Project Proposal For
ECE 445, SENIOR DESIGN

By

Chenxi Wang
Shihua Zeng
Zhuohao Xu
Zhizhan Li

March 2024

Context

1. Introduction

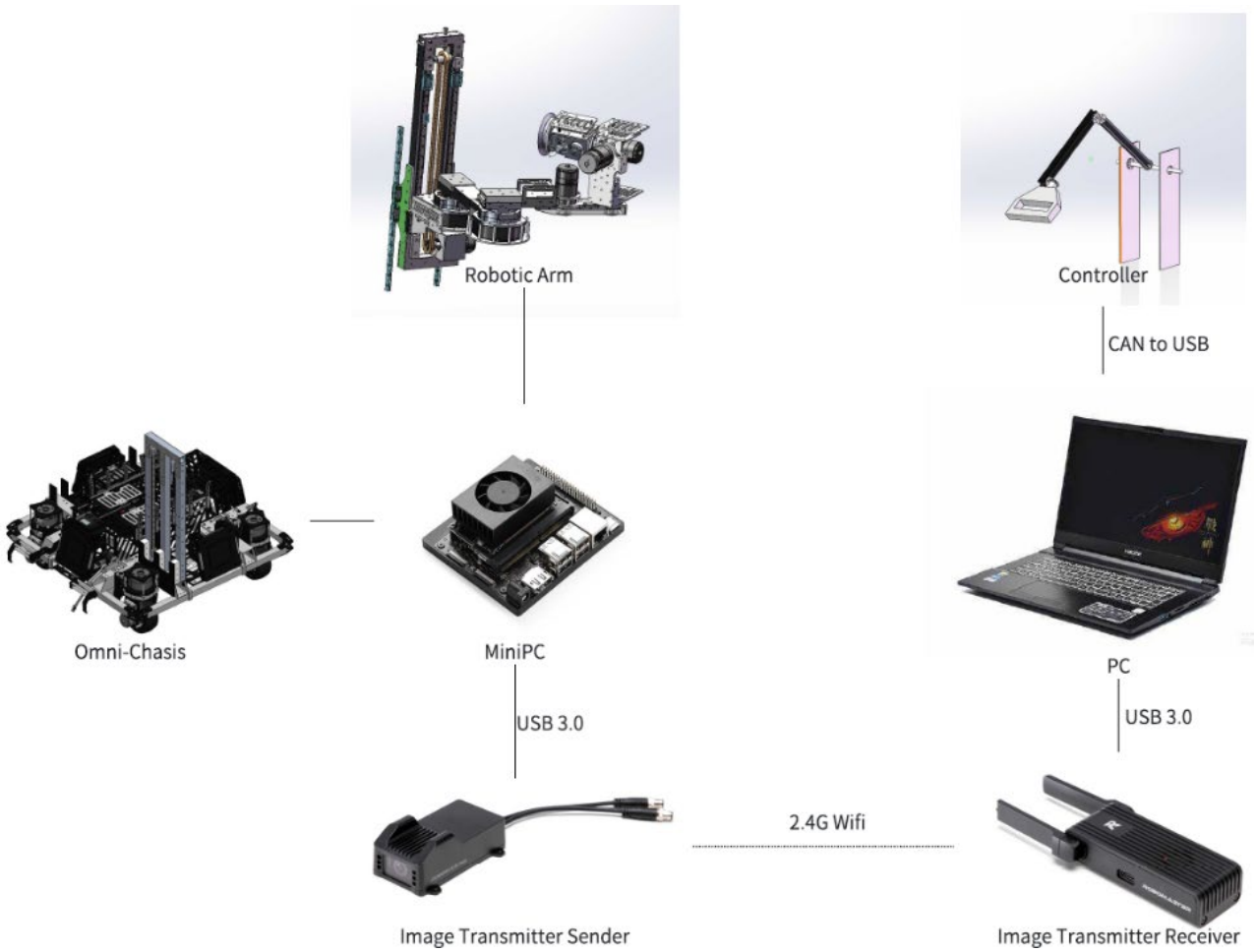
1.1 Problem

In today's rapidly advancing technological landscape, there is a growing need for solutions that can perform tasks in environments that are either hazardous or inaccessible to humans. This encompasses a wide array of scenarios, from executing precise maneuvers in dangerous industrial settings to conducting rescue operations in disaster-struck areas where human responders are at risk of injury or are physically unable to perform the necessary tasks. Traditional methods, such as direct human intervention or the use of cumbersome machinery, often fall short due to safety concerns, physical limitations, or the inability to execute the fine motor skills required in many such situations. Moreover, existing robotic solutions that mimic human dexterity often require the operator to wear specialized equipment, which may not fit all users comfortably or be quickly deployable in emergency situations.

