**Nawajyoti Boarding School (NJBS)**  
**ICT Club**

**Project Documentation**  
**“Building a Drone from Scratch – ICT Club Initiative”**

(A Documentation Submitted by the ICT Club)

Prepared by: **ICT Club, NJBS**

Nawa Jyoti Boarding School  
[Tilottama-08, Rupandehi]

# Acknowledgment

We would like to express our sincere gratitude to **Mr. Rajendra Pantha**, our Project Supervisor and Guide, for his invaluable guidance, encouragement, and continuous support throughout the development of this project. His expertise and mentorship have been vital to our learning and success.

We are equally thankful to our respected Principal, **Mr. Top Bahadur KC** and **Nawajyoti English Boarding School**, for providing us with the opportunity, resources, and motivation to carry out this project under the ICT Club of NJBS.

Finally, we extend our gratitude to all teachers, friends, and well-wishers who directly or indirectly contributed to the success of this project.

……………………..

Pramish Bhusal

**President**

(ICT Club of NJBS)

# Project Overview

## Introduction

Drones, also known as Unmanned Aerial Vehicles (UAVs), are aircraft that operate without a human pilot onboard. They are controlled remotely or can fly autonomously using pre-programmed flight paths guided by sensors and GPS. Drones have revolutionized various fields, including agriculture, surveillance, photography, delivery services, disaster management, and scientific research, due to their ability to access hard-to-reach areas and capture precise data efficiently.

Our drone project was chosen to explore and understand the potential of modern technology in practical applications. With the increasing importance of automation, robotics, and aerial technology in both academic and real-world scenarios, building a drone offers hands-on experience in electronics, programming, and aerodynamics. This project also encourages innovation and problem-solving skills among students.

## Objectives of the Project:

1. To design and build a functional quadcopter drone capable of stable flight.
2. To understand the principles of flight dynamics, control systems, and motor calibration.
3. To integrate sensors, such as gyroscopes and accelerometers, for balance and navigation.
4. To explore applications of drones in areas like surveillance, data collection, and environmental monitoring.
5. To provide practical experience in electronics, programming, and teamwork in a project-based learning environment.

This project not only helps in learning modern technology but also demonstrates the practical applications of drones in solving real-world problems.

# Team members and Roles

 **Rajendra Pantha** – Project Supervisor & Guide

 **Pramish Bhusal** – Project Coordinator

 **Sangam Kunwar** – Hardware Assembler

 **Abishek Dhakal** – Hardware Assembler

 **Sugam Bhusal** – Support and Documentation

 **Madhu Kunwar** – Support and Documentation

 **Puspa Bhattrai** – Documentation Assistant

 **Sonakshi Ranpal** – Documentation Assistant

 **Nischal Dhakal** – Documentation Assistant

 **Kishor Poudel** – Support Assistant

* **Sagar Ale Magar** - Support Assistant

# Material used and component

## Software

* **Arduino IDE –** Used for writing, compiling, and uploading code to the Arduino micro controller.

## Hardware Components

### A2212/13T BLDC Motors (x4)

* + **Function**: Provides thrust for flight.
  + **Quantity**: 4
  + **Price**: 1000 × 4 = 4000

### Arduino Uno R3

* + **Function**: Main flight controller that processes sensor data and controls the motors.
  + **Quantity**: 1
  + **Price**: 3000

### 30A ESC (Electronic Speed Controller) (x4)

* + **Function:** Regulates the speed and power of the BLDC motors.
  + **Quantity:** 4
  + **Price:** 500 × 4 = 2000

### MPU-6050 (Accelerometer + Gyroscope Sensor)

* + **Function:** Provides motion and orientation data for stability.
  + **Quantity:** 1
  + **Price:** 500

### 4-Channel RC Transmitter and Receiver

* + **Function:** Allows manual remote control of the drone.
  + **Quantity:** 1 set
  + **Price:** 500

### 11.1V 2200mAh 35C Li-Po Battery

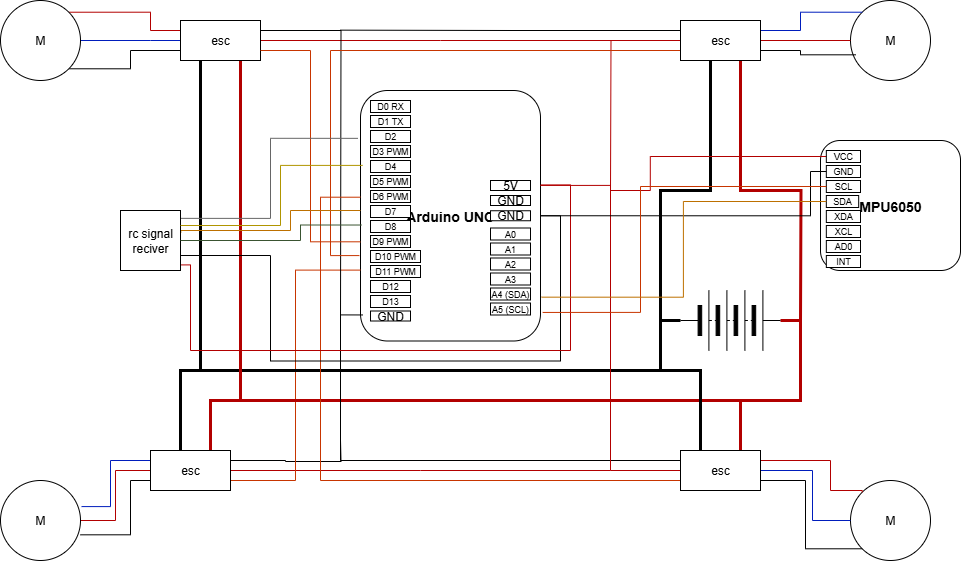
* + **Function:** Power supply for the drone.
  + **Quantity:** 1
  + **Price:** 4000

