Simulation and Design of The 2U CubeSat Project

Jiale Shi (石家乐)、Yujie Xiao (肖宇杰)、Zhiqing Yang (杨智钦)、Guanqi Li (李冠祺)、Yuqing Peng (彭毓卿)、Jingyi Zhou (周景怡)

***Abstract* – By establishing a three-axis and six-degree-of-freedom physical model, this project simulates part of 2U cubic star in orbit. The structure of the CubeSat was designed using Solidworks. Stress analysis and thermal analysis were performed using Solidorks Simulation. Combined with magnetometers and gyroscopes, the corrected nine-axis information was obtained. Attitude self-stabilization was achieved by controlling the three-axis motors through PID control. Wireless transmission of location information, six-axis information, and image data was accomplished using LoRa modules, GPS modules, and camera modules. So as to realize cubic stance since the stability, scan to determine the objectives and target tracking three core missions.**

Key Words: 2U CubeSat, CH32V307, Simulation Analysis, PID Control, Sensors.

I. BACKGROUND

A. Background and Significance of the Project

With the deepening of aerospace research, the demand for low-orbit space exploration and Earth observation missions continues to increase. Traditional satellites have drawbacks such as high cost and long development cycles. CubeSats are a low-cost spacecraft platform with multi-payload adaptability, suitable for such low-orbit space missions. Currently, CubeSat technology is rapidly developing and gradually being applied in practical scenarios[1].

CubeSats first appeared as part of a scientific research project at two American universities (California Institute of Technology and Stanford University) in 1999. The concept of CubeSats was proposed by Stanford University, specifying a mass of 1 kilogram and structural dimensions of 10 cm × 10 cm × 10 cm as one unit, making CubeSats a universal standard for nanosatellites[2].

Depending on mission needs, CubeSats can be expanded to 2 units, 3 units, 6 units, or even 12 units. Their emergence is attributed to advances in microelectronics technology, the development of lightweight materials, the advent of high-power solar cells, and the reduction in costs of satellite core components. CubeSats represent the modularization and standardization era of the micro-nano satellite industry.

Fig. 1. CubeSats Expansion Structure

In recent years, CubeSats have developed rapidly, with over 300 CubeSats launched in 2017 and over 800 in 2018. It is estimated that by 2020, more than 500 launches will occur annually, with the entire micro-nano satellite industry exceeding $2 billion, becoming an important branch of on-orbit spacecraft. Since their first launch in 2003, CubeSats have entered many practical application fields and were listed in 'Science' magazine's 'Top Ten Scientific Breakthroughs of 2014'[3].

B. Application Scenarios

CubeSats have many applications in the field of ocean observation, including the following aspects:

1) Marine Weather Forecasting: CubeSats can be equipped with meteorological sensors to observe parameters such as sea surface temperature, humidity, wind speed, and direction, providing accurate marine weather forecasting services through data analysis and model prediction[4].

2) Marine Environmental Monitoring: CubeSats can carry water quality sensors to monitor parameters such as seawater temperature, salinity, turbidity, nutrient content, and harmful algae, as well as sea surface conditions, wave height, and tides, providing data support for marine environmental protection and resource management.

3) Marine Resource Exploration: CubeSats can be equipped with high-resolution multispectral cameras to explore marine resources such as seabed minerals and fishing grounds, as well as conduct research on marine ecosystems and biodiversity conservation.

4) Maritime Traffic Management: CubeSats can carry AIS (Automatic Identification System) sensors to monitor maritime traffic in real-time, improving the efficiency and safety of maritime traffic management.

5) Marine Disaster Emergency Response: CubeSats can monitor and provide early warnings for marine disasters such as storm surges, tsunamis, and oil spills by carrying high-resolution optical sensors and radar sensors, offering data support for emergency response and rescue.

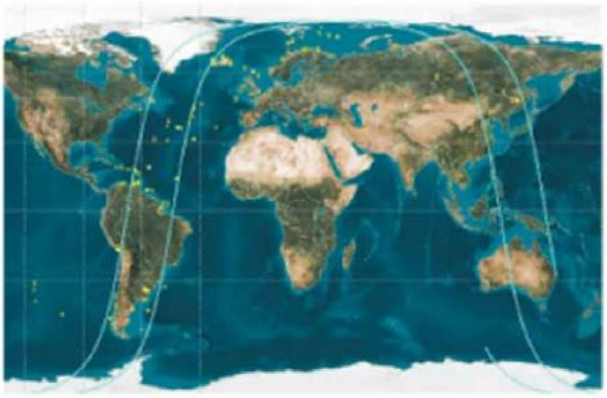


Fig. 2. CubeSats Marine Traffic Management

Fig. 3. CubeSats Networking System

C. Development Trends of CubeSats

With continuous technological advancements, CubeSats' functional density is increasing, and their capabilities are becoming more robust. Micro-nano satellites can form constellation formations, creating virtual large satellites to meet the growing and complex needs of space missions such as communications, remote sensing, Earth observation, and scientific experiments.

Therefore, in recent years, the design and application of CubeSats have gained significant attention from countries worldwide. With technological advancements, CubeSats have also become important tools for future space experiments and space exploration.

The development trends of CubeSat technology are as follows:

1) From Experimental Stage to Application Stage: CubeSats have begun transitioning from the experimental stage to the application stage, with increasing investment from governments and the military. In the early stages, CubeSat research was primarily conducted in higher education institutions, mainly for education, teaching, and technology demonstration and verification. As CubeSat technology develops, national governments are starting to support and research CubeSats for specific payload missions, especially the U.S. military, which highly values CubeSat development and invests in multiple CubeSat research projects[5].

2) Acceleration of Networking Process: Leveraging the inherent advantages of CubeSats, the networking process has accelerated, and application models continue to expand. Based on certain application requirements, CubeSats use single or multiple orbits and operate according to predetermined configurations. This improves instantaneous coverage areas, enhances time resolution, and achieves complex functions that traditional large satellites cannot. Additionally, relying on network systems, swarm operation modes, and redundancy design improves the reliability of all satellite systems.

3) Cost Reduction: Using commercial devices significantly reduces the research and production costs of CubeSats. Currently, in-depth research on the space application technology of commercial devices is being conducted abroad, including component screening, electromechanical thermal design, radiation resistance, and test verification. As the functionality of commercial devices improves and space application technology develops, their use in CubeSats will become increasingly common.

4) Cubesat's Intelligence: As CubeSat systems become more complex, their application and management also become more complex. Therefore, enhancing the intelligence and autonomous management of CubeSats is a crucial direction in modern CubeSat technology development.

D. Project Introduction

Therefore, we plan to design a 2U Cubesat with good structural strength and thermal control performance from four aspects of structure, circuit, hardware and software, and capable of achieving three tasks: attitude self-stabilization, target scanning and target tracking. In terms of structure, we will carry out material selection, static stress analysis and thermal analysis, and conduct a comprehensive evaluation and verification of the satellite structure to ensure the reliability and stability of the satellite. In the circuit, the use of a variety of software for circuit simulation and circuit board drawing; In terms of software and hardware, this project is divided into information monitoring attitude measurement module, accurate positioning module, real-time shooting module, wireless transmission module, etc. Finally, through module combination and algorithm coordination, attitude self-stabilization, target scanning and target service is realized.

II. Design Concept and Requirements

The envisioned structure needs to provide flexibility for satellite designers throughout the design, development, and testing cycles. Specifically, the structure should allow designers to change the position of subsystems or make

design modifications to subsystems without the need to redesign the main structure. The modular structure of this satellite also conforms to the standards set by numerous universities for CubeSats, thereby ensuring one-to-one compatibility with launch pods. In this process, we first introduce the design process, then review the requirements one by one from a) Launch vehicles and deployment systems b) Requirements analysis and definition.

A. Design Process

Satellite structure has gone through several phases starting with concept design and reaching to the final model. After every phase, the requirements and the constraints are checked whether they are fulfilled or not. The design flowchart which shows all the phases is given in Fig.4. For example, if the structural design cannot fulfill the launch requirements after Finite Element Analysis, we return back to design phase. While for some designs only modeling was sufficient, for some special designs prototyping were done to visual design concepts. In addition, computer programs were used to simulate space and launch environment[6]-[7].

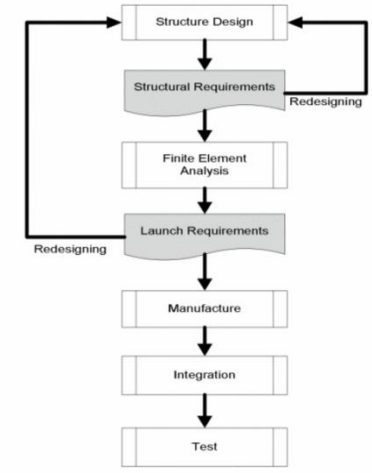


Fig. 4. Design Flow Chart

B. Launch Vehicle and Deployment

The selection of launch vehicle is one of the important steps to determine the launch scenario. Quasistatic launch loads and natural frequencies of rocket will determine the cubesat launch requirements. In Fig.5 the launch scenario and launch loads are demonstrated from launch to orbit for Dnepr Rocket as an example[8]. ITU pSAT II is planned to be launched via the most readily available Polar Satellite Launch Vehicle (PSLV) from India[9].

Predicting of loads that comes from the launch vehicle is one of the hardest steps of designing a spacecraft. Because of the complexity and high variety of mission environments little inaccuracies in the finite element models are capable of causing large errors[10]. During its launch, a satellite is subject to various external loads resulting from steady-state booster acceleration, vibro-acoustic noise, air turbulence, gusts, propulsion system engine vibrations, booster ignition and burn-out, stage separations, vehicle maneuvers, propellant slosh, payload fairing separation and ejection. These sources' characteristic feature is being random and independent [11].

Every event generates structural loads in the life of a spacecraft from launch to put on orbit. Even though launch causes the highest value loads for most spacecraft structures; any other event can be critical and significant for some parts of the structure, such as manufacturing, ground handling-testing, pre-launch preparations, payload separation, on-orbit operations, landing[12].

Deployment System is an important step to provide reliable and cost-effective launch. The adaptor is the interface structure between the satellite and launch vehicle. Cubesats should be well suited with the deployment mechanism to ensure safety and success of the mission. Generally, there are a rectangular aluminum box, a door and a spring mechanism inside the system. Furthermore, various adaptors that are single or multiple, are used to launch cubesats from launch vehicle to orbit. Poly-Picosatellite Orbital Deployer (PPOD) or Single Picosatellite Launcher (SPL) can be indicated as two of them[13]-[14].

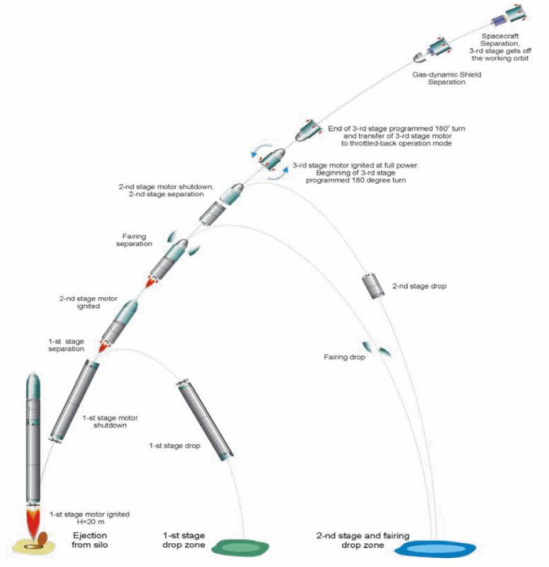


Fig. 5. Dnepr Rocket Launch Loads

One of the important notes is the orientation of the deployment system in the launch vehicle. Direction of deployment system is determined before launching and design of the cubesat should be complete according to this

requirements. In the case of indefiniteness, modeling and analyzing should be performed by considering the worst case scenario. PPOD will be used for ITU pSAT II as a deployment system.

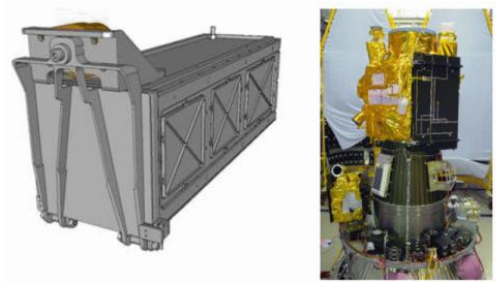


Fig. 6. PPOD and Its Allocation in Launch Vehicle

C. Requirement Analysis and Definition:

During the design process, we examined the requirements and constraints of the design scheme and consulted relevant papers and research findings in the field for guidance. These papers cover the key principles and best practices of satellite structural design, providing us with valuable information. The general requirements for satellite structural design in CubeSat standards are as follows:

1) Framework Structure: Consider adopting a lightweight but sturdy framework to support various satellite components and maintain overall stability. Materials such as aluminum alloy or carbon fiber can be used to construct the framework to ensure sufficient strength and rigidity.

2) Partition Design: Divide the internal space of the satellite into different functional areas, such as payload area, power area, control area, etc., to facilitate component installation and maintenance and reduce interference between components with different functions.

3) Connection Structure: Design appropriate connection structures to ensure that each component can be securely fixed to the framework and withstand the vibrations during launch and various forces in the space environment.

4) Vibration and Shock Analysis: Conduct vibration and shock analysis to evaluate the potential vibrations and shocks the satellite might encounter during launch and operation. This will help optimize the structural design and ensure the satellite's reliability and stability.

5) Thermal Design: Design suitable thermal channels or radiators within the structure to help regulate the internal temperature of the satellite and ensure that each component can operate within an appropriate temperature range.

6) Center of Gravity Control: Arrange the internal components of the satellite reasonably to control the position of the center of gravity, ensuring that the satellite can maintain a stable attitude during operation.

In addition, according to the course task requirements, we need to complete the following three core tasks:

1) Attitude Control Design: Attitude control includes attitude measurement, attitude stabilization control, and disturbance recovery testing.

2) Communication Design: Use a wireless communication module to simulate the satellite-to-ground communication task. Design serial communication programs for the satellite end (microcontroller) and the ground station (host computer). The ground station software also needs to include the design of a human-machine interface.

3) Payload Design: For each task requirement, we need to select appropriate components to accomplish the task to a high degree. For attitude control, satellite attitude control is crucial to ensure stable operation and task completion in orbit. To achieve attitude measurement, we use the MPU6050 chip to sense and measure the satellite's attitude, including parameters like angle and angular velocity. To ensure attitude stabilization control, we use brushless motors with built-in drivers to drive the momentum wheel, adjusting the satellite's attitude to maintain the required state.

For communication design, the satellite communication system is vital for data exchange and command transmission with the ground station. We use the LoRa communication module to simulate the satellite-to-ground communication task, ensuring reliable communication between the satellite and the ground station. For the satellite end communication program, we designed the serial communication program and utilized the Vofa+ host computer to achieve data transmission and command reception functions with the ground station.

For payload requirements, the satellite needs to carry a camera, rechargeable battery, GPS module, and circuit board. The TTL serial camera is mounted on the CubeSat shell as the mission payload, allowing zoom control and image file acquisition through the ground station software. A 12V rechargeable remote control switch battery is fixed to the shell with tie wraps and sponge double-sided tape. This type of rechargeable battery typically has more stable voltage output and longer service life, which helps ensure stable satellite operation. The GPS module is attached to the shell with sponge double-sided tape, providing high-precision location information to the ground receiver, ensuring the satellite accurately performs its tasks, including communication, Earth observation, and remote sensing.

III. Modelling and Analysing

2U Cubesats are a common small satellite, compact in size and simple in structure. The mechanical structure part of this

group aims to design a 2U cube frame with good structural strength and thermal control performance, and rationally arrange the internal structure, select suitable materials, and conduct simulation and verification.

A. External Structure Shape Design:

The design of the satellite's external structure is a critical factor in ensuring the satellite's stable operation in extreme environments and the safe functioning of its internal equipment. It not only bears the responsibility of protecting and supporting internal components but also directly affects the satellite's attitude control, stability, and operational efficiency. Through a rational design, the performance of the satellite can be enhanced, and the manufacturing and operational costs can be reduced, ensuring the satellite can successfully execute its mission. Therefore, the design of the satellite's external structure occupies a crucial position in the entire satellite project.

Based on design requirements and available information, the most significant part of the external structure design is the shell. The shell adopts a sliced modular design, with the overall structure assembled sequentially from six faces. Considering the loadable size range of the experimental platform, the satellite is 2U, with the shell dimensions designed to be 200mm×50mm×50mm. The CubeSat's core framework primarily uses an X-shaped and cruciform structure for hollow support, with detailed designs tailored to the installation components and requirements of each face, as shown in Fig.7. The advantages of this shell shape are as follows:

1) Improved Structural Strength And Stiffness: The X-shape and cruciform structure are used to hollow out support structure can significantly improve the structural strength and stiffness of the Cubesat kernel framework. The cross-braced layout increases the connection points between the supporting rods, creating a solid frame structure. This structure design can withstand greater torque and stress, thus improving the structural safety of the satellite.

