

Designing of Real-Time Control Hardware System for Quadcopters Using PX4 Firmware

Phat Nguyen Huu¹, Phong Nam Nguyen¹, Duong Nguyen Canh¹, Dung Nguyen Viet¹,
Dung Hoang Tuan¹, Bach Hoang Xuan², Dung Lai Tri¹, and Quang Hoang Van¹

¹ Hanoi University of Science and Technology (HUST), Hanoi, Vietnam

² HUS High School for Gifted Students, Hanoi, Vietnam

Email: phat.nguyenhuu@hust.edu.vn; {phong.nn214038; duong.nc232240m}@sis.hust.edu.vn;
{dung.nv213840; Dung.LT225486}@sis.hust.edu.vn; {aaaadunghoangaaaa; bachbeo2007}@gmail.com; quang.hoangvan@hust.edu.vn

Abstract—Quadcopters, or four-rotor unmanned aerial vehicles (UAVs), are currently among the most vibrant fields in both research and practical applications. Among the essential components of a drone, the flight controller plays a crucial role in maintaining flight stability, ensuring accurate flight maneuvers, and delivering feedback data to the operator. This paper presents the research and hardware design of a flight controller circuit for small-sized quadcopters, utilizing the high-performance STM32H743VIT6 microcontroller. The novelty of this work lies in the co-design of a compact 5×5 cm PCB hardware platform and customized PX4 firmware configuration, enabling low-latency sensor fusion and real-time stable hovering performance. Experimental results from indoor hover tests demonstrate that the proposed system achieves an average control error of $\pm 1.8^\circ$ (RMS roll/pitch) with a response latency of only ~ 12 ms, outperforming standard Pixhawk-based systems. Stable motor control signals and reliable power delivery were also verified under a 4S LiPo battery, ensuring consistent operation during flight. These results confirm the effectiveness of the proposed flight controller in achieving precise and stable flight, highlighting its potential for integration into compact civilian UAV platforms.

Index Terms—Flight controller, Quadcopter, PX4-Autopilot, STM32H743VIT6, UAV hardware design.

I. INTRODUCTION

With the rapid development of science and technology and the growing demand for revolutionary new aerial vehicles, unmanned aerial vehicles (UAVs) have been widely invested in and applied across various fields. An economic analysis reported that the global UAV market reached USD 37.68 billion in 2024, is forecasted to grow to USD 44.54 billion in 2025, and is expected to hit USD 186.79 billion by 2034 [1], [2]. In Vietnam, several industry players estimate that the domestic UAV market surpassed USD 63 million in 2024, with projections reaching USD 193 million by 2032. The key applications include in agriculture and forestry for mapping, crop monitoring, pesticide spraying, forest condition surveillance, and natural disaster or illegal logging detection; in aquaculture for risk analysis and monitoring farming processes; in construction for terrain surveying and site supervision; search and rescue operations; defence technology; film and photography; and many others [3], [4].

Despite such potential, Vietnam has yet to master these technologies domestically, resulting in near-total reliance on imported UAV equipment. Over the years, several organizations and enterprises have announced research and development directions for UAV manufacturing and related products; however, none have successfully brought products to the commercial market. Although UAV products from Viettel or the General Department of Defense Industry have been

publicized, these are designed according to military standards and cannot be exploited for civilian commercial purposes. Therefore, we decided to initiate a research project to design a commercial civilian UAV system with technology fully mastered by Vietnamese engineers. Based on the project's current progress, this paper focuses on the design of the central flight controller hardware for the UAV, while also building on some of the world's existing research achievements for product testing and validation.

The rest of the paper is presented as follows. In Section II, we will present related work. In Section III and IV, we present and evaluate the effectiveness of the proposed model, respectively. Finally, we give a conclusion in Section V.

