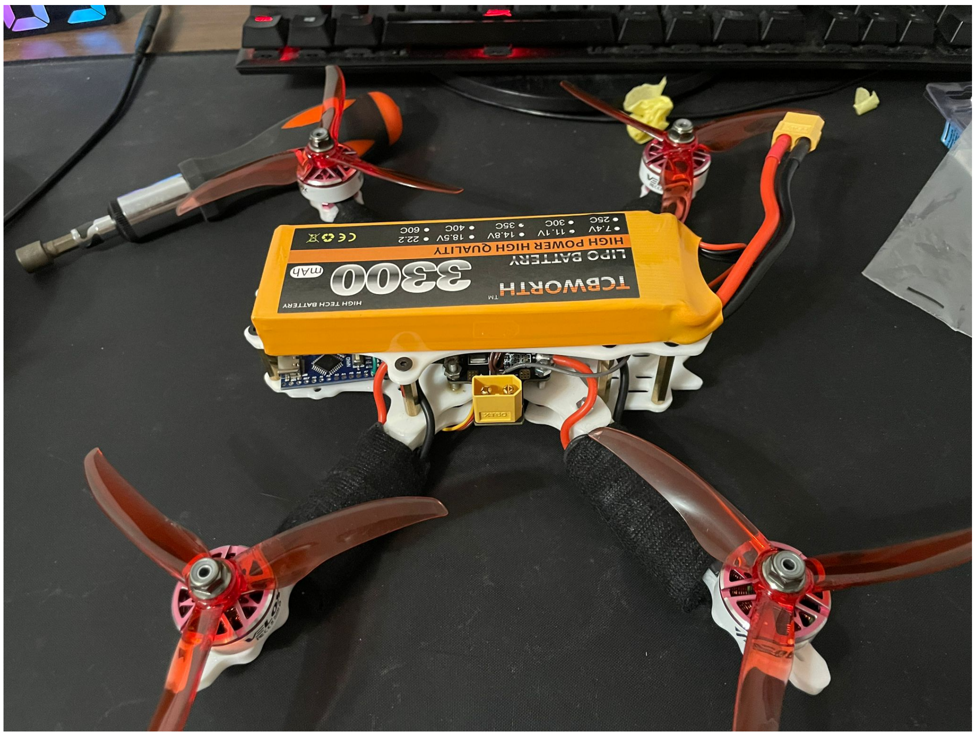
Drone

Abstract

This project presents the development of a manually built quadcopter drone equipped with basic altitude control and vertical movement capabilities. The drone is designed to ascend, descend, and maintain a stable hover at a desired height using a control algorithm implemented on an Arduino microcontroller. Built on a 5-inch frame, it combines lightweight construction with efficient lift from four brushless motors, each driven by an electronic speed controller (ESC), and is remotely controlled via Bluetooth communication from a laptop. The goal of this drone is to explore the fundamentals of drone control systems, from frame assembly to flight logic programming.



Buboacă Justin, Dincă Patrick Adrian, Pintilii Andrei-Valentin

May 2025

Contents

1 Domain Overview

1.1 Introduction

In recent years, unmanned aerial vehicles (UAVs), commonly known as drones, have gained significant popularity across various fields, including aerial imaging, agriculture, logistics, and research. This project falls within the domain of manually controlled aerial systems, with a specific focus on vertical flight control in a quadcopter configuration. The drone is built on a compact 5-inch frame and is powered by four brushless motors, each controlled via an electronic speed controller (ESC). An Arduino microcontroller is used for basic throttle control, while all flight maneuvers are performed manually through control commands sent from a laptop via Bluetooth communication.

1.2 The Importance of Drones

Drones are becoming essential tools in the efficient execution and management of tasks across a wide range of industries. Whether used in agriculture, logistics, filmmaking, disaster response, or environmental research, drones enhance access to difficult areas, reduce manual workload, and improve accuracy.

In agriculture, for instance, drones equipped with multispectral cameras help farmers monitor crop health, detect irrigation issues, and even spray fertilizers with high precision. In logistics, companies like Amazon and Zipline are developing drone delivery systems that can transport medical supplies, small packages, or food to remote areas in a matter of minutes. In filmmaking and real estate, drones offer smooth aerial shots and sweeping views that once required helicopters and expensive equipment, revolutionizing how visual content is produced. During natural disasters or search-and-rescue operations, drones can be quickly deployed to locate survivors, assess structural damage, or deliver essential supplies in areas inaccessible by ground.

By providing real-time aerial data, programmable flight control, and autonomous navigation, drones enable faster decision-making, increased safety, and smarter workflows. In large-scale operations—such as inspecting power lines, pipelines, or solar farms—drones minimize human risk while maximizing efficiency and coverage.

Moreover, drones support the rise of automation by acting as mobile sensing platforms and intelligent agents in data collection. As industries continue to adopt digital solutions, the role of drones is expanding rapidly, shaping a future where airborne systems are no longer optional tools but core components of intelligent infrastructure and service delivery.