2) Avoid Excessive Local Stress: The X-shape and cruciform structure are used to hollow out support structure can distribute the external load evenly to the whole structure, so as to avoid excessive local stress. This helps reduce the risk of structural fatigue and failure. In addition, the X-shaped support structure is designed to provide flexibility to adapt to different loads and operating conditions.

3) Excellent Stability: The X-shape and cruciform structure are used to hollow out support structure has excellent stability and can resist torque and torsion in different directions. This is very important for the stability of the satellite in various working environments and motion conditions. Therefore, we designed the Cubesat frame, in which the camera is placed, the position is specially left to allow the camera to take pictures without any occluders, so that it can better track the target and control the shooting task.

4) Reducing The Overall Weight: The hollow design using X-shaped and cruciform support structures can reduce material usage while maintaining sufficient strength, thereby reducing the overall weight of the satellite. Without the hollow structure, the total weight of the shell, according to simulation software calculations, is 37.6g. After adopting the hollow design, the total weight is 22.0g, reducing the weight by 37%. This lightweight design improves launch efficiency and reduces launch costs.

5) Excellent heat dissipation Performance: The hollow design using X-shaped and cruciform support structures can increase the surface area for heat dissipation, improving the satellite's thermal management. This helps maintain a stable internal temperature and prevent overheating in space.

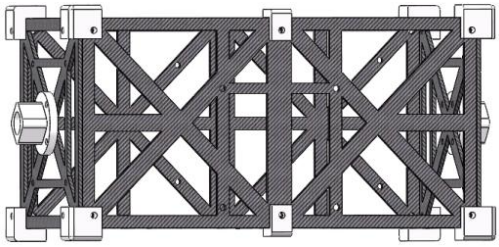


Fig. 7. CubeSat Kernel Framework

In the hollow design, to ensure the stability and resistance to vibration and impact of the framework, thickening operations are designed to prevent fractures and large deformations. Additionally, reinforcement structures are designed at the installation positions of the momentum wheel modules to enhance stability and strength, as shown in Fig.8.

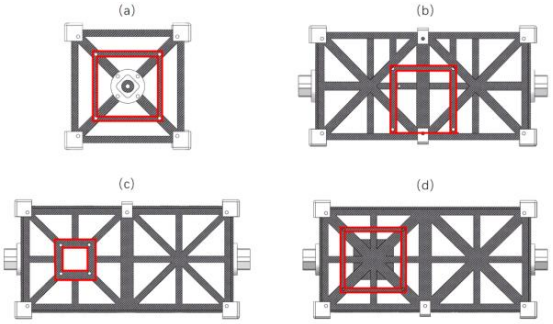


Fig. 8. Enhanced Bar Detail Diagram

For the connection parts design, we chose to use eight 16mm×16mm cubic corner connectors to fix the corners of the rectangular shell. Each connector fixes three faces with three screws. At the edges of the rectangle, L-shaped external connectors are used to connect adjacent faces. These connectors are also fastened with screws.

For the use of external fixtures: these components are easy to manufacture, install, and maintain, simplifying the assembly

process and reducing the risk of manufacturing errors, as shown in Fig.9. They optimize the utilization of internal space and enhance the mechanical strength of the shell, enabling the satellite to better withstand external environmental challenges, ensuring its stability and reliability.

(a)Angle fastener (b)L-shaped fastener

Fig. 9. Enlarged View of The Fixed Part

B. *Internal Structure Layout Design*

The internal structure layout design plays a critical role in satellite engineering. It directly affects the satellite's performance, reliability, and maintainability. By optimizing weight and volume, achieving effective thermal control, and ensuring electromagnetic compatibility, it ensures the satellite can operate stably and reliably. Therefore, a meticulously designed internal structure layout is one of the key guarantees for the successful completion of satellite missions.

1) Momentum wheel combination module layout design: To ensure the satellite's balance in the X, Y, and Z directions, the momentum wheel combination modules are arranged in these three directions. The specific layout is as follows:

a) The momentum wheel combination modules are fixed on three different faces of the satellite to achieve multi-axis momentum control.

b) Each momentum wheel combination module is connected to a motor through motor fixing parts, and the motor is fixed to the shell with four copper columns, ensuring the stability and precise positioning of the momentum wheel combination modules.

2) Battery Layout Design: To ensure the satellite's center of gravity is in a central position, the battery layout is as follows:

a) The battery is installed on the face in the Y direction without the momentum wheel combination module, positioned near the camera section. This maintains the balance of the center of gravity while optimizing space utilization.

b) The battery is secured with sponge double-sided tape and fastening straps, ensuring stability and preventing displacement under external forces.

3) Circuit Board Layout Design: In the circuit board layout design, the main considerations are circuit connections and the center of gravity. The specific design is as follows:

a) The circuit board is installed on the face in the Z direction without the momentum wheel combination module.

b) It is secured using two longer L-shaped fasteners in a clamping manner, ensuring stability and reliable connections.

4) Other Chip Layout Design: The layout design of other chips needs to consider circuit connections and interference from the momentum wheel. The specific design is as follows:

a) The MPU6050 chip is placed parallel to the center of the shell, secured with sponge double-sided tape, ensuring it remains stable under external forces and does not affect the functionality of other components.

b) Other chips are reasonably arranged according to their circuit connection requirements and interference with the momentum wheel, all secured with sponge double-sided tape on appropriate faces of the shell, ensuring their proper functionality.

Through the interference detection in SolidWorks, we can also see that the whole space design is very reasonable without any interference, as shown in Fig.10.

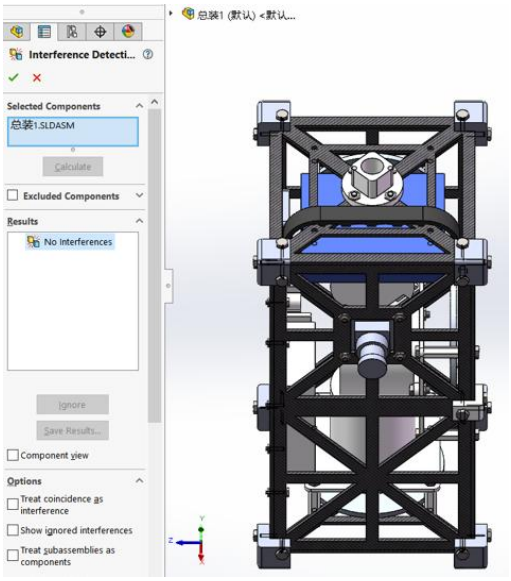
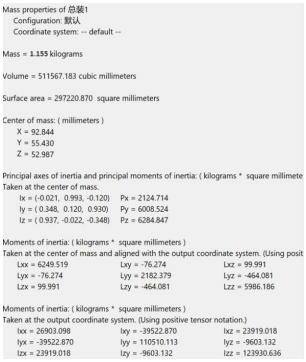


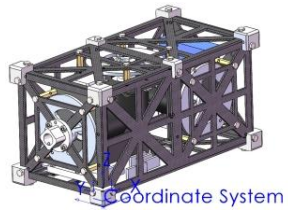
Fig. 10. Interference Analysis Diagram

Through the above layout design, the reasonable distribution of internal components is ensured, maintaining the balance and stability of the satellite's center of gravity. The final overall center of gravity of the satellite is estimated to be located at (92.8, 55.4 , 53.0), with a deviation of only 9.4 mm from the exact center, as shown in Fig.11(a). During the design process, circuit connections and potential interference

factors for each component were fully considered, ensuring the stable operation of the satellite.



(a)



(b)

Fig. 11(a) (b). Estimation of the Center of Gravity Position and Schematic Diagram of the Model.

C. CubeSat Structure Materials

The selection of materials for satellite design directly affects its performance, stability, and reliability in extreme space environments. Mass is a critical factor for objects in orbit. Especially for a 2U microsatellite, minor structural changes can provide valuable space for other subsystems and components. Reasonable material selection not only meets the functional and performance requirements of the satellite but also optimizes its weight and volume, enhances durability, effectively controls thermal properties, and ultimately ensures the satellite's long-term stable operation. The satellite design considers not only weight but also strength, stiffness, thermal conductivity, thermal expansion, manufacturability, and cost factors[15].

1) Carbon Fiber Shell Material Selection

The satellite shell adopts carbon fiber material, whose main advantages include high strength, light weight, and corrosion resistance. These characteristics of carbon fiber ensure the durability and stability of the satellite shell. The performance of the carbon fiber material used in the shell's manufacturing is shown in Table 1(a). Analyzing the data in the table, carbon fiber material, with its excellent mechanical properties and low density, becomes an ideal choice for the satellite shell. It can significantly reduce the structural load and improve the satellite's performance. Its high strength and corrosion resistance ensure the satellite's stability in harsh environments. However, the manufacturing process of carbon fiber is complex and costly, but it excels in tensile strength, thermal conductivity, and thermal expansion. Despite the high cost, its superior performance makes it widely used in demanding aerospace applications.

2) Connector Material Selection

The motor fastening parts, cubic corner connectors, L-shaped face connectors, and circuit board L-shaped connectors are all made of photosensitive resin 3D printing material. The performance of the photosensitive resin 3D printing material used in the manufacturing of connectors is shown in Table 1(b). Analyzing the data in the table, photosensitive resin material has the advantages of high precision and rapid prototyping, suitable for manufacturing complex parts. Photosensitive resin material's high precision and rapid prototyping features make it suitable for manufacturing complex connecting parts. Its low density and relatively high mechanical properties meet the structural needs of the satellite. Due to its ease of processing and low cost, photosensitive resin is suitable for parts with less stringent requirements, effectively reducing manufacturing costs and improving production efficiency. However, photosensitive resin is relatively brittle and has lower long-term durability, which needs to be considered in the design for potential lifespan limitations.

Through the selection and analysis of carbon fiber and photosensitive resin materials, the satellite structure's high strength, light weight, and high precision are ensured. The reasonable application of these materials provides a guarantee for the satellite's stability and functionality.

Table 1(a). Material Properties of The Frame

Carbon Fiber Material Properties	
$\rho(\mathrm{kg/m^3})$	1.6
σ_{UTS}(MPa)	3500
σ_B(MPa)	1000
$\lambda(\mathrm{W/m \cdot}K)$	1.5
$\alpha(10^{-6}/\mathrm{K})$	-0.5

Table 1(b). Material Properties of The Frame

Photosensitive Resin Materials Properties	
$\rho(\mathrm{kg/m^3})$	1.1 - 1.2
σ_{UTS}(MPa)	50 - 100
σ_B(MPa)	80 - 150
$\lambda(\mathrm{W/m \cdot}K)$	0.2-0.3
$\alpha(10^{-6}/\mathrm{K})$	80 - 120

The final mass of the entire 2U Cubesat is estimated to be 1.155 kg by searching the data.

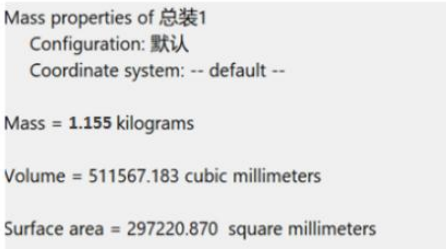


Fig. 12. Quality Simulation Prediction Results Diagram

D. Launch Environment and Scenario Analysis

A typical CubeSat device will be launched on various rockets. To meet acceptance conditions, the CubeSat structure must not fail under certain static and dynamic loads, which are calculated based on launch conditions. Launch rockets impart random excitations, and to avoid severe resonances in the structure, the first natural frequency of free vibration should be above 70-90 Hz.

Therefore, based on the existing analysis procedures[16], all launch load scenarios required for the QB50 mission are considered: modal analysis, static stress analysis. Details are given below. In addition, the thermal analysis of the low Earth orbit cube is carried out considering the thermal load during its operation.

1) Modal Analysis

The first step during a dynamic analysis is the determination of natural frequencies (eigenfrequencies) and mode shapes of structure, considering zero damping. The results of this analysis characterize the dynamic behavior of the structure and can show how the structure will respond under dynamic loads[17]. The most important modal characteristic of a space structure, like the CubeSat, is the natural frequency threshold—meaning that the first natural frequency of the structure must be above a specific value, which usually is determined by the launch vehicle. The typical range for such missions is between 50 and 90 Hz.

Firstly, a modal analysis was conducted under free-free boundary conditions. This served as an intermediate verification step for the connectivity of the developed finite element model. Then, another modal analysis was performed with the P-POD boundary conditions applied. This part reveals the real-case scenario of the structural system.

2) Static Stress Shock Simulation

Static stress Static stress shock simulation analysis is an important step to ensure that the satellite structure will not be damaged during operation. The finite element analysis method is used to simulate the satellite structure and evaluate the mechanical load. Through these analyses, we determine the strength and stiffness of the satellite structure to ensure that it can withstand loads under various operating conditions as well as external shocks during operation. The maximum acceleration received by the Cubesat can be obtained through the estimation of the above part of materials and by referring to relevant papers.

In order to simulate the bearing capacity of the cubesat under impact, the deformation simulation of the cubesat under given pressure is made. The deformation of the whole frame structure and internal parts of the cubesat under the pressure of Z and Y direction is simulated.

The simulation steps are as follows: First of all, we select the corresponding materials for each part of the satellite in the SolidWorks' model. For the materials not in the SW material library, we choose the materials similar to the real materials in weight, density and strength as far as possible.To simulate the displacement of stress in the Z direction, a surface in the Z direction of the cubesat is fixed first, and then a pressure of 100N is applied to the corresponding surface of the fixed surface.m^2. Then, the displacements of cubesat assembly in X, Y and Z directions were calculated respectively. It can be seen from the simulation results that the maximum displacement in Z direction is 0.17mm. To simulate the displacement under pressure in the Y direction, a surface in the Y direction is fixed, and then a pressure of 100N/m^2 is applied to the corresponding surface of the fixed surface to solve the displacement of the cubesat assembly in the X, Y and Z directions respectively.

3) Thermal Analysis

Since the CubeSat will be launched from the International Space Station (ISS), the thermal analysis considers the ISS's orbit. This is a low Earth orbit with the characteristics shown in Table 2.

Table 2. Orbit Characteristics

ISS Orbit	
Semimajor axis (km)	6780
Eccentricity (degrees)	0
Ascending node longitude (degrees)	-94.354
Inclination (degrees)	51.64
Mean anomaly (degrees)	254.215

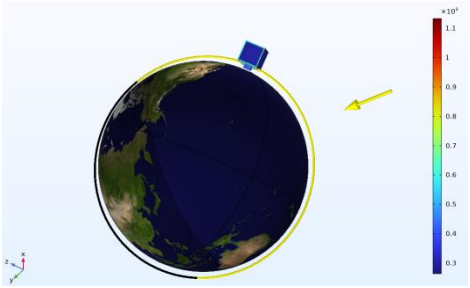


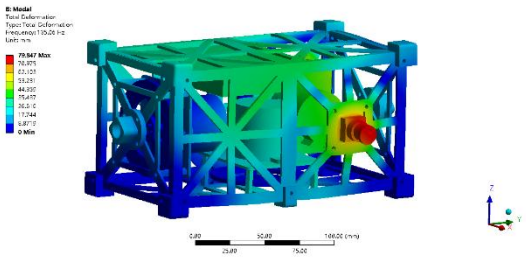
Fig. 13. Orbital Visualization Diagram

E. Verification by Analysis

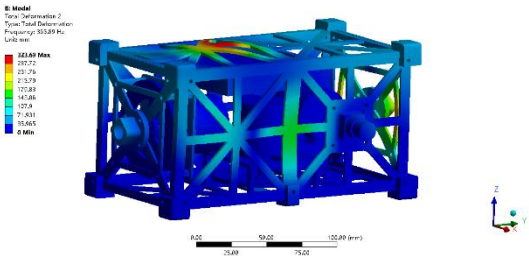
This section presents the structural analysis of the UPSat 2U CubeSat platform. Some data regarding static stress shock simulation and operational loads (trajectory for thermal analysis) are referenced from QB50 mission requirements[18] and pertain only to this specific mission.

1) Modal Analysis Results

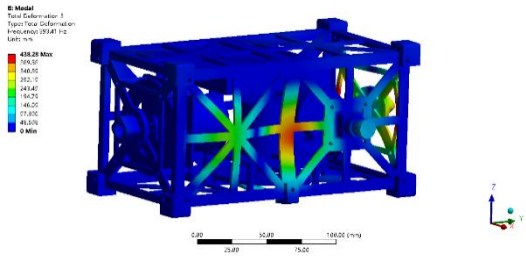
Fig 14 shows the first six natural frequencies of the satellite structure and the corresponding mode shapes (bending modes). The lower limit for the first natural frequency for this specific mission is set above 90 Hz. In this analysis, the first natural frequency is close to 135 Hz, as shown in Fig.14(a), indicating that the modal characteristics of the structure can be considered sufficiently robust.