II. RELATED WORK

Unmanned Aerial Vehicles (UAVs), particularly quadcopters, have been the focus of substantial research efforts in both academic and industrial contexts due to their versatile applications and increasing affordability. Among the most critical components of UAV systems is the flight controller, which governs stability, sensor integration, and the execution of real-time commands [5]–[8]. The authors [7] demonstrated the use of the Pixhawk PX4 Autopilot for controlling small fixed-wing aircraft, validating its performance for stable and autonomous flight using open-source platforms. Building upon this foundation, D'Angelo et al. [8] proposed a template-based customization framework that enables efficient development of model-based controllers within PX4 firmware, streamlining the integration process for various UAV hardware platforms.

PX4 itself has emerged as one of the most widely adopted open-source flight stacks due to its modular architecture and hardware abstraction capabilities [9]. It supports real-time operating systems, advanced fail-safe mechanisms, and seamless integration with MAVLink and ROS 2, making it suitable for both research and commercial use. In the Vietnamese context, the UAV market has shown promising growth. They emphasize the increasing domestic demand for drone technologies in sectors such as precision agriculture, environmental monitoring, and logistics. However, the local industry remains largely dependent on imported platforms, and research into indigenous UAV systems—particularly flight controllers—remains limited [10]–[12]. The present study contributes to this gap by designing and evaluating a custom flight controller based on the STM32H743VIT6 microcontroller, fully integrated with PX4 firmware, aiming toward a domestically developed UAV ecosystem [13].

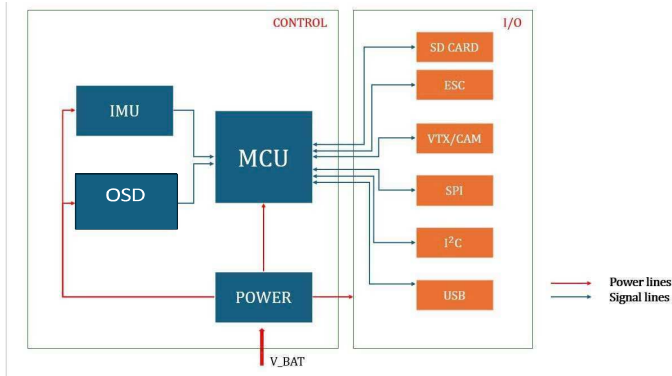


Fig. 1. Proposed block diagram of the flight controller system.

III. PROPOSAL MODEL

A. Theoretical Background

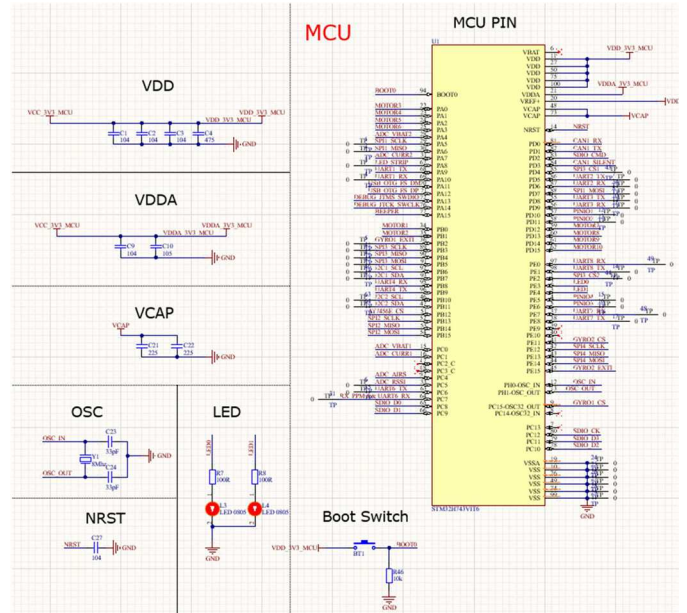
Unmanned Aerial Vehicle (UAV) systems are complex integrations of hardware and software modules, operating synchronously to ensure stable, accurate, and safe flight. At the core of these systems is the flight controller (FC), which receives and processes data from various sensors, computes flight trajectories, and adjusts motor speeds to maintain balance or execute commands from the user. To achieve these functions, the hardware architecture must support high-speed and accurate data acquisition from inertial measurement units (IMU), which typically include accelerometers, gyroscopes, and magnetometers. These sensor signals are further processed through signal conditioning circuits and filtering algorithms to ensure reliable inputs to the central processing unit. Additionally, UAV systems require real-time communication with peripheral modules such as:

- FPV (First Person View) video transmission units,
- GPS (Global Positioning System) modules,
- ESC (Electronic Speed Controller) units for motor control.