1.3 Similar Existing Solutions

Modern aerial systems often rely on a variety of technologies that share key features with quadcopters, particularly in terms of mobility, remote control, and task automation. Some commonly used aerial or robotic platforms that perform similar functions to manually controlled drones are:

Technology	**Application**
RC helicopters	Manually controlled rotorcraft used in hobby flying;
Ground-based RC vehicles	Remote-controlled cars or robotic rovers used in STEM education, competitions, and prototyping
Indoor airships	Lightweight aerial platforms for indoor navigation, used in advertising or surveillance

Table 1: Existing similar solutions and their applications

These systems share several characteristics with drones, especially in the areas of remote operation, motion control, and the use of microcontrollers or RC transmitters. RC helicopters, for example, use comparable propulsion systems but often lack the automatic stabilization offered by quadcopters.

1.4 Our Approach

Our approach focuses on building a manually controlled quadcopter drone that prioritizes simplicity, hardware understanding, and hands-on experience. Unlike commercial drones that rely on GPS, onboard sensors, and flight controllers for automated stability and navigation, our prototype is controlled entirely by the user through a laptop via Bluetooth communication. The goal was to create an aerial system that demonstrates the basic principles of quadcopter flight, including thrust control, motor coordination, and frame assembly.

The drone is constructed on a 5-inch frame and powered by four brushless motors, each connected to an electronic speed controller (ESC). An Arduino microcontroller is used to process throttle input and distribute PWM signals to the ESCs. By excluding complex sensor systems and automated features, our design allows for a deeper understanding of the mechanical and electronic components involved in basic drone operation. This hands-on approach also reinforces essential skills in embedded systems, communication protocols, and real-time signal control.

2 Project Objectives

The main objective of this project is to design and build a manually controlled quadcopter drone that demonstrates the basic principles of flight control and embedded systems. The drone is controlled via a Bluetooth-connected Xbox controller and focuses on simplicity, modularity, and hands-on learning.

Specific objectives include:

- Construct a lightweight and compact 5-inch drone frame.
- Mount and configure four brushless motors with electronic speed controllers (ESCs).
- Program an Arduino microcontroller to process input signals and generate PWM outputs for motor control.
- Establish Bluetooth communication between the Laptop and the Arduino.
- Design a safe and efficient power supply system for the motors and onboard electronics.
- Perform flight tests to evaluate responsiveness, thrust, and stability under manual control.

These objectives aim to provide a deeper understanding of quadcopter mechanics, wireless control systems, and basic UAV functionality.

3 Implementation

3.1 General Description

The project consists of a manually controlled quadcopter drone, developed to demonstrate the basic principles of multirotor flight control, hardware integration, and real-time command execution.

Here is how the system works:

The drone is powered by a 3-cell 3300mAh LiPo battery, which supplies energy to both the flight control logic and the propulsion system. A power distribution board (PDB) allocates electrical current from the battery to four electronic speed controllers (ESCs) and to a 5V regulator module, which delivers a stable voltage supply to the Arduino Nano microcontroller.

A Bluetooth serial module (HC-05) establishes a wireless communication link between the onboard Arduino and a ground-based laptop. The user transmits manual flight commands using Tera Term, a serial terminal emulator that allows real-time communication via Bluetooth. These commands are received by the Arduino and translated into PWM (pulse-width modulation) signals.

Each of the four brushless motors is connected to a dedicated ESC. The ESCs interpret the PWM signals and adjust motor speeds accordingly, generating lift and maintaining stability. The user can increase or decrease thrust to make the drone ascend, descend, or maintain a hover at a fixed altitude.

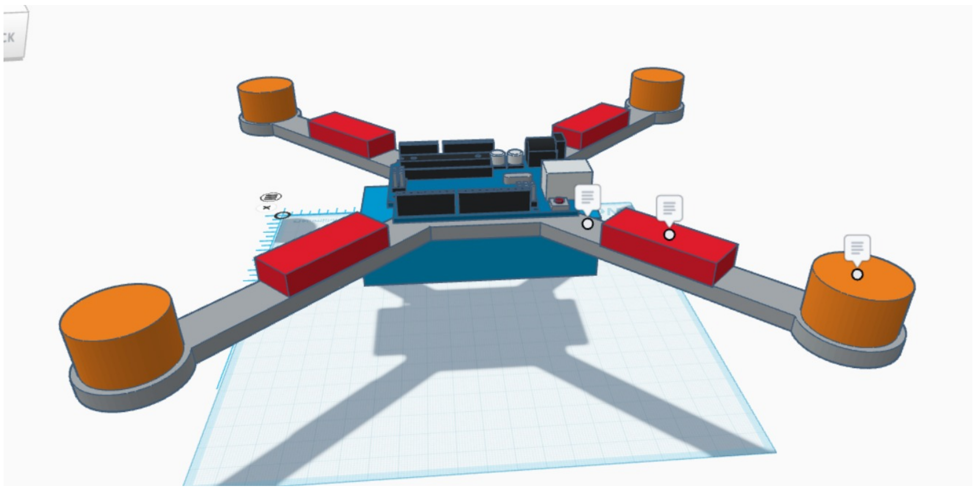
3.2 System Components

Component	**Type/Model**	**Function**
Brushless Motor	TMOTOR Velox 2550 KV	Provides lift and thrust for flight
ESC (Electronic Speed Controller)	40 A with XT60 plug	Controls motor speed by regulating power
Bluetooth Module	HC-05	Allows wireless input from Xbox controller
Power Distribution Board	XT60 PDB	Sends power from battery to ESCs and Arduino
Microcontroller	Arduino Nano	Sends PWM signals to ESCs based on user input
Propeller	T5143S Tri-blade	Converts motor rotation into upward thrust
Battery	3300 mAh 3 S TCBWORTH Li-Po	Powers motors and all onboard components
Frame	5-inch, 3D-printed ABS	Holds components and ensures structural stability