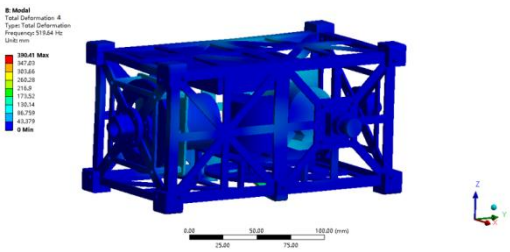
(a)1st Eigenmode (135.06 Hz)



(b)2nd Eigenmode (365.89 Hz)



(c)3rd Eigenmode (393.41 Hz)



(d)4th Eigenmode (519.64 Hz)

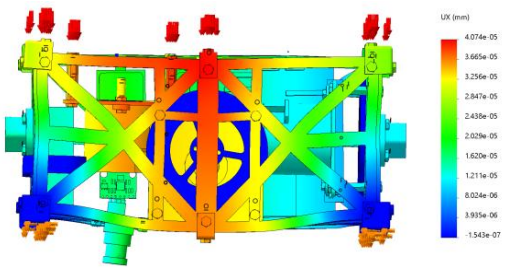
Fig. 14(a) (b) (c) (d). The First 4 Characteristic Frequencies And Modal Shapes Of The Satellite.

2) Static Stress Shock Analysis Results

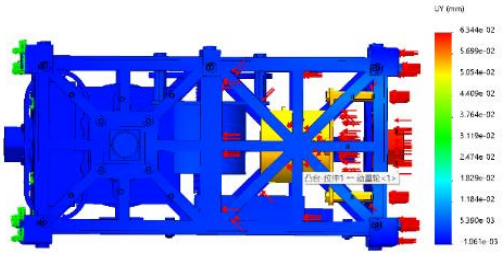
The displacement simulation in the Z direction shows that the deformation displacement in the Z direction is the largest, and the maximum is 0.17mm. The deformation part is mainly in the middle position of the upper X-coordinate about 50mm after the deformation displacement proportion amplification, indicating that the middle part of the cubesat is most prone to deformation when the cubesat is under high load. Therefore, the chip sensor and other components should be placed on both sides of the satellite supported by the frame.

From the simulation results of displacement in the Y direction, it can be seen that the deformation displacement in the Y direction is the largest, and the maximum is 6×10^{-2} mm, and it can be seen that the deformed part is mainly at the center position of the satellite x,z coordinates are about 50mm, which indicates that when the CubeSat is under high load, the middle part of the CubeSat is most prone to deformation, so the chip sensor and other components should be placed on both sides of the satellite with frame support.

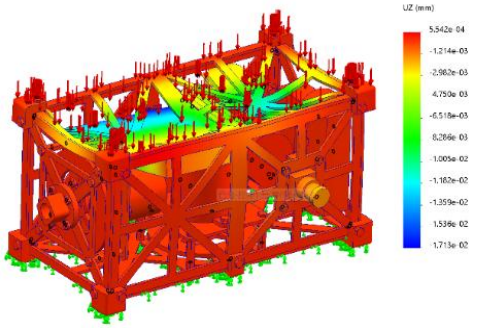
According to the analysis of simulation results, when the CubeSat bears a high load, its external frame will deform within a certain range, but within the gap reserved in the structural design, but it will have a certain impact on the stability of the internal components. Therefore, the external frame with large deformation should be avoided when designing the location of the internal components.



(a)



(b)



(c)

Fig. 15(a) (b) (c). The Amount Of Deformation Of The Satellite In The X, Y, And Z Directions After Shock

In addition, the analysis indicates that Von Mises stress is observed as 1.47 MPa, and this value is within the specifications since Carbon fiber material yield strength is 1500-3000 MPa.

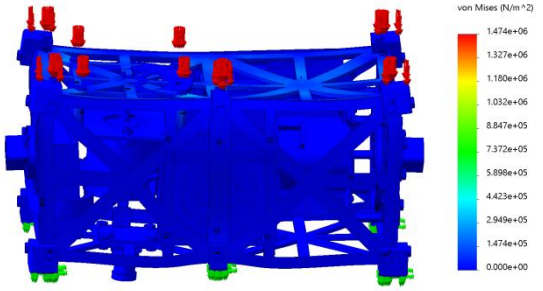


Fig. 16. Von Mises Stress Simulation Results Of Carbon Fiber Shell

3) Thermal Analysis Results

In addition to structural strength, thermal control is also an important consideration in satellite design. Thermal analysis was carried out to assess the heat distribution and temperature control of the satellite in operation. By referring to the data, we found that the heat received by the cubic star mainly comes from our solar radiation. In order to ensure the temperature stability of the satellite in various working environments, we carried out a thermodynamic analysis, namely, the temperature is equal to 423k. It can be seen that the temperature of titanium melting is far from reached under the influence of the maximum solar radiation, the melting temperature of the carbon fiber shell is far from reached.

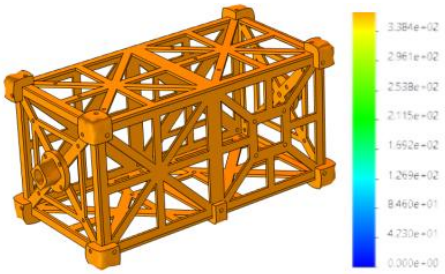


Fig. 17. Schematic Diagram Of Thermal Analysis

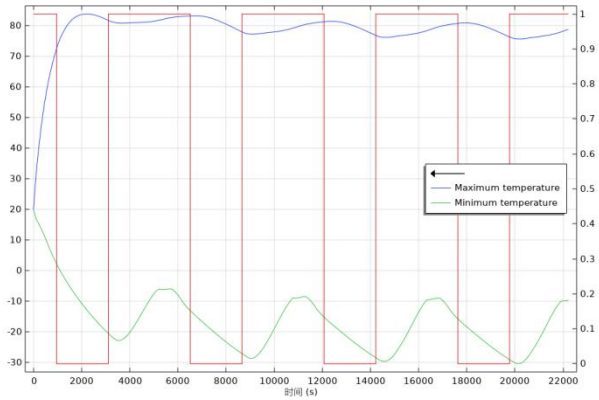


Fig. 18. Satellite Surface Temperature Vs. Time Diagram

F. Optimization scheme

1) Static stress analysis after material replacement

On the basis of static stress analysis, the simulation results of our carbon fiber material were mediocre, so the material was replaced. Titanium alloy was selected as the new material and the static stress analysis was carried out again. This step aims to verify the feasibility of the new material and evaluate its performance in the satellite structure. Titanium alloy material has excellent strength, light weight and corrosion resistance, which is suitable for the requirements of satellite structure. It also has excellent thermal conductivity and high temperature stability, which is crucial for satellite thermal control. Its main advantages are as follows:

a) Excellent strength: Titanium alloy has an excellent strength-to-weight ratio, stronger than many other metal materials. This makes titanium alloys ideal for the strength requirements of satellite structures.

b) Lightweight: Titanium alloy is a lightweight material with low density. This allows the use of titanium alloy in satellite design to effectively reduce the overall weight, contributing to improving satellite carrying capacity and reducing cost.

c) Corrosion resistance: Titanium alloy has good corrosion resistance and can resist corrosion and oxidation common in the space environment. This enables titanium

alloys to maintain long-term stability and reliability in satellite structures.

Titanium alloys provide superior strength and stiffness in structural design. Its high strength and excellent tensile strength enable the Cubic Star frame to withstand a variety of static and dynamic loads, including vibration, shock and acceleration. In addition, the low coefficient of thermal expansion and good thermal stability of the titanium alloy help ensure the normal operation of the satellite in different temperature and thermal environments. Static stress impact simulation after optimization

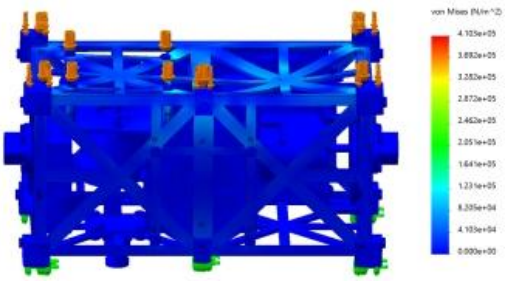
2) Static stress analysis after material replacement

Based on the simulation results, it was found that the von Mises stress of the titanium alloy shell is 0.41 MPa, and the maximum deformation in the x-direction is 6.505×10^{-6} mm, as shown in Fig.19. In contrast, the von Mises stress of the carbon fiber shell is 1.474 MPa, with a maximum deformation in the x-direction of 4.074×10^{-5} mm. After optimization, the selection of titanium alloy material for the satellite shell significantly outperforms the initially chosen carbon fiber material. This is evident in the following aspects:

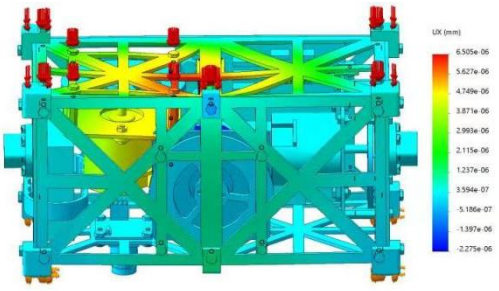
a) The von Mises stress of the titanium alloy shell is 0.41 MPa, significantly lower than the 1.474 MPa of the carbon fiber shell. This indicates that the titanium alloy material sustains less internal stress under the same conditions, better resisting external loads and environmental stresses, thereby enhancing the overall strength and reliability of the satellite shell.

b) The maximum shape variable of the titanium alloy shell in the x direction is 6.505×10^{-6}mm , while the maximum shape variable of the carbon fiber shell in the same direction is 4.074×10^{-5}mm. The reduction of shape variable indicates that the deformation degree of titanium alloy material is smaller under the same load, the rigidity of the shell is higher, and the structure is more stable.

Overall, the optimized titanium alloy shell exhibits superior performance in stress resistance and deformation resistance, ensuring the stability and reliability of the satellite shell under harsh environmental conditions. Therefore, selecting titanium alloy as the shell material is an effective optimization strategy that significantly enhances the performance of the satellite shell.



(a)



(b)

Fig. 19. Von Mises Stress (a) And Deformation In The X Direction(b) Simulation Results Of Titanium Alloy Shell

G. Actual assembly splicing

Finally, We assembled, connected and fixed the components of the satellite. We followed the relevant assembly specifications and process flow to ensure the integrity and stability of the satellite structure. Below is our finished product.

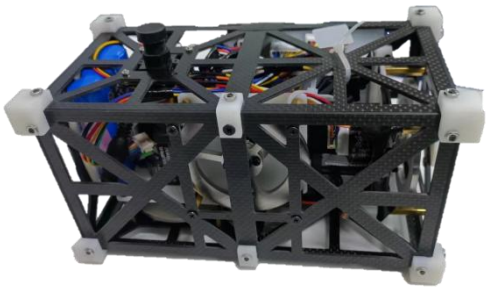


Fig. 20. Actual Assembly Picture

IV. HARDWARE SYSTEM

The core of this project's cube satellite design primarily consists of the upper computer and the lower computer, encompassing three functional areas: the energy module, the measurement module, and the execution module. In the lower computer section (measurement module), a CH32V307 core board is used as the main control board of the cube satellite, integrating the components of five measurement modules. Attitude measurement is conducted by the MPU6050 accelerometer and gyroscope, which transmit data via the I2C interface. The ATGM336H GPS module provides precise positioning through the UART

interface, while the HMC5883 magnetic sensor also uses the I2C interface to transmit magnetic field and attitude data. Additionally, the camera module transmits image data via the UART interface. The LoRa communication module interacts with the upper computer through the UART interface, enabling remote control and data transmission.

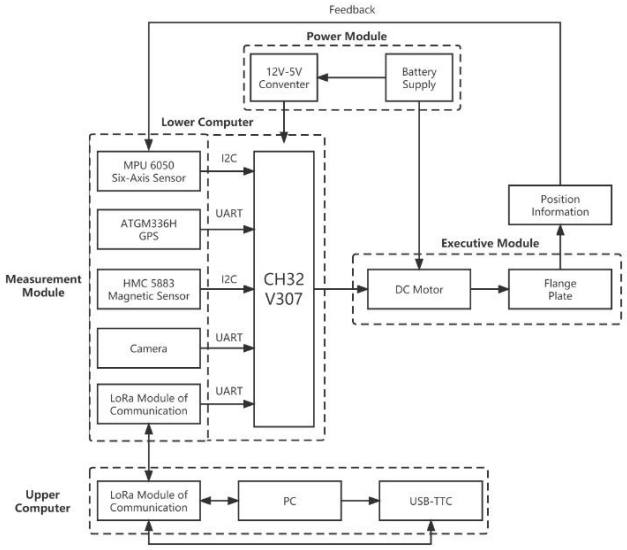


Fig. 21. The Flow Chart Of The Overall Design Of CubeSat

The upper computer utilizes a PC-USB TTC-LoRa combination module for communication management, ensuring stable and reliable data transmission. In the execution module, a DC motor is connected to the flange via a PWM control signal, allowing the adjustment of the cube satellite's attitude by regulating different directions and speeds. Regarding the energy module, a rechargeable 12V battery powers the DC motor, and a voltage regulator along with supporting capacitors converts the voltage to 5V to provide stable power support to the core board and measurement modules, ensuring the normal operation of each functional module. A remotely controlled switch is installed on the power supply, enabling contact-less and non-intrusive control of the satellite’s startup and shutdown.

A. Attitude Solving Module

1) Hardware Selection: MPU6050

The MPU6050 is a high-performance 6-axis motion sensor that integrates a 3-axis gyroscope and a 3-axis accelerometer. It uses the I2C interface for data transmission, allowing it to measure both angular velocity and linear acceleration simultaneously, thereby providing comprehensive motion information.

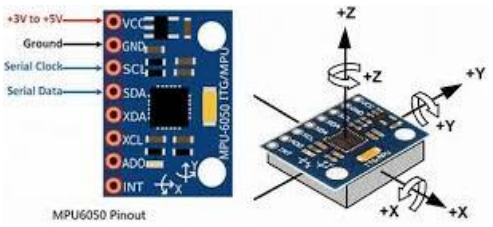


Fig. 22. MPU6050 Six-Axis Attitude Sensor

a) Principle of MPU6050

The accelerometer within the MPU6050 uses Micro-Electro-Mechanical Systems (MEMS) technology to measure an object's linear motion by sensing acceleration in three orthogonal directions. The accelerometer consists of tiny mass blocks and spring structures; when the object moves, the displacement of the mass blocks is detected and converted into electrical signals.

The gyroscope also employs MEMS technology, measuring rotational motion by detecting angular velocity in three orthogonal directions. Its internal structure comprises a vibrating mass block and detection electrodes. As the object rotates, the Coriolis force generated by the mass block causes a change in capacitance, which is then converted into angular velocity signals.

b) Advantages of MPU6050

1. High Integration: The MPU6050 integrates a 3-axis accelerometer and a 3-axis gyroscope into a single chip, significantly simplifying system design, reducing space requirements, and lowering development costs.

2. High Precision: The sensor features high precision and high resolution, capable of accurately capturing subtle motion changes, thus providing reliable data for high-precision attitude and motion control.

3. Low Power Consumption: With its advanced low-power design, the MPU6050 is highly suitable for battery-powered portable devices, extending the device's operational life.

4. Digital Interface Advantages: Supporting the I2C interface, it simplifies connections with various microcontrollers and processors, providing efficient data transmission and easy system integration.

c) MPU6050 Initialization and Data Acquisition

```
mpu6050_init();
mpu6050_get_acc();
mpu6050_get_gyro();
```

Fig. 23. MPU6050 Initialization Code

The mpu6050_init( ) function is used to configure various registers of the MPU6050, including the address register, GYRO_CONFIG register, ACCEL_CONFIG register, and others. The mpu6050_get_acc( ) function is employed to retrieve accelerometer data from the MPU6050, while the mpu6050_get_gyro( ) function is used to acquire gyroscope data from the MPU6050.

2) Hardware Selection: HMC5883

The HMC5883 is a high-precision 3-axis digital magnetic sensor used for measuring the intensity and direction of the Earth's magnetic field. It is commonly employed in electronic compasses and attitude detection systems. Communication with other devices is facilitated via the I2C interface.

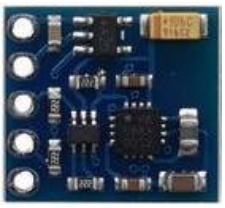


Fig. 24. HMC5883 Magnetic Sensor

a) Principle of HMC5883

The HMC5883 utilizes Hall Effect technology to measure the Earth's magnetic field by detecting the magnetic intensity in three orthogonal directions. The internal Hall elements generate minute voltage changes under the influence of an external magnetic field. These changes are then amplified, processed, and converted into digital signals.

The HMC5883 integrates signal conditioning circuits and analog-to-digital converters (ADC) to convert analog signals into digital signals, which are then output through the I2C interface.

b) Advantages of HMC5883

1. *Precision Measurement Capability:* The HMC5883 offers superior high-resolution magnetic field measurement, ensuring accuracy in navigation and attitude detection, making it suitable for demanding applications.

2. *Efficient Low Power Consumption:* Its optimized low-power design makes it ideal for portable and long-duration applications, enhancing overall device performance and reliability.

3. *Compact Packaging:* The small and compact packaging of the HMC5883 allows easy integration into various miniature devices, supporting more complex and diverse design requirements.

