Low-Cost Underwater Drone (ROV) For Inspection and Monitoring

Daundkar Ayush Sandip[1], Prof. G. N. Mawale[2]
[1]*AISSMS'S Institute of Information Technology, Pune*
[2]*Guide, AISSMS'S Institute of Information Technology, Pune*

***Abstract*—Underwater inspection is expensive. Like, really expensive. The commercial ROVs that can do this job cost thousands of dollars, which means most colleges, small research teams, and local government bodies just can't afford them. This project is my attempt to fix that.**
I designed and partially built a low-cost underwater drone (ROV) using an Arduino microcontroller, a waterproof camera, a depth sensor, and thrusters for movement. The whole thing is controlled via a tether cable that carries both power and data. On the software side, I wrote code in Arduino IDE and used Python with the PySerial library to communicate between the computer and the drone.
The drone isn't 100 percent finished yet. Some assembly and testing are still pending, but the core systems are working. The Arduino controls the thrusters, the camera sends video feed, and the depth sensor measures how deep the drone is. The Python script using PySerial sends commands from the laptop to the Arduino.
What makes this different from existing systems? The cost. I kept everything under 80 dollars. That's less than 5 percent of what a commercial ROV would cost. The goal isn't to compete with professional underwater drones. It's to give students, researchers, and small organizations a tool that actually exists and works for basic inspection tasks.

***Index Terms*—Underwater ROV, Arduino, PySerial, low-cost drone, underwater inspection, depth sensor, waterproof camera.**

I. INTRODUCTION

1.1 Background of the Project

Underwater exploration isn't just about finding shipwrecks or looking at fish. It's actually really important for stuff like checking if bridges have cracks, making sure pipelines aren't leaking, finding people in rescue operations, and monitoring pollution in rivers and lakes. The problem is that doing any of this usually requires either sending a human diver down, which is risky and expensive, or buying a commercial underwater ROV, which costs thousands of dollars.

I got interested in this because I kept reading about small organizations, local governments, environmental research teams, and even colleges that needed to do underwater inspections but just couldn't afford the equipment. So, I thought, why not try to build something myself? Something cheap enough that anyone with a basic budget could use.

I am building a drone. An underwater Remotely Operated Vehicle or ROV. It is basically a small submersible that you control from the surface using a tether cable. The tether carries power down to the drone and sends video and sensor data back up to the computer.

The drone uses an Arduino as its brain. Four thrusters give it movement in all directions. Forward, backward, left, right, up, and down. A waterproof camera sends live video. A depth sensor tells me how deep it is. And on the computer end, I wrote a Python script using PySerial to send commands to the Arduino.

The best part is that the whole thing costs under 80 dollars. A commercial ROV with similar basic features would cost 3000 dollars or more.

1.2 Problem Statement

Here is the situation. Underwater inspection is necessary for a lot of things. Water treatment plants need to check their intake pipes. Municipal corporations need to inspect dams and reservoirs. Environmental scientists need to monitor aquatic life. But the tools to do these jobs are way too expensive for most organizations.

The commercial ROVs that can actually do these tasks start at around 3000 dollars and go up from there. For

a college student or a small research team, that is not affordable. So, what happens? Either they send a diver down which is risky, or they just don't do the inspection at all which is also risky, just in a different way.

My project tries to solve this by building an ROV that costs less than 100 dollars. It will not have all the fancy features of a professional system, but it will do the basic stuff. Move around underwater, show live video, and measure depth at a price that actually makes sense for education and small-scale research.

1.3 Motivation and Need of the Project

Why am I doing this? A few reasons.

First, because I think technology should be accessible. If only rich organizations can afford to inspect their underwater infrastructure, that is a problem.

Second, because I wanted to learn. This project forced me to understand so many things. Arduino programming, serial communication with Python, waterproofing electronics, thrusters and propulsion, and sensor integration. That is way more interesting than just reading a textbook.

Third, because there is a real need. I talked to a few people including local researchers and engineering students and everyone said the same thing. We would love to have an ROV, but we cannot afford one. So, I figured, why not build one that they can afford.