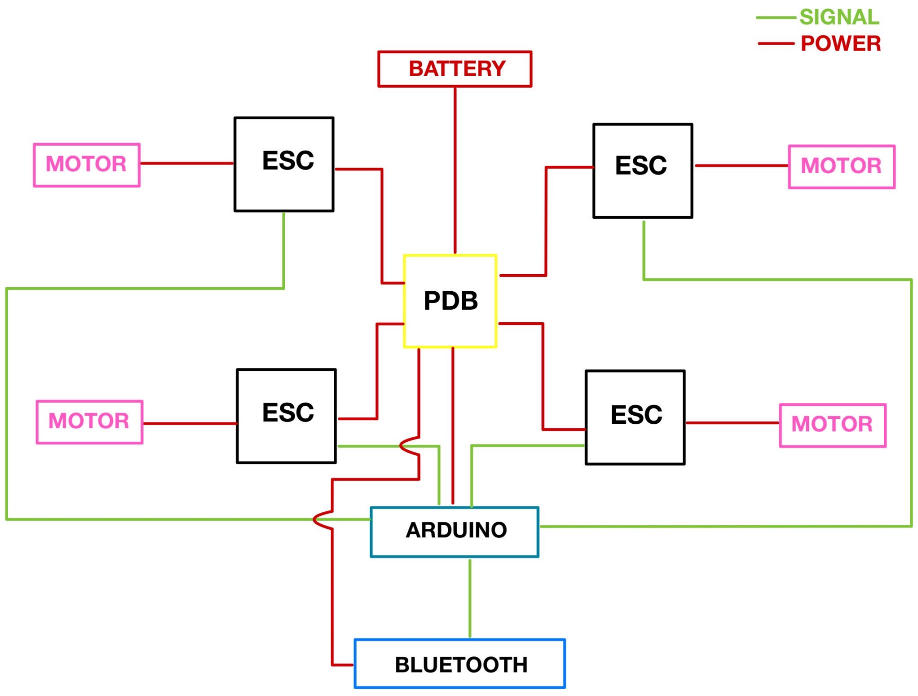
Table 2: Drone components, models, and their functions

3.3 Electrical Circuit Design

This section describes the electrical circuit design of the quadcopter drone. The system comprises several interconnected components responsible for power distribution, signal processing, and motor control to ensure stable and responsive flight. A simplified schematic of the drone's layout is presented below, followed by a detailed block diagram illustrating the flow of power and control signals throughout the system.

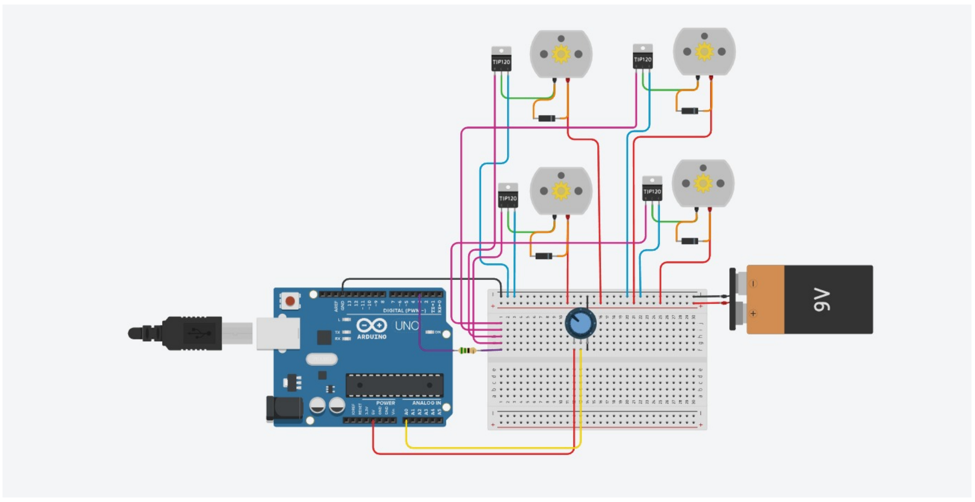


Simplified Schematic of the Drone - design stage



Block Diagram of the Drone

At the heart of the system is the Arduino Nano microcontroller, which receives flight commands from a laptop via Bluetooth communication using an HC-05 module. These commands are interpreted by the Arduino, which then generates PWM signals to control the speed of four brushless motors through their respective electronic speed controllers (ESCs). Power is supplied by a 3S (11.1V) 3300mAh LiPo battery. This battery connects to a power distribution board (PDB), which distributes raw power directly to each ESC. In the wiring diagram below, the functionality of the ESCs is represented by a simplified transistor-diode circuit, while the potentiometer simulates the role of a command input device, serving the same purpose as the Bluetooth module used in our actual drone setup.