4. *Simple Interface:* The I2C digital interface simplifies hardware connections, providing fast and stable data transmission, facilitating system integration and development.

c) HMC5883 Initialization and Data Acquisition

```
void HMC5883_Init(void)
{
    soft_iic_init(&HMC5883_iic_struct, HMC5883_ADDR,
                  HMC5883_SOFT_IIC_DELAY, HMC5883_SCL_PIN,
                  HMC5883_SDA_PIN);
    system_delay_ms(100);
    HMC5883_write_register(0x00, 0x58);
    HMC5883_write_register(0x01, 0x60);
    HMC5883_write_register(0x02, 0x00);
}
```

Fig. 25. HMC5883 Initialization Code

The HMC5883_init function configures various registers of the HMC5883, including the address register and various read/write registers.

```
void Multiple_Read_HMC5883(void)
{
    uint8_t buf[6];
    HMC5883_read_registers(HMC5883_REG_X_MSB, buf, 6);
    X = (int16)(((uint16)buf[0] << 8 | buf[1]));
    Y = (int16)(((uint16)buf[2] << 8 | buf[3]));
    Z = (int16)(((uint16)buf[4] << 8 | buf[5]));
    system_delay_ms(5);
}
```

Fig. 26. HMC5883 Data Read

Continuous read mode is utilized to acquire register data from the HMC5883, starting with the HMC5883_REG_X_MSB register and sequentially reading six registers.

Fig. 27. HMC5883 Data Reading Test

B. GPS Positioning Module ATGM336H

The ATGM336H is a high-precision GPS positioning module widely used in vehicle navigation, drones, handheld devices, and other scenarios requiring high-accuracy positioning. It communicates via the UART interface, providing precise location information.

Fig. 28. ATTGM336H GPS Module

1) Principle of ATTGM336H

The ATGM336H receives signals from multiple GPS satellites through its built-in antenna. An internal high-performance processor decodes and processes these signals, using triangulation to determine the module's exact position.

The ATGM336H outputs position information in RMC sentence format through the UART interface, including data such as time, longitude, and latitude.

2) Advantages of ATTGM336H

a) High-Precision Positioning: The module offers sub-meter level positioning accuracy, making it suitable for various high-precision applications

b) .Fast Signal Acquisition: It features rapid cold start, hot start, and re-acquisition capabilities, ensuring quick positioning in diverse environments.

c) Low Power Consumption: The optimized low-power design makes it ideal for battery-powered portable devices.

d) Strong Compatibility: The module supports multiple satellite navigation systems (such as GPS and GLONASS), enhancing positioning accuracy and reliability.

3) GPS Initialization and Data Acquisition

```
uart_init(GNSS_UART, 9600, GNSS_RX, GNSS_TX);

uart_write_buffer(GNSS_UART, (uint8 *)set_rate, sizeof(set_rate));
system_delay_ms(200);

uart_write_buffer(GNSS_UART, (uint8 *)open_rmc, sizeof(open_rmc));
system_delay_ms(50);
```

Fig. 29. GPS Initialization Code

The GPS module communicates with the chip using the UART protocol. The process involves writing to registers to initiate GPS communication, setting the GPS update rate to 10Hz, and enabling PMC sentences.

```
static uint8 gps_gnrmc_parse
(char *line, gnss_info_struct *gnss)
```

Fig. 30. GPS Data Read

The RMC sentences, with *char * line* containing the received sentence information and *gnss_info_struct * gnss* storing the parsed data.

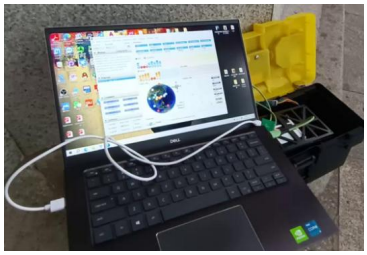


Fig. 31. GPS Data Reading Test

C. TTL Serial Camera Module

The TTL serial camera is a high-resolution 2-megapixel camera module widely used in image acquisition, surveillance, and robotic vision applications. It transmits image data via the UART interface.

Fig. 32. TTL Serial Camera Module

1) Principle of TTL Serial Camera

The camera features a built-in high-performance image sensor that converts light signals into electrical signals through photoelectric conversion. The integrated Image Signal Processor (ISP) processes these electrical signals and outputs high-quality image data. Image data is transmitted via the UART interface, enabling real-time communication with the main control board for image acquisition and processing.

2) Advantages of TTL Serial Camera

a) High Resolution: The camera provides 2-megapixel high-resolution images, offering clear and detailed visuals suitable for fine image acquisition.

b) High Sensitivity: It captures clear images even under low-light conditions, making it versatile for various applications.

c) Low Latency Transmission: Real-time image data transmission through the UART interface ensures low latency, maintaining real-time performance.

d) Compact Design: The camera's compact design allows easy integration into various devices.

3) TTL Serial Camera Initialization and Data Acquisition

```
void VC0706_TakePicture(void) {

    uint8_t Serial_TxPacket1[5];
    Serial_TxPacket1[0] = 0x56;
    Serial_TxPacket1[1] = 0x00;
    Serial_TxPacket1[2] = 0x36;
    Serial_TxPacket1[3] = 0x01;
    Serial_TxPacket1[4] = 0x00;
    uart_tx_interrupt(UART_2, ENABLE);
    uart_write_buffer(UART_2, Serial_TxPacket1, 5);
    uart_tx_interrupt(UART_2, DISABLE);
```

Fig. 33. TTL Serial Camera Data Read

The camera uses the VC0706 protocol for data transmission. By sending hexadecimal data control frames via the UART

protocol, one can stop the frame, refresh the frame, and retrieve image length and data.

Fig. 34. TTL Serial Camera Data Reading Test

D. SX1278 LoRa Wireless Transmission Module

The SX1278 is a high-performance LoRa wireless transmission module that utilizes LoRa spread spectrum technology. It is suitable for long-distance, low-power wireless communication applications. Data transmission is conducted through the UART interface, making it widely used in the Internet of Things (IoT) and wireless sensor networks.

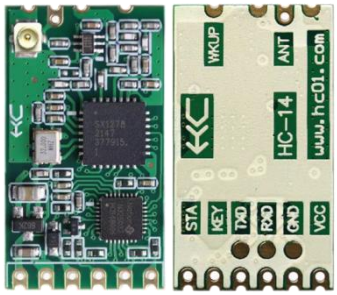


Fig. 35. SX1278 LoRa Wireless Transmission Module

1) Principle of SX1278 LoRa Module

The SX1278 LoRa wireless transmission module employs spread spectrum modulation technology, which enhances interference resistance and extends communication range by broadening the signal bandwidth. It features a built-in high-sensitivity RF receiver and a low-power RF transmitter, enabling long-distance data transmission. The module supports data communication via the UART interface, allowing for point-to-point or multipoint wireless communication.

2) Advantages of SX1278 LoRa Module

a) Long-Distance Transmission: Utilizing LoRa spread spectrum technology, the communication range can reach several kilometers, making it suitable for extensive wireless transmission applications.

b) Ultra-Low Power Consumption: Its optimized low-power design is ideal for battery-powered devices, extending the operational lifespan of the equipment.

c) Strong Anti-Interference Capability: The module boasts excellent anti-interference performance, ensuring stable communication in complex electromagnetic environments.

d) Strong Anti-Interference Capability: The module boasts excellent anti-interference performance, ensuring stable communication in complex electromagnetic environments.

3) SX1278 LoRa Module Initialization and Data Acquisition

```
void lora_Init(){
    uart_init(UART_3, 9600,
    UART3_MAP0_TX_B10, UART3_MAP0_RX_B11);
}
```

Fig. 36. SX1278 LoRa Initialization Code

```
void Vofa_JustFloat ()
{
    Float_to_Byte((mpu6050_gyro_x + 29 ) * 1.0, xbyte);
    Float_to_Byte((mpu6050_gyro_y - 15 ) * 1.0, ybyte);
    Float_to_Byte((mpu6050_gyro_z + 25 ) * 1.0, zbyte);
```

Fig. 37. SX1278 LoRa Data Send

The LoRa module communicates via the UART interface, with UART_3 configured as the data channel. The "Justfloat" protocol is used for transmitting data to the upper computer.

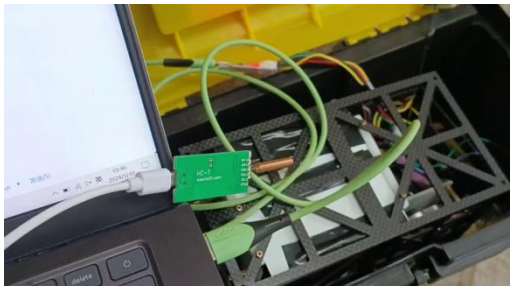


Fig. 38. SX1278 LoRa Data Sending Test

E. Motor Drive Module

The Mabuchi Brushless DC Motor from Japan is an efficient and reliable motor, widely used in drones, electric vehicles, and robots, where high power density and efficiency are crucial. It employs an electronic control system for commutation, eliminating the need for mechanical brushes, which results in longer lifespan and superior performance.

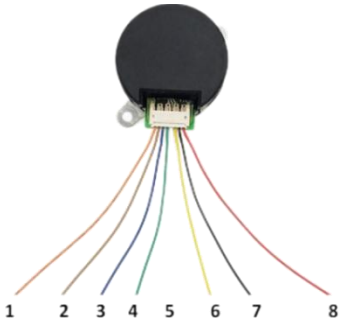


Fig. 39. Mabuchi Brushless DC Motor

Table. 3. Parameters of Mabuchi Brushless DC Motor

Parameter	Value
Max power	10W
Max torque	0.0385Nm
Motor drive voltage	12-24V
Control system voltage	5V
Encoder	100 lines
Diameter	42mm
Length	39mm
Weight	140g

1) Principle of Mabuchi Brushless DC Motor

The Mabuchi Brushless DC Motor uses an electronic controller instead of traditional brushes. The electronic commutator controls the energizing sequence of the stator windings, driving the rotor to rotate. The rotating magnetic field generated by the stator windings interacts with the permanent magnets on the rotor, causing the rotor to turn. To precisely control the motor's rotation, the Mabuchi motor is equipped with position sensors (such as Hall sensors) to detect the rotor's actual position. The controller adjusts the direction and magnitude of the current based on the sensor feedback, achieving precise speed and position control.The adjustment of motor speed and torque is achieved through Pulse Width Modulation technology. The controller regulates the pulse width of the current to control the motor's speed and output torque.

2) Advantages of Mabuchi Brushless DC Motor

a) Built-in Driver: The Mabuchi Brushless DC Motor, model A4931, includes an integrated driver. It provides a compact structure to save space, simplify wiring, and reduce installation complexity.

b) Enhanced Reliability: Features built-in protection functions (such as over-current, over-temperature, and under-voltage protection), extending the device's lifespan.

c) High Efficiency: Without brush friction losses, the motor has higher electrical energy conversion efficiency, making it suitable for applications requiring high efficiency.

d) High Power Density: The motor's small size and light weight provide substantial power output, ideal for applications with space and weight constraints.

e) Excellent Heat Dissipation: The stationary stator windings allow for more effective heat dissipation, ensuring stable operation during high power output.

f) Precise Control: The electronic controller enables precise control of speed, position, and torque, suitable for high-precision applications.

3) Mabuchi Brushless DC Motor Initialization and Data Acquisition

```
gpio_init(CW_DJ1, GPO, GPIO_LOW, GPO_PUSH_PULL);
gpio_init(CW_DJ2, GPO, GPIO_LOW, GPO_PUSH_PULL);
gpio_init(CW_DJ3, GPO, GPIO_LOW, GPO_PUSH_PULL);

pwm_init (PWM_CH1, 17000, 7200);
pwm_init (PWM_CH2, 17000, 7200);
pwm_init (PWM_CH3, 17000, 7200);

encoder_quad_init (ENCODER_QUADDEC1, ENCODER_QUADDEC1_A, ENCODER_QUADDEC1_B);
encoder_quad_init (ENCODER_QUADDEC2, ENCODER_QUADDEC2_A, ENCODER_QUADDEC2_B);
encoder_quad_init (ENCODER_QUADDEC3, ENCODER_QUADDEC3_A, ENCODER_QUADDEC3_B);

motorspeedx = encoder_data_xl / ENCODER_RESOLUTION * 10;
motorspeedy = encoder_data_yl / ENCODER_RESOLUTION * 10;
motorspeedz = encoder_data_zl / ENCODER_RESOLUTION * 10;
```

Fig. 40. Mabuchi Brushless DC Motor Initialization Code

Motor initialization involves configuring forward and reverse rotation control, PWM, and encoder pins. Additionally, encoder speed needs to be acquired. The final PWM values for the x, y, and z axes (finalpwmx_x, finalpwmx_y, finalpwmx_z) are calculated using a dual-loop PID algorithm and are assigned to the three-axis motor using the PWM_SetDuty function.

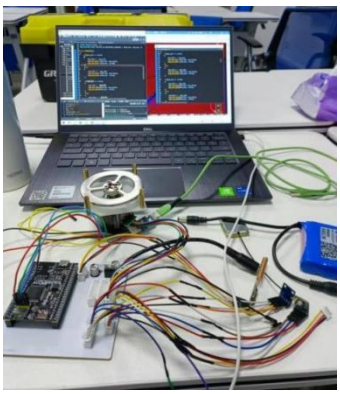


Fig. 41. Mabuchi Brushless DC Motor Running Test

V. CIRCUIT DESIGN

The circuit design for this project uses JLCPCB EDA software. It integrates the MPU6050 gyroscope and accelerometer, HMC5883 magnetic sensor, three Mabuchi Brushless DC Motors, SX1278 LoRa wireless transmission module, ATGM336H GPS positioning module, 12V rechargeable power supply, 12V to 5V voltage regulator LM117T-5.0, and TTL serial camera module on a single circuit board.

All components share a common ground (GND) with the circuit board. The 12V power supply connects directly to the onboard power interface, distributing to the positive lines of the three motors. The power interface connects to the LM117T-5.0 voltage regulator, which adjusts the voltage to 5V before supplying power to the MPU6050, HMC5883, GPS, camera, brushless motors (for the internal driver and encoder start), and the CH32V307 core board. The 3.3V pin of the core board connects to the LoRa module.

The CW/CCW lines for controlling the forward and reverse rotation of the three motors connect to the E2, E4, and E6 pins of the core board, respectively. The encoder A-phase signal lines of the three motors connect to the A0, A6, and B9 pins of the core board, while the encoder B-phase signal lines connect to the A1, A7, and B8 pins. The PWM signal lines connect to the D12, D13, and D14 pins of the core board.

The MPU6050 uses the I2C communication protocol, with the SCL pin connected to the B3 pin of the core board and the SDA pin connected to the B5 pin. The HMC5883 also uses the I2C communication protocol, with the SCL pin connected to the B4 pin and the SDA pin connected to the B6 pin of the core board. The LoRa module uses the UART communication protocol, with the RX pin connected to the B10 pin and the TX pin connected to the B11 pin of the core board. The GPS module uses the UART communication protocol, with the RX pin connected to the A9 pin and the TX pin connected to the A10 pin of the core board. The camera module uses the UART communication protocol, with the RX pin connected to the A2 pin and the TX pin connected to the A3 pin of the core board.

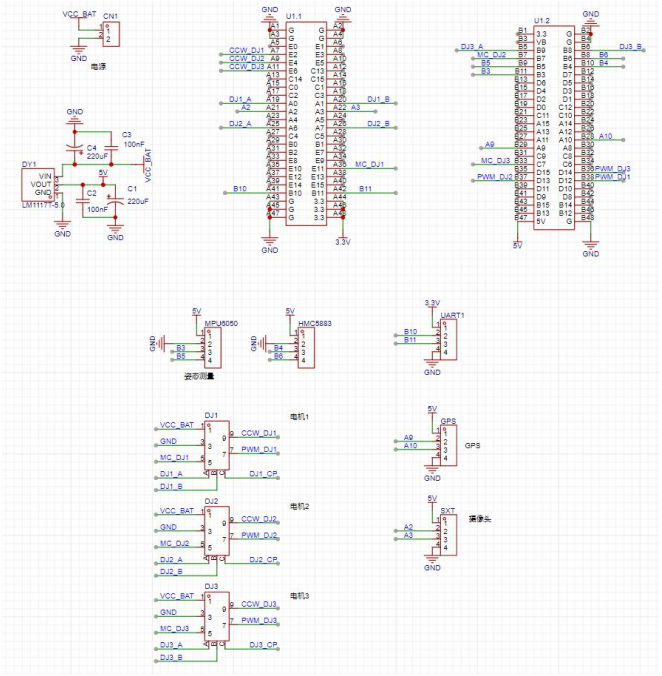


Fig. 42. PCB Components Schematic Diagram

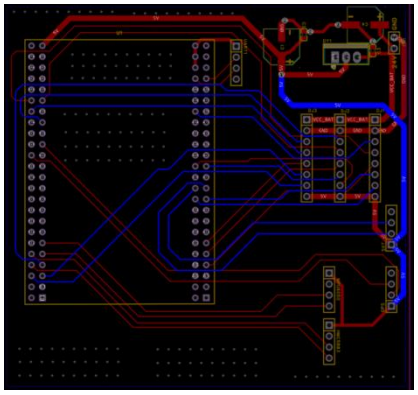


Fig. 43. PCB Circuit Diagram (Applied Version)

In the later stages of development, we made numerous improvements to the original design. We changed the positions of nearly all components, including the LoRa module, three Mabuchi brushless motors, TTL serial camera, ATGM336H GPS, MPU6050 attitude sensor, and HMC5883 magnetic sensor, and rewired the layout. The revised PCB layout is more rational, avoiding the component crowding and wiring congestion seen in the original version. Precise measurements confirmed that the improved circuit board could integrate the LoRa module, ATGM336H GPS, MPU6050 attitude sensor, and HMC5883 magnetic sensor directly onto the board, with only the three motors and camera needing to be external, connected via wiring. This design enhances the level of circuit integration, significantly saves the limited space inside the satellite, reduces weight, and improves aesthetics.

