TDPS Group Report

Team 44：Cao Yang, He Zhixin, Shen Bowen, Liang Yao, Kai Xin, Feng Zijie

Abstract *——This project aimed to develop an autonomous quadrotor drone system equipped with advanced sensors integration and intelligence capabilities, allowing it to autonomously maneuver and interact within a controlled environment. Utilizing a powerful flight control, OpenMV cameras, an ultrasonic sensor, and optical flow etc, the drone successfully executed tasks including autonomous take-off, arrow and block detection, block pick-up and drop-off, obstacle avoidance, and gate communication. Challenges such as environmental light sensitivity and hardware reliability issues like stability of solder joints were encountered, impacting the precision and effectiveness of the system. The project demonstrated the potential of integrating various technologies to enable intelligent drone flight. It also highlighted the need for future improvements in mechanical design, algorithm adaptability, and communication stability to enhance overall system performance.*

Keywords — Autonomous drone design, sensors integration, OpenMV, Optical Flow, ultrasonic sensor, communication, teamwork

I. INTRODUCTION

A. Literature Survey

Autonomous quadrotor drones have recently become a key research focus, driven by advances in embedded systems, vision processing, and sensor fusion. By integrating flight control, computer vision, and real-time feedback, these drones can now perform tasks like target recognition and autonomous landing with high precision and autonomy. Foundational concepts of quadrotor flight dynamics, including thrust coordination and attitude control, were drawn from the work of Dimililer et al. [1], who detailed the mathematical modeling and stability criteria for quadrotors. This provided us with detailed background knowledge on how to assemble the hardware and implement basic control.

Dynamic submodule for flight control: To realize stable flight, a Proportional-Integral-Derivative (PID) controller was employed for attitude and velocity regulation. Qingqing et al. [2] highlighted the sensitivity of quadrotor performance to PID gain values, noting that poor tuning can lead to instability or sluggishness. Given practical constraints, the PID controller in this project was tuned manually, guided by the heuristic method proposed by Sahrir et al. [3], which proved effective for achieving acceptable stability in constrained testing environments. Furthermore, control strategy design drew from hybrid open-loop and closed-loop approaches. Informed by Chen et al. [4], an open-loop control phase was utilized for initial vertical ascent, followed by closed-loop stabilization to maintain controlled flight during key maneuvers. These studies collectively informed the algorithmic design and hardware configuration implemented in the project, ensuring reliable drone operation in autonomous scenarios.

Visual processing submodule: There are two instances in the project tasks where the visual module is required to guide the drone's movement. The first occurs at the end of Task 2, where the drone needs to identify the arrows on the wall, determine the direction, and move in that direction to enter the next task. The second time, the drone needs to identify the black objects on the green background below and perform positioning and picking up. After moving a certain distance, it needs to identify the cross again and position for landing. Based on this, the drone needs to install two cameras, namely cameras installed at the front and bottom of the drone for analysis. After evaluation by the seller and analysis of the materials [5], we chose OpenMV-H7-PLUS camera module for its compact size, high-performance ARM Cortex-M7 processor (480MHz), and 640×480 CMOS sensor [6]. It supported real-time image processing and Python programming, enabling quick development of vision algorithms. Communication with the drone's main controller is via UART at 115200 baud, ensuring fast, stable data transfer [6].

Other sensors: The reason we ultimately chose the HC-SR04 ultrasonic module as our final solution for obstacle avoidance was mainly due to three factors. First, it could work normally within a range of 2 cm to 4 m [7] from the module, which met our task requirements. Second, as a module that worked with sound waves, it was not affected by light and had good data stability without random fluctuations. Third, the module was economically priced, meeting our economic demands for the task. However, it is worth mentioning that the module did have some drawbacks. It could still interfere with measurement accuracy in the presence of noise, and its measurement range is usually fixed at a certain angle, almost unidirectional. Therefore, there is often only one HC-SR04 module in one direction.

Our project built upon these advanced research studies by combining multiple sensing modalities, such as the optical flow for stable flight, the OpenMV camera for vision-based tasks, ultrasonic sensors for distance measurement, and communication. The integration of these components aimed to create a robust system capable of performing complex tasks autonomously.

B. Scope and Limitations

The scope of our project includes conducting the following tasks in the patio shown in Figure 1:

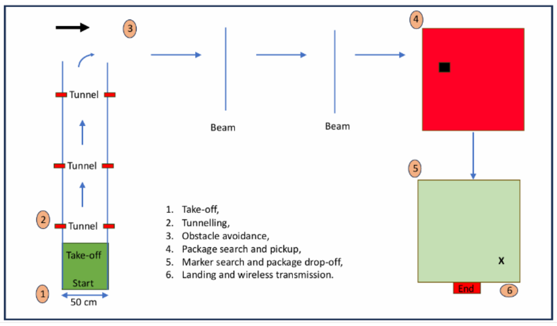

1
2
3
4
5
6
Tunnel
Tunnel
Tunnel
Take-off
Start
50 cm
Beam
Beam
X
End
1. Take-off,
2. Tunnelling,
3. Obstacle avoidance,
4. Package search and pickup,
5. Marker search and package drop-off,
6. Landing and wireless transmission.

Figure 1.Overall Project Tasks

Autonomous Take-off: Implementing stable autonomous take-off using altitude and positional controls.

Tunnel Traversal: Navigating autonomously through a designated tunnel using PID controllers for accurate positioning and stability.

Arrow Recognition: Detecting and interpreting arrow signs to determine direction.

Obstacle Avoidance: Detecting and navigating around three consecutive columns to maintain a collision-free flight path.

Package Search and Pickup: Identifying the target object using visual module and executing precise grasping with the onboard grabbing mechanism.

Marker Search and Package Drop-off: Locating the designated drop-off marker and releasing the package at the correct position based on visual alignment.

Landing and Wireless Transmission: Performing stable autonomous landing and transmitting task completion data wirelessly via the HC-12 module.

While the project completed significant tasks, it also faced several limitations:

Environmental Sensitivity: The performance of the stabilization module, arrow recognition and object detection modules were significantly affected by ambient lighting. Sudden changes in light intensity, especially in outdoor or semi-lit areas, led to decreased accuracy in visual processing.

Mechanical Reliability: Due to the improper selection of soldering components and insufficient soldering skills of the team members at the beginning, issues such as cold solder joints and desoldering frequently occurred during the first few rounds of tests. This led to unnecessary damage to certain components and caused instability during flight. In some cases, short circuits in components even resulted in thick smoke and sparks.

Technical Limitation: During the tunnel traversal task, we were unable to develop a fully robust solution for navigating through the narrow tunnel space.

Variations In Sensor Accuracy: Especially during the task of tunnelling, variations in sensor accuracy occasionally impacted positional stability.

Latency of Data Transfer: Occasional latency in UART communication between the OpenMV module and flight controller caused minor control inaccuracies.

Frequent recalibration of parameters: While structural constraints reduced maneuverability, it required frequent recalibration of control parameters.

Nevertheless, our project successfully demonstrated the integration of sophisticated control strategies and sensor feedback systems, laying a solid foundation for future advancements in drone autonomy.