1.2 Solution

Our project proposes a versatile robotic arm system equipped with a camera to capture and analyze human hand movements, enabling it to replicate these actions with high precision. This robotic arm is mounted on a wheeled base, allowing for autonomous or remote-controlled navigation across various terrains and environments. Our innovative approach circumvents the limitations of direct human involvement and the need for wearing potentially cumbersome control gear. The system's design focuses on flexibility and ease of use, making it adaptable to a wide range of operators and scenarios, from intricate tasks in unsafe industrial environments to swift response in post-disaster rescue missions. By combining delicate and powerful operations, our solution addresses the apparent contradiction of requiring both finesse and strength in critical tasks, such as removing heavy debris to rescue earthquake victims. The robotic arm's design emphasizes user-friendly interaction, with the potential for operators to switch control seamlessly, enhancing its utility across diverse applications.

1.3 Visual Aid



1.4 High-level requirements list

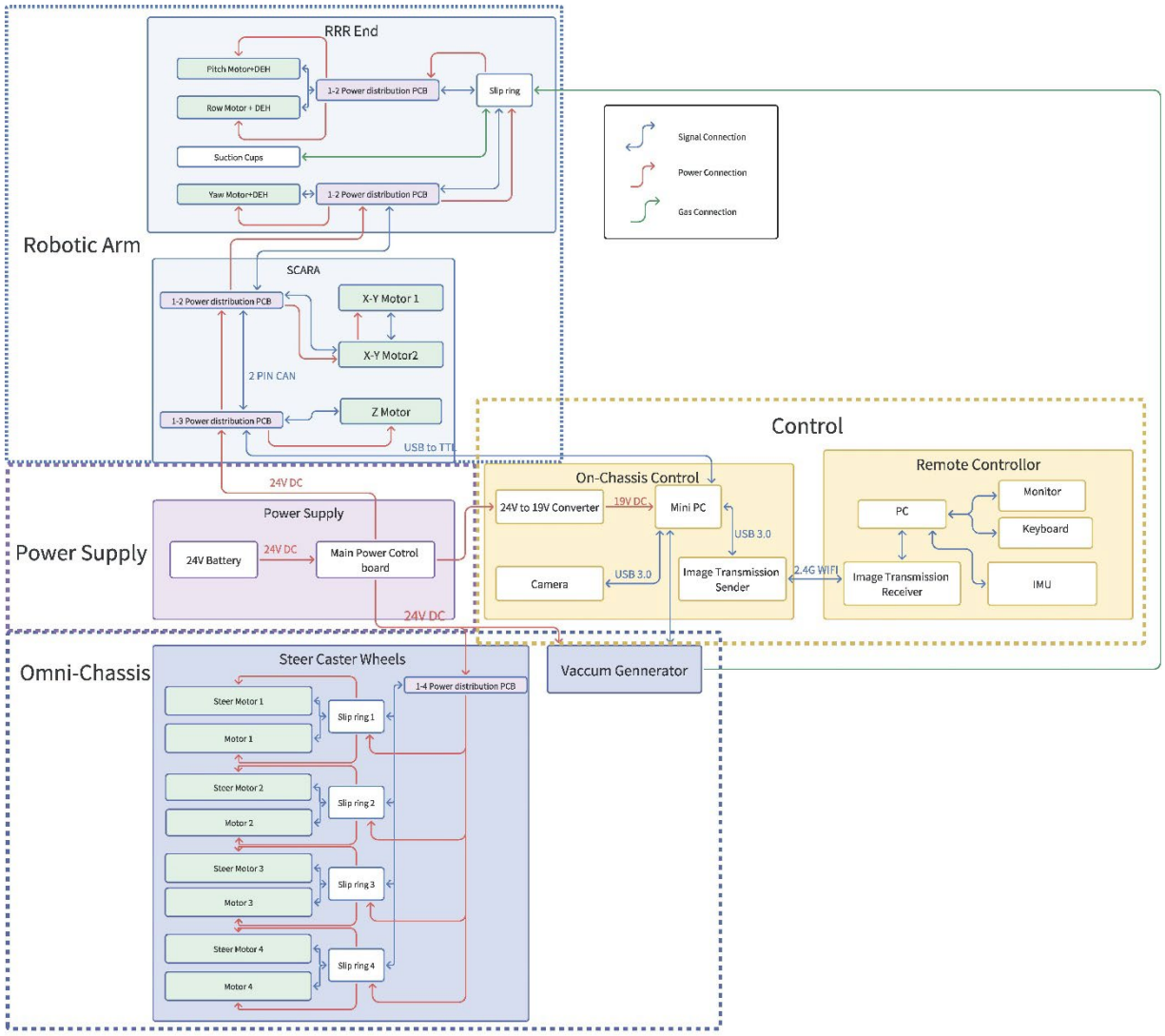
- The robotic arm must accurately replicate human hand movements with a fidelity rate of over 95%, ensuring precise manipulation of objects or tools as captured by the camera system.