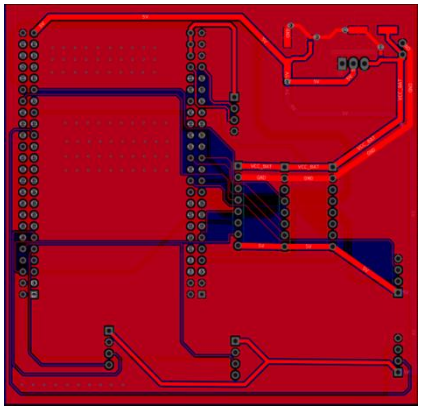


Fig. 44. PCB Copper Plating Diagram (Upgraded Version)

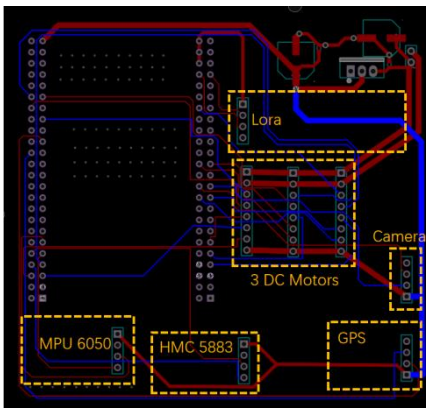


Fig. 45. PCB Circuit Diagram (Upgraded Version)

VI. Upper Computer

A. Vofa+ Software Introduction

Vofa+ is a powerful upper computer software widely used for debugging and monitoring embedded systems. Its main functions include real-time data display, waveform plotting, and serial communication configuration. For the debugging and operation of this project's self-balancing cube satellite, Vofa+ provides a convenient and intuitive graphical interface, greatly facilitating data collection and analysis.

In the practical application of this project, Vofa+ software provides substantial support. Through the attitude display and data transmission interface, we can monitor the cube satellite's working state in real-time, perform precise attitude control, and analyze data. It not only enhances our work efficiency but also improves data visualization and operational convenience. Vofa+ plays a crucial role in the project's debugging and operation.

B. Vofa+ Interface Explanation

1) Attitude Display and Data Transmission Interface

a) Attitude Display: The central 3D attitude display module shows the cube satellite's attitude in space in real-time, helping us intuitively understand its current state. This functionality is achieved through data interaction with the satellite's sensors, ensuring real-time and accurate attitude display.

b) Real-Time 3-Axis Data Plot: The right side of the interface displays real-time waveform graphs showing the cube satellite's attitude changes along the x, y, and z axes, helping us analyze its motion trajectory and stability. This data allows us to determine whether the satellite is balanced and make corresponding adjustments.

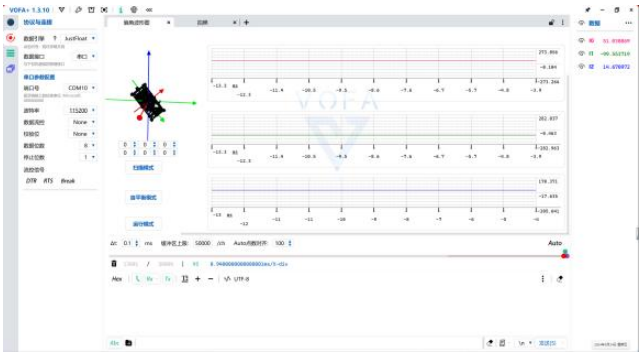


Fig. 46. Attitude Display and Data Transmission Interface

c) Real-Time Attitude Angle Data: This feature displays real-time graphs of the cube satellite's attitude angles over time, including roll, pitch, and yaw angles. These data are crucial for accurately controlling and adjusting the satellite's attitude, ensuring stability in various modes.

d) Protocol and Serial Configuration: The left side of the interface is dedicated to protocol and serial configuration, allowing adjustments to data engine, serial port number, baud rate, data bits, parity bits, and stop bits. Proper configuration ensures stable and accurate communication with the satellite, avoiding errors and delays in data transmission.

e) Mode Control: This interface allows switching the cube satellite among three different operating modes: scan mode, self-balancing mode, and operational mode, to meet different functional requirements. For example, in scan mode, the satellite rotates to collect environmental data; in self-balancing mode, it automatically returns to a balanced position to verify its balancing performance; in operational mode, it sends attitude angle data to the upper computer, which displays the changes over time and the satellite's attitude in real-time.

f) Manual Data Transmission: The bottom section supports sending custom commands to the cube satellite for debugging and functionality verification. During debugging,

we can observe the satellite's response to specific commands to make targeted adjustments and optimizations.

2) Camera Control Interface

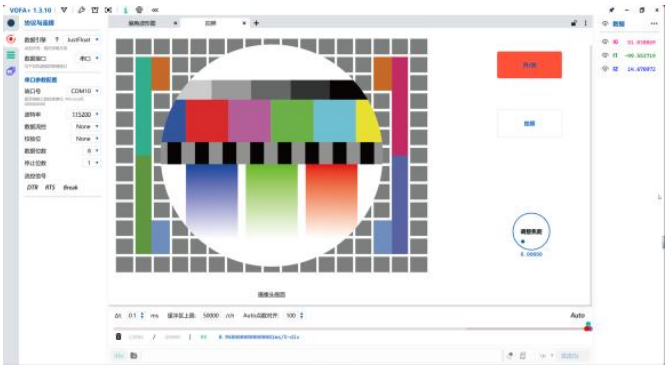


Fig. 47. Camera Control Interface

a) Display Captured Photos: The central image display area shows photos taken by the cube satellite's camera, facilitating image quality inspection and content analysis. This functionality relies on effective communication between the camera module and the upper computer software.

b) Camera Control: Buttons on the right allow control of the camera's power, photo capture, and focus adjustment, ensuring clear and satisfactory images. In practical operation, these control buttons enable flexible management of the camera's working state to meet different shooting needs.

c) Protocol and Serial Configuration: Similar to the previous interface, the left side is for protocol and serial configuration, ensuring communication stability. By configuring different parameters, we can adapt to communication needs in various working environments, ensuring accurate data transmission.

C. Vofa+ 'JustFloat' Introduction

This protocol is a byte stream protocol in the form of little-endian floating-point arrays, transmitting pure hexadecimal floating points to save bandwidth. It is suitable for scenarios with a large number of channels and high transmission frequencies. Given the need for continuous and precise sensor data for the attitude control of the CubeSat, the Justfloat protocol is selected for this project.

The data parsing process of this protocol includes both sensor data and frame tail data, transmitted via UART. During task execution, the chip first sends the sensor data, followed by the frame tail data. Upon receiving both data sets, Vofa+ begins parsing and visually displaying the data on the interface.

```
for (int t_test = 0; t_test < 12; t_test++)
{
    uart_write_byte(UART_3, byte[t_test]);
}

sent_byte[0] = 0X00;
sent_byte[1] = 0X00;
sent_byte[2] = 0X80;
sent_byte[3] = 0X7f;

for (int t_test = 0; t_test < 4; t_test++)
{
    uart_write_byte(UART_3, sent_byte[t_test]);
}
```

Fig. 48. Vofa+ 'JustFloat' Parsing

VII. CONTROL ALGORITHM DESIGN

With the deepening of aerospace research, the demand for low-orbit space exploration and Earth observation missions continues to increase. Traditional satel Based on the predetermined flight missions and control requirements of the control system, we first conduct satellite attitude dynamics modeling. Using the modeling results, we design attitude measurement and control algorithms for the control system.

A. Satellite Attitude Dynamics Modeling

The motion of a space object can be considered as two parts: one part is the translational motion of its equivalent mass point under the influence of external forces, and the other part is the rotational motion around the center of mass caused by external torques. For a satellite, the translational motion is within the scope of orbital dynamics research, while the rotational motion around its center of mass pertains to attitude dynamics.

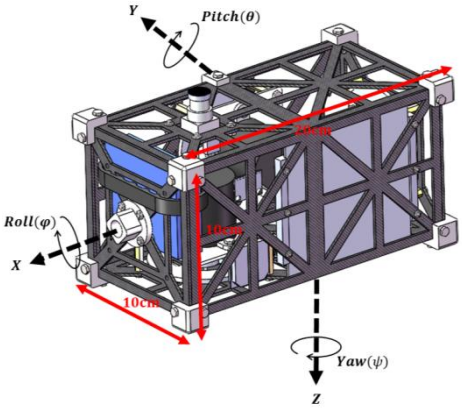


Fig. 49. Cubesat Model Showing Dimensions And Euler Angle Representations

Firstly, the general form of the attitude dynamics equation for a satellite equipped with a momentum wheel is:

$$\boldsymbol{I\dot{\omega} + \omega \times (I\omega + h) = T_c + T_g}$$

The moment of inertia tensor of the satellite is denoted as **I** ,with the principal axes of inertia of the satellite body

serving as the body coordinate system. The moment of inertia tensor is represented as $\mathbf{I} = \mathbf{diag}\left(\mathbf{I_x}, \mathbf{I_y}, \mathbf{I_z}\right)$, and the components of the angular momentum of the momentum wheel in the body coordinate system are represented as $\mathbf{h} = \left(\mathbf{h_x}, \mathbf{h_y}, \mathbf{h_z}\right)$, $\boldsymbol{\omega}$ denotes the absolute angular velocity of the satellite body, the second term on the left-hand side of the equation represents the gyroscopic torque, while $\mathbf{T}$ on the right-hand side represents the external torque. This equation can be expanded as follows:

$$\begin{cases} I_x\dot{\omega}_x + (I_z - I_y)\omega_y\omega_z = -\dot{h}_x + h_y\omega_z - h_z\omega_y + T_x \\ I_y\dot{\omega}_y + (I_x - I_z)\omega_z\omega_x = -\dot{h}_y + h_z\omega_x - h_x\omega_z + T_y \\ I_z\dot{\omega}_z + (I_y - I_x)\omega_x\omega_y = -\dot{h}_z + h_x\omega_y - h_y\omega_x + T_z \end{cases}$$

Where T_x, T_x, T_x represent the external torque acting on the satellite body.

In the case of three-axis stabilized control, the attitude Euler angles are small. The absolute angular velocity of the satellite in the inertial frame can be expressed in the body coordinate system as follows:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 1 & \psi & -\theta \\ -\psi & 1 & \varphi \\ \theta & -\varphi & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -\omega_0 \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{\varphi} - \omega_0\psi \\ \dot{\theta} - \omega_0 \\ \dot{\psi} + \omega_0\varphi \end{bmatrix}$$

Substituting into the dynamic equations, the attitude dynamic equations expressed in terms of the attitude angles are derived as follows:

$$\begin{cases} I_x\ddot{\varphi} + [(I_y - I_z)\omega_o^2 - \omega_o h_y]\varphi + [(I_y - I_z - I_x)\omega_o - h_y]\dot{\psi} = -\dot{h}_x + \omega_o h_z + T_x \\ I_y\ddot{\theta} + h_x(\dot{\psi} + \omega_o\varphi) - h_z(\varphi - \omega_o\psi) = -\dot{h}_y + T_y \\ I_z\ddot{\psi} + [(I_y - I_x)\omega_o^2 - \omega_o h_y]\psi - [(I_y - I_z - I_x)\omega_o - h_y]\varphi = -\dot{h}_z - \omega_o h_z + T_z \end{cases}$$

The angular momentum distribution characteristics of the momentum wheel are as follows:

$$\begin{cases} |h_y| \gg \mathbf{max}(I_x\omega_o, I_y\omega_o, I_z\omega_o) \\ |h_y| \gg (|h_x|, |h_z|) \end{cases}$$

From this, the dynamic equations for a biased momentum satellite can be derived as:

$$\begin{cases} I_x\ddot{\varphi} + \omega_o h\varphi + h\dot{\psi} = T_x \\ I_y\ddot{\theta} = T_y \\ I_z\ddot{\psi} + \omega_o h\psi - h\dot{\varphi} = T_z \end{cases}$$

From the above equation, it can be seen that the roll and yaw channels are coupled, while the pitch channel is decoupled from the roll and yaw channels. The motion of the roll and yaw channels can be analyzed, and the characteristic equation of the roll-yaw coupled motion can be written as:

$$\begin{bmatrix} I_x s^2 + \omega_o h & hs \\ -hs & I_z s^2 + \omega_0 h \end{bmatrix} = 0$$

Namely:

$$\Delta(s) = I_x I_z s^4 + \omega_o h(I_x + I_z)s^2 + h^2 s^2 + \omega_o^2 h^2 \approx I_x I_z(s^2 + \omega_o^2)\left(s^2 + \frac{h^2}{I_x I_z}\right)$$

From the above equation, it can be inferred that the free motion of the satellite exhibits a superposition of two periodic motions. One is the precession motion caused by rotation along the orbit, with an angular frequency of ω_o; the other is the nutation motion caused by the biased momentum and satellite inertia, with an angular frequency of $h/\sqrt{I_x I_z}$. Typically, nutation is a short-period motion, while precession is a long-period motion.

B. Three-Axis Attitude Control

The function of attitude control is to achieve the determination of the satellite's attitude in its orbit and the control of the satellite's three-axis attitude according to the requirements of the flight mission at various stages.

The attitude control system must go through the following stages:

1) Angular Rate Damping. After the separation of the satellite and the rocket, it enters this mode by default. Using information from the magnetometer, the momentum wheel maintains a central angular momentum (1000 rpm). During damping, the GPS receiver is activated to provide position information for comparison with the magnetic field information collected by the magnetometer. It can also receive tracking and control from the ground station, perform orbital parameter updates, and calibration, and start the pitch filter when appropriate. The magnetometer, magnetorquer, and momentum wheel remain powered on, and the gyroscope is remotely controlled to switch on and off according to ground commands.

2) Attitude Acquisition. After the satellite's angular rate damping ends, the momentum wheel and magnetometer are used to achieve Earth acquisition, initially establishing Earth-pointing. The magnetometer and momentum wheel remain powered on, with the momentum wheel maintaining its central rotational speed. The pitch filter estimates the pitch angle and pitch rate. When the pitch filter indicates that the angle has entered the threshold for active pitch control, the magnetorquer actively controls the pitch direction. After maneuvering to the Earth-pointing state, the attitude is maintained, completing Earth acquisition.

3) Three-Axis Stabilized Control. This phase begins after the attitude acquisition stage ends. The satellite's inertial orientation is maintained using the angular

momentum of the momentum wheel, with the magnetorquer as an autonomous control component to complete the satellite's precession and pitch attitude control. The magnetometer and momentum wheel remain powered on, and the GPS receiver stays on continuously.

The control system of the micro-satellite uses a constant-speed momentum wheel to achieve inertial orientation, making the control of the satellite's three-axis attitude a critical task.

The following section will use pitch attitude control as an example. Here, the magnetometer can be used to estimate the satellite's pitch attitude and pitch rate changes. Using the biased momentum control method, the dynamic equations can be simplified, decoupling the pitch channel from the roll/yaw channels. A pitch channel controller can be designed separately, followed by precession control for the roll/yaw channels. The satellite's attitude dynamics in the Y axis direction, decoupled from the X and Z axes, can be expressed as:

$$\boldsymbol{\theta = \tan^{-1}\left\{\frac{-b_x b_{zo} + b_z b_{xo}}{b_x b_{xo} + b_z b_{zo}}\right\}}$$

where $\boldsymbol{b_i}$ represents the measured geomagnetic components and $\boldsymbol{b_{io}}$ represents the geomagnetic components in the orbit reference frame.

