Implementation and Customization of a YMFC-AL Arduino Quadcopter Based on Joop Brokking's Code • Built and Tuned by :

- Safaet Islam Pranto
- Junaid Bagdady
- Rabiul Awal Sahan
- Fida Zaman

Department of Electrical and Electronic Engineering
Mymensingh Engineering College, Bangladesh

Acknowledgments

I gratefully acknowledge Joop Brokking for developing the original YMFC-AL flight-controller code and tutorials, which formed the foundation of this work. Thanks also to the Mechatronics & Embedded System Lab (MESL) at Mymensingh Engineering College for providing workspace, components, and technical guidance.

Table of Contents

1. Abstract

This document details the design, assembly, firmware adaptation, calibration, and flight testing of a quadcopter built on the YMFC-AL (Your Multicopter Flight Controller–Auto Level) framework by Joop Brokking. Important elements include an Arduino Uno, an MPU-6050 IMU for attitude sensing, four 30 A ESCs with 10" propellers on an F450 frame, and a 3S 11.1 V 1100 mAh LiPo battery. Key outcomes are a stable hover, documented PID tuning values, and a record of real-world troubleshooting (motor orientation, IMU offset, center of gravity). Future enhancements are outlined (GPS waypoint navigation, failsafe Return-to-Home, PC route control). This concise report is suitable for inclusion in a CV or portfolio to showcase practical skills in embedded control systems and UAV assembly.

2. Introduction

Background:

Quadcopters rely on a flight controller that fuses inertial data (gyroscope + accelerometer) and runs a PID control loop to maintain level flight. Joop Brokking's YMFC-AL code is a popular open-source Arduino-based controller that implements these algorithms using an MPU-6050 IMU.

Project Rationale:

While tutorials exist, each hardware build presents unique challenges in wiring, calibration, and tuning. The aim of this project was to:

- Reproduce YMFC-AL functionality on local hardware
- Document every step (assembly, wiring, code adaptation)
- Systematically troubleshoot and solve real-world issues
- Achieve stable, flyable performance
- Lay out a clear roadmap for advanced features (GPS, failsafe, PC control)

3. Objectives

Implement & Understand YMFC-AL:

- Flash and run Joop Brokking's core code on an Arduino Uno
- Study IMU readout, complementary-filter attitude estimation, and PID loops

Assemble & Wire Hardware:

- Integrate Arduino Uno, MPU-6050, ESCs, brushless motors, and power distribution on an F450 frame
- Verify correct motor rotation and balanced propeller installation

Calibrate & Tune:

- Calibrate MPU-6050 gyro offsets on a level surface
- Calibrate ESC throttle range (1000–2000 µs) for consistent motor response
- Perform iterative PID tuning (Kp, Ki, Kd) to minimize oscillations and drift

Document Troubleshooting:

- Record all challenges (motor orientation errors, IMU offsets, CoG imbalance)
- Explain the root cause for each and how they were fixed

Conduct Flight Tests:

- Bench testing of motors and ESCs
- Tethered hover to refine PID gains
- Free flight to evaluate stability, drift, and flight time

Outline Future Enhancements:

- GPS-based waypoint navigation and Return-to-Home (RTH)
- PC-based route control (telemetry link, ground station)
- Hardware upgrades (barometer, magnetometer, faster MCU)

---

4. Materials & Bill of Materials

Item	Qty	Specification / Notes
Arduino Uno	1	ATmega328P, 16 MHz
MPU-6050 (IMU)	1	6-axis gyro + accel breakout (I^2C interface)
ESCs (30 A)	4	BLHeli-compatible, 3S LiPo, 5 V BEC
Brushless Motors (1000 KV)	4	10” propeller compatible (outrunner)
Propellers (10”: 2 CW + 2 CCW)	4	Nylon or Carbon Fiber
Frame: F450 Quadcopter	1	450 mm diagonal, built-in PDB
LiPo Battery (3S,11.1 V, 1100 mAh, 20 C)	1	XT60 connector
LiPo Charger (3S Balanced)	1	AC/DC powered, with balance lead input
RC Transmitter & Receiver	1	FlySky FS-i6 (6 channels) (optional for manual control)
Jumper Wires	-	Male-to-male & female-to-female (22 AWG)
XT60 Connectors	2	Male + Female for battery-to-PDB connection
Rubber/Foam Damping Pads	-	For isolating IMU from frame vibrations
Zip Ties & Heat Shrink	-	For cable management and insulating solder joints

Battery Strap	1	Nylon strap with buckle
Screws	-	M3 hardware to secure Arduino & IMU

Estimated Cost: Approximately **5600 BDT** (excluding FlySky transmitter and receiver). Actual cost may vary depending on local prices and component availability.

