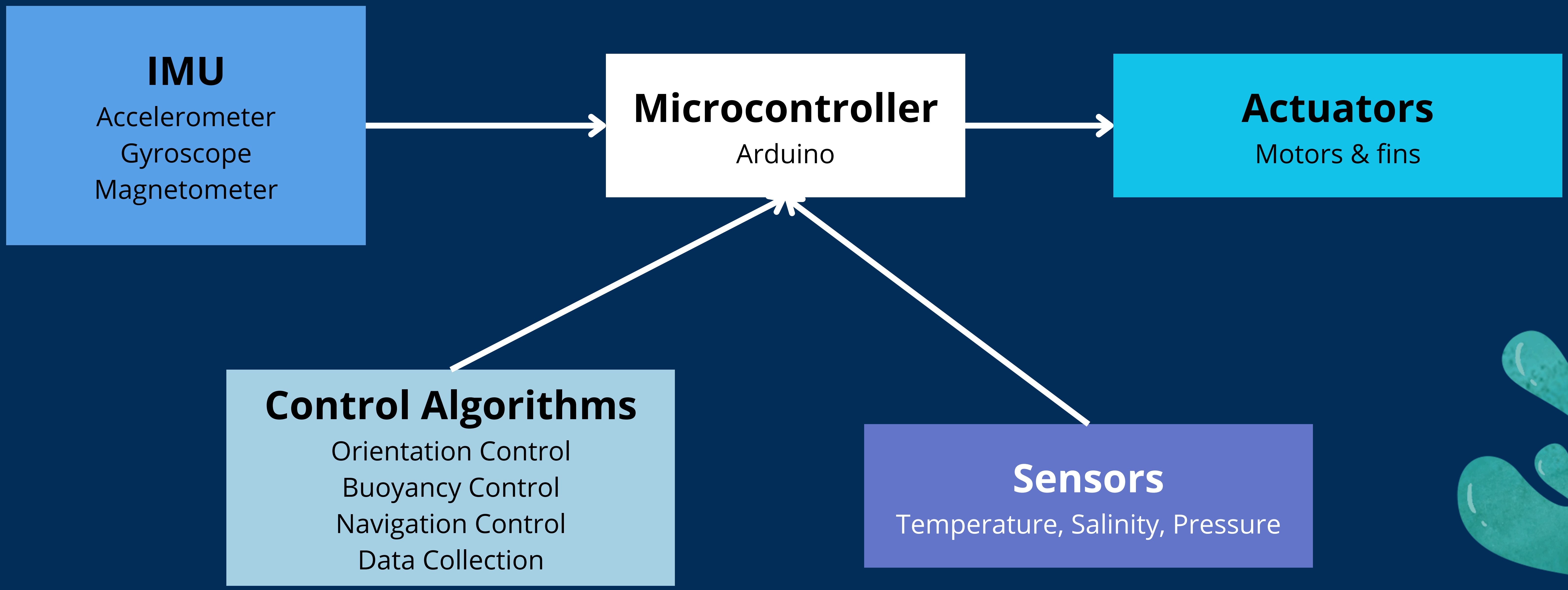


Sensors

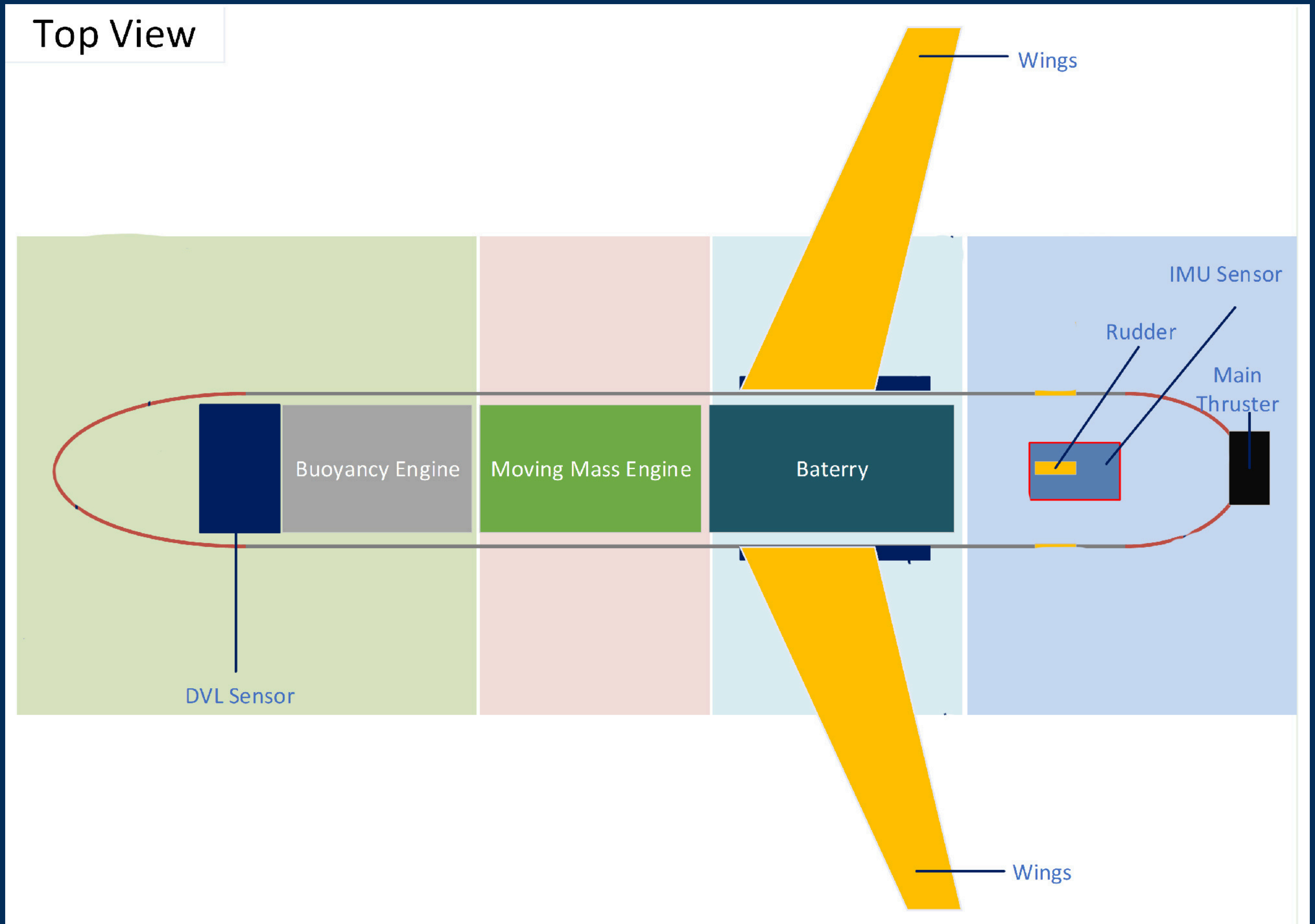