C. Report Structure

In this report, the “Overall System Design Approach” section will showcase the system-level design of our drone, detailing how each task is achieved within the system. Each subsystem involved in the system-level design will be clearly described in the “Experimental Design- Subsystem Design and Solution(s)” section. Following this, the “Results” section will present the outcomes of conducting tasks with our integrated system. Finally, the “Analysis, Conclusions, and Future Work” section will present our results and design analysis, draw conclusions from the project, and provide recommendations for future work.

II. OVERALL SYSTEM DESIGN APPROACH

This section details the overall system-level design and the technical approach adopted by our team to complete the specified tasks. The system architecture and design are structured to integrate multiple modules, each responsible for specific tasks.

A. System Architecture and Design

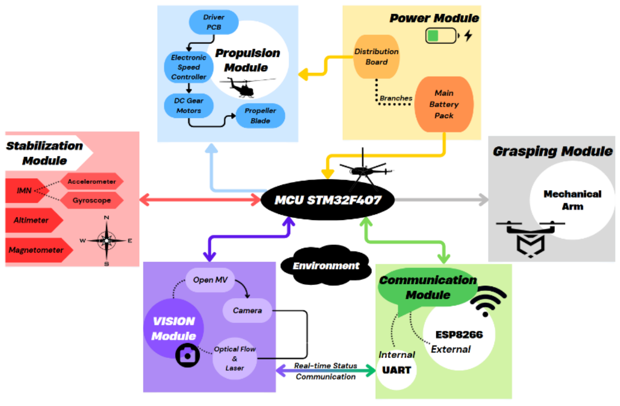


Figure 2. System-level Design Chart

The system-level design and task management structure of our autonomous drone project is shown in Figure 2.

1). Main Control System

Objective: Coordinate each functional submodule to ensure reliable flight execution and maintain stable communication between the flight controller and vision modules.

Method: A robust flight controller based on STM32F407VET6 is deployed as the central processor, executing flight logic using polling methods to sequentially manage tasks such as take-off, path adjustment, and landing. The controller communicates with the OpenMV vision module through UART at 115200 baud, ensuring low-latency, stable command and data transmission.

2). Propulsion and Stabilization Module

Objective: Deliver stable lift and precise attitude control to enable maneuverability in confined and dynamic environments.

Method: Four RS2205 2300KV brushless motors paired with tri-blade propellers are managed by a 4-in-1 BLHeli-S 45A ESC supporting DSHOT600. This setup provides rapid response to control inputs. The PID control loop is tuned to maintain attitude stability during sharp turns and vertical adjustments, such as tunnel traversal or descent beneath obstacles.

3). Obstacle Detection Module

Objective: To detect whether there are obstacles on the forward path of the drone, and then through flight control, adjust the height of the drone to avoid the obstacles to ensure the safe passage of the drone.

Method: An HC-SR04 ultrasonic sensor was installed on the right side of the drone to continuously measure the distance to the objects in front. If an obstacle is detected within the preset threshold (for example, less than 0.5 meters), the flight controller will gradually and dynamically adjust the height to descend and avoid the obstacle. Once the flight is stable, it will directly move forward in a straight line towards the front.

4). Visual Module

Objective: Arrow recognition after tunneling to decide the later direction of drone flight, and object and color detection before the final communication submodule.

Method: Use two OpenMV-H7-PLUS cameras on the front of the drone to distinguish the direction of the arrow and at the bottom of the drone to detect the object, respectively.

5). Grasping Module

Objective: Complete the retrieval and drop-off of a designated object during flight.

Method: A lightweight servo-driven grabbing arm is installed beneath the drone and triggered based on visual confirmation from the bottom camera. The system ensures alignment before executing grasp or release actions and is coordinated with the flight logic for mid-air stabilization during grasping operations.

6). Wireless Transmission Module

Objective: After all tasks are completed, send the attached time, group number, and information that the task has been completed to the designated external receiving module.

Method: Use the HC-12 wireless module equipped on the drone as the transmitting end, operating at a 473 MHz frequency. Send a message every 5 seconds 15 seconds after the task is completed to ensure that the receiving end can stably receive the required message.

Collectively, these integrated modules form a robust, responsive, and adaptable drone system, capable of autonomous flight, stable operation, real-time environmental interaction, and seamless communication, suitable for advanced operational tasks.

B. Project Planning

The streamlined sequence which is shown in Figure 3 ensured that take-off, landing, and flight control tasks are executed in parallel with other module developments, yet testing for take-off, landing, and flight control maintains a finish-to-start relationship with other subsystem tests. Overall, all phases were completed smoothly. However, during the hardware phase, we discovered that hardware debugging required significantly more time than initially anticipated, severely impacting the subsequent project schedule.

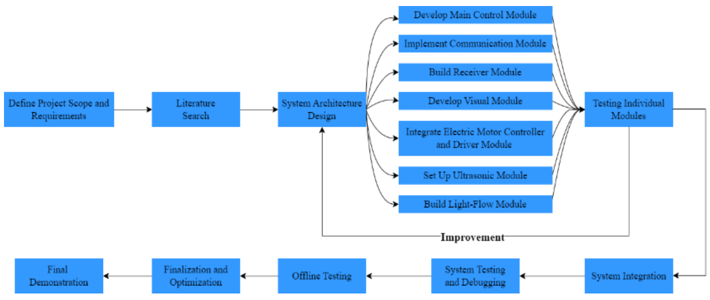

Define Project Scope and Requirements
Literature Search
System Architecture Design
Develop Main Control Module
Implement Communication Module
Build Receiver Module
Develop Visual Module
Integrate Electric Motor Controller and Driver Module
Set Up Ultrasonic Module
Build Light-Flow Module
Testing Individual Modules
Improvement
System Integration
System Testing and Debugging
Offline Testing
Finalization and Optimization
Final Demonstration

Figure 3. Flow Chart of Project Planning

During the planning phase, careful consideration was given to both financial and time constraints. Due to a limited budget, we clearly outlined the components required for the drone, prioritizing essential modules such as the flight controller, motors, and sensors. Recognizing the potential impact of hardware failures and shipping delays on our timeline, we proactively purchased backup units for critical and fragile components to reduce downtime. Ultimately, the project budget was planned to remain around 3000 CNY, allowing for both primary purchases and necessary contingencies while maintaining cost-effectiveness.

III. EXPERIMENTAL DESIGN: SUBSYSTEM DESIGN AND SOLUTION(S)

This section describes the hardware and software implemented in each module.

A. Drone Structure

Our drone's airframe was carefully designed to balance payload capacity, maneuverability, and tunnel navigation compatibility. Standard frames such as the F450, with a 450-millimeter wheelbase weighing between 400 and 600 grams, were too bulky for confined spaces. In contrast, micro-class frames like the Tiny Whoop, typically less than 100 millimeters in wheelbase, lacked the structural integrity and space to carry functional payloads. To address this, we adopted a classic burger-style carbon fiber structure commonly used in racing drones, featuring a 265-millimeter diagonal wheelbase. This configuration strikes a balance between size and performance. The sandwich-style frame integrates vibration-damping foam between interlocked carbon fiber plates, reducing the total weight to 280 grams, about 30 percent lighter than the F330, while increasing joint strength by 200 percent. The compact and robust design enables smooth passage through 60-centimeter-wide tunnels without compromising the mounting space required for vision and grasping modules, ensuring both spatial efficiency and functional versatility.

Based on the chosen airframe, we assembled the complete drone structure, as is displayed in Figure 4.