### **Total Cost**

4000 (Motors) + 3000 (Arduino) + 2000 (ESCs) + 500 (MPU6050) + 500 (RC set) + 4000 (Battery)

**Total** = 14,000

# Circuit Diagram

****

## Explanation of the Circuit Diagram

The diagram represents the wiring and control system of a Quadcopter Drone using an Arduino UNO, an MPU6050 sensor, ESCs (Electronic Speed Controllers), BLDC motors, and an RC receiver.

## Main Components and Connections

### **Arduino UNO**

* Acts as the central flight controller.
* Receives input signals from the RC receiver and sensor data from the MPU6050.
* Processes this data and generates PWM signals to control the ESCs and motors.

**MPU6050 (Gyroscope + Accelerometer)**

* Connected to Arduino via I2C communication:
* SDA → A4 (Arduino)
* SCL → A5 (Arduino)
* Provides orientation, tilt, and acceleration data for stabilization.
* VCC → 5V, GND → GND.

**RC Signal Receiver**

* Provides user input (throttle, pitch, roll, yaw).
* Connected to Arduino digital pins (e.g., D2–D7).
* Allows manual control of the drone.

**ESCs (Electronic Speed Controllers)**

* Four ESCs, each connected to one motor.
* ESC input signals are controlled by Arduino PWM pins (D9, D10, D11, D6).
* ESCs are powered directly from the battery (positive and negative).
* Each ESC powers one BLDC motor.

**Motors (BLDC Motors)**

* Four motors (M1, M2, M3, M4) fixed on the quad-copter frame.
* Controlled via ESCs to adjust speed and maintain balance.

**Battery Pack**

* Provides power to the ESCs and Arduino.
* Positive terminal → ESCs + Arduino VIN.
* Negative terminal → ESCs and Arduino GND.

**Assembly Process for the Drone**

**Frame Setup**

* Mount the quad-copter frame securely.
* Fix the four BLDC motors on each arm of the frame.

**Motor and ESC Connection**

* Connect each BLDC motor to its corresponding ESC (3-phase wires).
* Mount the ESCs on the frame close to the motors.

**Power Wiring**

* Connect the battery’s positive and negative terminals to the ESCs (via a power distribution board if available).
* Connect ESC power lines (red = +, black = –) to the Arduino GND and VIN for common grounding.

**Arduino Wiring**

* Place the Arduino UNO at the center of the frame.
* Connect:
  + ESC signal pins → Arduino PWM pins (D6, D9, D10, D11).
  + RC receiver channels → Arduino digital pins (D2–D7).
  + MPU6050 SDA → A4, SCL → A5.
  + MPU6050 VCC → 5V, GND → GND.

**Sensor and Receiver Setup**

* Secure the MPU6050 on the frame’s center for accurate measurement.
* Fix the RC receiver to the frame and connect its channels properly.

**Final Assembly**

* Ensure all wiring is neat and insulated to prevent short circuits.
* Attach propellers to motors (make sure clockwise and counter-clockwise props are placed correctly).
* Fix the battery with a strap at the drone’s center of gravity.

**Testing and Calibration**

* Upload Arduino flight control code.
* Calibrate the MPU6050 sensor and ESCs.
* Test motor directions (two should spin clockwise, two counter-clockwise).
* Perform a hover test in a safe, open area.

# Code explained/psudo code

## Explanation

This Arduino code controls a quadcopter (drone) using an MPU6050 gyro/accelerometer sensor for stabilization and a radio controller for user input. The code combines RC signal processing, sensor fusion for angle estimation, altitude control, and motor mixing to achieve stable flight.

### **Key Components**

#### Libraries and Pin Definitions

* Uses Wire.h for I2C, MPU6050.h for the gyro/accelerometer, and Servo.h for controlling ESCs (Electronic Speed Controllers).
* Defines pins for radio channels and motors.

#### Motor and Servo Setup

* Declares 4 servo objects for motors (m1–m4).
* Assigns digital pins to each motor.