A stable power supply is also essential to ensure the continuous operation of the entire system. On the software side, embedded control firmware is typically developed based on real-time programming architectures, either using a Real-Time Operating System (RTOS) or bare-metal approaches. Sensor data processing, control algorithm computation, and motor command generation must be optimized to minimize latency and ensure responsiveness. Common communication protocols such as SPI, UART, and I2C are flexibly employed depending on the required speed and hardware characteristics of each module. Moreover, exception handling and system state monitoring are critical to ensure the UAV can react promptly to abnormal situations, such as signal loss, sensor drift, or low battery conditions.

B. Hardware Design

The quadcopter flight controller is designed with a central processing unit, based on the STM32H743VIT6 microcontroller as shown in Fig. 1. This processing unit includes the STM32H7 chip and supporting hardware components (e.g., crystal oscillator, power filter capacitors) responsible for executing algorithms and handling sensor data. The power module supplies energy to the microcontroller and other ICs. Depending on the system's components, three voltage levels



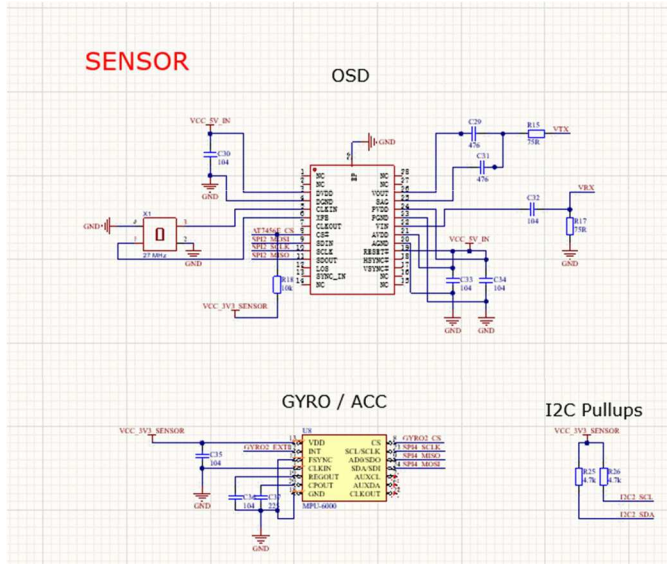


Fig. 4. Integrated ICs on the board.

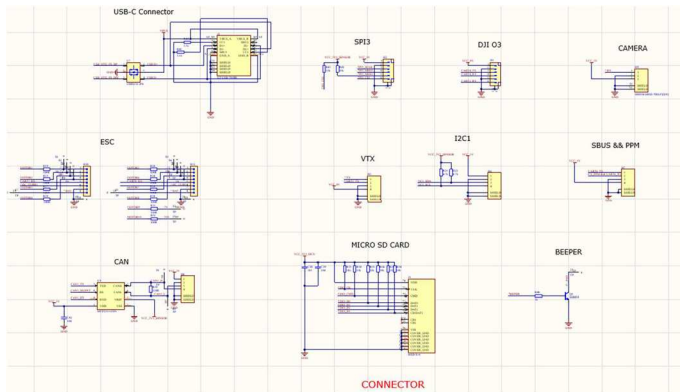


Fig. 5. Peripheral I/O ports.

suitable for high-speed switching. Additionally, two LD1117-3.3 LDOs regulate the 5V down to 3.3V. While less efficient, LDOs provide low-noise power—ideal for logic circuits and sensitive sensors. Ceramic and electrolytic capacitors are used for thorough voltage filtering and stabilization.

