Sanity: An Agile Brushless Quadrotor for Multi-Agent Experiments

Heedo Woo, Kazi R. I. Sanim, Keisuke Okumura, Guang Yang, Ajay Shankar, Amanda Prorok
Department of Computer Science and Technology
University of Cambridge, United Kingdom
{hw527, kris2, ko393, gy268, as3233, asp45}@cst.cam.ac.uk

Abstract: This paper introduces *Sanity*, a quadrotor platform designed for rapid multi-agent research. Building on off-the-shelf components, the 72g brushless-motor platform with WiFi and dual-processor compute achieves a 6.9 thrust-to-weight ratio and speeds over 6m/s indoors. It offers ease of deployment, with its reliability demonstrated through successful missions involving teams of up to 20 agents flying simultaneously at high speeds.

1 Introduction

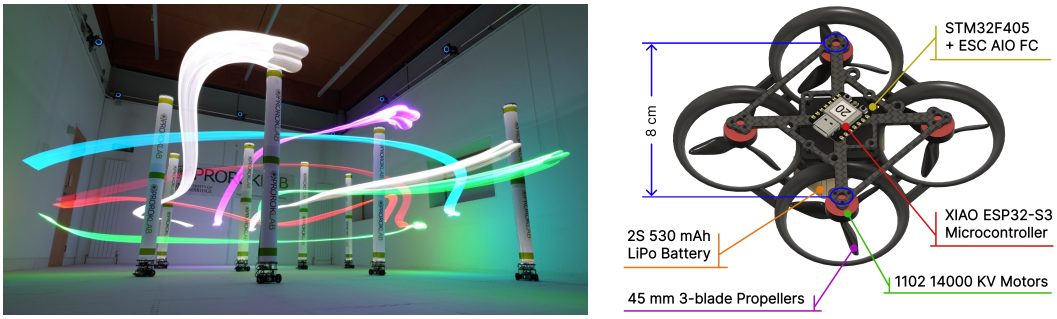


Figure 1: (left) Long exposure photograph showing eight LED-equipped *Sanity* quadrotors during coordinated flight operation in an indoor motion-capture environment, with light trails illustrating their flight trajectories. (right) CAD overview of the platform, highlighting key components.

Multi-agent robotics research necessitates platforms that can serve as testbeds for rapid development and deployment of navigation algorithms in challenging environments. Aerial platforms are particularly valuable as they enable testing of multi-agent coordination algorithms that exploit three-dimensional space and execute agile maneuvers, providing significantly more complex dynamics than ground-based systems. However, developing such platforms presents substantial engineering challenges, especially when constrained to small indoor operational spaces where Size, Weight, and Power (SWaP) limitations become critical. The selection of an appropriate platform is fundamentally driven by several factors, most prominently the size of the operational workspace, the number of agents constituting a 'team', and the computational demands of the algorithms.

This paper introduces a minimalist quadrotor platform, *Sanity*, purpose-built to facilitate efficient testing and validation of proof-of-concept algorithm designs.

Our system prioritizes small size, high agility, and a very low cost to deploy. We build upon off-the-shelf components and a pre-assembled ready-to-fly (RTF) platform and make substantial improvements to swarm communication reliability, crash resilience, and overall ease of implementation. Our approach dramatically reduces the time, cost and engineering overhead to build, maintain, deploy, and regularly operate a multi-agent fleet—thus enabling more test cycles for multi-robot algorithms.

A popular testbed platform with comparable form factor is the Crazyflie 2.1 [1], widely adopted in multi-agent research [2, 3, 4]. While this platform operates effectively in similar physical spaces,

we identified opportunities to significantly enhance key performance characteristics. Our platform aims to improve kinodynamics through higher thrust-to-weight ratios, simplify the interface through streamlined communication protocols, and expand computational capabilities with a dual-processor architecture. These improvements address common limitations in existing platforms while maintaining the small form factor essential for dense multi-agent operations.

2 Technical Details

The following sections detail the platform build, the control stack, and the communication system.