---

5. Hardware Assembly

5.1. Frame & Motor Placement

F450 Frame Setup

- Attach four carbon-fiber arms to the bottom plate (with integrated PDB) using M3 screws.
- Secure the top plate via nylon standoffs. Ensure the PDB is accessible for soldering.

Motor Mounting & Orientation

- Front-Left (Motor 1): Mount a CW motor; wiring phase order set accordingly.
- Front-Right (Motor 2): Mount a CCW motor.
- Rear-Right (Motor 3): Mount a CW motor.
- Rear-Left (Motor 4): Mount a CCW motor.
- Use M3×10 mm screws to secure motors. If spin direction is wrong after ESC calibration, swap any two of the three bullet wires.

Landing Gear (Optional)

- Attach landing legs to the bottom plate to ensure ≥ 30 mm prop-to-ground clearance.

5.2. Electrical Wiring

ESC Power Wiring

- Solder each ESC’s red (+)'état

System: and black (–) leads to the PDB’s corresponding rails.

- Insulate solder joints with heat-shrink tubing. Confirm polarity carefully.

ESC → Arduino (PWM) Connections

- ESC 1 (Front-Left) → D7
- ESC 2 (Front-Right) → D4
- ESC 3 (Rear-Right) → D5
- ESC 4 (Rear-Left) → D6
- Tie all ESC grounds together on the PDB ground rail, then run a single GND wire to Arduino GND.

MPU-6050 → Arduino (I²C)

- Connect VCC → 5 V (module's onboard regulator handles 5 V→3.3 V).
- Connect GND → GND.
- SDA → A4, SCL → A5.
- Use a small damping pad underneath (see 5.3).

Battery Connection

- Plug the 3S LiPo's XT60 male into the PDB's XT60 female. Leave battery unplugged until all wiring is complete.

(Optional) RC Receiver Wiring

- Use interrupt-capable pins (e.g., D8, D9, D10, D11) to read PWM channels for Throttle, Roll, Pitch, Yaw.
- Tie receiver ground to PDB ground to maintain a common reference.

5.3. IMU Mounting & Vibration Damping

- Pad Material: Use 3 mm–5 mm dense rubber or foam.
- Placement: Centered on the top plate to coincide with the frame's center of gravity.
- Attachment: Double-sided tape or a small dab of hot glue; avoid rigid mounting.
- Alignment: Ensure MPU axes align:
  - "X" axis → left side
  - "Y" axis → forward
  - "Z" axis → downward

5.4. Battery Placement & Balance

- Position: Strap the LiPo battery lengthwise above the PDB at the geometric center.
- Balancing:
  - Place the assembled quadcopter (battery attached) on a cylindrical pivot (e.g., a pencil).
  - Adjust battery position until the craft remains level fore-aft and left-right.
  - Confirm that on a flat table (powered on, no motors spinning), the IMU reports roll/pitch ≈ 0° (see Section 7.1).