Wiring Diagram

3.4 Technical Specifications

Parameter	Specification
Dimensions (w/o propellers)	22 cm / 18 cm / 8 cm
Dimensions (w/ propellers)	32 cm $\times$ 28 cm $\times$ 8 cm
Weight	622 g
Power Consumption	$\approx$ 275 W (hover) / $\approx$ 600 W (full-throttle)
Estimated Max Thrust	$\approx$ 1 - 1.2 kg per motor
Power Supply	TCBWORTH 3300mAh 3S 11.1V LiPo Battery
Max Flight Time	$\approx$ 6 - 9 minutes (depends on throttle usage)

Table 3: System Specifications

3.5 Programming and Microcontroller Code

The control logic of the drone is implemented on an Arduino Nano microcontroller using the Arduino programming environment. Written in C++, the code is responsible for receiving flight commands via Bluetooth, interpreting them in real time, and generating appropriate PWM signals to control the speed of each motor through the ESCs. It ensures responsive manual control over vertical motion and forms the core of the drone's flight behavior. The complete source code is presented below.

```
#include <Servo.h>
#include <SoftwareSerial.h>

// Pini pentru SoftwareSerial (Bluetooth)
#define BT_RX 11
#define BT_TX 10
SoftwareSerial BT(BT_RX, BT_TX); // RX, TX

Servo esc1; // ESC rosu
Servo esc2; // ESC rosu
Servo esc3; // ESC rosu
Servo esc4; // ESC verde

int pwm_common = 1020;
int pwm_step = 10;
int pwm_min = 1000;
int pwm_max = 2000;
int corectie_esc4 = 10;

bool motors_started = false;

void setup() {
  Serial.begin(9600);  // Pentru monitor serial (USB)
  BT.begin(9600);      // Pentru Bluetooth HC-05

  esc1.attach(3);
  esc2.attach(5);
  esc3.attach(6);
  esc4.attach(9);

  esc1.writeMicroseconds(pwm_min);
  esc2.writeMicroseconds(pwm_min);
  esc3.writeMicroseconds(pwm_min);
  esc4.writeMicroseconds(pwm_min);

  Serial.println("Sistem gata. Comenzi: s = start | + = add pwm | - =
      minus pwm | x = stop | c = compensatie");
  BT.println("Sistem gata. Comenzi: s = start | + = add pwm | - =
     minus pwm | x = stop | c = compensatie");
}

void loop() {
  if (BT.available()) {
    char c = BT.read();
    Serial.print("Tasta primit : "); Serial.println(c);

    switch (c) {
      case 's': // start
        pwm_common = 1050;
        esc1.writeMicroseconds(pwm_common);
        esc2.writeMicroseconds(pwm_common);
        esc3.writeMicroseconds(pwm_common);
        esc4.writeMicroseconds(pwm_common + corectie_esc4);
        motors_started = true;
        Serial.println("Motoare armate  i  pornite.");
        BT.println("Motoare armate  i  pornite.");
        break;

      case '+': // add pwm
        if (motors_started) {
          pwm_common += pwm_step;
          if (pwm_common > pwm_max) pwm_common = pwm_max;
          updateMotors();
          Serial.println("PWM crescut.");
          BT.println("PWM crescut.");
        }
        break;

      case '-': // minus pwm
        if (motors_started) {
          pwm_common -= pwm_step;
          if (pwm_common < pwm_min) pwm_common = pwm_min;
          updateMotors();
          Serial.println("PWM sc zut.");
          BT.println("PWM sc zut.");
        }
        break;

      case 'x': // stop
        if (motors_started) {
          Serial.println("Oprire motoare gradual ...");
          BT.println("Oprire motoare gradual ...");
          for (int val = pwm_common; val >= pwm_min; val -= pwm_step)
              {
            esc1.writeMicroseconds(val);
            esc2.writeMicroseconds(val);
            esc3.writeMicroseconds(val);
            esc4.writeMicroseconds(val + corectie_esc4);
            delay(100);
          }
          motors_started = false;
          Serial.println("Motoare oprite.");
          BT.println("Motoare oprite.");
        }
        break;

      case 'c': // compensatie
        corectie_esc4 += pwm_step;
        if ((pwm_common + corectie_esc4) > pwm_max) {
          corectie_esc4 = pwm_max - pwm_common;
        }
        updateMotors();
        Serial.println("Compensare ESC 4 crescut .");
        BT.println("Compensare ESC 4 crescut .");
        break;

      default:
        Serial.print("Comand  necunoscut : "); Serial.println(c);
        BT.print("Comand  necunoscut : "); BT.println(c);
        break;
    }
    printPWMValues();
  }
}

void updateMotors() {
  esc1.writeMicroseconds(pwm_common);
  esc2.writeMicroseconds(pwm_common);
  esc3.writeMicroseconds(pwm_common);
  esc4.writeMicroseconds(pwm_common + corectie_esc4);
}

void printPWMValues() {
  Serial.print("ESC1: "); Serial.print(pwm_common);
  Serial.print(" | ESC2: "); Serial.print(pwm_common);
  Serial.print(" | ESC3: "); Serial.print(pwm_common);
  Serial.print(" | ESC4: "); Serial.print(pwm_common + corectie_esc4)
     ;
  Serial.print(" | Compensatie ESC4: "); Serial.println(corectie_esc4
     );

  BT.print("ESC1: "); BT.print(pwm_common);
  BT.print(" | ESC2: "); BT.print(pwm_common);
  BT.print(" | ESC3: "); BT.print(pwm_common);
  BT.print(" | ESC4: "); BT.print(pwm_common + corectie_esc4);
  BT.print(" | Compensatie ESC4: "); BT.println(corectie_esc4);
}
```