Four brushless motors with tri-blade propellers provide the main thrust, while individual electronic speed controllers ensure responsive attitude control. The flight controller, housed in a protective shell, works with a bottom-mounted optical flow module for accurate position and velocity estimation. A top-mounted lithium battery delivers stable power to all modules. OpenMV cameras and obstacle sensors are positioned on the front, bottom, and sides, forming the core system for visual recognition and obstacle avoidance. Custom landing gear enhances stability during takeoff and landing and offers a reliable platform for sensor mounting. This modular design ensures the Glo-fly drone is compact, stable, and adaptable for autonomous flight in confined environments.

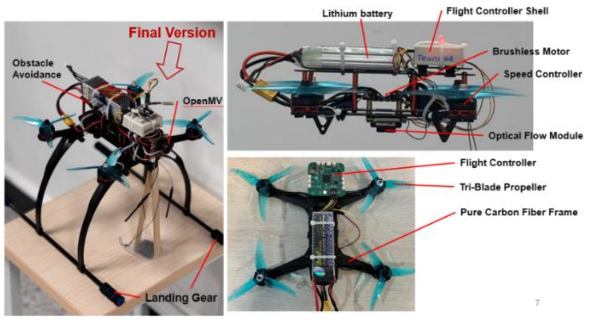


Figure 4. Whole drone's structure

B. Power Supply

Our Glo-fly drone uses a 3-cell 2200mAh Li-Po battery as its primary power source, delivering energy to both propulsion and control systems. At the core of the power distribution is the BLHeli-S DSHOT600 45A 4-in-1 electronic speed controller, which supports 2–6S input and offers precise, low-latency motor control through the DSHOT600 digital protocol. This ESC regulates and distributes power to four RS2205 2300KV brushless motors, a popular choice for FPV racing drones known for their high thrust-to-weight ratio and responsiveness. In parallel, the ESC supplies regulated power to the onboard microcontroller, enabling real-time control signal processing and flight stabilization. This integrated setup ensures efficient power delivery, high-performance thrust control, and reliable operation suitable for high-speed, confined-space navigation tasks. The clearer connection figure is shown in Figure 5 below.

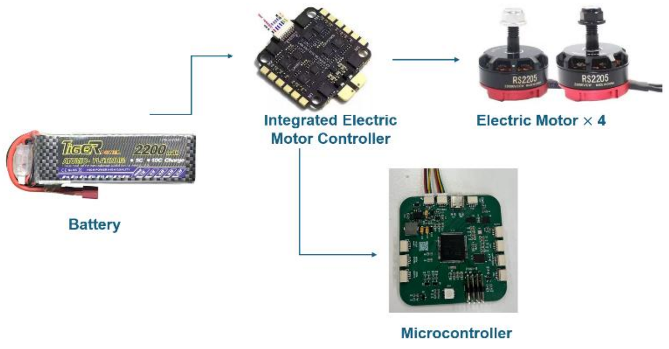


Figure 5. Power supply through the drone system

C. Flight Control PCB Design

Due to initial funding constraints, we sought a cost-effective solution by thoroughly researching the functions of various PCB components and contacting the original developers of an open-source flight controller PCB. Through this collaboration, we were able to customize a version tailored to our needs, effectively reducing the PCB cost to nearly half the price of commercially available alternatives. This approach ensured that our design maintained both functionality and affordability, while meeting the performance demands of the flight control system.

Overall, the STM32F407VET6 acted as the core controller, responsible for running flight algorithms and coordinating sensor data. It featured a high-performance ARM Cortex-M4 core with floating-point support, enabling efficient execution of real-time tasks such as PID control and sensor fusion. Its rich peripheral interfaces (SPI, I2C, UART) supported communication with multiple sensors and motor drivers.

We selected the following sensors to provide comprehensive feedback for flight stabilization:

TEAM 44-Glo-fly

BMI088 (6-axis IMU): Combine a gyroscope and accelerometer to estimate pitch, roll, and yaw, which communicates with the MCU via **SPI**, offering low drift and fast response.

SPL06-001 (Barometer): Provide precise altitude measurements essential for take-off, landing, and vertical control, which communicates over **I2C**.

AK8975 (Magnetometer): Offer absolute heading information to correct gyroscope drift, which also communicates through I2C.

Together, these sensors enabled accurate attitude estimation and height control. Combined with appropriate routing design and noise isolation practices on the PCB, this sensor suite allowed our drone to maintain stable autonomous flight under various environmental conditions.

Based on this understanding, we communicated with the manufacturer to confirm the required functionalities and gained a clearer grasp of the basic principles behind UAV take-off. Accordingly, we customized the PCB design to meet our specific needs and incorporated our team name onto the board. The finalized PCB is shown in Figure 6 below.

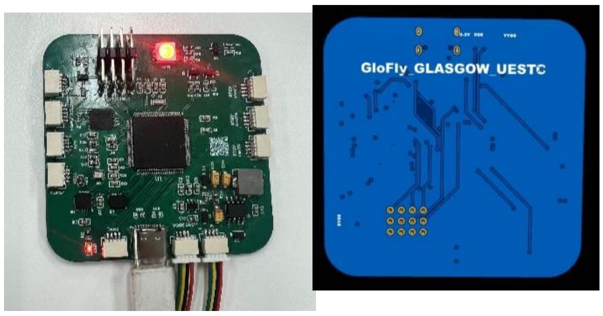


Figure 6: Overall exihibition of our Flight Control and the special design on it

D. Main control

Since the flight control had the ability to deal with other calculations, we used the flight controller as the mainboard as well. It could also save space and weight on the drone, which saved the electrical energy of the battery, meeting the design requirement of sustainable development. The communication method between each sensor and the flight controller was UART due to its ease of software designing. Meanwhile, its full duplex mode enabled it to transfer and receive signals at the same time.

Receiving and Executing Scheme Selection

During the drone's flight, it would simultaneously receive instructions from various sensors, which could lead to information confusion and affect the stability of the aircraft's flight. Therefore, it is necessary to set the priority of each sub-module during the code design process to ensure that each sub-module works in an orderly manner.

At the beginning of the task, the FS-IA6 Receiver received a starting signal from the remote control to indicate the beginning of the project. We shut it down afterwards to save power. During the flight, the Light-Flow Sensor stabilized the flight altitude. Its function was to ensure that the aircraft remained at the same altitude after being set at a certain height. It could be checked for altitude regularly at fixed intervals. The Ultrasound and OpenMV modules detected obstacles and adjusted flight directions in real time. The Driver executed commands to change movement, whose adjustment must be in real time. At last, the Communication module was triggered by a Timer after approximately 4 minutes and 30 seconds. After that, the module was shut down and the total project was completed. Based on the process above, the receiving and executing scheme selection of each submodule is summarized in TABLE I below.

TABLE I. THE RECEIVING AND EXECUTING SCHEME SELECTION OF EACH SUBMODULE.

Scheme Selection	Priority	Real-time	Submodule Sensors
Interrupt	High	High	OpenMV, Ultrasound Sensor, Driver
Timer	Middle	Middle	Light Flow
Loop	Low	Low	Receiver, Communication

Hardware Connections:

The hardware connecting method was described in the following graph. Among them, the direction of the arrows indicated the transmission direction of the signal. A single arrow represented a one-way propagation of the signal, while a double arrow indicated a two-way propagation of the signal. Each arrow represents that the modules on both sides of the arrow will be connected using the corresponding data line, which is shown in Figure 7.