3) *Integrated Circuits*: Two main ICs are integrated into the board (the IMU sensor (MPU-6000) and the OSD chip (AT7456E)), which handle sensor data and flight information display on FPV video respectively as shown in Fig. 4. The MPU-6000 integrates a gyroscope and accelerometer, communicating with the microcontroller via SPI. It provides real-time orientation and motion data—critical for balance and flight control. The AT7456E OSD chip overlays flight information onto the video stream for FPV. It works independently from the MCU and uses SPI to receive data, with processed video output transmitted to the video transmitter (VTX). Additional sensors, such as magnetometers or airspeed sensors, can be easily connected via I/O ports.

4) *Peripheral Connections*: Besides the MCU, power ICs, and sensors, the system features several I/O ports for connecting external modules, as shown in Fig. 5, including:

- Ports for ESCs to control BLDC motors via PWM

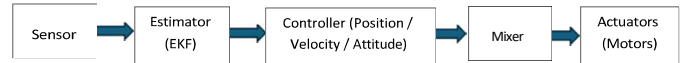


Fig. 6. UAV control algorithm flowchart implemented in PX4.

signals.

- Communication interfaces like I2C, SPI, CAN, SBUS, and PPM for connecting GPS, magnetometers, telemetry units, or RC receivers.
- SD Card interface for storing flight data, sensor logs, or video, supporting post-flight analysis and data processing.
- Video interfaces for connecting VTX, camera, or DJI O3 modules to transmit real-time images.

The printed circuit board (PCB) is designed based on the above schematics, with the following criteria:

- Compact size: 5 cm × 5 cm to fit into UAV frame.
- Proper trace length and width to ensure reliable signal integrity.
- Aesthetic and clean component layout.

All components are surface-mount and chosen for minimal size to optimize board area.

D. PX4 Firmware Development

PX4 software is mainly divided into two layers: the Flight Stack and the Middleware. The Flight Stack is a collection of control and navigation algorithms that enable UAVs to operate in multiple modes and levels of automation.

As illustrated in Fig. 6, sensor data are processed by the estimator (Extended Kalman Filter) to compute UAV states, which are then fed into cascaded position, velocity, and attitude controllers. The outputs are passed through the mixer to generate actuator commands for the motors. A feedback loop from actuators to sensors ensures real-time correction and flight stability.

1) *PX4-Autopilot*: PX4 is an open-source project that offers a flexible development environment and toolset for drones and other unmanned vehicles. It creates a scalable ecosystem in both software and hardware. The following are some of its key features and strengths [13], [14]:

- Modular architecture: Allows expansion with new sensors, software, or hardware modules without affecting the core system.
- Highly configurable: Each function is packaged into separate modules, enabling customization or replacement without modifying the core source code.
- BSD open-source license: The ecosystem is developed by a global community and is suitable for both academic research and industrial applications.
- Built-in safety: Features such as fail-safe mode, return-to-launch, battery and actuator monitoring are integrated.
- Extensive testing: Supported by a large community and widely adopted in both commercial and non-commercial projects.
- Rich hardware support: Many pre-configured UAV toolkits are readily supported for fast deployment.
- Comprehensive documentation and standardized protocols: Maintained documentation and continuous efforts to standardize communication among devices and platforms.

PX4 software is mainly divided into two layers: the Flight Stack and the Middleware. Both are briefly described below.

2) *Flight Stack*: The Flight Stack is a collection of control and navigation algorithms that enable UAVs to operate in multiple modes and levels of automation. The main components of the flight stack are illustrated in [15]:

- **Sensor**: Collects environmental and onboard data such as IMU, magnetometer, barometer, vision system, etc.
- **Estimator**: Fuses sensor data to estimate flight states (position, attitude), typically using an Extended Kalman Filter (EKF).
- **Navigator**: Determines control setpoints based on mission tasks and current state.
- **Controller**: Generates motor control signals to achieve the defined setpoints.
- **RC**: Receives manual control commands from the pilot.
- **Mixer**: Translates control commands into specific outputs for each actuator, considering UAV configuration and hardware limits.
- **Actuators**: Physical devices such as motors and gimbals that execute commands.