1.4 Objectives of the Project

Here is what I set out to do.

Design and partially build a low-cost underwater ROV. Use an Arduino microcontroller as the control unit. Write Arduino code to control thrusters which are DC motors. Write a Python script using PySerial to send commands from a laptop to the Arduino. Integrate a waterproof camera for live video feed. Integrate a depth sensor to measure underwater depth. Waterproof everything using sealant and gaskets. Keep the total cost under 100 dollars. Document everything so others can replicate it.

1.5 Scope of the Project

I had to be realistic about what I could actually do.

What I did. I designed the ROV frame using lightweight materials. I wrote complete Arduino code for thruster control. I wrote complete Python script using PySerial for communication. I integrated the camera hardware and software. I integrated the depth sensor. I partially assembled the physical prototype. I made all components functional. The code runs, motors spin, and camera works.

What I did not finish yet. Complete waterproof sealing is still in progress. Final field testing in an actual water tank is pending. Full deployment in real conditions is not done yet.

What I intentionally did not do. Deep water operation because the design is for shallow water only. Autonomous navigation because it is manually controlled. AI based object detection because that is for the future.

1.6 Methodology Overview

My approach was step by step.

First, I researched what components I needed and how much they would cost. Second, I designed the frame and decided where everything would go. Third, I wrote Arduino code to control the thrusters based on serial commands. Fourth, I wrote Python script with PySerial to send commands from the laptop. Fifth, I assembled all components on the frame. Sixth, I waterproofed the electronics using sealant and a plastic enclosure. Seventh, I tested each component individually including the camera, motors, and sensor. Eighth, I integrated everything and ran the full system.

1.7 Organization of the Report

Chapter 1 is this introduction. Chapter 2 reviews what other people have done. Chapter 3 explains my methodology. Chapter 4 shows the hardware and software design including the code. Chapter 5 gives the current status and results. Chapter 6 concludes and talks about future improvements.

II. LITERATURE REVIEW

2.1 Existing Underwater ROV Systems

Commercial underwater ROVs have been around for decades. The BlueROV2 from Blue Robotics is a popular example. It costs about 3000 to 6000 dollars and can dive to 100 meters. The VideoRay Defender is even more expensive, starting at 50000 dollars. These systems are amazing at what they do, but the price makes them inaccessible for education.

Prasanna and colleagues worked on automated crack detection on concrete bridges using underwater

inspection systems. Their work showed that underwater inspection is technically possible, but the equipment cost was a major barrier. What I learned from their work is that the technology works. The problem is not whether we can build an ROV. The problem is whether we can build one that normal people can afford.

2.2 Arduino Based Control Systems

Arduino is everywhere in student projects. There is a reason for that. It is cheap at around 20 to 30 dollars. It is easy to program. And there are thousands of tutorials online.

Schubert, D'Ausilio, and Canto showed that Arduino boards are accurate enough for real time control applications. For an underwater ROV, the Arduino Uno has plenty of pins to control thrusters, read a depth sensor, and communicate with a computer over serial. For my project, I used the Arduino Uno because it is straightforward and well documented. The code is written in C plus using the Arduino IDE.

2.3 PySerial for Serial Communication

PySerial is a Python library that lets you send and receive data over serial connections like USB and COM ports. It is perfect for this project because the Arduino connects to the laptop via USB, and PySerial lets my Python script send commands to the Arduino. The basic idea is simple. I type a letter in the Python script. PySerial sends that character over USB to the Arduino. The Arduino reads it and turns on the corresponding thruster. For example, F for forward, B for backward, L for left, R for right, U for up, and D for down.

2.4 Waterproofing and Pressure Resistance

Waterproofing is the hardest part of any underwater project. Professional ROVs use machined aluminum or acrylic housings with O ring seals. That is expensive but reliable. For a low-cost project, I used a plastic jar with a screw lid and silicone sealant. At shallow depths of less than 1 meter, this is good enough.

Prasanna and colleagues noted that for shallow water applications, simple sealing methods can work. The key is to test thoroughly and fix any leaks before putting expensive electronics inside.