4 Testing

4.1 Bluetooth Communication Test

To ensure reliable wireless control, we paired the HC-05 Bluetooth module with a phone and laptop. A serial terminal app was used to send predefined control commands (‘s‘, ‘+‘, ‘-‘, ‘x‘, and ‘c‘). Successful communication was confirmed by observing the appropriate feedback via the serial monitor and the motors responding accordingly.

4.2 Motor Arming and Starting Test

Upon powering the system, we sent the ‘s‘ command to initialize and arm the Electronic Speed Controllers (ESCs). This set all ESCs to a starting PWM of 1050 μs. The motors began spinning uniformly, indicating successful arming and proper power delivery. ESC 4 was configured with an additional compensation value to ensure uniform thrust.

4.3 PWM Increase/Decrease Test

To test motor throttle control, the ‘+‘ and ‘-‘ commands were used to increment or decrement the PWM signal in steps of 10 μs. The motors responded immediately, with their speed increasing or decreasing proportionally. Limits were enforced between 1000 μs and 2000 μs to protect the ESCs and motors from unsafe operation.

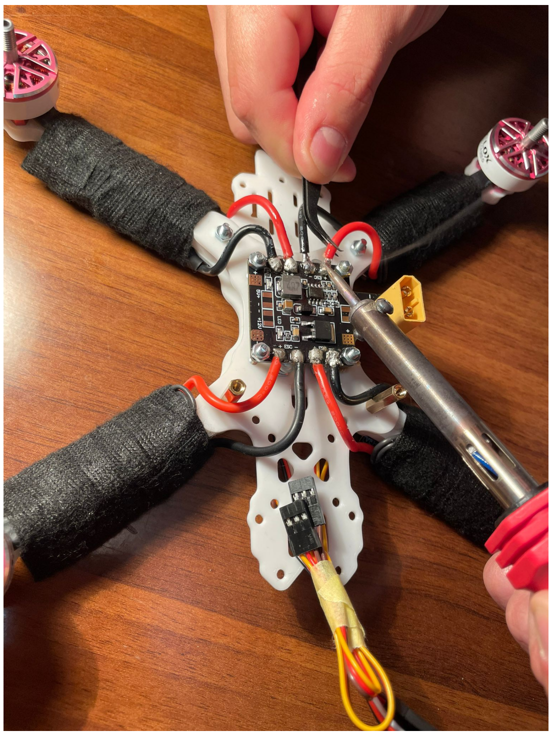
4.4 ESC4 Compensation Test

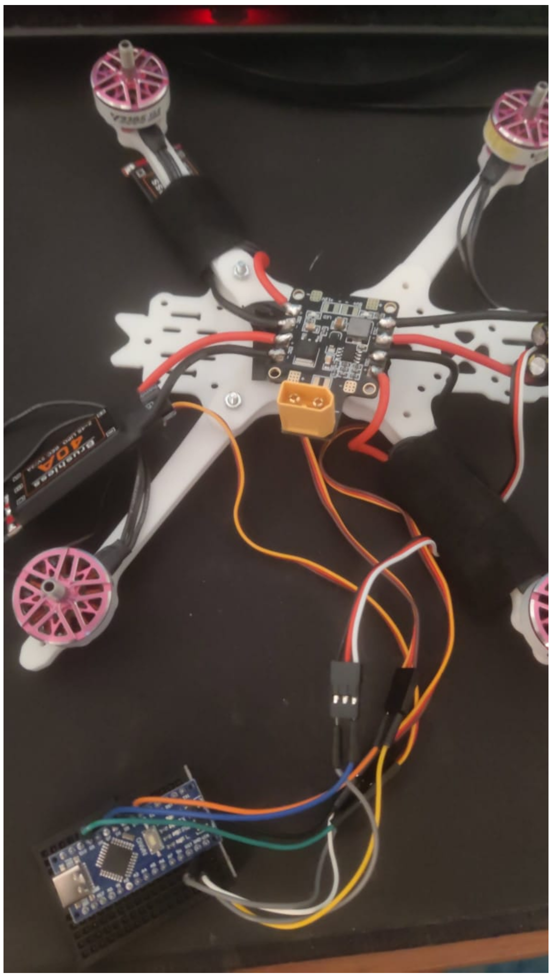
The ‘c‘ command was tested to manually increase the PWM of ESC 4, which received a small correction value to match the behavior of the other motors. This was verified through observed motor response and printed feedback over the serial interface. Compensation ensured consistent lift from all propellers.