5.5 Circuit Diagram

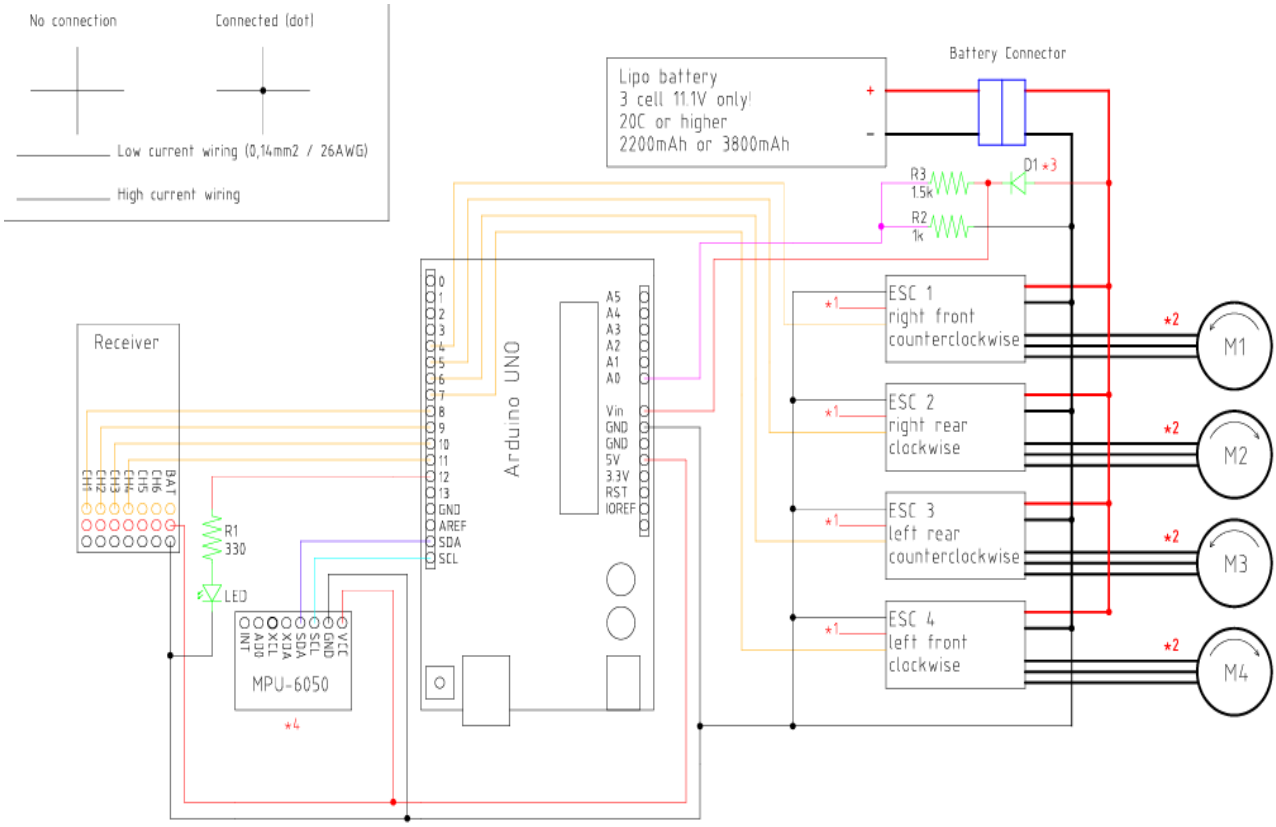

Figure 1: Circuit diagram of the drone

---

6. Software Overview

Arduino IDE Setup

- Install Arduino IDE (≥ 1.8.x).
- Add libraries:
  - I2Cdev & MPU6050 (Jeff Rowberg's I²Cdevlib)
  - Servo (built into Arduino IDE for ESC PWM)

Core Flight Code

- Based on Joop Brokking's YMFC-AL project — visit his website for details: http://www.brokking.net/ymfc-al_downloads.html

  Key modules:
  - IMU Initialization & Calibration (MPU-6050 offsets)
  - Complementary Filter (fuses gyro + accel to compute roll/pitch angles)
  - PID Controllers (for roll & pitch stabilization)
  - Motor Mixer (translates PID outputs into individual ESC signals)

Custom Adaptations

- Added serial debug prints for IMU angles and PID values (for tuning).
- Trimmed code to disable yaw control (deferred to future work).
- Removed on-screen calibration prompts for faster startup.

Code Repository Link

A cleaned-up version of the code is hosted at:
https://github.com/Safaet-islam/ymfc-al-arduino-quadcopter

---

7. Calibration & PID Tuning

7.1. IMU Zero-Offset Calibration

- Purpose: Remove gyroscope bias so that stationary IMU reads 0 °/s.
- Procedure:
  - Place the quadcopter (no props spinning) on a perfectly level, vibration-free surface.
  - Run a calibration routine that collects 2000 gyro samples at ~500 Hz.
  - Compute average raw gyro readings on X, Y (and Z) axes → convert to °/s offsets.
  - Subtract these offsets in subsequent measurements.
- Outcome: After calibration, on a level table, roll and pitch angles read within ± 0.5 ° of zero.

7.2. ESC Throttle Calibration

- Purpose: Ensure each ESC recognizes the same minimum and maximum PWM range.
- Procedure:
  - Props removed; power on Arduino + ESCs.
  - Send 2000 µs pulse (max throttle) for ~2 s → ESC beeps indicating “max.”
  - Send 1100 µs pulse (min throttle) for ~2 s → ESC beeps indicating “min.”
  - ESCs store these values and use them as endpoints.
- Outcome: All four ESCs now respond consistently to throttle commands between 1100 µs and 2000 µs.