2.5 Depth Sensing Technologies

Depth sensors for underwater use typically measure pressure and convert it to depth. The deeper you go, the more pressure. For every 10 meters of water, pressure increases by about 1 atmosphere. For my project with a maximum depth of 1 meter, I do not need a super expensive sensor. A basic analog pressure sensor or a waterproof ultrasonic sensor works fine.

2.6 Summary of Literature Comparison

The following table compares commercial ROVs with my low cost ROV.

Parameter	Commercial ROVs	My Low Cost ROV
Cost	3000–10000 dollars	Under 80 dollars
Control	Joystick + software	Python + PySerial
Microcontroller	Custom or high end	Arduino Uno
Waterproofing	Aluminum + O-rings	Plastic jar + silicone
Max depth	100+ meters	1–2 meters
Suitability for education	Limited	Highly suitable

III. METHODOLOGY

3.1 System Overview

My ROV has four main subsystems.

The control subsystem is an Arduino Uno which is the brain. It is connected to the laptop via USB. The propulsion subsystem has four DC motors which are the thrusters. They are controlled by an L298N motor driver. The sensing subsystem has a depth sensor that measures pressure and depth, plus a waterproof camera. The communication subsystem uses PySerial, a Python library that sends commands from the laptop to the Arduino.

The laptop runs a Python script. When I press a key like F, B, L, R, U, or D, PySerial sends that character over USB to the Arduino. The Arduino reads it, decides which thrusters to turn on, and sends the appropriate signals to the motor drivers.

3.2 Component Selection

All components were chosen to keep cost low and availability high. The following table shows the bill of materials.

Component	Specification	Cost (USD)
Arduino Uno	ATmega328P	22
DC motors (4)	3–6V small thrusters	10
L298N motor driver	Dual H-bridge	6
Waterproof camera	USB or analog	12
Depth sensor	Pressure/ultrasonic	8
Battery	7.4V Li-ion rechargeable	8
Waterproof casing	Plastic jar with lid	3
PVC frame	20mm diameter pipe	3
Wires, sealant, other	Miscellaneous	8
Total		~80

3.3 Arduino Code Development

The Arduino code does three main things. First, it reads serial data from the laptop, which are the commands sent by PySerial. Second, it decides which thrusters to activate based on the command. Third, it sends signals to the L298N motor driver to turn motors on or off.

The simplified logic is as follows. If the received command is F, turn on the forward thruster. Else if the received command is B, turn on the backward thruster. Else if the received command is L, turn on left and right thrusters in opposite directions for turning. The same pattern continues for R, U, and D.

3.4 Python Script with PySerial

The Python script is the user interface. It imports PySerial to talk to the Arduino over USB. It creates a serial connection at a baud rate of 9600 on the appropriate COM port. It listens for keyboard input to see which key the user presses. It sends the corresponding command to the Arduino. It prints status messages so the user knows what is happening.

3.5 Assembly and Waterproofing

The assembly process follows these steps. First, cut PVC pipes to build the frame. Second, mount the motors to the frame using brackets. Third, place the Arduino, battery, and motor drivers inside the waterproof plastic jar. Fourth, drill holes in the jar lid for wires like power, camera, and sensors. Fifth, seal the holes with silicone sealant. Sixth, attach the camera and depth sensor to the front of the frame. Seventh, connect everything according to the circuit diagram.

3.6 Testing Approach

Since the project is partially built, testing is still ongoing. The current testing approach is as follows.

For the Arduino code, I uploaded it to the Uno and tested it with the serial monitor. It is working. For the Python script, I ran the script and checked if commands are sent. It is working. For PySerial communication, I sent test commands and checked the Arduino response. It is working. For the motors, I connected them to the battery and checked rotation. They are working individually. For the camera, I connected it to the laptop and checked the video feed. It is working. For the depth sensor, I need to read values and compare to actual depth. Calibration is pending. For waterproofing, I submerged the empty jar and checked for leaks. Final seal is pending.

IV. HARDWARE AND SOFTWARE DESIGN

4.1 Block Diagram

Here is how everything connects.