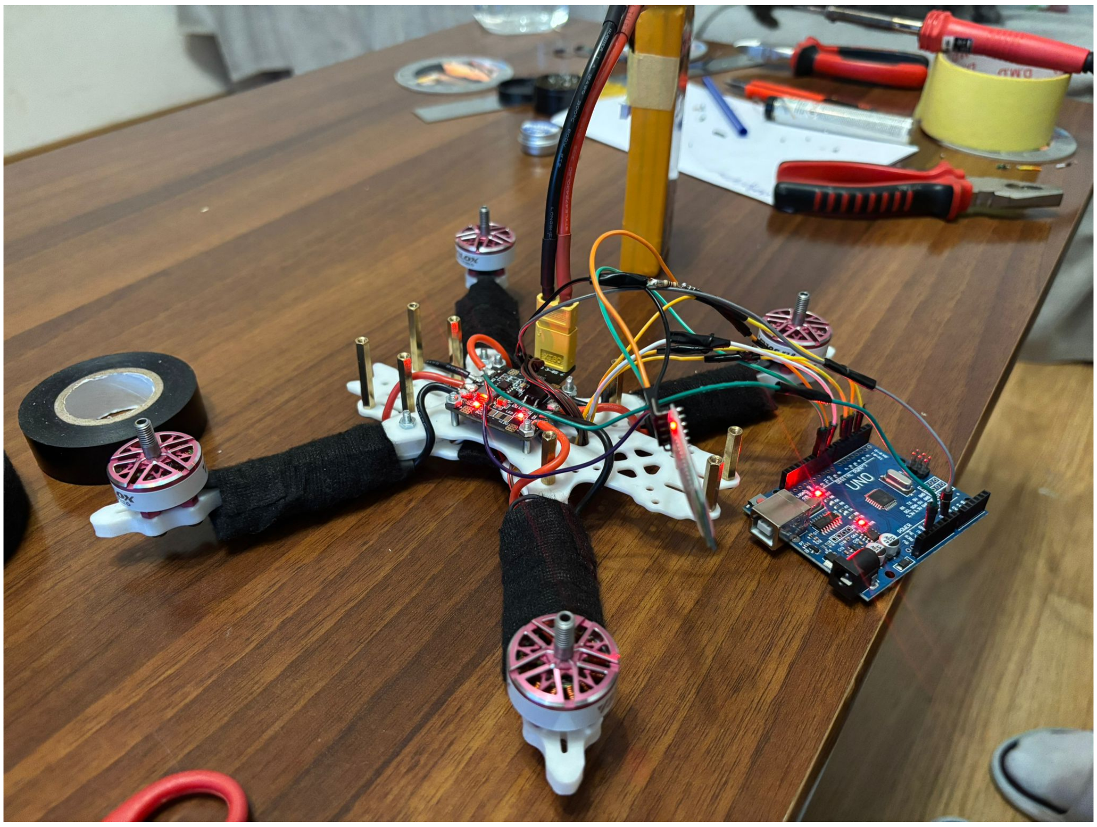
4.5 Motor Stop Test

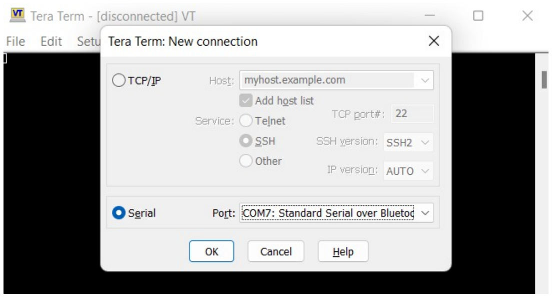
Safety and controlled shutdown were tested using the ‘x‘ command. This gradually decreased the PWM signal in steps down to the minimum (1000 μs), rather than stopping the motors abruptly. This behavior prevents sudden loss of control and mimics the disarming process in real drones.