#### MPU6050 Initialization

* Sets up the gyro/accelerometer for reading pitch and roll.

#### RC Input Processing

* Uses hardware interrupts to accurately measure PWM signals from the radio controller (throttle, yaw, pitch, roll).
* Stores the incoming values in rcValue[].

#### Main Control Loop

* Reads sensor data (gyro and accelerometer).
* Calculates pitch and roll using a complementary filter (combines gyro and accelerometer for better accuracy).
* Reads RC commands for movement (thrust up/down, yaw, pitch, roll).
* Smooths thrust changes to avoid sudden jolts.
* Implements a simple PID controller for altitude control using vertical acceleration (Z-axis).
* Calculates corrections for each motor based on current and target angles.
* Balances the motors to compensate for drift.
* Outputs calculated speeds to the motors.

#### Flight Control Logic

* User commands (from RC transmitter) are interpreted as movement or yaw commands.
* Altitude is controlled using the Z-acceleration from the MPU6050.
* Pitch and roll are stabilized automatically using sensor readings.
* Motor speeds are adjusted accordingly to keep the drone level and respond to user commands.

### **Pseudocode**

START

INITIALIZE serial, I2C, MPU6050 sensor

ATTACH four motors using Servo library

ARM ESCs by sending minimum PWM signals

ATTACH interrupts for four RC receiver pins

// Calibrate accelerometer Z offset

FOR 400 samples:

READ accelerometer Z (az)

ACCUMULATE az into sum

SET accZ\_offset = average of accumulated az

// Tiny startup lift (soft start for motors)

FOR 0.2 seconds:

SET all motors to low thrust (startupThrust)

WAIT 20ms

LOOP FOREVER:

now = current time in microseconds

dt = time difference from last loop

// Read RC button states from rcValue array

rclk = right button pressed?

lclk = left button pressed?

fclk = forward button pressed?

bclk = backward button pressed?

// Read sensor values

READ accelerometer (ax, ay, az) and gyro (gx, gy, gz)

// Calculate orientation (pitch, roll) using complementary filter

accPitch = angle from ax, ay, az

accRoll = angle from ay, ax, az

gyroPitch += gyro Y \* dt

gyroRoll -= gyro X \* dt

pitch = 0.98 \* gyroPitch + 0.02 \* accPitch

roll = 0.98 \* gyroRoll + 0.02 \* accRoll

// Altitude (vertical acceleration)

accZ = (az - Z offset) \* gain

accZ\_filt = filtered accZ

// Interpret RC commands

IF left+right pressed:

targetThrust = descend

ELSE IF forward+back pressed:

targetThrust = ascend

ELSE IF forward+right pressed:

yaw = positive

ELSE IF back+left pressed:

yaw = negative

ELSE IF forward pressed:

targetPitch = forward tilt

ELSE IF back pressed:

targetPitch = backward tilt

ELSE IF right pressed:

targetRoll = right tilt

ELSE IF left pressed:

targetRoll = left tilt

// Smoothly transition to target thrust

IF currentThrust < targetThrust:

currentThrust += thrustStep

ELSE IF currentThrust > targetThrust:

currentThrust -= thrustStep

// Altitude PID controller (only if thrust command)

IF targetThrust != 0:

accZ\_error = (direction) - accZ\_filt/scale

accZ\_integral += accZ\_error \* dt

accZ\_out = Kp \* error + Ki \* integral

LIMIT accZ\_out to max/min

ELSE:

RESET accZ\_integral

thrust = currentThrust + accZ\_out

// Calculate base motor speeds

base1..base4 = mid + thrust

// Apply yaw mixing

base1 += yaw; base4 += yaw

base2 -= yaw; base3 -= yaw

// Apply pitch and roll corrections

pitchCorrection = Kp \* (pitch - targetPitch)

rollCorrection = Kp \* (roll - targetRoll)

sp1 = LIMIT(base1 - pitchCorrection + rollCorrection)

sp2 = LIMIT(base2 - pitchCorrection - rollCorrection)

sp3 = LIMIT(base3 + pitchCorrection + rollCorrection)

sp4 = LIMIT(base4 + pitchCorrection - rollCorrection)

// In-flight drift compensation

pitchDrift = pitch - targetPitch

rollDrift = roll - targetRoll

sp1 -= pitchDrift \* driftGain - rollDrift \* driftGain

sp2 -= pitchDrift \* driftGain + rollDrift \* driftGain

sp3 += pitchDrift \* driftGain - rollDrift \* driftGain

sp4 += pitchDrift \* driftGain + rollDrift \* driftGain

// Update motors at fixed interval

IF enough time has passed:

SEND sp1..sp4 as PWM to motors

END LOOP

### **Summary:**

This code enables a drone to interpret radio commands, self-level using IMU (gyro/accel) data, control altitude, and adjust individual motor outputs for stable and responsive flight. The pseudocode outlines the main logic flow for your report. Let me know if you need a diagram or more details!