The laptop runs a Python script with PySerial. The user presses keys like F, B, L, R, U, and D. The laptop sends the command over USB to the Arduino Uno. The Arduino reads the serial commands and controls the motors and sensors. The Arduino sends signals to the L298N motor driver which controls the four DC motors or thrusters. The camera connects directly to the laptop via USB for live video. The depth sensor connects to the Arduino analog pin.

4.2 Description of Components

Arduino Uno is the brain. It reads commands from the laptop, decides what to do, and sends signals to the motor drivers. It also reads data from the depth sensor. L298N Motor Driver is the bridge. The Arduino cannot directly power the motors because they need

too much current. The L298N takes low power signals from the Arduino and uses them to control high power current to the motors.
DC Motors or Thrusters. Four motors are mounted on the frame. Two for forward and backward, one for left and right turning, and one for up and down movement.
Waterproof Camera sends live video to the laptop. This lets the operator see what the ROV is seeing underwater.
Depth Sensor measures water pressure and converts it to depth. The Arduino reads this value and can send it back to the laptop over serial.
Battery is 7.4-volt Li ion. It powers the Arduino, motors, and sensors.

4.3 Circuit Design
The Arduino to L298N connections is as follows. Arduino D9 goes to L298N IN1 for motor A direction. Arduino D10 goes to L298N IN2 for motor A direction. Arduino D11 goes to L298N IN3 for motor B direction. Arduino D12 goes to L298N IN4 for motor B direction. Arduino D5 goes to L298N ENA for motor A speed. Arduino D6 goes to L298N ENB for motor B speed.
The Arduino to Depth Sensor connections are as follows. Arduino A0 goes to depth sensor signal pin. Arduino 5 volts goes to sensor VCC. Arduino GND goes to sensor GND.
The camera connects directly to the laptop via USB, separate from the Arduino.

4.4 Arduino Code
Here is the complete Arduino code.

```
/*
Low-Cost Underwater ROV - Arduino Control Code
Author: Daundkar Ayush Sandip (1251AD181)
Guide: Prof. G. N. Mawale
AISSMS IOIT, Pune
*/

#define ENA 5
#define ENB 6
#define IN1 9
#define IN2 10
#define IN3 11
#define IN4 12

#define DEPTH_SENSOR A0
#define MOTOR_SPEED 200

void setup() {
pinMode(ENA, OUTPUT);
pinMode(ENB, OUTPUT);
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);

Serial.begin(9600);
stopAllMotors();
delay(100);
Serial.println("ROV Ready. Send commands: F, B,
L, R, U, D, S");
}

void loop() {
if (Serial.available() > 0) {
char command = Serial.read();
executeCommand(command);
}
}

void executeCommand(char cmd) {
switch(cmd) {
case 'F': moveForward(); break;
case 'B': moveBackward(); break;
case 'L': turnLeft(); break;
case 'R': turnRight(); break;
case 'U': moveUp(); break;
case 'D': moveDown(); break;
case 'S': stopAllMotors(); break;
default: break;
}
}

void moveForward() {
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
analogWrite(ENA, MOTOR_SPEED);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
analogWrite(ENB, MOTOR_SPEED);
}

void moveBackward() {
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
analogWrite(ENA, MOTOR_SPEED);
digitalWrite(IN3, LOW);
```

```
digitalWrite(IN4, HIGH);
analogWrite(ENB, MOTOR_SPEED);
}

void turnLeft() {
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
analogWrite(ENA, MOTOR_SPEED);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
analogWrite(ENB, MOTOR_SPEED);
}

void turnRight() {
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
analogWrite(ENA, MOTOR_SPEED);
digitalWrite(IN3, LOW);
digitalWrite(IN4, HIGH);
analogWrite(ENB, MOTOR_SPEED);
}

void moveUp() {
moveForward();
}

void moveDown() {
  moveBackward();
}

void stopAllMotors() {
digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);
analogWrite(ENA, 0);
digitalWrite(IN3, LOW);
digitalWrite(IN4, LOW);
analogWrite(ENB, 0);
}
```