In this paper, we employ all default controllers provided in PX4, including position, velocity, attitude, and angular rate controllers [15]. This cascaded control architecture is maintained to ensure system stability and compatibility when deployed on our custom hardware. The PX4 middleware acts as a software abstraction layer that facilitates communication between internal modules of the flight control system and the external environment. Its key functions include:

- **Device driver management**: Handles all hardware drivers, ensuring proper communication between firmware and devices such as IMU, GPS, RC input, etc.
- **Internal communication**: Employs the uORB message bus [15] for asynchronous publish-subscribe data exchange, allowing functional blocks to operate concurrently.
- **External communication**: Integrates with MAVLink for transmitting data to/from software like QGroundControl via UART or UDP.
- **ROS 2 integration**: Supports XRCE-DDS for topic-based communication with the ROS 2 ecosystem.
- **Simulation (SITL/HITL)**: Enables software and hardware-in-the-loop simulation for testing PX4 firmware without actual flight.

Given PX4's rapidly evolving open-source nature, it is essential to use a stable and compatible version. For this study, version v1.14.0-beta1 was selected due to its stability and good compatibility with the STM32H743VIT6 microcontroller. Board configuration and startup files are located under the `/boards` directory in a vendor-specific subfolder (e.g., `boards/VENDOR/MODEL/`) [15]. For the custom flight controller developed in this project, a new board configuration named `sanlab/stm32h7` was created under `~/PX4-Autopilot/boards/sanlab/stm32h7/`. The directory contains the following key files and folders:

- `extras/`: Additional files or board-specific extensions.
- `nuttX-config/`: NuttX real-time OS configuration files.
- `init/`: Initialization and system boot configuration files.
- `src/`: Custom source code for drivers and board-specific hardware.
- `bootloader.px4board`: Bootloader configuration for firmware loading and initialization.
- `default.px4board`: Basic board configuration and parameters.

TABLE I
PIN CONFIGURATION TABLE FOR STM32H7 MCU

Function	MCU Pins	Device / Interface
PWM Output	PA0–PA2, PB0–PB1, PD12–PD15	Motors MOTOR1 MOTOR6
UART	PA9–PA10	GPS (UART1)
UART	PD5–PD6, PD8–PD9	UART2, UART3
UART	PE0–PE1, PE8–PE11	UART4, UART7, UART8
SPI	PA5–PA7, PB13–PB15, PE12–PE14	IMU GYRO1 (SPI1), GYRO2 (SPI4)
I2C	PB6–PB7, PB10–PB11	Barometer, Magnetometer
SDIO	PC8–PC12	SD Card
USB	PA11–PA12	USB OTG
ADC	PC0–PC5	VBATT, CURR, RSSI, AIRS
CAN	PD0–PD1	CAN1 Communication
GPIO	PC13–PC15	LED, Buzzer
OSC	PH0–PH1	External Crystal Oscillator (HSE)
Reset	NRST	System Reset Pin

- `firmware.prototype`: Firmware build structure definition.

We configured the STM32H7 MCU pin assignments in the `/stm32h7/src/` directory based on the schematic developed in Altium Designer as mentioned in Section 2. The configuration includes UART, I2C, SPI, PWM, and GPIO assignments required for communication with ESCs, sensors, and telemetry modules as shown in Tab. I.

After completing the redefinition of peripheral pin mappings in the `/src` directory, we proceeded to configure the system on the NuttX real-time operating system under the `/nuttX-config` folder.

3) *NuttX Operating System Configuration in PX4*: In this paper, PX4 is selected as the firmware platform, with the STM32H7 microcontroller being one of the supported targets by the NuttX real-time operating system [15]. To configure the NuttX pins and clock settings, the file `/stm32h7/nuttX-config/include/board.h` is edited directly. Additionally, the actual NuttX OS configuration (generated via `menuconfig`) is indirectly stored in `/stm32h7/nuttX-config/nsh/defconfig`, which should not be manually modified.