- The wheeled base must be capable of navigating diverse terrains with stability and agility, achieving a mobility efficiency that allows for operation in varied environments, including industrial sites and disaster zones.

- The system must provide an intuitive control interface that can be adapted for use by different operators without the need for specialized fitting or extensive training, ensuring rapid deployment and versatility in emergency and routine scenarios.

2. Design

2.1 Block Diagram



3. Subsystem

3.1 Robotic Arm

3.1.1 Overview

Robotic Arm System consist of two parts: Six-axis robotic arm and End-effector

Six-axis robotic arm	The front three axes use a non-traditional SCARA configuration (PRR, with the first joint as P), and the rear three axes use a classical orthogonal spherical wrist configuration (RRR). Using the SCARA configuration instead of the traditional RRR configuration reduces the maximum torque at the joints, and the self-weight of the robotic arm is

	carried by the frame without the constant torque output from the motors. The solution of the six-axis inverse kinematics can be accomplished through a number of strategies.
End-effector	Negative pressure generator choose to use a vacuum generator, can avoid the vibration of the air pump on the mechanical arm of the impact; the end of a single suction cup, plus a rotary degree of freedom, so that the end of the end of the suction cup can be reached in both horizontal and vertical state, the horizontal state is used for the exchange of the vertical state is used for the extraction of the ore.

3.1.2 Six-axis robotic arm

3.1.2.1 Overview

The Six-axis robotic arm consists of two main components: the RRR end and the SCARA subsystem, connected by two 1-2 power distribution PCB boards.

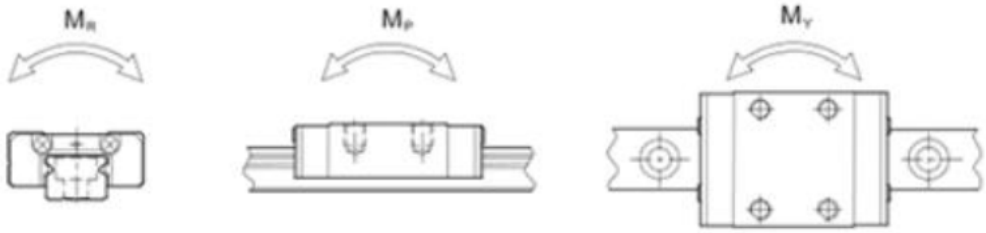
The first three axles of the robotic arm adopt a non-traditional SCARA configuration, while the rest three axes adopt the classic orthogonal spherical wrist configuration (RRR) because using SCARA configuration instead of the traditional RRR configuration can reduce the maximum torque of the joint. Additionally, the dead weight of the arm can be supported by the frame instead of the motor, which needs to continuously output torque and can lead to unnecessary energy consumption.

This hybrid design allows for the RRR end to be used for heavy lifting and the SCARA end to be used for precise manipulation.