2.1 Platform

Sanity is based on the BetaFPV Pavo Pico micro-sized quadrotor with an 80mm motor-to-motor wheelbase and a diameter of 150mm. It features 1102 14000Kv brushless motors paired with 45mm tri-blade propellers, delivering a TWR of 6.9 with demonstrated flight speeds up to 8 m/s indoors and 17 m/s outdoors. The lightweight carbon fiber frame with integrated nylon ducts protects the platform during collisions while keeping the total weight to just 72g. A 2S 530mAh LiPo battery provides approximately 6 minutes of flight time per charge. At £85 for the base platform and £6 for the ESP32S3 controller, the system offers an affordable solution for research applications.

Specification	**Sanity**
Mass (g)	72
Wheelbase / Diameter (mm)	80 / 150
Thrust-to-Weight Ratio (TWR)	6.9
Flight Controller	STM32 F405 (168 MHz)
Motor	BetaFPV 1102 14000 Kv Brushless Motor
Propeller	Gemfan 45 mm 3 blade
Battery	2S (7.4V) 530 mAh

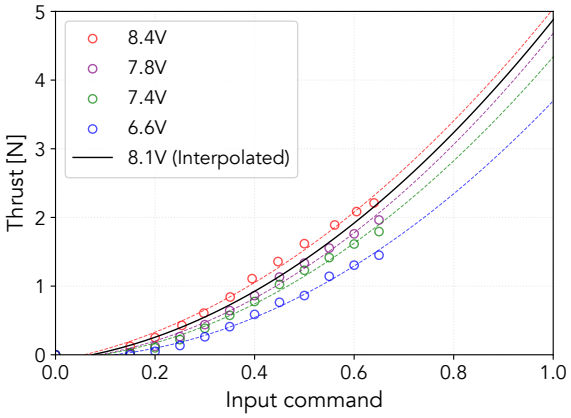
2.2 Compute & Control

The stock platform is already equipped with an STM32F405 microcontroller that interfaces with the onboard sensors— 3-axis accelerometer, 3-axis gyroscope and motor controllers— and performs low-level control and stabilization. Several open-source autopilot stacks support this SoC (System on Chip); we use *Betaflight* for its minimalist and light-weight implementation. We additionally mount an ESP32S3 (Seeed Studio XIAO ESP32S3) to this platform to serve as a second layer of compute. This interfaces with the low-level SoC, provides wireless connectivity, and can potentially run higher-level control and inference. The interface to the flight control SoC is a light-weight wireless communication module (see 2.3) which runs automatically at startup. The majority of the processor and memory space on the secondary compute is left available for custom user-level code.

Our architecture utilizes Betaflight's `ANGLE` mode, a self-stabilising flight mode that uses the onboard 3-axis accelerometer to maintain level flight. Then, a high-level control architecture generates RPYT commands, the desired roll angle, pitch angle, yaw rate, and collective thrust, which are sent as standardized inputs to the flight controller. This modular interface allows replacement of high-level controllers as long as they support these input parameters.

We use the SBUS digital RC protocol to send RPYT commands to the flight controller, which are mapped from physical units to a scaled range as described next.

Angle Mapping. For roll and pitch targets, we use a straightforward linear mapping between $[-\texttt{ANGLE\_MAX}, +\texttt{ANGLE\_MAX}]$ into the range supported by the digital protocol. The parameter $\texttt{ANGLE\_MAX} = 60°$, along with the linearization of its response curve is set manually within Betaflight. A similar $\texttt{ANGLE\_RATE} = 600°/\text{s}$ profile is also set for angular rates, including yaw-rate targets.



$$f = \alpha \cdot \texttt{cmd}^2 + \beta \cdot \texttt{cmd} + \gamma$$
$$\alpha = 4.054 \times 10^{-6}$$
$$\beta = -7.194 \times 10^{-3}$$
$$\gamma = 3.051$$

Figure 2: Thrust calibration curves at different voltage levels. The plot shows measured data points marked with circles, dashed lines representing second-degree polynomial fits for each voltage level (6.6 V, 7.4 V, 7.8 V, and 8.4 V), and a solid line indicating the final interpolated polynomial near 8.1 V, used for flight control.

Thrust Mapping. To map the collective thrust commands from the high-level controller to input values, we mounted the *Sanity* platform on a load stand equipped with a single load cell. This setup allowed us to record the vertically downward collective thrust generated in response to different input commands. This experiment was repeated at multiple voltage levels: 6.6 V, 7.4 V, 7.8 V, and 8.4 V, to emulate different battery conditions and a second-degree polynomial was fitted to the data collected at each voltage. Among them, the interpolated curve near the 8.1 V level was chosen as it empirically minimized the error between the desired (output by high-level controller) and measured acceleration (from mocap).