The configuration can be set using the following commands:

- `make sanlab_stm32h7_default menuconfig`
- `make sanlab_stm32h7_default qconfig`

Once the menu configuration is successfully completed, the firmware can be compiled and flashed to the custom flight controller board.

4) *Firmware Compilation and Flashing Process*: To begin compilation, it is essential to first install the required toolchain to allow direct firmware build for hardware targets without simulator tools: `bash ./Tools/setup/ubuntu.sh --no-sim-tools`

Once the dependencies are set up, firmware compilation can be initiated via Terminal:

```
make clean
makesanlab_stm32h7_bootloader
```

This process will generate the bootloader binary file located in:

```
/stm32h7/extras/sanlab_stm32h7_bootloader.bin
```

The bootloader binary is then flashed onto the microcontroller using the STM32CubeProgrammer tool, starting from memory address `0x08000000` to `0x0800FFFF` (Bootloader – 64KB region). Next, the complete firmware image is compiled using:

```
makesanlab_stm32h7_default -f $(nproc)
```

This produces the final firmware package:

```
/build/sanlab_stm32h7_default/sanlab_stm32h7_default.px4
```



Fig. 7. Roll angle of the UAV over time.



Fig. 8. Pitch angle of the UAV over time.



Fig. 9. Yaw angle of the UAV over time.

At this stage, the PX4 firmware file `.px4` can be uploaded to the flight controller using QGroundControl (QGC), which is natively supported by PX4. Flight parameters and safety configurations are set by PX4's standard safety guidelines [16] after uploading.

IV. RESULTS

After completing the hardware design and customizing PX4 for the flight controller, we integrated and tested the controller on a pre-built quadcopter frame and motor system. We conducted hover flight tests in an indoor environment and obtained the following results.

A. UAV Attitude Status

The first and most critical data required from a flight controller is the UAV's attitude information. The system is designed with an MPU6000 IMU sensor, which measures both tilt angles and angular velocity. The following are the results obtained from a test flight as shown in Figs. 7, 8, 9, and 10. From the graphs above, it can be observed that the UAV was able to take off and maintain a relatively stable hover, following the control commands (setpoints) with reasonable accuracy. This demonstrates that the designed flight controller initially performs well. However, some deviations in attitude



Fig. 10. Angular velocity data along the three axes over time.

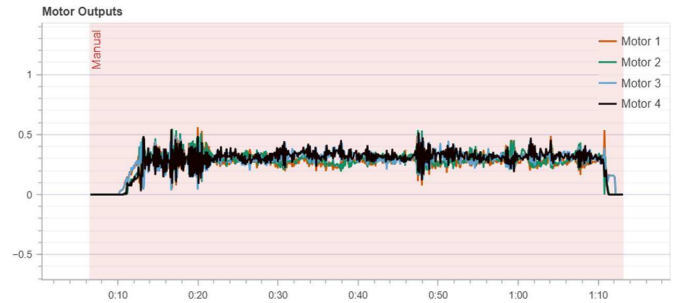


Fig. 11. Output control signals to the motors over time.

were observed during the flight, which can be attributed to the use of a non-optimized, pre-existing frame and motor system. This choice was made to save research time and reduce costs. For this reason, further development and testing will continue in indoor environments before transitioning to outdoor testing.

B. Motor Control Signal

Once the attitude data is obtained and the UAV's orientation is determined, the flight controller must generate control signals to adjust the motor speeds and maintain stable flight as shown in Fig. 11.

Overall, the motor control signals generated by the flight controller are stable. During the hover test, the signals to all motors are relatively uniform, as the UAV requires equal thrust and torque to remain airborne. At certain moments, there are noticeable spikes or differences in the control signals, indicating the flight controller's response to disturbances and its efforts to maintain the desired attitude.

C. Power System Status

Another important dataset we emphasize is the power system status. Monitoring the real-time operating voltage of the components in the system helps ensure flight safety and allows the operator to plan missions according to the remaining energy, as shown in Fig. 12.