Using the above equations and a Kalman filter, an estimate of the pitch attitude angle and its rate of change can be achieved. Given $\boldsymbol{\theta}$ and $\boldsymbol{\dot{\theta}}$, pitch attitude control of the satellite can be accomplished using a PD control law:

$$\boldsymbol{N_y = -kp \cdot \theta - kd \cdot \dot{\theta}}$$

Therefore, the characteristic equation of the system is:

$$\boldsymbol{s^2 + Ds + P = 0}$$

Where, $\boldsymbol{D = \frac{kp}{I_{yy}}, P = \frac{kp}{I_{yy}}}$

The attitude error due to the disturbance torque in the pitch direction is:

$$\boldsymbol{\frac{\theta}{N_\varepsilon} = \frac{1}{I_{yy} \cdot s^2 + kd \cdot s + kp}}$$

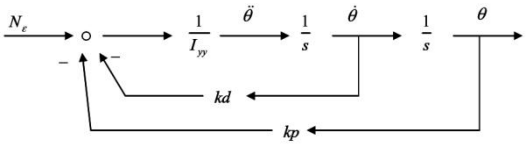


Fig. 50. Pitch Loop Control Block Diagram

In momentum control mode, the command torque is integrated before driving the flywheel. The flywheel speed is measured by a tachometer, forming a feedback loop for flywheel speed. The input to the flywheel is the desired control angular velocity, which is adjusted by varying the voltage to achieve the desired speed. The mathematical model of the flywheel in momentum control mode is shown in the figure below:

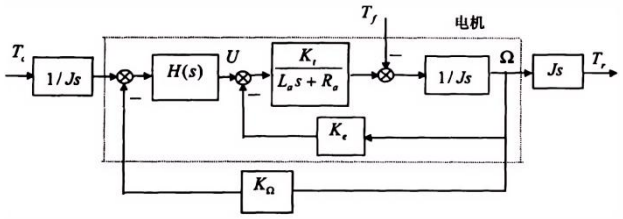


Fig. 51. Flywheel Control System Model

In Fig 50, $\boldsymbol{J}$ is the flywheel moment of inertia; $\boldsymbol{K_\Omega}$ is the flywheel speed feedback coefficient.

Thus, the input-output relationship of the flywheel system is:

$$\boldsymbol{\frac{\Omega}{T_c} = \frac{K_3}{Js(T_2 s + 1)}}$$

Where $\boldsymbol{K_3}$ is the sensitivity of the reaction wheel speed control characteristics, and $\boldsymbol{K_3 = \frac{K_1}{1+K_1 K_\Omega}}$; $\boldsymbol{T_2}$ is the time constant of the reaction wheel, and $\boldsymbol{T_2 = \frac{T_1}{1+K_1 K_\Omega}}$ 。

When only disturbance torque acts on the system, the transfer function between the output and the disturbance torque is:

$$\boldsymbol{\frac{\Omega}{T_f} = \frac{K_4}{T_3 s + 1}}$$

Where $\boldsymbol{K_4}$ is the system sensitivity to disturbance torque, and $\boldsymbol{K_4 = \frac{R_a}{K_t K_e + H(s) K_\Omega}, T_3}$ is the time constant of the reaction wheel under disturbance torque, and $\boldsymbol{T_3 = \frac{JK_\Omega}{K_l(K_e + H(s)K_\Omega)}}$. When the flywheel is in steady operation, the flywheel speed is:

$$\boldsymbol{\Omega = K_4 T_f}$$

For the effect of $\boldsymbol{T_f}$ under disturbance torque, the system's sensitivity to disturbances in torque mode is:

$$\boldsymbol{K_2 = \frac{R_a}{K_t K_e}}$$

In momentum mode, the system's sensitivity to disturbances is:

$$\boldsymbol{K_4 = \frac{R_a}{K_t K_e + K_t H(s) K_\Omega}}$$

If parameters $\boldsymbol{H(s)}$、$\boldsymbol{K_\Omega}$ are chosen such that $\boldsymbol{K_t[(H(s)K_\Omega)+K_e] \gg K_tK_e}$, making $\boldsymbol{K_4 \ll K_2}$, then in momentum control mode, by introducing speed feedback and appropriately selecting parameters, the effect of disturbance torque is greatly suppressed. This indicates that the insensitivity region of the system in momentum control mode is greatly reduced, and the linearity of the flywheel speed control characteristics is significantly improved. It can also be observed that as the values of parameters $\boldsymbol{H(s)}$、$\boldsymbol{K_\Omega}$ increase, the system's ability to counteract disturbance torque improves. However, if these parameters are too large, limit cycle oscillations may occur, so they must be chosen appropriately.

VIII. Software System (Code Design)

The code design is based on the analysis of the aforementioned control algorithms, aiming to achieve attitude control. Attitude control includes attitude measurement, self-balancing stabilization control, and disturbance recovery testing.

Attitude Measurement uses data from sensors such as sun sensors, magnetometers, and star sensors for attitude determination.

Self-Balancing Stabilization Control involves using gyroscope data to control flywheels as attitude control actuators for closed-loop angular velocity control. During attitude stabilization, the satellite is controlled to maintain a specific attitude angle. During mission attitude transitions, the satellite is maneuvered from its current attitude to the target attitude at a specified angular velocity, or a certain axis is scanned continuously at a set rate.

Disturbance Recovery Testing involves applying a simulated disturbance to the satellite at a set attitude and testing its ability to recover its attitude, including recovery accuracy, time, and dynamic performance.

A. Self-Balancing Stabilization Control:

To achieve attitude control of the cube satellite and quickly return to a balanced position under slight disturbances, this project employs a PID algorithm for precise self-balancing attitude control.

1) Algorithm Principle:

The PID (Proportional, Integral, Derivative) control algorithm is a classic method for attitude control. It measures the error between the current and target attitudes and calculates a control signal to adjust the attitude. The PID control algorithm can be tuned to achieve stable and accurate attitude control based on practical needs.

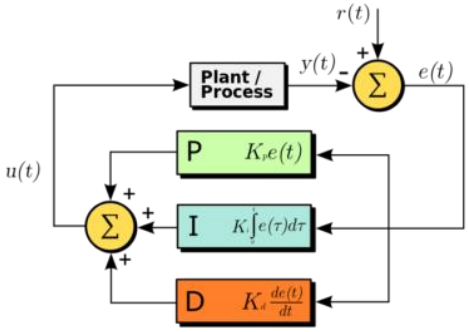


Fig. 52. PID Principle Structure Diagram

$$u(t) = K_P\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de(t)}{dt}\right]$$

Proportional gain (Kp) directly affects the response speed and amplitude of the control system to the current error. If too high, the system may oscillate; if too low, the response to error will be sluggish.

Integral gain (Ki) accumulates the error to help eliminate steady-state error. If too high, it can cause excessive adjustment or instability; if too low, it slows down the recovery speed.

Derivative gain (Kd) predicts error trends to smooth the system's response. If set improperly, it can cause the system to overreact to error changes, leading to instability.

PID control can be subdivided into single-loop and dual-loop control. The following content first introduces the single-loop PID control algorithm and analyzes its effects, then introduces the dual-loop cascade PID control algorithm to improve control performance.

2) Single-Loop PID Control:

The key to single-loop control is tuning the three parameters Kp, Ki, and Kd. The basic tuning principles are: Increase Kp until the output does not oscillate. Decrease Ki until the output does not oscillate. Increase Kd until the output does not oscillate.

Typical tuning steps are as follows:

a) Determining the Proportional Gain Kp: First, remove the integral and derivative terms from the PID controller, setting Ki=0 and Kd=0, thus making the PID control purely proportional. Set the input to 60%-70% of the system's maximum allowable value. Gradually increase Kp from zero until the system starts to oscillate. Then, decrease Kp from this point until the oscillations disappear. Record this value of Kp and set the PID proportional gain Kp to

60%-70% of this value. The tuning of the proportional gain Kp is now complete.

b) Determine Integral Time Constant Ki: After setting the proportional gain Kp, set a large initial value for the integral time constant Ki. Gradually decrease Ki until the system starts to oscillate. Then, increase Ki until the oscillations disappear. Record this value of Ki and set the PID integral time constant Ki to 150%-180% of this value. The tuning of the integral time constant Ki is now complete.

c) Determine Derivative Time Constant Kd: The derivative time constant Kd is generally not set and can remain at 0. If setting it is necessary, follow the same method used for Kp and Ki, setting it to 30% of the value where no oscillation occurs.

Based on these principles and hundreds of experiments, we determined the single-loop PID control parameters:

```
// PID controller parameters
float kp1 = 50;
float ki1 = 0.1;
float kd1 = 2;
float kp2 = 50;
float ki2 = 0.1;
float kd2 = 2;
float kp3 = 50;
float ki3 = 0.1;
float kd3 = 2;
```

Fig. 53. Kp,Ki,Kd Parameters

After tuning, we implemented the single-loop PID control code in MounRiver Studio for motor control:

Define PID controller parameters, proportional factor, integral coefficient, and derivative coefficient. The PID controller calculates the input values from the sensor outputs (MPU6050 and HMC5883) to update attitude angles and angular velocity information in real-time, using the PID output to control the motor PWM and achieve cube satellite attitude control.

```
void pid_controller(float target_roll, float target_pitch, float target_yaw,
                    float current_roll, float current_pitch, float current_yaw,
                    float* roll_output, float* pitch_output, float* yaw_output) {
  error[0] = target_roll - current_roll;
  error[1] = target_pitch - current_pitch;
  error[2] = target_yaw - current_yaw;

  integral[0] += error[0];
  integral[1] += error[1];
  integral[2] += error[2];

  derivative[0] = error[0] - prev_error[0];
  derivative[1] = error[1] - prev_error[1];
  derivative[2] = error[2] - prev_error[2];

  *roll_output = kp * error[0] + ki * integral[0] + kd * derivative[0];
  *pitch_output = kp * error[1] + ki * integral[1] + kd * derivative[1];
  *yaw_output = kp * error[2] + ki * integral[2] + kd * derivative[2];

  prev_error[0] = error[0];
  prev_error[1] = error[1];
  prev_error[2] = error[2];
}
```

Fig. 54. Single Loop PID Control Motor Code

Through tuning and testing experiments, we found that the single-loop PID algorithm could not meet the high-precision, high-sensitivity response requirements for the cube satellite. Thus, the project team decided to use a dual-loop cascade PID control.

3) Dual-Loop Cascade PID Control (Position Loop + Velocity Loop)：

The dual-loop cascade PID control system consists of an inner loop and an outer loop to achieve precise control. The inner loop is a fast-responding velocity control loop, while the outer loop is a slower position control loop. This design allows the system to quickly respond to external disturbances while maintaining precise final control objectives.

Inner Loop ensures the system can quickly and accurately follow the speed set by the outer loop. The inner loop's response speed is much faster, quickly eliminating disturbance effects.

Outer Loop responsible for the system's position control, setting the final target of the system's movement. The outer loop adjusts slowly, providing stable and accurate control instructions to the inner loop. Adjusting the outer loop's PID parameters affects the control system's target-following performance, optimizing stability and lag characteristics.

The input parameters for the inner loop (position loop) are derived from the gyroscope data measured by the MPU6050 and processed by the MahonyAHRS algorithm to obtain pitch, roll, and yaw angles. The outer loop (velocity loop) receives input data from both the position loop output and the encoder speed measured by the TIM_2 timer.

The project team tested two different structures of dual-loop PID algorithms and, after comparing speed, accuracy, and error results, determined the following PID control algorithm.

```
int balance_x(float Angle,float gyro) //倾角PD控制 入口参数：角度 返回  值：倾角控制PWM
{
    float Balance_KP = 270,Balance_KI = 0,Balance_KD = -2.2;
    float Bias;                                         //倾角偏差
    static float D_Bias,Integral_bias;                  //PID相关变量
    int balance;                                        //PWM返回值
    Bias = Angle - 90;                                    //求出平衡的角度中值 和机械相关
    Integral_bias += Bias;
    if(Integral_bias>30000)Integral_bias=30000;
    if(Integral_bias<-30000)Integral_bias=-30000;
//  D_Bias=gyro;                                        //求出偏差的微分 进行微分控制
    D_Bias *= 0.2;                                          // 一阶低通滤波器
    D_Bias += gyro*0.8;                                   //求出偏差的微分 进行微分控制
    balance=Balance_KP*Bias+Balance_KI*Integral_bias+D_Bias*Balance_KD;   //计算倾角控制的电机PWM  PD控制
    return balance;
}
```

Fig. 55. PID Inner Ring Algorithm Code

```
int velocity_x(int encoder)   //位置式PID控制器 入口参数：编码器测量位置信息，目标位置  返回  值：电机PWM
{
    float Speed_KP=65,Speed_KI=0.05,Speed_KD=0;
    static float Pwm,Integral_bias,Last_Bias,Encoder;
    Encoder *= 0.65;                                        //一阶低通滤波器
    Encoder += encoder*0.35;                              //一阶低通滤波器
    Integral_bias+=Encoder;                               //求出偏差的积分
    if(Integral_bias>20000)Integral_bias=20000;
    if(Integral_bias<-20000)Integral_bias=-20000;
    Pwm=Speed_KP*Encoder+Speed_KI*Integral_bias+Speed_KD*(Encoder-Last_Bias);       //位置式PID控制器
    Last_Bias=Encoder;                                    //保存上一次偏差
    return Pwm;                                           //增量输出
}
```

Fig. 56. PID Outer Ring Algorithm Code

4) Dual-Loop Cascade PID Tuning:

a) Angle control:

Cube satellite self-balancing angle control uses a PD(Proportional-Derivative) controller. The tuning process includes determining the polarity and magnitude of Kp and Kd values.

1. Determine Kp Polarity and Magnitude (with Kd=0):

Estimate the range of Kp. For example, if PWM set to 7200 represents 100% duty cycle, setting Kp to 360 would result in maximum output at a 20-degree deviation. Start with Kp=−200; if the satellite tilts further, the polarity is reversed. Set Kp=200; if the satellite begins to stabilize. Increasing Kp until low-frequency oscillation appears. Final value: Kp=350.

2. Determine Kd Polarity and Magnitude (with Kp=0):

Observing the raw angular velocity data from MPU6050, estimate Kd to be within 10. Start with Kd=−5; if oscillation accelerates, reverse polarity. Set Kd=5; if the wheel generates resisting force, adjust Kd to minimize high-frequency vibration. Final value: Kd=2.

b) Velocity control:

After adjusting the upright control, the experiment found that although the momentum wheel can provide reverse torque to keep the satellite self-balancing for a period of time, the momentum wheel will eventually accelerate, and after reaching the speed limit of the motor, the motor can not further accelerate to provide reverse torque, and ultimately cause the satellite to be unbalanced. Therefore, it is necessary to add the motor speed PI controller.

1. Determine the Polarity and Magnitude of Kp and Ki Values:

By using the CH32V307 timer's encoder interface mode to quadruple the frequency of the encoder, and employing the M-method for speed measurement (pulse count every 6ms), we obtain the speed information of the momentum wheel. This speed control is positive feedback. When the satellite tilts, the momentum wheel needs to provide greater counter-torque to correct it, thus exhibiting a positive feedback effect.

First, set Kp=20 and Ki=0.05. Rotate the satellite's momentum wheel, and the wheel will accelerate to the motor's maximum speed. This is positive feedback, which is expected. Hence, the signs of Kp and Ki should be positive.

2. Determining the Magnitude of Kp and Ki values:

The ideal control state achieved through velocity PI control is for the satellite to maintain self-balance while the momentum wheel speed approaches zero. However, in reality, the satellite is subjected to continuous external disturbances, requiring the momentum wheel to constantly rotate to provide counter-torque for balance. The goal of speed control is to keep the momentum wheel speed from becoming too high. If it exceeds 3000 rpm, the motor can no longer accelerate to provide counter-torque.

Based on engineering experience, the Ki value is approximately 1/200th of the Kp value. Initially, set Kp=20 and Ki=0.05. The momentum wheel speed control is weak, making it difficult to maintain a low wheel speed.

Next, set Kp=60 and Ki=0.1. The response of the momentum wheel speed control improves, but the speed oscillations are still significant, making it insufficient to maintain a low wheel speed. Then, set Kp=50 and Ki=0.1. At this point, the performance of the momentum wheel is nearly perfect. Finally, attempt to increase Kp. Set Kp=100 and Ki=0.2. The positive feedback becomes too strong, causing the wheel speed to diverge. Therefore, the final values are determined as Kp=50 and Ki=0.1.

Through the test experiment, the effect of double-loop cascade PID control is obviously better than that of single-loop PID control, which improves the dynamic performance and anti-interference ability of the system. Through the fast response of the inner loop and the precise adjustment of the outer loop, the system can still maintain stability under the external disturbance. The overshoot and adjustment time of the system are reduced, and the robustness and adaptability of the control system are enhanced.

B. Disturbance Recovery Testing

In the disturbance recovery testing, the attitude control system based on the dual-loop cascade PID control algorithm undergoes simulated disturbance tests on a test bench to verify its ability to restore the self-balance state. During the experiment, a certain disturbance force is artificially applied to deviate the satellite from its initial balanced attitude. The dynamic process of the satellite gradually returning to the balanced state through the adjustment of the attitude control system is measured and recorded. The dual-loop cascade PID control algorithm coordinates the inner and outer loops, with the inner loop

controlling angular velocity and the outer loop controlling angles, allowing the control system to quickly respond and precisely adjust the attitude. This test verifies the response speed, stability, and accuracy of the dual-loop cascade PID control algorithm under disturbance conditions and provides data support for further optimization of the attitude control system.