5 Photos from the construction of the project


Wiring the PDB

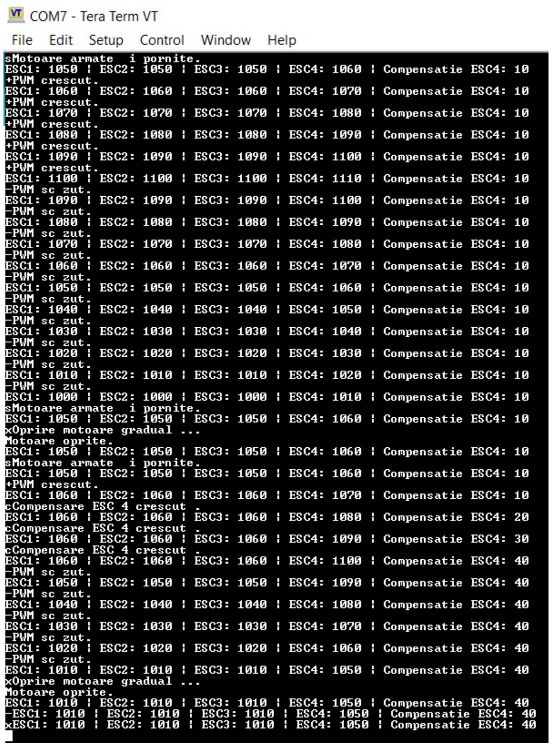

Initial test of speed control


Bluetooth connectivity test



TCP/IP Host: myhost.example.com
[✓ Add host list]
Service: Telnet TCP port#: 22
[✓ SSH] SSH version: SSH2
Other IP version: AUTO
[✓ Serial] Port: COM7: Standard Serial over Blueto

Bluetooth connection setup



```
sMotoare armate  i pornite.
ESC1: 1050 ¦ ESC2: 1050 ¦ ESC3: 1050 ¦ ESC4: 1060 ¦ Compensatie ESC4: 10
+PWM crescut.
ESC1: 1060 ¦ ESC2: 1060 ¦ ESC3: 1060 ¦ ESC4: 1070 ¦ Compensatie ESC4: 10
+PWM crescut.
ESC1: 1070 ¦ ESC2: 1070 ¦ ESC3: 1070 ¦ ESC4: 1080 ¦ Compensatie ESC4: 10
+PWM crescut.
ESC1: 1080 ¦ ESC2: 1080 ¦ ESC3: 1080 ¦ ESC4: 1090 ¦ Compensatie ESC4: 10
+PWM crescut.
ESC1: 1090 ¦ ESC2: 1090 ¦ ESC3: 1090 ¦ ESC4: 1100 ¦ Compensatie ESC4: 10
+PWM crescut.
ESC1: 1100 ¦ ESC2: 1100 ¦ ESC3: 1100 ¦ ESC4: 1110 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1090 ¦ ESC2: 1090 ¦ ESC3: 1090 ¦ ESC4: 1100 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1080 ¦ ESC2: 1080 ¦ ESC3: 1080 ¦ ESC4: 1090 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1070 ¦ ESC2: 1070 ¦ ESC3: 1070 ¦ ESC4: 1080 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1060 ¦ ESC2: 1060 ¦ ESC3: 1060 ¦ ESC4: 1070 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1050 ¦ ESC2: 1050 ¦ ESC3: 1050 ¦ ESC4: 1060 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1040 ¦ ESC2: 1040 ¦ ESC3: 1040 ¦ ESC4: 1050 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1030 ¦ ESC2: 1030 ¦ ESC3: 1030 ¦ ESC4: 1040 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1020 ¦ ESC2: 1020 ¦ ESC3: 1020 ¦ ESC4: 1030 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1010 ¦ ESC2: 1010 ¦ ESC3: 1010 ¦ ESC4: 1020 ¦ Compensatie ESC4: 10
-PWM sc zut.
ESC1: 1000 ¦ ESC2: 1000 ¦ ESC3: 1000 ¦ ESC4: 1010 ¦ Compensatie ESC4: 10
sMotoare armate  i pornite.
ESC1: 1050 ¦ ESC2: 1050 ¦ ESC3: 1050 ¦ ESC4: 1060 ¦ Compensatie ESC4: 10
xOprire motoare gradual ...
Motoare oprite.
ESC1: 1050 ¦ ESC2: 1050 ¦ ESC3: 1050 ¦ ESC4: 1060 ¦ Compensatie ESC4: 10
sMotoare armate  i pornite.
ESC1: 1050 ¦ ESC2: 1050 ¦ ESC3: 1050 ¦ ESC4: 1060 ¦ Compensatie ESC4: 10
+PWM crescut.
ESC1: 1060 ¦ ESC2: 1060 ¦ ESC3: 1060 ¦ ESC4: 1070 ¦ Compensatie ESC4: 10
cCompensare ESC 4 crescut .
ESC1: 1060 ¦ ESC2: 1060 ¦ ESC3: 1060 ¦ ESC4: 1080 ¦ Compensatie ESC4: 20
cCompensare ESC 4 crescut .
ESC1: 1060 ¦ ESC2: 1060 ¦ ESC3: 1060 ¦ ESC4: 1090 ¦ Compensatie ESC4: 30
cCompensare ESC 4 crescut .
ESC1: 1060 ¦ ESC2: 1060 ¦ ESC3: 1060 ¦ ESC4: 1100 ¦ Compensatie ESC4: 40
-PWM sc zut.
ESC1: 1050 ¦ ESC2: 1050 ¦ ESC3: 1050 ¦ ESC4: 1090 ¦ Compensatie ESC4: 40
-PWM sc zut.
ESC1: 1040 ¦ ESC2: 1040 ¦ ESC3: 1040 ¦ ESC4: 1080 ¦ Compensatie ESC4: 40
-PWM sc zut.
ESC1: 1030 ¦ ESC2: 1030 ¦ ESC3: 1030 ¦ ESC4: 1070 ¦ Compensatie ESC4: 40
-PWM sc zut.
ESC1: 1020 ¦ ESC2: 1020 ¦ ESC3: 1020 ¦ ESC4: 1060 ¦ Compensatie ESC4: 40
-PWM sc zut.
ESC1: 1010 ¦ ESC2: 1010 ¦ ESC3: 1010 ¦ ESC4: 1050 ¦ Compensatie ESC4: 40
xOprire motoare gradual ...
Motoare oprite.
ESC1: 1010 ¦ ESC2: 1010 ¦ ESC3: 1010 ¦ ESC4: 1050 ¦ Compensatie ESC4: 40
-ESC1: 1010 ¦ ESC2: 1010 ¦ ESC3: 1010 ¦ ESC4: 1050 ¦ Compensatie ESC4: 40
xESC1: 1010 ¦ ESC2: 1010 ¦ ESC3: 1010 ¦ ESC4: 1050 ¦ Compensatie ESC4: 40
```