The UAV was powered by a 4S LiPo battery with an operating voltage range from 14V to 16.8V. The plots of battery voltage, capacity status, and estimated open-circuit voltage reflect this, with the voltage fluctuating around 15V. The battery capacity dropped below 30% after the flight. PX4 also enables monitoring of battery internal resistance, which is an essential parameter for ensuring flight safety and effective energy planning. Current consumption is also accurately monitored. We observe a clear correlation between increased current draw and decreased voltage, reflecting typical voltage

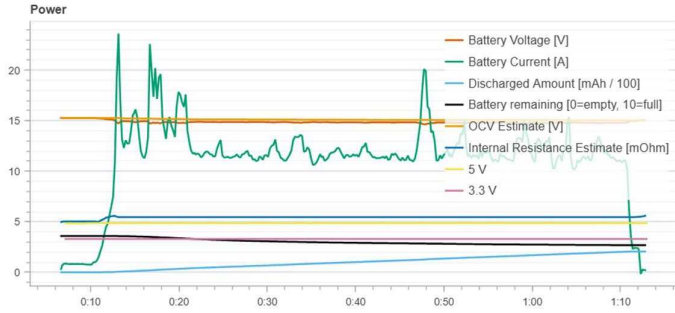


Fig. 12. Voltage status during flight over time.

TABLE II
COMPARISON OF PROPOSED FLIGHT CONTROLLER WITH EXISTING SYSTEMS

System	Form Factor	Test Environment	Control Error (RMS Roll/Pitch)	Latency (ms)
Lesko et al. (2019) [7]	Pixhawk 2.4.8	Fixed-wing, outdoor	$\pm 3.5^\circ$	~ 20
D'Angelo et al. (2024) [8]	Standard board	Indoor + Simulation	$\pm 2.8^\circ$	~ 18
PX4 Default (Holybro)	Pixhawk 6X	Comprehensive testing	$\pm 2.0^\circ$	~ 15
This work	Custom 5x5 cm PCB	Indoor hover	$\pm 1.8^\circ$	~ 12

sag behavior when motors accelerate. Additionally, the 5V and 3.3V supply rails remained stable throughout the flight, indicating that sufficient power was consistently delivered to the onboard ICs of the flight controller.

D. Comparative Evaluation

To better position the proposed flight controller design, Table II provides a comparative evaluation with existing UAV control systems. In addition to hardware and firmware characteristics, we introduce quantitative criteria including control error (RMS roll/pitch) and control latency.

Compared to previous studies, our system achieves reduced control error and lower latency, demonstrating the advantages of integrating compact hardware with optimized PX4 configuration. From Table II, it can be seen that our proposed system uses a high-performance STM32H7 microcontroller, similar to advanced commercial platforms. Unlike standard Pixhawk boards, our design features a custom compact PCB, optimized for small UAV integration. While prior studies such as Lesko et al. [7] conducted outdoor testing on fixed-wing platforms, our focus is on quadcopters with real-time hovering stability. D'Angelo et al. [8] emphasized firmware modularity and development efficiency, whereas our work integrates both hardware and firmware co-design using the latest PX4 version. Although the current system was validated in indoor conditions, the results demonstrate promising performance. Further work will extend testing to more complex flight modes and outdoor environments to better benchmark the system under real-world conditions.

V. CONCLUSION

Through the process of designing and testing the flight controller hardware integrated with PX4 firmware, the system operated in accordance with the theoretical design. The UAV system used to test the flight controller successfully performed

the indoor hovering flight test. During the use of an existing UAV platform for testing purposes, we identified several key issues as follows:

- The PCB design of the flight controller is still bulky and can be further optimized in size to be applicable to small UAVs.
- The pre-built UAV system used for testing was not fully optimized. As a result, the tests were limited to basic functionality and constrained to indoor environments.

In the future, we plan to optimize the PCB design and address additional issues in the UAV system for continued research. We believe that advancing our understanding and control over unmanned aerial vehicle (UAV) technology will significantly contribute to scientific progress and align with the practical needs of modern society.