7.3. PID Tuning Process

- Tethered Setup:

    - Attach a harness or have a helper hold the drone so it cannot lift off. Install props.
    - Power on, confirm IMU and ESC calibration complete.
- Roll Axis (Pitch is analogous):
    - Set Ki = 0, Kd = 0.
    - Increase Kp from 1.0 upward in steps of 0.5 until oscillations appear around the 0° setpoint.
    - Back off Kp by ~ 10% from the oscillation threshold.
    - Introduce Ki = 0.05; observe for steady-state error correction. Increase if small bias remains.
    - Add Kd = 0.5 to dampen residual oscillations. Increase slowly until response is crisp but not overly sluggish.
    - Final example: Kp_roll = 4.5, Ki_roll = 0.10, Kd_roll = 2.00.
- Pitch Axis:
    - Repeat the above procedure using Kp_pitch, Ki_pitch, Kd_pitch.
    - Final example: Kp_pitch = 4.5, Ki_pitch = 0.10, Kd_pitch = 2.00.
- Validation:
    - With tuned gains, a 50 % throttle hover should hold level within ± 2 ° roll/pitch.
    - Minor oscillations under wind gusts should damp out in < 1 s.

---

8. Flight Testing & Results

8.1. Bench Tests

- Objective: Verify motor/ESC response without props.
- Procedure:
    - Power on system (props off). painless
    - Run a simple sketch that sweeps throttle from 1100 µs → 1900 µs in 200 µs increments every 2 s.
    - Observe each motor individually; confirm smooth spool-up/spool-down and correct spin direction.
- Result:
    - All motors responded smoothly.
    - Motor 2 initially spun CCW instead of CW; corrected by swapping two motor wires.

8.2. Tethered Hover

- Objective: Validate PID loops and auto-level behavior.
- Procedure:
  - Install props; secure the drone under a harness so it cannot fully take off.
  - Power on; allow ESCs to initialize.
  - Increase throttle to ~ 50 %.
  - Observe whether the drone remains level without rolling/pitching.
- Observations:
  - With initial gains, the drone oscillated at ~ 3 Hz.
  - After tuning (Kp = 4.5, Ki = 0.10, Kd = 2.00), the drone held level with only minor ripples under disturbance.
- Conclusion: PID values yielded stable level-hold in restricted hover.

8.3. Free Flight

- Objective: Demonstrate stable hover, low-altitude flight, and overall performance.
- Procedure:
  - Choose a calm, open field (wind < 5 km/h).
  - Place quadcopter on leveled ground.
  - Power on; after ESC beeps, slowly increase throttle until lift-off.
  - Hover at ~ 1 m for 15–20 s.
  - Perform minor roll/pitch corrections (if RC integrated) to test responsiveness.
- Metrics & Observations:
  - Hover Stability: Maintained ± 5 cm height variation at ~ 60 % throttle.
  - Lateral Drift: < 10 cm/s in calm conditions; auto-level corrected drift in < 0.5 s.
  - Flight Time: ~ 5 minutes on a single 1100 mAh battery (hover at ~ 55 % throttle).
  - Wind Performance: Minor forward drift under mild breeze (< 2 m/s); PID corrected within ~ 1 s.
- Outcome:
  - A stable hover was achieved.
  - Slight yaw drift observed (no yaw PID).

8.4. Drone Build Photos

Figure 2: Fully assembled quadcopter

---

9. Challenges & Solutions

Challenge	Cause Analysis	Solution & Result
Motor Spin Direction Errors	Factory ESC calibration yielded random spin directions; mixed up motor wire order.	Swapped two motor wires on mis-spinning motors. Verified spin on 10 % throttle (props off). Corrected orientation for CW/CCW pair.
IMU Nonzero Roll/Pitch on Level Surface	Gyro had a bias (~±1.5 °/s). Accelerometer bias was unaccounted.	Conducted 2000-sample calibration on a fool-proofly level table. Computed and applied gyro offsets in code. Verified roll/pitch ≈ 0° at rest.
Forward Drift During Hover	Center of gravity was slightly off; battery	Repositioned battery to geometric center. Performed static balance test on a pencil pivot.