Live PWM testing feedback

6 Problems Encountered

During the development of our Drone, we have encountered in our journey 3 major challenges:

- **Different ESCs:** to properly build the drone, we decided to replace the ESC wires by soldering the motor wires directly onto the ESCs. During this process, we had to remove the insulating tape to access the solder points. That's when we discovered that one of the ESCs was different from the other three, which were identical. The real issue became apparent during testing: the different ESC behaved inconsistently because it had brake mode enabled by default. Since we had no way to disable the brake mode or access the ESC's firmware, we compensated by adding a delay in the code for the motor connected to the different ESC. This helped ensure consistent motor behavior across all four ESCs.
- **Undocumented PWM mapping:** Another challenge we faced was the lack of documentation indicating what PWM values correspond to specific motor speeds. Since there was no reference table or clear guidance, we had to approximate the necessary PWM values through trial and error in order to achieve the desired RPM. This made the calibration process slower and less precise.
- **Dead Arduino Nano:** while making test, at some point of our journey, our arduino happened to stop working properly, overheating and also not accepting uploading any other code. We think it got entirely broke and decided to change it.

7 Potential Improvements

- **Add a gyroscope and accelerometer (IMU)**
  Integrating a sensor like the MPU6050 allows real-time stabilization and orientation tracking, which is essential for smooth FPV (First-Person View) flight, where rapid, jerky motion can disorient the pilot.
- **Upgrade the frame material**
  Switching to a lightweight carbon fiber or composite frame not only improves durability and flight efficiency but also provides a stable platform for mounting an FPV camera system without excessive vibration.
- **Implement PID control for motor balancing**
  A PID algorithm enables smooth, responsive flight, which is crucial for FPV piloting, where visual stability translates directly into control comfort and video clarity.
- **Obstacle avoidance system**
  Add ultrasonic or time-of-flight sensors to detect and avoid obstacles. This helps protect the drone and FPV camera during close-quarters navigation, such as through trees or structures.
- **Switch to joystick or gamepad input**
  Replace keyboard input with a gamepad or RC-style controller connected to the laptop for more analog control, which is critical for nuanced FPV maneuvers.

8 Conclusion

The project resulted in a functional, manually controlled quadcopter drone capable of stable vertical flight. Using an Arduino Nano, Bluetooth communication via Tera Term, and PWM motor control, the system demonstrated essential drone mechanics.

8.1 Individual Contributions

- Buboacă Justin: Primarily responsible for the project documentation and for modifying the 3D printed frame to fit the components.
- Dincă Patrick Adrian: Led the wiring and hardware assembly, ensuring all electronic components were properly connected and integrated.
- Pintilii Andrei-Valentin: Focused on programming the control code and performing the system testing to validate functionality.
- All team members participated equally in purchasing the drone components and collaborated together on coding tasks.

8.2 Bill of Materials

#	Item	Price (USD)	Qty	Supplier/Link
1	Brushless Motor – TMOTOR Velox 2550KV	15.94	4	AliExpress
2	40A ESC with XT60 plug	8.26	4	AliExpress
3	T5143S Tri-blade Propeller 2pcs	3.41	2	AliExpress
4	Arduino Nano	6.00	1	Optimus Digital
5	HC-05 Bluetooth Module	3.00	1	AliExpress
6	XT60 Power Distribution Board	4.90	1	AliExpress
7	3300mAh 3S TCBWORTH LiPo Battery	48.74	1	AliExpress
8	M3 Pressnuts	0.30	6	FixKit
9	M3 × 12mm Steel Screws	0.15	6	FixKit
10	M3 × 16mm Steel Screws	0.17	2	FixKit
11	M3 × 6mm Steel Screws	0.10	10	FixKit
12	M3 × 8mm Steel Screws	0.12	20	FixKit
13	M3 × 20mm Standoffs	0.20	4	Optimus Digital
14	M3 × 30mm Standoffs	0.35	4	Optimus Digital
15	Fabric Tape – TESA 51608	3.90	1	Dedeman
16	WU TANK MAX - 5 and 6 inch freestyle frame	6.00	1	Printables

Table 4: Final Bill of Materials with prices, quantities, and supplier links

Total Price: 184.80$