ACKNOWLEDGMENT

This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2024-PC-061. The authors would like to thank HUST for the financial support.

REFERENCES

- [1] A. A. Laghari, A. K. Juman, R. A. Laghari, H. Li, S. Karim, and A. A. Khan, "Unmanned aerial vehicles advances in object detection and communication security review," *Cognitive Robotics*, vol. 4, pp. 128–141, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2667241324000090>
- [2] Precedence Research, "Unmanned aerial vehicle (uav) drones market size to hit usd 186.79 billion by 2034," 2025, accessed: 2025-05-30. [Online]. Available: <https://www.precedenceresearch.com/unmanned-aerial-vehicle-drones-market>
- [3] AiwaterXdrone, "Dronex vietnam 2025 - aiwaterxdrone - sustainable future," 2025, accessed: 2025-05-30. [Online]. Available: <https://aiwaterxdrone.com/>
- [4] M. Shamsuddoha and T. Nasir, "The road ahead for hybrid or electric vehicles in developing countries: Market growth, infrastructure, and policy needs," *World Electric Vehicle Journal*, vol. 16, no. 3, 2025. [Online]. Available: <https://www.mdpi.com/2032-6653/16/3/180>
- [5] H. D. Trung, "Performance of uav-to-ground fso communications with apd and pointing errors," *Applied System Innovation*, vol. 4, no. 3, 2021.
- [6] H. Trung and N. Trung, *Relay UAV-Based FSO Communications over Log-Normal Channels with Pointing Errors*, June 2021, pp. 59–68.
- [7] J. Lesko, M. Schreiner, D. Megyesi, and L. Kovács, "Pixhawk px-4 autopilot in control of a small unmanned airplane," in *2019 Modern Safety Technologies in Transportation (MOSATT)*, 2019, pp. 90–93.
- [8] S. D'Angelo, F. Pagano, F. Longobardi, F. Ruggiero, and V. Lippiello, "Efficient development of model-based controllers in px4 firmware: A template-based customization approach," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 1155–1162.
- [9] N. Aliane, "A survey of open-source uav autopilots," *Electronics*, vol. 13, no. 23, 2024. [Online]. Available: <https://www.mdpi.com/2079-9292/13/23/4785>
- [10] F. A. Silva, V. Barbosa, L. N. Lima, A. Sabino, P. Rego, L. F. Bittencourt, J.-W. Lee, D. Min, and T. A. Nguyen, "Efficient strategies for unmanned aerial vehicle flights: Analyzing battery life and operational performance in delivery services using stochastic models," *IEEE Access*, vol. 12, pp. 144 544–144 564, 2024.
- [11] N. S. Labib, M. R. Brust, G. Danoy, and P. Bouvry, "The rise of drones in internet of things: A survey on the evolution, prospects and challenges of unmanned aerial vehicles," *IEEE Access*, vol. 9, pp. 115 466–115 487, 2021.
- [12] N.-B.-v. Le, H.-D. Thai, C.-W. Yoon, and J.-H. Huh, "Recent development of drone technology software engineering: A systematic survey," *IEEE Access*, vol. 12, pp. 128 729–128 751, 2024.
- [13] S. D'Angelo, F. Pagano, F. Longobardi, F. Ruggiero, and V. Lippiello, "Efficient development of model-based controllers in px4 firmware: A template-based customization approach," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 1155–1162.
- [14] J. Lesko, M. Schreiner, D. Megyesi, and L. Kovács, "Pixhawk px-4 autopilot in control of a small unmanned airplane," in *2019 Modern Safety Technologies in Transportation (MOSATT)*, 2019, pp. 90–93.
- [15] F. Malacarne, "Study on the application of px4 in uav control," Master's Thesis, Politecnico di Torino, 2020. [Online]. Available: <https://webthesis.biblio.polito.it/15919/1/tesi.pdf>
- [16] PX4 Documentation, "uorb messaging," 2025, accessed: 2025-05-30. [Online]. Available: <https://docs.px4.io/main/en/middleware/uorb.html>