Quadcopter Flight Controller

Toms Jiji Varghese, DESE, Indian Institute of Science, Bangalore-560012

I Team Details

1. Toms Jiji Varghese (22567)
2. Sanidhya Saxena (22586)

II Introduction

Quadcopters, also known as multirotors or drones, are popular flying machines with diverse applications. This report outlines the development of a quadcopter flight controller using the STM32F407 microcontroller, MPU6050 inertial measurement unit (IMU), Flysky FS-I6 remote control, and sensor fusion techniques for flight stabilization.

III Hardware

- Microcontroller (MCU): STM32F407 - An ARM-based MCU with sufficient processing power for real-time floating point flight control algorithms and motor control.
- Inertial Measurement Unit (IMU): MPU6050 - A 6-DOF sensor which provides 3-axis gyroscope and accelerometer data crucial for attitude estimation and stabilization.
- Remote Control (RC): Flysky FS-I6 - A widely used RC system with a transmitter and receiver.
- Electronic Speed Controllers: Commonly available BLDC motor speed controllers based on SimonK firmware.
- Brushless DC Motors/Batteries: 1000KV A2212/13T BLDC motors and Lipo Batteries

IV Software Development

4.1 Development Environment:

- An STM32CubeIDE will be used for code development, compilation, and debugging. along with serial terminals programs like teraterm/matlab programs for PID tuning.

4.2 Key Software Modules:

- Remote Calibration: This module calibrates the end-points of the Received control signal and maps them to standard values

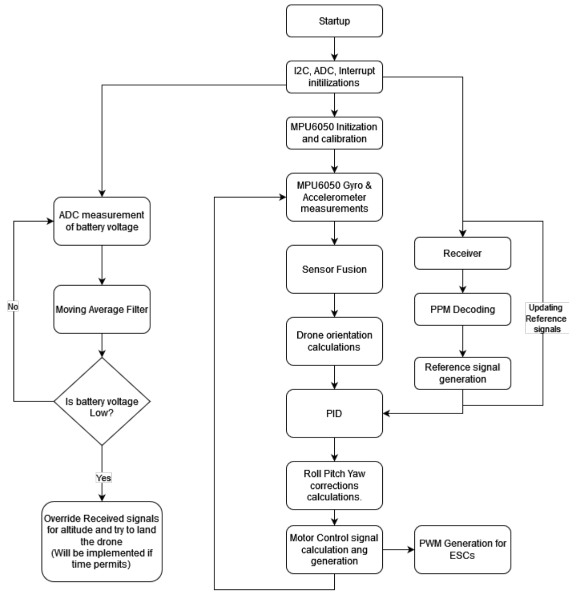

Figure 1: Flow Diagram

- ESC Calibration: This module calibrates the ESCs for standard input pwm values so that all the motors start at the same time.
- Gyro Calibration: This module removes the offset from the gyro reading so that its error is minimized in the subsequent integration opteration used to determine the attitute of the quadcopter.
- Arming/Unarming: These modules are used to arm and unarm the quadcopter so that unintended motors rotations does not take place.
- Debug Mode: This mode is used to map the throttle values to all four motors so that the ESC calirbation can be verified.
- Sensor Driver: This module acts as the bridge between the flight controller and the MPU6050 IMU. It handles I2C communication to initialize the sensor, reads raw gyroscope and accelerometer data, and applies calibration offsets to eliminate inherent biases.
- Sensor Fusion: Since raw sensor data can be noisy and prone to drift, this module employs a sensor fusion technique (complementary filter) to combine gyroscope and accelerometer data. This results in a more accurate and robust estimation of the quadcopter's orientation (pitch, roll, and yaw).
- PPM Decoder: The PPM signal received from the Flysky FS-I6 remote control contains all control

channel information. This module will efficiently decode the signal with the help from interrupts to extract individual control inputs for pitch, roll, throttle, and yaw.

- PID Control Loops (Core for Stability): Proportional-Integral-Derivative (PID) control loops are the backbone of flight stability. These control loops continuously analyze sensor data (obtained from sensor fusion) and compare it to desired flight attitudes (setpoints received from the RC). Based on these comparisons, the PID loops adjust motor control signals in real-time to maintain a stable flight.
- Motor Control: This module translates the processed control signals (from PID loops and throttle) into PWM (Pulse Width Modulation) signals for each motor. Utilizing the STM32F407's dedicated PWM peripherals, these signals are sent to the ESCs, ultimately controlling the speed of each brushless DC motor.

4.3 Optimization for Real-time Performance:

A crucial aspect of the software development is ensuring real-time performance. All software routines will be implemented efficiently, aiming for motor speed updates in the millisecond range. This rapid update rate minimizes control lag and enables a more responsive flight experience.

V Development Considerations

Challenge 1: Balancing Processing Demands

The flight controller juggles several real-time tasks: sensor data acquisition, control algorithm execution, and motor control signal generation. Balancing these demands on the STM32F407 MCU is crucial.

Solution: Optimization Techniques

Code Optimization: Implementing efficient coding practices and algorithms will minimize processing overhead. Prioritization: Prioritizing critical tasks like sensor data processing and motor control within the main loop ensures timely execution. Peripheral Utilization: Employing hardware peripherals like timers and the I2C interface offloads processing burden from the CPU, allowing it to focus on core control algorithms.

Challenge 2: Minimizing Control Latency

Latency between sensor data acquisition and motor response can significantly impact flight stability.

Solution: Interrupt Handling

Timer Interrupts: Configuring timers to generate interrupts at precise intervals ensures timely sensor data sampling and motor control updates. I2C Interrupts: Enabling I2C interrupts allows efficient data exchange with the MPU6050, minimizing delays in retrieving sensor readings.

Challenge 3: Precise Motor Control

The accuracy of PWM signals determines the responsiveness and efficiency of motor control.

Solution: Timer Configuration

High-Resolution Timers: Utilizing high-resolution timers on the STM32F407 allows for generating PWM signals with fine-grained control over motor speed adjustments. Calibration: Implementing a calibration routine during startup removes biases from the motor control signals, ensuring accurate motor response based on the control loops' outputs.

VI Final Quadcopter

Figure 2: Quadcopter

References

[1] Project YMFC-32 (http://www.brokking.net/ymfc-32_main.html)

[2] https://github.com/CarbonAeronautics

[3] Quadcopter Design MATLAB (https://www.youtube.com/playlist?list=PLPNM6NzYyzYqMYNc5e4_xip-yEu1jiVrr)