	was not perfectly centered.	After reposition drift reduced to < 5 cm/s and easily corrected by PID.
Excessive Roll/Pitch Oscillations (3–5 Hz) in Tethered Hover	Proportional gain was too high; no derivative damping.	Reduced Kp from 1.3 → 0.8 Introduced Kd = 20.0 Added Ki = 0.03 Resulted in damped oscillations within 0.5 s under disturbance.
High-Frequency IMU Noise from Motor Vibrations	Propeller and motor imbalance transmitted vibrations to IMU.	Added a dense rubber/foam pad under the MPU-6050. Secured IMU loosely with hot glue. Adjusted complementary filter to α = 0.96. Noise reduced by ~ 30 %.
Yaw Drift/Lack of Yaw Stabilization	Project deferred yaw PID (requires magnetometer).	Used RC yaw input for manual correction; in future plan to integrate HMC5883L magnetometer and add yaw PID.
Voltage Sag Under Brass-and-Heavy-Thrust Conditions	1100 mAh 20 C LiPo battery sagged under sustained high throttle.	For next iteration plan to upgrade to 2200 mAh 30 C LiPo for better voltage stability and longer flight time.

---

10. Future Work

GPS-Based Waypoint Navigation & Return-to-Home (RTH)

- Hardware: Add a Ublox NEO-6M (or NEO-M8N) GPS module wired to Arduino's UART (SoftwareSerial). Introduce a magnetometer (HMC5883L) for accurate yaw heading.
- Software:
    - Parse GPS NMEA data via TinyGPS++ library.
    - Implement bearing & distance calculations (Great-Circle formula).
    - Create a waypoint scheduler: fly to each coordinate in sequence.
    - On failsafe trigger (RC signal loss or low battery), switch to RTH: climb to safe altitude, navigate home, descend, and land.

Failsafe & Emergency Landing

- Triggers: Loss of RC link (interrupt), telemetry timeout, low battery voltage (< 10.5 V).

- Behavior:
    - Hold altitude/position (GPS + barometer).
    - Navigate to home coordinate at ~ 5 m/s.
    - Descend at ~ 0.5 m/s when within 5 m of home.
    - Auto-land if battery < 3.4 V per cell.

PC-Based Route Control & Telemetry

- Telemetry Link: Use 433 MHz SiK radios (57.6 kbps) between Arduino (UART) and PC (USB radio).
- Ground Station (PC Software):
    - Either adapt open-source GCS (e.g., Mission Planner) by implementing MAVLink on Arduino (requires major code changes),
    - Or develop a lightweight Python GUI (PyQt) that sends simple ASCII/JSON waypoint commands.
- Arduino Firmware: Implement a packet parser to receive waypoints, store them, and switch between modes (Manual, Auto, Failsafe).

Hardware Upgrades for Performance

- Microcontroller: Upgrade from Arduino Uno (8-bit, 16 MHz) to a 32-bit MCU (Arduino DUE, STM32) for faster loop rates (≥ 400 Hz).
- ESC: Switch to a 4-in-1 BLHeli ESC for lighter weight and unified wiring.
- Frame Improvements: Use carbon-fiber plates to reduce weight.
- Additional Sensors:
    - Barometer (BMP280) for precise altitude hold.
    - Optical Flow Sensor (Px4FLOW) for indoor position hold.

---

11. Conclusion

This concise report documents the successful implementation of an Arduino-based quadcopter using Joop Brokking's YMFC-AL framework. Through methodical hardware assembly, IMU calibration, ESC tuning, and iterative PID adjustments, a stable, 5-minute hover was achieved. Key challenges—motor orientation, IMU biases, forward drift from CoG misalignment, and vibration-induced noise—were identified and resolved. Future enhancements (GPS waypoints, failsafe RTH, PC control) create a clear roadmap for advancing this platform toward semi-autonomous flight. This project exemplifies practical skills in embedded control systems, sensor integration, and UAV operation.

This document, once populated with images and circuit diagrams, can be exported to Word or PDF and appended to a CV or portfolio as evidence of hands-on expertise.

---

12. References

Joop Brokking. (2018). YMFC-AL — Arduino Quadcopter Flight Controller Code.
Website: http://www.brokking.net/ymfc-al_main.html

Rowberg, Jeff. (2021). I2Cdevlib: I2C Device Library for Arduino & Teensy.
GitHub: https://github.com/jrowberg/i2cdevlib

InvenSense. (2013). MPU-6000/MPU-6050 Register Map & Detailed Descriptions (Rev. 3.4).

Arduino.cc. (2025). Arduino Uno Board Reference.
Online: https://www.arduino.cc/en/Main/ArduinoBoardUno

Ublox. (2017). NEO-6M GPS Module Datasheet.

Mikal Hart. (2020). TinyGPS++ Library.
GitHub: https://github.com/mikalhart/TinyGPSPlus