Figure 2 presents the thrust calibration data, including measured thrust values at different voltages, their corresponding polynomial fits, and the final interpolated calibration curve. The resulting mapping function between the input command `cmd` and total generated thrust f is also shown in the figure, where the coefficients α, β, and γ were derived from the interpolated polynomial fit.

2.3 Communication System

The ESP32S3 microcontroller wirelessly receives control commands over a dedicated WiFi channel, and relays them to the flight controller over UART with SBUS, enabling remote control.

2.3.1 UDP Broadcast

Control commands are transmitted to the entire quadrotor fleet using a UDP broadcast protocol, where each quadrotor extracts its designated 9-byte control segment from a shared packet using its pre-assigned index. As illustrated in Figure 3, each control segment contains four 2-byte fields encoding roll, pitch, yaw rate, and thrust commands, along with a 1-byte auxiliary flag field. To improve communication reliability and mitigate packet loss, the broadcaster transmits packets at approximately twice the rate at which control commands are generated.

2.3.2 Serial Communication

SBUS is a digital serial communication protocol for transmitting remote control signals over a single UART line. It operates at an inverted 100,000 baud rate with an 8E2 (8 data bits, even parity, 2 stop bits) frame structure. Each SBUS packet, sent every 20 milliseconds (50 Hz), encapsulates up to 16 analog and 2 digital channels as 11-bit values, densely packed into a 25-byte frame. We leverage this protocol to transmit decoded UDP control commands to the flight controller, effectively emulating an external SBUS receiver through a custom serial encoder implemented on the ESP32S3.

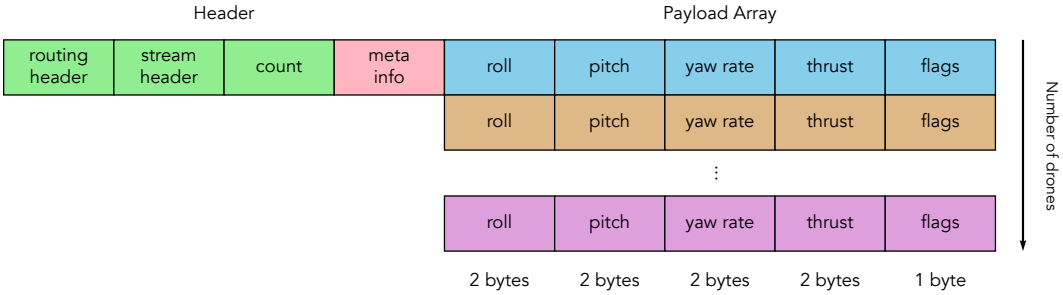


Figure 3: Structure of the UDP broadcast control packet used to command all quadrotors in the fleet. The packet contains a sequence of 9-byte control payloads, each corresponding to a single quadrotor. A payload contains four 2-byte fields and one 1-byte field, encoding remote control and auxiliary flags, respectively.

3 Sample Evaluations

We conclude this paper with experimental profiles showcasing the platform's capabilities in our research applications. We demonstrate its ability to track prescribed trajectories in an indoor motion-capture lab environment.

To evaluate trajectory tracking performance, we conducted tests using circular and figure-eight (lemniscate) patterns at varying speeds to provide insight into the platform's handling characteristics across different dynamic conditions. Figure 4 presents the trajectory tracking results, with horizontal plane visualizations showing executed paths colored by velocity magnitude. The accompanying time-series data illustrate velocity profiles and the corresponding acceleration profiles throughout each maneuver. Results demonstrate expected performance trade-offs between speed and precision, with the platform maintaining stable flight characteristics across the tested conditions. The high-level controller used in these experiments was an LQG controller implementation based on [5].