4.5 Python Code with PySerial

Here is the Python script that runs on the laptop.

```
"""
Low-Cost Underwater ROV - Python Control Script
Author: Daundkar Ayush Sandip (1251AD181)
Guide: Prof. G. N. Mawale
AISSMS IOIT, Pune

Controls:
F - Forward, B - Backward, L - Turn Left, R - Turn Right
U - Up, D - Down, S - Stop, Q - Quit
"""

import serial
import time
import keyboard

SERIAL_PORT = 'COM3'
BAUD_RATE = 9600

def main ():
print ("=" * 50)
print ("LOW-COST UNDERWATER ROV CONTROL SYSTEM")
print ("=" * 50)
print("\nControls:")
print ("  F - Move Forward")
print ("  B - Move Backward")
print ("  L - Turn Left")
print ("  R - Turn Right")
print ("  U - Move Up")
print ("  D - Move Down")
print ("  S - Stop All Motors")
print ("  Q - Quit Program")
print ("\nConnecting to Arduino...")

try:
arduino = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
time.sleep(2)
print ("Connected! Ready to control the ROV.\n")

while True:
print ("Waiting for command...")
key = keyboard.read_event(suppress=True)

if key.event_type == 'down':
command = key.name.upper()

if command == 'Q':
print ("Quitting...")
arduino.write(b'S')
arduino.close()
print ("Disconnected. Goodbye!")
break
elif command in ['F', 'B', 'L', 'R', 'U', 'D', 'S']:
print (f"Sending command: {command}")
arduino.write(command.encode())
else:
```

```
print (f"Invalid command: {command}")

except serial.SerialException:
print (f"ERROR: Could not connect to Arduino on port {SERIAL_PORT}")
print ("Check that the Arduino is plugged in and the COM port is correct.")
except Exception as e:
print (f"An error occurred: {e}")

if __name__ == "__main__":
main ()
```

V. RESULTS AND ANALYSIS

5.1 Current Status of the Project

The project is partially complete. Here is exactly where things stand.

Component	Status	Notes
Arduino code	Complete and working	Tested with serial monitor
Python script with PySerial	Complete and working	Sends commands successfully
PySerial communication	Working	Laptop talks to Arduino
DC motors (thrusters)	Working individually	Need to finalize mounting
Camera	Working	Video feed works
Depth sensor	Code written	Calibration pending
Frame assembly	Partially assembled	Need final mounting
Waterproofing	Pending final seal	Testing ongoing
Full system test	Not yet done	Waiting for final assembly

5.2 Component Functionality

What works. The Arduino reads serial commands correctly. The Python script sends F, B, L, R, U, D, and S commands. PySerial establishes the USB connection without errors. Motors spin in the correct direction for each command. The camera shows live video on the laptop.

What needs more work. The depth sensor is connected but calibration values are not final. Waterproof sealing has not been fully tested. The physical frame needs final assembly.

5.3 Code Functionality

Both the Arduino code and Python script are fully functional. I have tested them separately and together. When I run the Python script and press F, the Arduino receives F and turns on the forward thruster. The same happens for all other commands. The serial communication works at 9600 baud, which is fast enough for this application. There is no noticeable lag between pressing a key and the motor responding.

5.4 Challenges Faced

Challenge 1 was PySerial not connecting. At first, PySerial could not find the Arduino. The Arduino IDE was using the same COM port. Closing the IDE fixed the problem.

Challenge 2 was motors not responding. The L298N needs both ENA and ENB pins set to high or PWM and the direction pins set correctly. I forgot to set ENA at first. The motors did nothing. It took me a while to figure that out.

Challenge 3 was waterproofing. Silicone sealant works but it is messy. Getting a perfect seal without blocking wires or connectors is tricky. I am still working on this.

Challenge 4 was depth sensor readings being noisy. The analog reading from the depth sensor fluctuates a lot. I will need to add averaging in the Arduino code to smooth it out.

VI. CONCLUSIONS AND FUTURE SCOPE

6.1 Conclusion

So, did I succeed? Partially. Here is what I can say.

First, the software works. The Arduino code is complete. The Python script with PySerial works perfectly. The laptop talks to the Arduino over USB, and the Arduino controls the thrusters based on keyboard commands. That part is done.