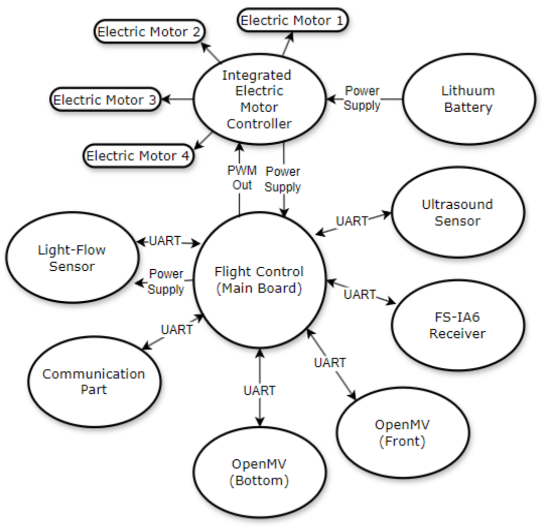


Figure 7. The demonstration of each submodule and its connection

E. Stabilization and Propulsion Module

Take-off and Landing Program Design

Motor Control

Each of the drone's four brushless motors was independently controlled using PWM signals from the flight controller to the ESCs. Proper propeller rotation direction was enforced based on a predefined logic pattern, as incorrect rotation—even with opposing pairs—led to instability.

PID Control

After configuring the motor outputs to ensure correct propeller rotation, the PID controller was manually tuned to achieve stable flight by adjusting motor speeds in response to attitude errors across the roll, pitch, and yaw axes. Following control theory, the inner loop was tuned first, beginning with the proportional gain. Within just three adjustments, a stable flight was achieved. Further tuning was deemed unnecessary, as it risked slower response or degraded performance. During the initial lift-off, the drone showed horizontal oscillation and uncontrolled yaw. Setting the roll-axis Kp=50 reduced oscillation, and increasing Ki to 58 eliminated lateral drift, stabilizing horizontal motion. Finally, adjusting the yaw-axis Kp=23 resolved the self-spin issue. This systematic tuning process enabled precise hovering and orientation, laying the groundwork for more complex tasks like tunnel traversal.

Open-loop position control

To implement autonomous take-off, closed-loop control was first applied to manage the drone's attitude and altitude, ensuring it remained upright and gained height in a stable manner. However, the horizontal (X-Y) position control was initially implemented using an open-loop method, where fixed velocity commands were issued without feedback correction.

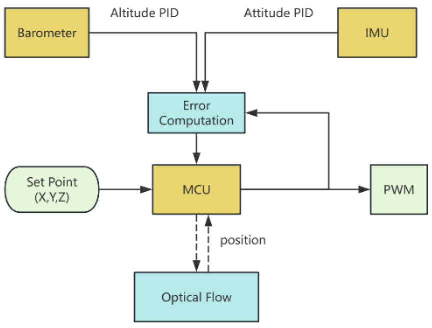


Figure 8. Position control logic design

As illustrated in the flow diagram in Figure 8, this design separates outer-loop position control from inner-loop stabilization. In the initial phase, the optical flow sensor, which is typically used to provide real-time horizontal position feedback, was deliberately disabled as an open-loop control.

This decision reduced the system's sensitivity to environmental factors such as variable lighting and uneven textures, which often degrade optical flow performance. As a result, this strategy allowed progressive tuning and verification. Under this configuration, the PID tuning process was carried out as described earlier. Initial adjustments were performed using a PID debugging interface, allowing for real-time observation and parameter refinement. Subsequently, flight tests were conducted to evaluate the drone's in-air stability. Based on observed behavior, further tuning was applied to enhance control precision and robustness. However, the limitations of open-loop control quickly became evident during testing. For example, external airflow disturbances introduced lateral drift that could not be corrected without positional feedback. Therefore, once inner-loop stability was confirmed, the optical flow module was reintroduced, enabling closed-loop control in the horizontal plane and completing the spatial control loop.

Closed-loop system control

After achieving stable altitude and attitude performance using open-loop position control, the system was upgraded to a fully closed-loop velocity control scheme for vertical take-off. In this configuration, the drone's altitude was continuously measured using onboard sensors, including optical flow, and any deviation from the desired ascent velocity was immediately corrected by the controller.

This dynamic response of the closed-loop system significantly improved overall take-off consistency compared to the open-loop implementation.

Crossing Tunnels

Following the successful stabilization and take-off, the next objective was to enable the drone to autonomously fly through a series of narrow tunnels. As shown in Figure 9, the tunnel dimensions were only slightly wider than the drone, which meant that positional errors had to be kept to a minimum. This task required precise height control and accurate trajectory maintenance throughout the 3-meter flight span separating each tunnel.

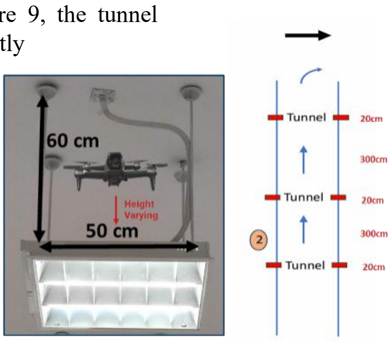


Figure 9. Tunnelling Realization Schematic

To address this challenge, both closed-loop and open-loop velocity control strategies were tested, with the former relying on continuous sensor feedback for adjustment and the latter executing preset velocity commands without correction.

Both approaches were evaluated under flight conditions. However, neither strategy proved fully reliable within the constrained tunnel environment. The closed-loop mode encountered sensor instability, which was likely caused by optical flow noise resulting from height fluctuations or reflections near tunnel surfaces. This led to disordered sensor data and strong oscillations during flight. In contrast, the open-loop configuration provided smoother motion but lacked feedback correction, which caused drift to accumulate over time and ultimately resulted in a collision with the tunnel boundary during one of the trials.

F. *Visual Module*

Arrow Detection

For the implementation of the arrow recognition function, our approach can be summarized as follows: Take a large number of photos of two different arrow directions, conduct training on the dedicated training website of OpenMV, and export the file. Using this file, OpenMV will recognize the arrow direction in real-time and give the probability that the arrow is pointing left or right. The direction with a probability greater than 50% is regarded as the final direction of the arrow, and the task can be completed. The following will show the specific steps of achievement.

Firstly, we use OpenMV connected on the PC to take photos of the right arrow and left arrow separately on OpenMV-IDE, with more than 200 photos of each data set.

Then, to train the arrow recognition model, the dataset was split into two parts: 90% for training and 10% for testing. This data was uploaded to 'Edge Impulse', which provides a mature neural network training pipeline widely validated by existing projects. During training, Edge Impulse invoked its internal neural network algorithms to process the training data and validate performance using the testing set.[8] We found that trained model achieved a testing accuracy of 100%, demonstrating strong generalization on unseen data. Upon completion, Edge Impulse exported the model as a '.tflite' file, encapsulating all learned parameters for deployment.[9] This file was then imported into OpenMV-IDE, where it was used to run real-time inference. The final code outputs the probability of the arrow pointing left or right and selects the direction with a probability exceeding 50% as the predicted result.

Block Detection

The object identification module consists of three key components: target recognition, precise localization and communication with main controller. The following section outlines the design principles and implementation of each component.