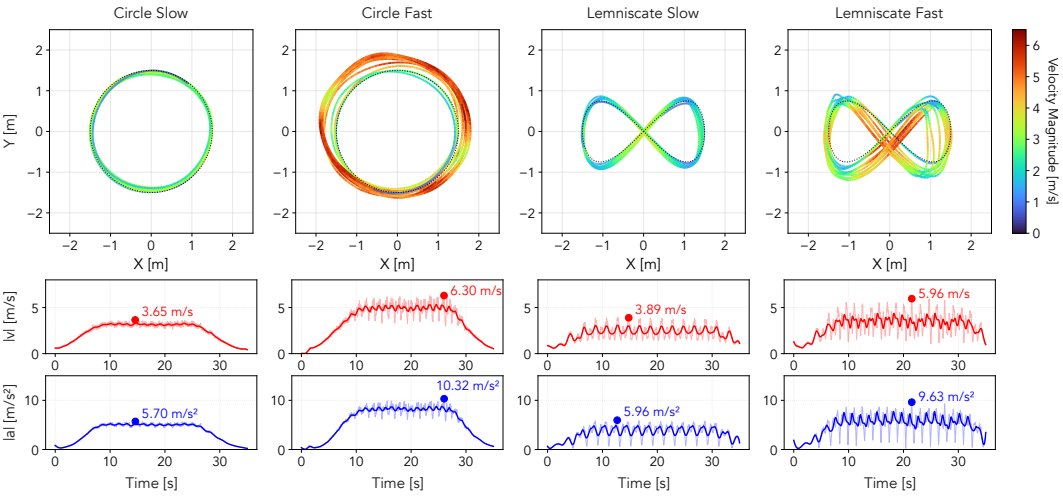


Figure 4: Trajectory tracking results showing executed paths in the horizontal plane colored by velocity magnitude. Time-series plots illustrate velocity and acceleration profiles for each maneuver.

The platform enables large-scale indoor deployments, with testing conducted using up to 20 quadrotors operating simultaneously. Figure 5 shows representative deployments with 8–10 agents flying at speeds near 6 m/s, as well as a still from a 20-agent mission. These snapshots highlight the system's robust multi-agent coordination capabilities in constrained indoor environments.

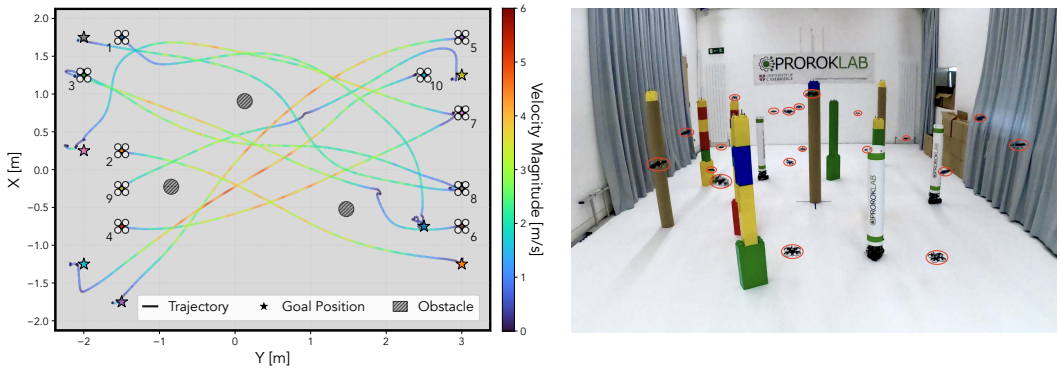


Figure 5: (left) Multi-agent deployment showing an overview plot of 10 quadrotors navigating around obstacles, and (right) a still image from a 20-quadrotor mission, where the locations of the quadrotors are highlighted in red.

References

[1] W. Giernacki, M. Skwierczynski, W. Witwicki, P. Wronski, and P. Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, Aug. 2017. doi:10.1109/mmar.2017.8046794. URL `http://dx.doi.org/10.1109/MMAR.2017.8046794`.

[2] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304, 2017. doi:10.1109/ICRA.2017.7989376.

[3] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig. Amswarm: An alternating minimization approach for safe motion planning of quadrotor swarms in cluttered environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1421–1427, 2023. doi:10.1109/ICRA48891.2023.10161063.

[4] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung. Neural-swarm: Decentralized close-proximity multirotor control using learned interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247, 2020. doi:10.1109/ICRA40945.2020.9196800.

[5] A. Shankar, S. Elbaum, and C. Detweiler. Freyja: A full multirotor system for agile & precise outdoor flights. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 217–223. IEEE, 2021.