Temperature | Salinity | Pressure

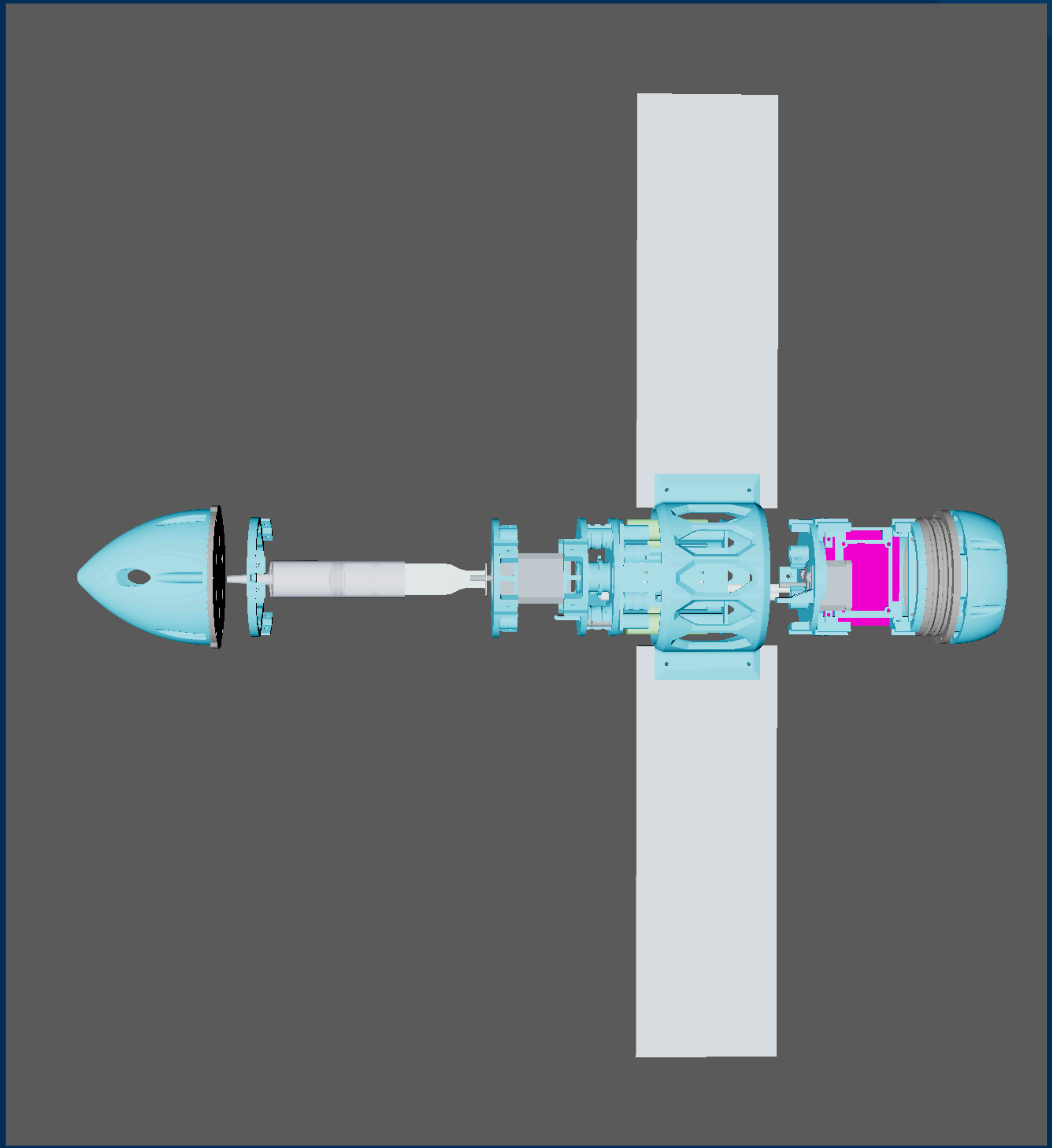


Control System

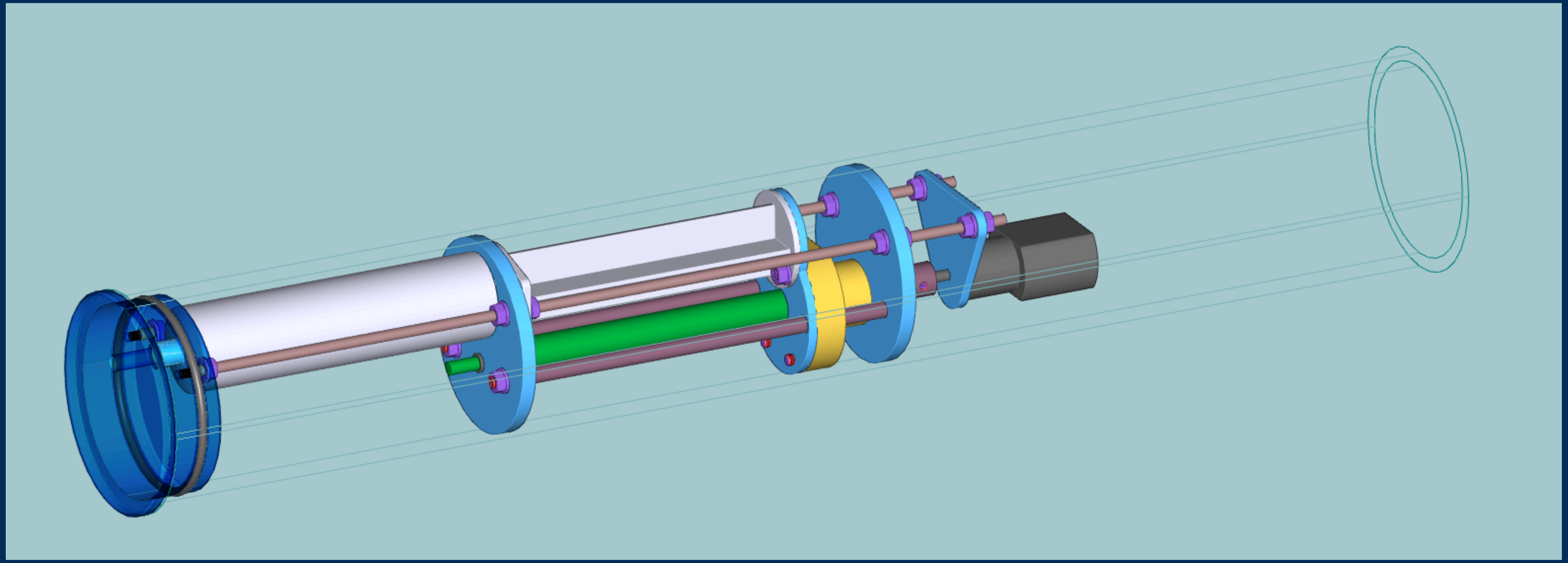