Target identification

Target recognition mainly involves computer vision, and the mainstream recognition method is machine learning algorithms which is used in arrow recognition such as CNN [10,11]. However, given the device's limited computing power and the simplicity of the target as shown in Figure 10, machine learning algorithms were omitted in favor of a more efficient traditional method. Instead, color recognition and shape recognition methods were used.

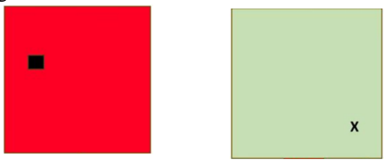


Figure 10. Identified targets
(black rectangles indicate objects, black crosses indicate landings)

The recognition algorithm built using OpenMV mainly consists of the following two steps [12]. The first step is to identify the target area, such as the red and green areas shown in Figure 10 above. Based on this, the object and landing target are identified. Area identification is performed in a loop, while object and landing identification are performed in interrupt mode. The main structure of the code is shown in Figure 11 below.

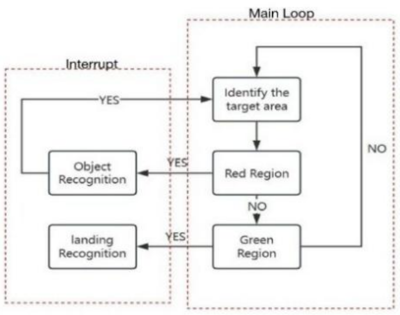


Figure 11. Code framework for the Target identification

Red & Green LAB Detection

To accurately identify targets, the camera captures RGB images and converts them into the LAB color space. LAB separates color information from brightness, making color recognition more stable and less affected by lighting changes [12]. Through experiments, we determined threshold ranges for red and green targets in the LAB space—for example, red targets have L values between 30 and 80, A values between 40 and 80, and B values between 20 and 60. Using these thresholds, the image is filtered to isolate pixels matching the target color. We then use OpenMV's find_blobs() function to detect color blobs within these ranges.

Shape Recognition & Area Size Detection

To ensure detected blobs correspond to actual targets, we further filter by requiring the blob area to be larger than 1000 pixels and the width-to-height ratio to be between 0.8 and 1.2, filtering out small or irregular shapes. For the landing marker, which is a black cross, we detect black regions by thresholding LAB values accordingly and confirm the shape by checking the area proportion of the cross to ensure accuracy

Color & Shape Recognition

The second step is the tracking of the black object and the tracking of the landing mark after the area has been successfully recognized. The main idea is the same as the identification of the area, which is both targeting through the method of color identification first, followed by shape identification. For shape recognition, the black object in the capture task uses the same rectangular recognition code as for region recognition, with a tolerance of 20%.

Landing Mark Tracking

For landing marks, the shape recognition of black crosses has been supplemented with an area restriction in addition to the width-to-length ratio restriction. The black cross has an area that accounts for approximately 45% of the rectangle with the cross as its side. Based on this, an area restriction with an upper and lower tolerance of approximately 15% is applied.

Filtering algorithm design

To improve the stability of recognition results, we incorporate a filtering algorithm. Due to drone vibrations and lighting variations, recognition data can jitter or exhibit sudden jumps. We first discard any detected positions that differ from the previous frame by more than 15 pixels, treating them as outliers. This is a simple maximum threshold filtering, which to some extent avoids the influence of minor jitter.

Then, a simplified Kalman filter is applied to smooth the target position over time. The Kalman filter predicts the next position based on previous estimates and corrects it with new measurements, reducing noise and providing a stable, reliable output [13]. This filtered position data is essential for precise drone navigation and control. The core formula is shown in Figure 12 below.

$$\hat{x}_t^- = F\hat{x}_{t-1} + Bu_{t-1} \qquad K_t = P_t^- H^T \left(HP_t^- H^T + R\right)^{-1}$$

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - H\hat{x}_t^-)$$

$$P_t^- = FP_{t-1}F^T + Q \qquad P_t = (I - K_t H)P_t^-$$

Figure 12. Kalman filter formula

The left side of the Kalman filter shows the predicted value of the highest priority and its covariance, while the right side, from top to bottom, respectively presents the Kalman coefficient, the maximum estimated value and its covariance. The core principle is to continuously output the optimal estimate value through the iterative update of the left and right formulas: The prior estimate on the right side is obtained by combining the optimal estimate at the previous moment with the control signal, and calculated through the state transition matrix F and the control matrix B. The optimal estimation on the left side is obtained by fusing the prior estimation with the measured value zt and the measurement transition matrix H through the Kalman coefficient. This method has a small amount of calculation and good prediction effect. In practical applications, by adjusting hyperparameters such as the measurement error variance R and the prediction error variance Q, and combining with specific scenarios, efficient prediction of the target value can be achieved [13,14].

Target location

After detecting the target, we calculate its position relative to the drone. The center of the camera image corresponds to the drone's forward direction, and the target's position is given in pixel coordinates. Experiments show that at a flight altitude of 1.2 meters, the camera's field of view is approximately 70 degrees, and each pixel corresponds to about 1.5 centimeters in the real world. By calculating the pixel offset between the target center and the image center and multiplying by 1.5 cm, we obtain the horizontal and vertical distances of the target relative to the drone. This localization information helps the drone adjust its flight attitude and approach the target accurately.

Communication

The camera module communicates with the drone's main control board via UART serial interface at a baud rate of 115200, ensuring fast and stable data transmission. The detected target position is sent as a string, upon receiving this data, the main control board parses the coordinates and adjusts the drone's flight direction and distance accordingly. The camera sends data 10 times per second to ensure real-time responsiveness to target position changes.

G. Grab Module Design

The drone's grasping module usually adopts a complex mechanical structure [15,16]. This design follows the principle of simplicity and efficiency, and its design is as follows. It is made from 5 mm thick lightweight plywood cut into an equilateral triangle with 15 cm sides. Double-sided tape is applied along the edges to provide adhesive force for grabbing objects. The grabbing module is mounted under the drone using 3D-printed plastic brackets designed to be sturdy without affecting flight performance.

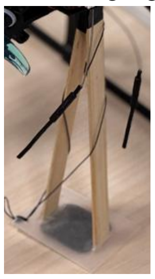


Figure 13. Grab Module

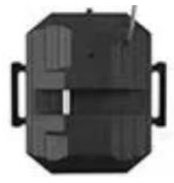


Figure 14. Release Device

To release grabbed objects, a small servo motor (model SG90) is used as the release mechanism. The drone's main control board controls the servo via PWM signals, rotating it 90 degrees to open the release hook and drop the object. The servo responds quickly, completing the action within about 0.2 seconds, ensuring a fast and reliable release.

This grabbing system is simple, lightweight, and effective, meeting the drone's task requirements for grasping and releasing objects.

H. Obstacle Avoidance

In the initial design of the obstacle avoidance module, we considered infrared, ultrasonic, and vision-based detection methods. Infrared was initially favored for its simplicity and low cost, but tests revealed major flaws: it was highly sensitive to light,[17] unreliable under sunlight, had poor accuracy, a limited range of about 0.5 meters, and failed entirely with dark or light-absorbing surfaces.[18] We then explored a vision-based solution using OpenMV, but image processing proved too computationally intensive, and the module was already assigned other tasks. Budget constraints also made additional hardware unfeasible. Ultimately, we chose the **HC-SR04 ultrasonic sensor** for three key reasons: it offers a wide detection range from 2 cm to 4 meters,[19] is unaffected by lighting conditions, and provides stable and accurate readings. Moreover, it is inexpensive, easy to integrate, and well-documented. However, it has limitations: it may be affected by acoustic noise and has a narrow detection angle, typically requiring one module per direction.