Second, the hardware is partially built. The frame exists. The motors work. The camera sends video. The depth sensor is connected. I just need to finish the assembly and waterproofing.

Third, the cost target was met. Total component cost is under 80 dollars. That is less than 5 percent of what a commercial ROV would cost. Mission accomplished on the affordability front.

Fourth, the project taught me a lot. I learned Arduino programming, PySerial communication, motor control, waterproofing techniques, and how to integrate multiple subsystems into one working system. That is more valuable than any textbook.
The bottom line is that I built a working prototype of a low cost underwater ROV. It is not fully assembled yet, and I have not done final field testing, but the core systems are functional. The code runs. The motors spin. The camera shows video. The depth sensor reads values. For a first-year student project with a tiny budget, I would call that a win.

6.2 Future Scope
If I had more time or if someone wants to continue this project, here is what I would do next.
First, finish the assembly and waterproofing. This is the immediate next step. Get the frame fully built, seal everything properly, and test in an actual water tank.

Second, add wireless control. Right now, the ROV is tethered with USB and power cable. A wireless version would be more practical. Maybe Bluetooth or RF modules.

Third, improve the Python interface. Instead of just keyboard commands, I could build a GUI with buttons, a video window, and a depth display. Something like Tkinter or PyQt.

Fourth, add more sensors. Temperature sensor, leak detection sensor, and IMU for orientation.

Fifth, implement autonomous features. Simple things like hold this depth using the depth sensor and feedback control with PID.

Sixth, test in real conditions. Take it to a swimming pool, a lake, or a water tank at a treatment plant and see how it performs.

Seventh, document everything for others. Write a detailed assembly guide, share the code on GitHub, and make it easy for other students to build their own.

ACKNOWLEDGEMENT

Honestly, this project taught me more than any textbook ever could. I want to start by thanking my guide Prof. G. N. Mawale. He didn't just give me directions; he actually made me think through every problem instead of just handing me solutions. That mattered a lot.
Thanks to Dr. M. P. Gajre, Head of Department, and all the faculty in First Year Engineering Science. They let me borrow equipment, use the lab whenever I needed, and never said no when I asked for help.
Our principal Dr. P. B. Mane. His encouragement kept me going when things weren't working.

REFERENCES

[1] P. Prasanna, K. J. Dana, N. Gucunski, B. B. Basily, H. M. La, R. S. Lim, and H. Parvardeh, “Automated crack detection on concrete bridges,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 591–599, Apr. 2016.
[2] T. W. Schubert, A. D’Ausilio, and R. Canto, “Using Arduino microcontroller boards to measure response latencies,” *Behavior Research Methods*, vol. 45, no. 4, pp. 1332–1346, Dec. 2013.
[3] S. Toxopeus, M. Kerkvliet, R. Vogels, F. Quadvlieg, and B. Nienhuis, “Submarine hydrodynamics for off-design conditions,” *Journal of Ocean Engineering and Marine Energy*, vol. 8, no. 4, pp. 499–511, Nov. 2022.
[4] Arduino Official Website
[5] PySerial Documentation
[6] Blue Robotics, “BlueROV2 Specifications.”

APPENDIX

Plagiarism Report
(Attach Turnitin or Grammarly report here)
Project Title Mapping with POs and PSOs

Project Title: Low-Cost Underwater Drone (ROV) for Inspection and Monitoring

Gr. No	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
1	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓		

Justification:

PO1 Engineering Knowledge: Applied principles of electronics, programming, and fluid mechanics.

PO2 Problem Analysis: Analyzed the problem of expensive underwater inspection systems.

PO3 Design and Development: Designed the ROV frame, circuit, and control software.

PO5 Modern Tools: Used Arduino IDE, Python with PySerial, and camera modules.

PO9 Individual and Team Work: Completed the project individually.

PO10 Communication: Documented the project in this report.

PO12 Life Long Learning: Learned new skills including Arduino programming and PySerial.

PSO1 Mechanical Systems: Designed thrusters and waterproof housing.

PSO2 Software Integration: Integrated Arduino code with Python script.