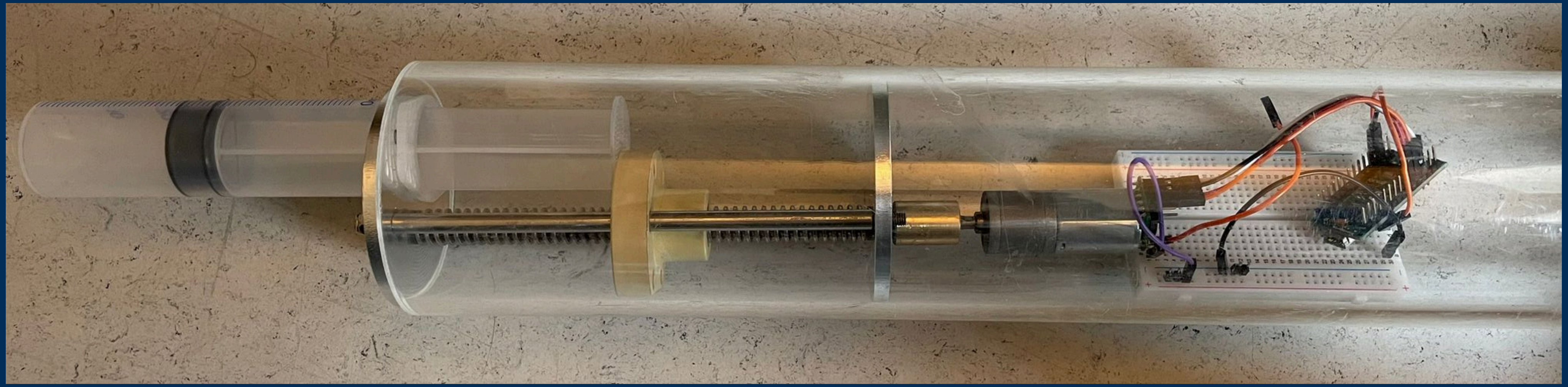
Glider Design





Why single syringe





Easier maintainence

Using a single syringe simplifies the mechanical design and reduces the number of moving parts