Working Principle of the HC-SR04 Module

The core working principle of this module is to determine the distance of the obstacle from the drone by calculating the time taken for the ultrasonic wave to be transmitted and received. There are a total of four pins for operation which are illustrated in Figure 17, namely *VCC, Trig, Echo, and GND*. When the module is powered on, the controller sends a 10-20us high-level pulse to the Trig pin. After the module detects the rising edge, it automatically sends 8 cycles of 40kHz [20] ultrasonic waves to the outside. Subsequently, when the ultrasonic wave encounters the obstacle and reflects, the receiving transducer captures the echo signal and processes it through the internal amplification and filtering circuit. Then, the distance between the drone and the obstacle can be calculated using the formula. The calculation formula (1) is as follows:

$$d = \frac{v\cdot(t-\Delta t_{circuit})}{2}(unit{:}m) \qquad (1)$$

where $\Delta t_{circuit}$ represents the fixed delay of the circuit (usually less than 100 microseconds) [21], d represents the target distance, and v represents the speed of sound.

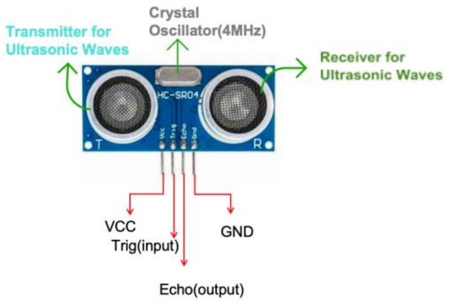


Figure 15. HC-SR04

Specific functional implementation

When the measured distance falls below the minimum safe threshold set in the program, the drone is instructed to move away from the obstacle. At the beginning of its flight path, the drone typically starts at a higher altitude and may be positioned close to an obstacle. To handle this, an ultrasonic sensor is placed in the forward direction. If it detects insufficient clearance, the drone descends until the measured distance exceeds the minimum allowable value. Once a safe distance is confirmed, the drone continues forward along a fixed straight path. The code ensures that the drone descends adequately, typically to a height between 0.5 and 0.8 meters, to avoid the obstacle while maintaining an altitude of at least 2.5 meters above the ground, as required by the task. Flowchart Figure 16 illustrates this control logic.

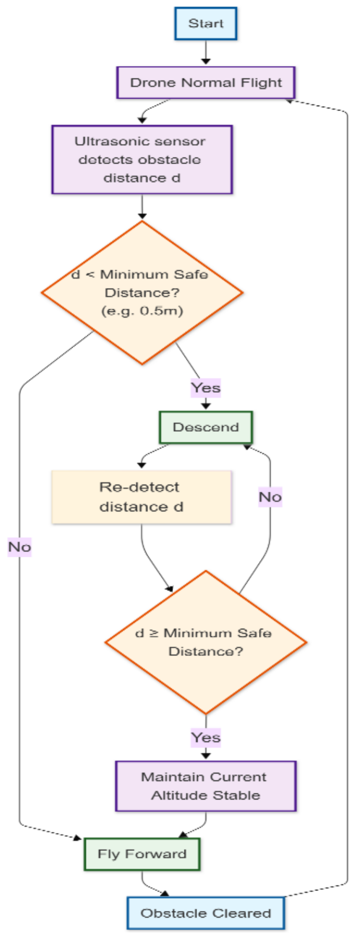

Start
Drone Normal Flight
Ultrasonic sensor detects obstacle distance d
d < Minimum Safe Distance? (e.g. 0.5m)
Yes
No
Descend
Re-detect distance d
No
d ≥ Minimum Safe Distance?
Yes
Maintain Current Altitude Stable
Fly Forward
Obstacle Cleared

Figure 16. Flowchart

I. Gate Communication

Due to project requirements and hardware compatibility constraints, we were limited to using the HC-12 wireless serial communication module for long-range data transmission. HC-12 was chosen as a reliable and low-power solution for point-to-point communication between the drone and external devices. Its picture is shown in Figure 17.

Figure 17. HC-12

Assembly of the Module and Principle of the HC-12

The HC-12 has two main components: the module itself and its accompanying antenna. We manually soldered both the antenna and pin headers to complete the assembly. To maximize transmission efficiency, the antenna was positioned horizontally during installation.[22]

The core working principle of the HC-12 module is to use wireless radio frequency signals instead of physical lines to achieve serial data transmission between devices. The working pins consist of four key interfaces: VCC, TXD, RXD, and GND. When the module is powered on, the controller sends serial data (such as sensor readings) to the RXD pin. The radio chip inside the module will automatically convert the data into radio waves in the 433 MHz frequency band [23] and transmit them into the air. Subsequently, when the other HC-12 module at the remote end receives the wireless signal, it will decode and demodulate through the internal circuit to restore the original data and output it from its TXD pin, ultimately achieving two-way communication.

Specific functional implementation

The HC-12 module's code design centers around two core functions: message transmission and timeline control. The transmission function handles packaging and repeatedly sending messages to ensure a successful reception. Although simple to implement, reliable communication is enhanced by using a looped send mechanism. Additionally, since HC-12 supports half-duplex transmission, both ends can alternate roles as sender and receiver. To improve efficiency, a 3ACK strategy is used, ensuring message acknowledgment without excessive retransmission, thus optimizing bandwidth and processing load.

The second function is timeline management. The code estimates the expected start time of the demonstration and sets it as the base reference. Once initiated, the timeline progresses in sync with real time. After completing all tasks except Task 6, the system waits approximately 15 seconds for an external trigger (from the flight controller) to begin data transmission. Each transmitted message is time-to-do meet real-time communication requirements.

Verification

Regarding the verification part, we prepared two hc-12 modules in advance. One was used as the transmitting end and the other as the receiving end. To ensure that they could work properly together, we first needed to ensure that the working frequencies of these two modules were consistent. Since they were all set to the default initial working frequency of 433 MHz when purchased, we initially did not perform any debugging but directly used them. However, after determining the test classroom and the given working frequency, we specifically adjusted both modules to 473 MHz. Regarding the adjustment method, we first connected them to the computer using a USB to TTL converter and then used a dedicated serial port debugging assistant to modify their default working frequencies. After consulting the literature, it was found that the frequency conversion was achieved by sending the AT command AT+C100. Subsequently, after burning the code onto the transmitting end (in conjunction with the STM32 development board), the receiving end did not require any additional processing. We successfully completed the verification of the function.

IV. RESULTS

A. Stabilization and Propulsion Module

After PID tuning and initial open-loop tests, the drone achieved stable flight but experienced drift due to the absence of optical flow feedback. Enabling the optical flow sensor enabled full closed-loop control, significantly improving stability and enabling consistent autonomous take-off.