Fig. 57. Disturbance Recovery Testing Setup

C. Measurement and control experiment

The project team conducted measurement and control experiments for the cube satellite. After the satellite's activation, relevant modules send telemetry data to the ground station. The ground measurement and control equipment receive the telemetry data, parse it, and display it on the upper computer. The data is transmitted in real-time to the server, allowing terminal devices to view the telemetry results via Vofa+ software.

We designed a human-machine interaction interface for measurement and control, enabling observation of telemetry status, sending remote commands, and uploading data. The test results show that the ground measurement and control equipment can accurately receive and parse telemetry signals, and the satellite can correctly receive ground commands and perform corresponding actions.

Fig. 58. Measurement and Control Experiment Test

IX. CONCLUSION

In this project, we successfully designed and simulated a 2U CubeSat with the primary objectives of attitude self-stabilization, target scanning, and target tracking. Utilizing a combination of SolidWorks for structural design, SolidWorks Simulation for stress and thermal analysis, and various sensors and modules for control and data transmission, we achieved a robust and functional satellite model. The CubeSat's structure, made from lightweight materials such as carbon fiber and photosensitive resin, demonstrated high strength and effective thermal management in simulations. Our integration of the MPU6050 accelerometer and gyroscope, HMC5883 magnetic sensor, and a GPS module enabled precise attitude measurement and control, while the implementation of PID control with brushless DC motors ensured effective attitude self-stabilization. The use of LoRa modules for wireless communication facilitated real-time transmission of location, six-axis information, and image data, with Vofa+ software providing an intuitive interface for monitoring and control. We proposed an optimization scheme using titanium alloy materials, which can further enhance the CubeSat's stress resistance and reduce deformation. Future work should focus on advanced material research, enhanced sensor integration, improved control algorithms, extensive real-world testing, and cost reduction. This project demonstrated the feasibility and effectiveness of our CubeSat design, laying a solid foundation for future research and development in CubeSat technology, contributing to the broader goals of space exploration and Earth observation missions.

X. REFERENCE

[1] Shin Y ,Yoon S, Seo Y, et al. Radiation effect for a CubeSat in slow transition from the Earth to the Moon[J]. Advances in Space Research, 2015,55(7): 1792-1798.

[2] Spangelo S, Longmier B. Optimization of cubesat system-level design and propulsion systems for earth-escape missions[J]. Journal of Spacecraft and Rockets, 2015, 52(4): 1009-1020.

[3] Song Y J, Ho J, Kim B Y.Preliminary analysis of delta-v requirements for a lunar cubesat Impactor with deployment altitude variations[J]. Journal of Astronomy and Space Sciences,2015,32(3):257-268.

[4] Boshuizen C, Mason J, Klupar P, et al.Results from the planet labs flock constellation[C]//28th Annual AIAA/USU Conference.Boshuizen,USA:AIAA, 2014:1-8.

[5] Bowen J, Villa M, Williams A. CubeSat based rendezvous, proximity operations, and docking in the CPOD mission[C]//29th Annual AIAA/USU Conference on Small Satellites. Bowen,USA:AIAA, 2015:17.
[6] Chuen-Feng Hu1, Shao-Tai Lu1, S. T. Jenq1, Jyh-Ching Juang2 and Jiun-Jih Miau3, "Analysis and Design of the LEAP Structure System", The Fourth Asian Space Conference, 2008.
[7] G.F. Brouwer, W.J. Ubbels, A.A. Vaartjes and F. te Hennepe, "Assembly Integration and Testing of Delfi-C3 Nanosatellite", 59th International Astronautical Congress, 28 September _ 3 October 2008.
[8] "Space Launch System Dnepr User Manuel", no. 2, November 2001.
[9] "Polar Satellite Launch Vehicle User Guide", no. 4, March 2005.
[10] M. Cihan, "A Methodology for the Structural Analysis of Cubesat", Department of Space Engineering, 2008.
[11] W.J. Larson and J.R. Wertz, "Space Mission Analysis and Design" in , CA:Microcosm Inc., 1997.
[12] Melahat Cihan, Ozan Oguz Haktanir, Ylke Akbulut and A. Rüstem Aslan, "Flight Dynamic Analysis of ITUpSATl", International Workshop on Small Satellites, 2008.
[13] Armen Toorian, "Redesign of the Poly Picosatellite Orbital Deployer for the Dnepr Launch Vehicle", California Polytechnic State University, 2007.
[14] SPL - Single Picosatellite Launcher User Manuel, [online] Available: http://www.astrofein.com/2728/dwnld/Datenblatt_SPL.pdf.
[15] Randy Ting and Alex Yang, "Material Selection and Finite Element Analysis for Structural Support of UCI CubeSat Spacecraft" in Senior Design Project University of California, Irvine, 2009.
[16] ECSS Secretariat, ECSS-E-ST-32-03C- Space Engineering: Structural Finite Element Models, ESA-ESTEC, The Netherlands,2008.
[17] European Space Agency and M. Appolloni, Final YGT Report -Structure, European Space Agency, 2004.
[18] "Amandine Denis, QB50 Systems Requirements and Recommendations (Issue 7)," 2015, February 2003, https://www.qb50.eu/index.php/tech-docs/category/25-up-to-date-docs. html.

Appendix:

1. MPU6050 Initialization

```
uint8 mpu6050_init(void)
{
   uint8 return_state = 0;
#if MPU6050_USE_SOFT_IIC
   soft_iic_init(&mpu6050_iic_struct,MPU6050_DEV_ADDR,MPU6050_SOFT_IIC_DELAY,MPU6050_SCL_PIN,MPU6050_SDA_PIN);
   system_delay_ms（100）;
do
{
   if(mpu6050_self1_check())
   {
   zf_log(0, "MPU6050 self check error.");
   return_state = 1;
   break;
   }
mpu6050_write_register(MPU6050_PWR_MGMT_1, 0x00);
mpu6050_write_register(MPU6050_SMPLRT_DIV, 0x07);
mpu6050_write_register(MPU6050_CONFIG, 0x04);
mpu6050_write_register(MPU6050_GYRO_CONFIG, MPU6050_GYR_SAMPLE);
mpu6050_write_register(MPU6050_ACCEL_CONFIG, MPU6050_ACC_SAMPLE);
mpu6050_write_register(MPU6050_USER_CONTROL, 0x00);
mpu6050_write_register(MPU6050_INT_PIN_CFG, 0x02);
}while(0);
   return return_state;
}
```

2. MPU6050 Read Data

```
void mpu6050_get_acc(void)
{
uint8 dat[6];
mpu6050_read_registers(MPU6050_ACCEL_XOUT_H, dat, 6);
mpu6050_acc_x=(int16)(((uint16(dat[0]<<8 | dat[1]));
mpu6050_acc_y=(int16)(((uint16(dat[2]<<8 | dat[3]));
mpu6050_acc_z=(int16)(((uint16(dat[4]<<8 | dat[5]));
}
void mpu6050_get_gyro(void)
{
uint8 dat[6];
mpu6050_read_registers(MPU6050_GYRO_XOUT_H, dat, 6);
mpu6050_acc_x=(int16)(((uint16(dat[0]<<8 | dat[1]));
mpu6050_acc_y=(int16)(((uint16(dat[2]<<8 | dat[3]));
mpu6050_acc_z=(int16)(((uint16(dat[4]<<8 | dat[5]));
}
```

3. HMC5883 Initialization

```
void HMC5883_Init(void)
{
soft_iic_init(&HMC5883_iic_struct,HMC5883_ADDR,HMC5883_SOFT_IIC_DELAY,HMC5883_SCL_PIN,HMC5883_SDA_PIN);
system_delay_ms(100);
HMC5883_write_register(0x00, 0x58);
HMC5883_write_register(0x01, 0x60);
HMC5883_write_register(0x02, 0x00);
}
```

4. HMC5883 Read Data

```
void Multiple_Read_HMC5883(void)
{
uint8_t buf[6];
HMC5883_read_registers(HMC5883_REG_X_MSB, buf, 6);
X = (int16)(((uint16)buf[0] << 8 | buf[1]));
Y = (int16)(((uint16)buf[2] << 8 | buf[3]));
Z = (int16)(((uint16)buf[4] << 8 | buf[5]));
system_delay_ms(5);
}
```

5. GPS Initialization

```
void gnss_init (gps_device_enum gps_device)
{
    const uint8 set_rate[] = {0xF1, 0xD9, 0x06, 0x42, 0x14, 0x00, 0x00, 0x0A, 0x05, 0x00, 0x64, 0x00, 0x00, 0x00, 0x60, 0xEA, 0x00, 0x00, 0xD0, 0x07, 0x00, 0x00, 0xC8, 0x00, 0x00, 0x00, 0xB8, 0xED};
     const uint8 open_rmc[] = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x05, 0x01, 0x00, 0x1A};
    const uint8 close_gll[] = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x01, 0x00, 0xFB, 0x11};
    const uint8 close_gsa[]      = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x02, 0x00, 0xFC, 0x13};
    const uint8 close_grs[]      = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x03, 0x00, 0xFD, 0x15};
    const uint8 close_gsv[]      = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x04, 0x00, 0xFE, 0x17};
    const uint8 close_vtg[]      = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x06, 0x00, 0x00, 0x1B};
    const uint8 close_zda[]      = {0xF1, 0xD9, 0x06, 0x01, 0x03, 0x00, 0xF0, 0x07, 0x00, 0x01, 0x1D};
```

```
    const uint8 close_gst[] = {0xF1, 0xD9, 0x06, 0x01,
0x03, 0x00, 0xF0, 0x08, 0x00, 0x02, 0x1F};
    const uint8 close_txt[] = {0xF1, 0xD9, 0x06, 0x01,
0x03, 0x00, 0xF0, 0x40, 0x00, 0x3A, 0x8F};
    const uint8 close_txt_ant[] = {0xF1, 0xD9, 0x06, 0x01,
0x03, 0x00, 0xF0, 0x20, 0x00, 0x1A, 0x4F};
    if((TAU1201 == gps_device) || (GN42A ==
gps_device))
    {
    fifo_init(&gnss_receiver_fifo, FIFO_DATA_8BIT,
    gnss_receiver_buffer, GNSS_BUFFER_SIZE);
            system_delay_ms(500);
            uart_init(GNSS_UART, 9600, GNSS_RX,
    GNSS_TX);
            uart_write_buffer(GNSS_UART, (uint8
    *)set_rate, sizeof(set_rate));
                system_delay_ms(200);
    uart_write_buffer(GNSS_UART, (uint8 *)open_rmc,
    sizeof(open_rmc));
    system_delay_ms(50);
     uart_write_buffer(GNSS_UART, (uint8 *)open_gga,
    sizeof(open_gga));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_gll,
    sizeof(close_gll));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_gsa,
    sizeof(close_gsa));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_grs,
    sizeof(close_grs));
    system_delay_ms(50);
                uart_write_buffer(GNSS_UART, (uint8
    *)close_gsv, sizeof(close_gsv));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_vtg,
    sizeof(close_vtg));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_zda,
    sizeof(close_zda));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_gst,
    sizeof(close_gst));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8 *)close_txt,
    sizeof(close_txt));
    system_delay_ms(50);
    uart_write_buffer(GNSS_UART, (uint8
    *)close_txt_ant, sizeof(close_txt_ant));
    system_delay_ms(50);
    gnss_state = 1;
    uart_rx_interrupt(GNSS_UART, 1);
}
}
```

6. GPS Read Data Using RMC phasing

```
static uint8 gps_gnrmc_parse (char *line,
gnss_info_struct *gnss)
{
    uint8 state = 0, temp = 0;
    double    latitude = 0;
    double    longitude = 0;
    double lati_cent_tmp = 0, lati_second_tmp = 0;
    double long_cent_tmp = 0, long_second_tmp = 0;
    float speed_tmp = 0;
    char *buf = line;
    uint8 return_state = 0;
state = buf[get_parameter_index(2, buf)];
    if('A' == state)
    {
        return_state = 1;
        gnss->state = 1;
            gnss -> ns = buf[get_parameter_index(4,
    buf)];
            gnss -> ew = buf[get_parameter_index(6,
    buf)];
            latitude =
    get_double_number(&buf[get_parameter_index(3,
    buf)]);
            longitude =
    get_double_number(&buf[get_parameter_index(5,
    buf)]);
            gnss->latitude_degree = (int)latitude / 100;
    lati_cent_tmp    = (latitude - gnss->latitude_degree *
    100);
    gnss->latitude_cent = (int)lati_cent_tmp;
    lati_second_tmp = (lati_cent_tmp - gnss->latitude_cent)
    * 6000;
    gnss->latitude_second = (int)lati_second_tmp;
    gnss->longitude_degree = (int)longitude / 100;
    long_cent_tmp    = (longitude -
    gnss->longitude_degree * 100);
    gnss->longitude_cent = (int)long_cent_tmp;
    long_second_tmp = (long_cent_tmp -
    gnss->longitude_cent) * 6000;
```

```
    gnss->longitude_second = (int)long_second_tmp;
    gnss->latitude = gnss->latitude_degree + lati_cent_tmp
    / 60;
    gnss->longitude = gnss->longitude_degree +
    long_cent_tmp / 60;
    speed_tmp =
    get_float_number(&buf[get_parameter_index(7,
    buf)]);
    gnss->speed = speed_tmp * 1.85f;
    gnss->direction    =
    get_float_number(&buf[get_parameter_index(8,
    buf)]);
    }
    else
    {
        gnss->state = 0;
    }
    gnss->time.hour       = (buf[7] - '0') * 10 + (buf[8] -
    '0');
    gnss->time.minute     = (buf[9] - '0') * 10 + (buf[10] -
    '0');
    gnss->time.second     = (buf[11] - '0') * 10 + (buf[12] -
    '0');
    temp = get_parameter_index(9, buf);
    gnss->time.day = (buf[temp + 0] - '0') * 10 + (buf[temp
    + 1] - '0');
    gnss->time.month = (buf[temp + 2] - '0') * 10 +
    (buf[temp + 3] - '0');
    gnss->time.year = (buf[temp + 4] - '0') * 10 +
    (buf[temp + 5] - '0') + 2000;
    utc_to_btc(&gnss->time);
    return return_state;
}
```

7. Camera Data Read

```
void VC0706_TakePicture(void) {
    uint8_t Serial_TxPacket1[5];
    Serial_TxPacket1[0] = 0x56;
    Serial_TxPacket1[1] = 0x00;
    Serial_TxPacket1[2] = 0x36;
    Serial_TxPacket1[3] = 0x01;
    Serial_TxPacket1[4] = 0x00;
    uart_tx_interrupt(UART_2, ENABLE);
    uart_write_buffer(UART_2, Serial_TxPacket1, 5);
    uart_tx_interrupt(UART_2, DISABLE);
    uint8_t Serial_TxPacket2[5];
    uint8_t Serial_RxPacket1[9];
    uint8_t PictureLength[4];
    Serial_TxPacket2[0] = 0x56;
    Serial_TxPacket2[1] = 0x00;
    Serial_TxPacket2[2] = 0x34;
    Serial_TxPacket2[3] = 0x01;
    Serial_TxPacket2[4] = 0x00;
    uart_tx_interrupt(UART_2, ENABLE);
    uart_write_buffer(UART_2, Serial_TxPacket2, 5);
    uart_tx_interrupt(UART_2, DISABLE);
    uart_rx_interrupt(UART_2, ENABLE);
    uart_query_byte(UART_2, Serial_RxPacket1);
    uart_rx_interrupt(UART_2, DISABLE);
    PictureLength[0] = Serial_RxPacket1[5];
    PictureLength[1] = Serial_RxPacket1[6];
    PictureLength[2] = Serial_RxPacket1[7];
    PictureLength[3] = Serial_RxPacket1[8];

    uint8_t Serial_TxPacket3[16];
    uint8_t Serial_RxPacket2[10000];
    Serial_TxPacket3[0] = 0x56;
    Serial_TxPacket3[1] = 0x00;
    Serial_TxPacket3[2] = 0x32;
    Serial_TxPacket3[3] = 0x0C;
    Serial_TxPacket3[4] = 0x00;
    Serial_TxPacket3[5] = 0x0A;
    Serial_TxPacket3[6] = 0x00;
    Serial_TxPacket3[7] = 0x00;
    Serial_TxPacket3[8] = 0x00;
    Serial_TxPacket3[9] = 0x00;
    Serial_TxPacket3[10] = PictureLength[0];
    Serial_TxPacket3[11] =    PictureLength[1];
    Serial_TxPacket3[12] = PictureLength[2];
    Serial_TxPacket3[13] = PictureLength[3];
    Serial_TxPacket3[14] = 0x00;
    Serial_TxPacket3[15] = 0xFF;
    uart_tx_interrupt(UART_2, ENABLE);
    uart_write_buffer(UART_2, Serial_TxPacket3, 16);
    uart_tx_interrupt(UART_2, DISABLE);
    uart_rx_interrupt(UART_2, ENABLE);
    uart_query_byte(UART_2, Serial_RxPacket2);
    uart_rx_interrupt(UART_2, DISABLE);

    uint8_t Serial_TxPacket4[5];
    Serial_TxPacket4[0] = 0x56;
    Serial_TxPacket4[1] = 0x00;
    Serial_TxPacket4[2] = 0x36;
    Serial_TxPacket4[3] = 0x01;
    Serial_TxPacket4[4] = 0x02;
```

```
    uart_tx_interrupt(UART_2, ENABLE);
    uart_write_buffer(UART_2, Serial_TxPacket4, 5);
    uart_tx_interrupt(UART_2, DISABLE);
}
```