Cost Effective

Making the glider more accessible for educational and research purposes

Reliability

Reducing potential points of failure

Applications

Student Researchers

Engage Student researchers in data collection and monitoring efforts. Community members can help deploy gliders and analyze the collected data.

Pollution Tracking

Utilize the glider to detect and monitor pollution levels in various water bodies. Sensors can measure parameters like chemical concentrations, oil spills, and other pollutants.

01

02

04

03

Station-Keeping

Maintaining a fixed position in the water column for extended monitoring or data collection

Breadboard - Future Development

The device in its current form could be used as a breadboard for future development. By easily encorporating various sensors and other mechanisms

Testing

Arduino Uno

sketch_jun3a.ino

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_MS5803.h>

// Motor control pins
int motor1pin1 = 3;
int motor1pin2 = 4;
int pwmPin = 9;

// Pressure sensor
Adafruit_MS5803 pressureSensor = Adafruit_MS5803();

// IMU sensor
Adafruit_MPU6050 mpu;

// Threshold values
float pressureThreshold = 500.0; // Example threshold value for depth

void setup() {
  // Initialize motor control pins as outputs
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(pwmPin, OUTPUT);

  // Initialize serial communication for debugging
  Serial.begin(9600);

  // Initialize pressure sensor
  if (!pressureSensor.begin()) {
    Serial.println("Failed to find MS5803 chip");
    while (1) { delay(10); }
  }

  // Initialize IMU
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) { delay(10); }
  }

  Serial.println("Setup complete");
}

void loop() {
  // Read the pressure value from the sensor
  pressureSensor.read();
  float pressureValue = pressureSensor.pressure();

  // Read IMU values
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // Print the pressure and IMU values for debugging
  Serial.print("Pressure: "); Serial.println(pressureValue);
  Serial.print("Accel X: "); Serial.print(a.acceleration.x);
  Serial.print(" Y: "); Serial.print(a.acceleration.y);
  Serial.print(" Z: "); Serial.println(a.acceleration.z);
  Serial.print("Gyro X: "); Serial.print(g.gyro.x);
  Serial.print(" Y: "); Serial.print(g.gyro.y);
  Serial.print(" Z: "); Serial.println(g.gyro.z);

  // Check if pressure exceeds the threshold
  if (pressureValue > pressureThreshold) {
    // Rotate motor in one direction
    digitalWrite(motor1pin1, HIGH);
    digitalWrite(motor1pin2, LOW);
    analogWrite(pwmPin, 255); // Set motor speed
  } else {
    // Rotate motor in the opposite direction
    digitalWrite(motor1pin1, LOW);
    digitalWrite(motor1pin2, HIGH);
    analogWrite(pwmPin, 255); // Set motor speed
  }

  delay(1000); // Small delay to prevent too rapid switching
```

IMU

Successfully controlled the motor and fins of the glider to maintain a desired orientation and path

Accelerometer

Determined pitch and tilt of the gllider

Gyroscope

Tracked the rotational movements and stabilize the glider's movement

Buoyancy Control

Buoyancy engine responded correctly to commands sent by the microcontroller

Challenges

Pressure Resistance

The system must withstand high pressures at greater depths

Control Precision

Requires precise control to achieve desired depth changes accurately

Energy Usage

Must operate efficiently to conserve battery power

Wear and Tear

Components must be robust and designed for longevity

Next Steps

EXPANSION + PROJECTS

Engaging students in projects involving environmental monitoring and data analysis. Encourage them to design experiments and analyze real-world data collected by the glider.

THANK YOU!