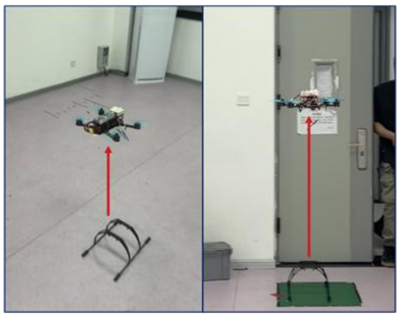


Figure 18. Successful take-off

As shown in TABLE II, all 10 closed-loop take-off attempts executed, with 8 successful flights. The two failures were traced to battery undervoltage and blade wear, which caused insufficient thrust and asymmetric lift, respectively.

TABLE II

Target	Trial time	Successful time	Successful Ratio
Open-loop	7	/	/
Closed-loop	10	8	80%

Tunnel navigation required high precision, as altitude deviations beyond ±5 cm caused failures in the 60 cm-high tunnel. Although the drone succeeded in passing the first tunnel in optimal conditions, inconsistent altitude control—due to sensor noise and control delays—led to unreliable performance across the full tunnel sequence.

B. Vision Module

Arrow Detection

Finally, we manually drew left and right arrows on the Pad and used OpenMV for real-time recognition. The command terminal would output the direction of the arrows in Chinese in real time, as shown in the figure below. This proves the successful implementation of this module, as shown in Figure 19, which is an example of the right direction.

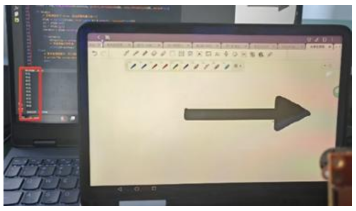


Figure 19. Initial test for arrow direction

Building on this success, we mounted a recognized arrow image in front of the drone for in-flight testing. The results demonstrated that when the drone approached within one meter of the arrow, it could reliably identify the direction and correctly turn accordingly.

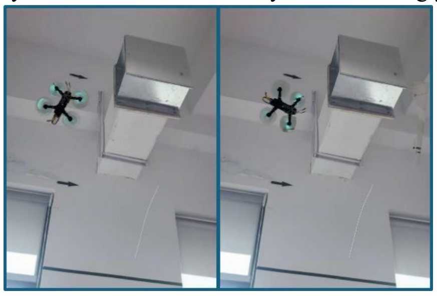


Figure 20. Successfully achieved the recognition of the right arrow

Object Recognition

The visual module has undergone multiple optimizations. Its initial test condition was to imitate the black squares in the red curtain built in the test site. The test results are shown in Figure 21 as follows.

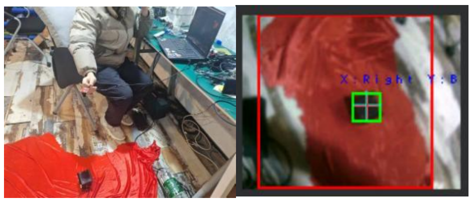


Figure 21. Initial test result

In the scenario of recognizing black squares against a simulated red background, the visual module uses color thresholds (red LAB threshold: [30, 100, 15, 60, 15, 60], green LAB threshold: [30, 100, -60, -10, 15, 60] and shape constraints (with a width-to-height ratio tolerance of 20%) are used to achieve target area detection. The initial test shows that the success rate of target area recognition in the static environment reaches 100%, but there is approximately 15% coordinate offset error in dynamic tracking.

In a field environment containing obstacles and dynamic disturbances, 10 repeated tests were conducted which are shown in Figure 22 below.

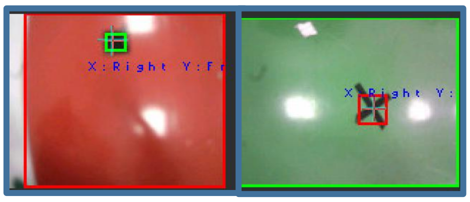


Figure 22. Test results of the test site

TEAM 44-Glo-fly

On this basis, a total of ten tests were conducted at the test site. The success rate and average time consumption are shown in TABLE III below.

TABLE III. TEST RESULT STATISTICS OF VISION MODULE

Target	Test Times	Accuracy	Average Test Time
Black Box	10	60%	2m14s
Landing Mark	10	50%	1m58s

Further analysis shows that after Kalman filtering optimization, the coordinate prediction error of target tracking is reduced by approximately 30%, but misidentification still exists in complex backgrounds (such as interference objects similar in shape to the black square).

C. Grab Module

Grasping performance

The grasping device based on the triangular mechanical structure and double-sided tape design successfully adhered to the target objects in six contact grasping tests, achieving a success rate of 70%. A slight offset always occurs during each grasping attempt. The device has the best grasping stability for objects under 100g. When the weight exceeds this limit, it may fall off due to a shift in the center of gravity.

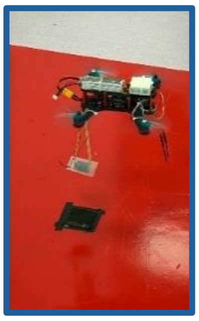


Figure 23. Test of Grab Module

Figure 23 above shows the testing process of the grasping module in the field.

Release function

The release device adopts a signal-triggered lock structure. However, due to signal interference and the stability of mechanical connection, only two of the five release tests were successful, with a success rate of 40%. Among the failed cases, three times the trigger was delayed due to the communication delay of UART, and one time the mechanical lock was stuck and failed to unlock.

The number of its tests is calculated based on the success of the visual test. Its statistical data are shown in TABLE IV as follows**.**

TABLE IV. TEST RESULT STATISTICS OF GRAB MODULE

Function	Test Times	Accuracy
Grab	6	70%
Release	5	40%

D. Obstacle Avoidance

According to the result display diagrams, we can see that after the aircraft recognized the arrow, it hovered at a high altitude for a period. The ultrasonic module then detected an obstacle (beam), triggering the descent procedure. After descending to a safe height, the drone was commanded to move forward in a straight line, successfully completing Task 3. As shown in the figure, the drone accurately detected the obstacle ahead and flew underneath it, effectively demonstrating the obstacle avoidance functionality.

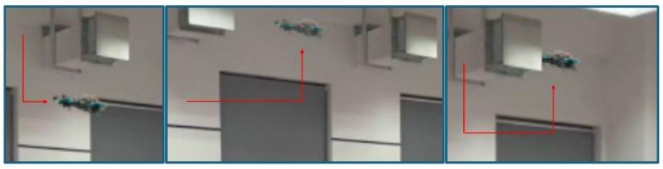
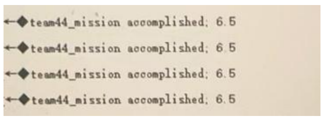


Figure 24. Finish the process of obstacle avoidance

E. Gate Communication

As shown in the figure, the HC-12 module on the sending end can successfully send information to the HC-12 module on the receiving end, and the computer on the receiving end can also successfully receive the sent message. The message contains the information to be transmitted, time, task completion, and group number.



```
←◆team44_mission accomplished; 6.5
←◆team44_mission accomplished; 6.5
←◆team44_mission accomplished; 6.5
←◆team44_mission accomplished; 6.5
```

Figure 25. Successful communication

F. Overall Result

Our team has made substantial progress in implementing the drone's functionalities. TABLE V summarizes the success rates of all tests, along with the possible causes of any errors encountered.