8. Main Control Code

```
#include "zf_common_headfile.h"
#include "MahonyAHRS.h"
#include "hmc5883.h"
#include <math.h>
#include <Vofa.h>
#define PIT_CH                  (TIM2_PIT )
#define PIT_PRIORITY            (TIM2_IRQn)
#define UART_INDEX              (UART_3)
#define UART_BAUDRATE           (115200)
#define UART_TX_PIN (UART3_MAP0_TX_B10)
#define UART_RX_PIN (UART3_MAP0_RX_B11)
#define UART_PRIORITY       (USART3_IRQn)
#define LED2                    (B4)
#define CW_DJ1                  (E2)
#define CW_DJ2                  (E4)
#define CW_DJ3                  (E6)
#define PIN_DJMC1               (D12)
#define PIN_DJMC2               (D13)
#define PIN_DJMC3               (D14)
#define ENCODER_RESOLUTION      (100)
#define ENCODER_QUADDEC1
TIM5_ENCOEDER
#define ENCODER_QUADDEC1_A
TIM5_ENCOEDER_MAP0_CH1_A0
#define ENCODER_QUADDEC1_B
TIM5_ENCOEDER_MAP0_CH2_A1
#define ENCODER_QUADDEC2
TIM3_ENCOEDER
#define ENCODER_QUADDEC2_A
TIM3_ENCOEDER_MAP0_CH1_A6
#define ENCODER_QUADDEC2_B
TIM3_ENCOEDER_MAP0_CH2_A7
#define ENCODER_QUADDEC3
TIM10_ENCOEDER
#define ENCODER_QUADDEC3_A
TIM10_ENCOEDER_MAP0_CH2_B9
#define ENCODER_QUADDEC3_B
TIM10_ENCOEDER_MAP0_CH1_B8
#define PWM_CH1
(TIM4_PWM_MAP1_CH1_D12)
#define PWM_CH2
(TIM4_PWM_MAP1_CH2_D13)
#define PWM_CH3
(TIM4_PWM_MAP1_CH3_D14)
#define PWM_CH4
(TIM4_PWM_MAP1_CH4_D15)
float fgx, fgy, fgz, fax, fay, faz;
float hxj, fyj, hgj, fyj_a0, fyj_a1;
float roll_output;
float pitch_output;
float yaw_output;
unsigned char byte[12];
unsigned char xbyte[4];
unsigned char ybyte[4];
unsigned char zbyte[4];
uint16_t Gyro[3];
int16 duty = 6900;
unsigned char sent_byte[4];
char Gyrox[25];
char Gyroy[25];
char Gyroz[25];
uint8       uart_get_data[64];
uint8       fifo_get_data[64];
uint8       get_data = 0;
uint32      fifo_data_count = 0;
fifo_struct uart_data_fifo;
float kp1 = 50;
float ki1 = 0.1;
float kd1 = 2;
float kp2 = 50;
float ki2 = 0.1;
float kd2 = 2;
float kp3 = 50;
float ki3 = 0.1;
float kd3 = 2;
float e1;
float e2;
float e3;
float i1;
float i2;
float i3;
float d1;
float d2;
float d3;
float p1;
float p2;
float p3;
float Angle_Balance_x;
float Gyro_Balance_x;
```

```
float Angle_Balance_y;
float Gyro_Balance_y;
float Angle_Balance_z;
float Gyro_Balance_z;
int Balance_Pwm_x=0,velocity_Pwm_x=0;
int Balance_Pwm_y=0,velocity_Pwm_y=0;
int Balance_Pwm_z=0,velocity_Pwm_z=0;
float finalpwmx, finalpwmy, finalpwmz;
int16 encoder_data_xl = 0;
int16 encoder_data_yl = 0;
int16 encoder_data_zl = 0;
int16 motorspeedx = 0;
int16 motorspeedy = 0;
int16 motorspeedz = 0;
uint8 pit_state = 0;
void lora_Init()
{
    uart_init(UART_3, 115200, UART3_MAP0_TX_B10,
    UART3_MAP0_RX_B11);
}
void Float_to_Byte (float f, unsigned char byte[])
{
    FloatLongType fl;
    fl.fdata = f;
    byte[0] = (unsigned char) fl.ldata;
    byte[1] = (unsigned char) (fl.ldata >> 8);
    byte[2] = (unsigned char) (fl.ldata >> 16);
    byte[3] = (unsigned char) (fl.ldata >> 24);
}
void Vofa_JustFloat ()
{
    Float_to_Byte((fyj/100 + 29 ) * 1.0, xbyte);
    Float_to_Byte((hgj/100 - 15 ) * 1.0, ybyte);
    Float_to_Byte((hxj/100 + 25 ) * 1.0, zbyte);
    int i = 0;
    int k = 4;
    int m = 8;
    for (i = 0; i < 4; ++i) {
        byte[i] =   xbyte[i];
    }
    for (k = 4; k < 8; ++k) {
        byte[k] =   ybyte[k-4];
    }
    for (m = 8; m < 12; ++m) {
        byte[m] = zbyte[m-8];
    }
    for (int t_test = 0; t_test < 12; t_test++)
    {
        uart_write_byte(UART_3, byte[t_test]);
    }
    sent_byte[0] = 0X00;
    sent_byte[1] = 0X00;
    sent_byte[2] = 0X80;
    sent_byte[3] = 0X7f;
    for (int t_test = 0; t_test < 4; t_test++)
    {
        uart_write_byte(UART_3, sent_byte[t_test]);
    }
}
void Yijielvbo_X(float angle_m, float gyro_m)
{
    Angle_Balance_x = 0.06 * angle_m + (1-0.06) *
    (Angle_Balance_x + gyro_m * 0.01);
}
void Yijielvbo_Y(float angle_m, float gyro_m)
{
    Angle_Balance_y = 0.06 * angle_m + (1-0.06) *
    (Angle_Balance_y + gyro_m * 0.01);
}
void Yijielvbo_Z(float angle_m, float gyro_m)
{
    Angle_Balance_z = 0.06 * angle_m + (1-0.06) *
    (Angle_Balance_z + gyro_m * 0.01);
}
void Get_Angle(void)
{
    float Gyro_Y, Gyro_Z, Accel_X, Accel_Z, Accel_Y,
      Gyro_X;
    Gyro_Y = mpu6050_gyro_y / 16.4;
    Gyro_Z = mpu6050_gyro_z;
    Accel_X = mpu6050_acc_x;
    Accel_Z = mpu6050_acc_z;
    if(Gyro_Y>32768)   Gyro_Y-=65536;
    if(Gyro_Z>32768)   Gyro_Z-=65536;
    if(Accel_X>32768) Accel_X-=65536;
    if(Accel_Z>32768) Accel_Z-=65536;
    Accel_Y = atan2(Accel_X,Accel_Z) * 180 / PI;
    Gyro_X = mpu6050_gyro_x / 32.8;
    Accel_Y = mpu6050_acc_y;
    if(Gyro_X>32768)   Gyro_X-=65536;
    if(Accel_Y>32768) Accel_Y-=65536;
    Gyro_Balance_x = -Gyro_X;
    Accel_X= (atan2(Accel_Z , Accel_Y)) * 180 / PI;
    Accel_Y= (atan2(Accel_X , Accel_Z)) * 180 / PI;
```

```
        Accel_Z= (atan2(Accel_X , Accel_Y)) * 180 / PI;
        Yijielvbo_X(Accel_X,Gyro_X);
        Yijielvbo_Y(Accel_Y,Gyro_Y);
        Yijielvbo_Z(Accel_Z,Gyro_Z);
}
void PID_Postionx ()
{
    float iIncpid1;
    int get_point=0;
    float in_pid1, out_pid1;
    e1 = 0 - fyj;
    d1 = -fyj - p1;
    i1 += e1;
    if (i1>20000) i1 = 20000;
    if (i1<-20000) i1 = -20000;
    p1 = e1;
    iIncpid1 = kp1 * e1 + ki1 * i1 + kd1 * d1;
    out_pid1 = iIncpid1;
    in_pid1 = out_pid1-get_point;
    roll_output = in_pid1;
}
void PID_Postiony ()
{
    float iIncpid2;
    int get_point=0;
    float in_pid2, out_pid2;
    e2 = 0 - hgj;
    d2 = -hgj - p2;
    i2 += e2;
    if (i2>20000) i2 = 20000;
    if (i2<-20000) i2 = -20000;
    p2 = e2;
    iIncpid2 = kp2 * e2 + ki2 * i2 + kd2 * d2;
    out_pid2 = iIncpid2;
    in_pid2 = out_pid2-get_point;
    pitch_output = in_pid2;
}
void PID_Postionz ()
{
    float iIncpid3;
    int get_point = 0;
    float in_pid3, out_pid3;
    e3 = 0 - hxj;
    d3 = -hxj - p3;
    i3 += e3;
    if (i2>20000) i2 = 20000;
    if (i2<-20000) i2 = -20000;
    p3 = e3;
    yaw_output = kp3 * e3 + ki3 * i3 + kd3 * d3;
    iIncpid3 = kp3 * e3 + ki3 * i3 + kd3 * d3;
    out_pid3 = iIncpid3;
    in_pid3 = out_pid3-get_point;
    yaw_output = in_pid3;
}
void Xianfu_Pwm(void)
{
  int
    Amplitude_x=6900,Amplitude_y=6900,Amplitude_z=
    6900;
  int
    LOWERAmplitude_x=2000,LOWERAmplitude_y=20
    00,LOWERAmplitude_z=2000;
    if(finalpwmx<-Amplitude_x)
    finalpwmx=-Amplitude_x;
    if(finalpwmx>Amplitude_x)
    finalpwmx=Amplitude_x;
    if(finalpwmy<-Amplitude_y)
    finalpwmy=-Amplitude_y;
    if(finalpwmy>Amplitude_y)
    finalpwmy=Amplitude_y;
    if(finalpwmz<-Amplitude_z)
    finalpwmz=-Amplitude_z;
    if(finalpwmz>Amplitude_z)
    finalpwmz=Amplitude_z;
}
void PWMSET ()
{
    if(finalpwmy < 0)
    {
     gpio_set_level (E2,0);
    }
     else
    {
      gpio_set_level (E2,1);
    }
    pwm_set_duty (PWM_CH1, -finalpwmy);
     if(finalpwmz < 0)
     {
      gpio_set_level(E4, 0);
     }
     else
     {
      gpio_set_level(E4, 1);
     }
```

```
        pwm_set_duty (PWM_CH2, -finalpwmz);
          if(finalpwmx < 0)
        {
         gpio_set_level(E6,0);
        }
         else
        {
         gpio_set_level(E6,1);
        }
        pwm_set_duty (PWM_CH3, -finalpwmx);
        pit_state = 0;
}
void PIT2_Run(void)
{
        static int cnt=0;
        cnt++;
        if(cnt>=9)
        {
              cnt=0;
              gpio_toggle_level(LED2);
        }
              mpu6050_get_acc();
              mpu6050_get_gyro();
        fgx=-mpu6050_gyro_x*0.00106414676712212148299
        493466139;
        fgy=-mpu6050_gyro_y*0.00106414676712212148299
        493466139;
        fgz=mpu6050_gyro_z*0.00106414676712212148299
        493466139;
        fax=-mpu6050_acc_x*0.000244140625;
        fay=-mpu6050_acc_y*0.000244140625;
        faz= mpu6050_acc_z*0.000244140625;
        MahonyAHRSupdateIMU(fgx,fgy,fgz,fax,fay,faz);
        fyj=asinf(2*(q0*q2-q1*q3))*5730;
        hgj=atan2f((q0*q1+q2*q3),(1-2*(q1*q1+q2*q2)))*57
        30;
        hxj=atan2f((q0*q3+q1*q2),(1-2*(q2*q2+q3*q3)))*57
        30;
        encoder_data_xl =
        encoder_get_count(ENCODER_QUADDEC1);
        encoder_clear_count(ENCODER_QUADDEC1);
        encoder_data_yl =
        -encoder_get_count(ENCODER_QUADDEC2);
        encoder_clear_count(ENCODER_QUADDEC2);
        encoder_data_zl=encoder_get_count(ENCODER_QU
        ADDEC3);
        encoder_clear_count(ENCODER_QUADDEC3);
        motorspeedx = encoder_data_xl /
        ENCODER_RESOLUTION * 10;
        motorspeedy = encoder_data_yl /
        ENCODER_RESOLUTION * 10;
        motorspeedz = encoder_data_zl /
        ENCODER_RESOLUTION * 10;
        Get_Angle();
        Balance_Pwm_x = balance_x(mpu6050_gyro_x,
        hxj/100);
        velocity_Pwm_x = velocity_x(motorspeedx);
        Balance_Pwm_y = balance_y(mpu6050_gyro_y,
        fyj/100);
        velocity_Pwm_y = velocity_y(motorspeedy);
        Balance_Pwm_z = balance_z(mpu6050_gyro_z,
        hgj/100);
        velocity_Pwm_z = velocity_z(motorspeedz);
        finalpwmx = Balance_Pwm_x + velocity_Pwm_x;
        finalpwmy = Balance_Pwm_y + velocity_Pwm_y;
        finalpwmz = Balance_Pwm_z + velocity_Pwm_z;
        Xianfu_Pwm();
}
int main (void)
{
        clock_init(SYSTEM_CLOCK_144M);
        debug_init();
        gpio_init(LED2, GPO, GPIO_LOW,
        GPO_PUSH_PULL);
        gpio_init(CW_DJ1, GPO, GPIO_LOW,
        GPO_PUSH_PULL);
        gpio_init(CW_DJ2, GPO, GPIO_LOW,
        GPO_PUSH_PULL);
        gpio_init(CW_DJ3, GPO, GPIO_LOW,
        GPO_PUSH_PULL);
        pit_ms_init(PIT_CH, 100);
      interrupt_set_priority(PIT_PRIORITY, 0);
        fifo_init(&uart_data_fifo, FIFO_DATA_8BIT,
        uart_get_data, 64);
        lora_Init();
        uart_rx_interrupt(UART_INDEX, ZF_ENABLE);
    interrupt_set_priority(UART_PRIORITY, (0<<5) || 1);
        mpu6050_init();
        encoder_quad_init (ENCODER_QUADDEC1,
        ENCODER_QUADDEC1_A,
        ENCODER_QUADDEC1_B);
        encoder_quad_init (ENCODER_QUADDEC2,
        ENCODER_QUADDEC2_A,
        ENCODER_QUADDEC2_B);
```

```
    encoder_quad_init (ENCODER_QUADDEC3,
    ENCODER_QUADDEC3_A,
    ENCODER_QUADDEC3_B);
    pwm_init (PWM_CH1, 17000, 7200);
    pwm_init (PWM_CH2, 17000, 7200);
    pwm_init (PWM_CH3, 17000, 7200);
    pwm_init (PWM_CH4, 17000, 7200);
    while(1)
    {
      if(pit_state)
      {
        PIT2_Run();
        PWMSET();
      }
                  fifo_data_count =
    fifo_used(&uart_data_fifo);
    if(fifo_data_count != 0)
    {
    fifo_read_buffer(&uart_data_fifo, fifo_get_data,
    &fifo_data_count, FIFO_READ_AND_CLEAN);
    uart_write_string(UART_INDEX, "UART get
    data:");
    uart_write_buffer(UART_INDEX, fifo_get_data,
    fifo_data_count);
        if (fifo_data_count == 4) {
        pit_disable(TIM2_PIT);
        pwm_set_duty (PWM_CH1, 7200);
        int16 i = 0;
        for (i = 0; i < 10; ++i)
       {
        gpio_set_level(E4,1);
        pwm_set_duty (PWM_CH2, 4000);
         system_delay_ms(100);
        }
         int16 k = 0;
        for (k = 0; k < 10; ++k)
        {
         gpio_set_level(E4,0);
         pwm_set_duty (PWM_CH2, 4000);
         system_delay_ms(100);
         }
         pwm_set_duty(PWM_CH2, 4000);
         pwm_set_duty (PWM_CH3, 7200);
}
        if (fifo_data_count == 5) {
         pit_enable(TIM2_PIT);
         pwm_set_duty (PWM_CH1, 6000);
         pwm_set_duty (PWM_CH2, 6000);
         pwm_set_duty (PWM_CH3, 6000);
         }
        if (fifo_data_count == 2) {
        while(1){
        pit_enable(TIM2_PIT);
        PIT2_Run();
        Vofa_JustFloat();
        system_delay_ms(1);
        }
      }
    }
  }
}
void uart_rx_interrupt_handler (void)
{
    uart_query_byte(UART_INDEX, &get_data);
    fifo_write_buffer(&uart_data_fifo, &get_data, 1);
}
void pit_handler (void)
{
    pit_state = 1;
}
```