TABLE V. OVERALL RESULT PERFORMANCE

Required tasks	Success Rate	Potential Causes of the Error
Take-off	80%	Different environmental condition
Tunneling	30%	Technical lack in positioning
Arrow Detection	100%	/
Obstacle Avoidance	50%	Different environmental condition
Block Pick-up & Drop-off	50%	Latency in uart communication
Search & Landing	70%	Latency in uart communication

V. ANALYSIS, CONCLUSIONS & FUTURE WORK

Our core deficiencies lie in the insufficient stability of tunnelling, high misjudgment rate in dynamic target recognition, and low reliability of the grasping and releasing mechanism. The fundamental bottlenecks of these problems stem from multiple constraints: environmental sensitivity causing continuous weakening of navigation accuracy and visual recognition stability due to airflow disturbance and background interference; hardware shortcomings manifested as sudden power loss caused by battery management defects, frequent jamming of mechanical locks, and UART communication delay affecting command response; and algorithm limitations where the visual module relying on traditional LAB color thresholds and shape constraints has weak anti-interference ability in dynamic scenes. The mutual coupling of environmental noise, hardware failures, and algorithm defects further amplifies the overall performance degradation of the system.

Our core achievements include completing the full-process verification of closed-loop autonomous takeoff, arrow guidance turning, basic obstacle avoidance, and wireless communication links. We have achieved phased success in basic functionality, but the three bottlenecks of environmental adaptability, hardware reliability, and dynamic scene accuracy restrict the practical application process. In the future, we need to focus on breakthroughs in sensor fusion algorithms and anti-interference design and enhance the system's fault tolerance through modular redundancy.

To provide more reliable solutions for tunnelling, future work may focus on enhancing sensor stability and redundancy. This could include sensor fusion techniques involving optical flow, downward-facing LiDAR, or stereo depth cameras, as well as real-time SLAM modules for spatial awareness. Additionally, incorporating vision-based detection for tunnel entrances and adaptive trajectory correction may significantly increase task success rates.

TEAM 44-Glo-fly

REFERENCES

[1] R. Toulson and T. Wilmshurst, Fast and Effective Embedded Systems Design: Applying the ARM Mbed. 2012.

[2] A. Meroth and P. Sora, Sensor Networks in Theory and Practice: Successfully Realize Embedded Systems Projects. 2023. DOI: 10.1007/978-3-658-39576-6.

[3] H. Budzier, G. Gerlach and D. Müller, Thermal Infrared Sensors: Theory, Optimisation and Practice. 2011. DOI: 10.1002/9780470976913.

[4] Z. Wei-Peng, C. Li-Sha, L. Er-Min and Z. Feng-Chun, "Design and Production of Tracking System based on OpenMV Image Recognition," 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2020, pp. 1198-1202, doi: 10.1109/ITOEC49072.2020.9141560.

[5] Z. Wei-Peng, C. Li-Sha, L. Er-Min, and Z. Feng-Chun, "Design and production of tracking system based on OpenMV image recognition," Proc. 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2020, pp. 1198–1202, doi: 10.1109/ITOEC49072.2020.9141560.

[6] OpenMV, "OpenMV Cam H7 Plus," OpenMV Store, [Online]. Available: https://openmv.io/collections/cameras/products/openmv-cam-h7 plus?variant=31180638224478 [Accessed: Jun. 22, 2025].

[7] SparkFun Electronics, "Ultrasonic distance sensor - HC-SR04," SparkFun, 2025. [Online]. Available: https://www.sparkfun.com/ ultrasonic-distance-sensor-hc-sr04.html

[8] OpenMV, "OpenMV - Edge Impulse Documentation," Edge Impulse Docs, 2025. [Online]. Available: https://docs.edgeimpulse.com/docs/runinference/running-your-impulse-openmv [Accessed: Jun. 22, 2025].

[9] redcrow, " Image Classification with EdgeImpulse – poor performance," OpenMV Forums, Sept. 11, 2021. [Online]. Available: https://forums.openmv.io/t/image-classification-with-edgeimpulse-poorperformance/6332 [Accessed: Jun. 22, 2025].

[10] J. Weon, T. Yong and J. Kim, "Camera-based Virtual Drone Control System Using Two-handed Gestures," 2024 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom), Hong Kong, China, 2024, pp. 291-296, doi: 10.1109/MetaCom62920.2024.00054.

[11] J. Kim, Q. Zhang, E. T. Matson and M. Y. Wang, "Improving Drone Classification with Audio-Derived Visual Features: A Vision Model Comparison," 2024 Eighth IEEE International Conference on Robotic Computing (IRC), Tokyo, Japan, 2024, pp. 41-45, doi: 10.1109/IRC63610.2024.00013.

[12] Z. Wei-Peng, C. Li-Sha, L. Er-Min and Z. Feng-Chun, "Design and Production of Tracking System based on OpenMV Image Recognition," 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2020, pp. 1198 1202, doi: 10.1109/ITOEC49072.2020.9141560.

[13] Z. -x. Liu, W. -x. Xie and P. Wang, "Tracking a target using a cubature Kalman filter versus unbiased converted measurements," 2012 IEEE 11th International Conference on Signal Processing, Beijing, China, 2012, pp. 2130-2133, doi: 10.1109/ICoSP.2012.6492002.

[14] C. Liu, P. Shui and S. Li, "Unscented extended Kalman filter for target tracking," in Journal of Systems Engineering and Electronics, vol. 22, no. 2, pp. 188-192, April 2011, doi: 10.3969/j.issn.1004-4132.2011.02.002.

[15] J. Wang and H. Peng, "Object Grabbing of Robotic Arm Based on OpenMV Module Positioning," 2023 2nd International Conference on Artificial Intelligence and Computer Information Technology (AICIT), Yichang, China, 2023, pp. 1-4, doi: 10.1109/AICIT59054.2023.10277796.

[16] A. Alsawy, A. Hicks, D. Moss and S. Mckeever, "An Image Processing Based Classifier to Support Safe Dropping for Delivery-by-Drone," 2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS), Genova, Italy, 2022, pp. 1-5, doi: 10.1109/IPAS55744.2022.10052868.

[17] Safe and Sound, "Infrared sensor: Definition, how it works, and its applications," Safe and Sound Security, May 2025. [Online]. Available: https://getsafeandsound.com/blog/infrared-sensor/

[18] P. Kumar, "IR sensor: Working principle, types and applications," Electronics Hub, Mar. 2024. [Online]. Available: https://www. electronicshub.org/ir-sensor/

[19] SparkFun Electronics, "Ultrasonic distance sensor - HC-SR04," SparkFun, 2025. [Online]. Available: https://www.sparkfun.com/ ultrasonic-distance-sensor-hc-sr04.html

[20] SparkFun Electronics, "HC-SR04 ultrasonic sensor datasheet," SparkFun, 2025. [Online]. Available: https://cdn.sparkfun.com/datasheets/ Sensors/Proximity/HCSR04.pdf

[21] B. Huang, "Getting started with the ESP8266 ESP-01," Instructables, Jan. 2016. [Online]. Available: https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01/

[22] Elecrow, "HC-12 wireless serial port communication module user manual V2.3," Elecrow, 2025. [Online]. Available: https://www.elecrow.com/ download/HC-12.pdf

[23] M. Khan, "Understanding and implementing the HC-12 wireless transceiver module," All About Circuits, Nov. 2016. [Online]. Available: https://www.allaboutcircuits.com/projects/ understanding-and-implementing-the-hc-12-wireless-transceiver-module/