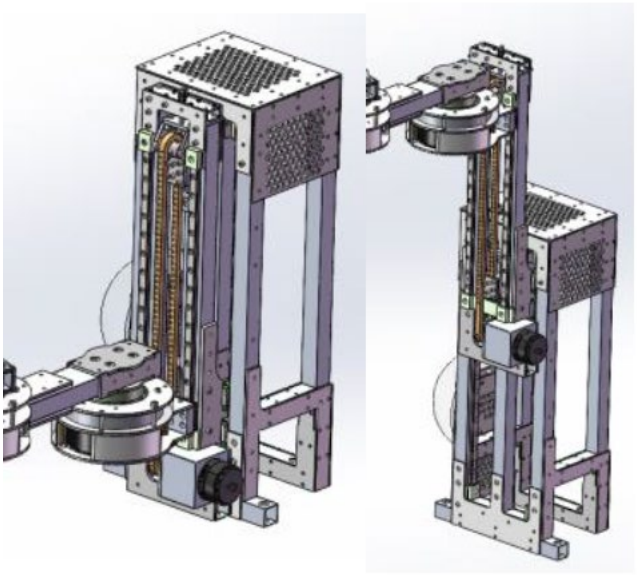
3.1.2.2 Structural design: First joint

The joint is a vertically oriented translational joint. The sum of the masses of the backward joints is about 5kg, and there may be impact working conditions, so a chain drive with a large load is used to realize the lifting function; at the same time, two sets of miniature linear guides are used to ensure the stability of the linear motion and to enhance the rigidity of the structure.

Guideway and slider we use MGN9 series products, each level for two rails side by side, each rail placed on two sliders to avoid individual slider torque, while ensuring the accuracy of the movement. Because the static rated torque and the data of the two directions are not much different, we do not pay much attention to the installation direction, and adopt the arrangement which is more favorable to the space arrangement.

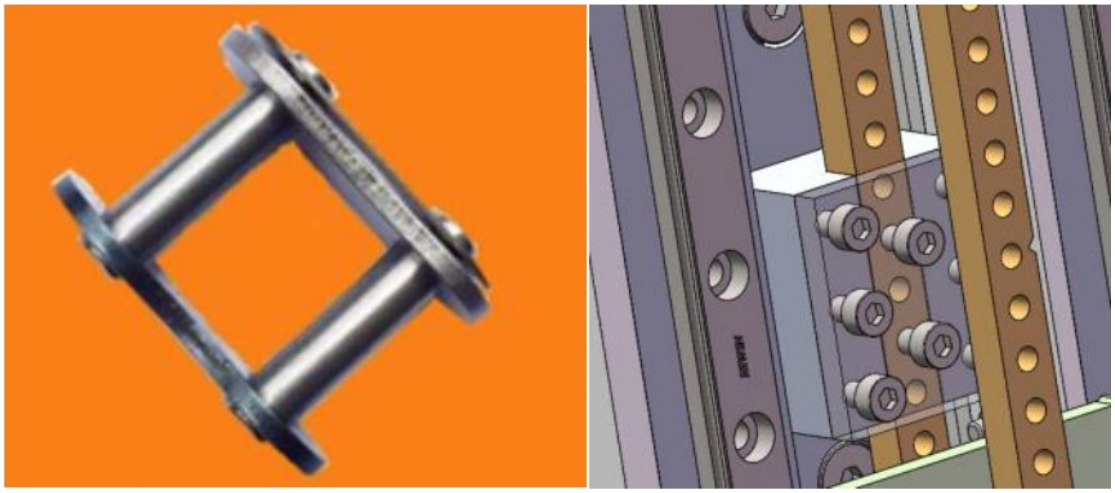


Schematic definition of different static rated torque directions

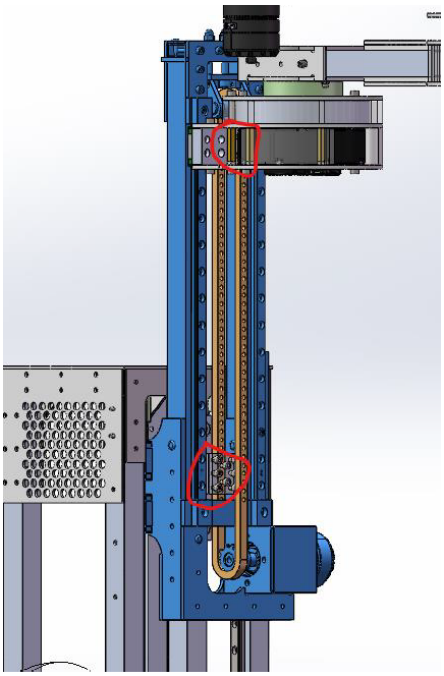


The two slides are synchronized to raise the slide

For the chain we chose 04c, a piece that was not calibrated very carefully, as it was an ancestral solution from the team and should be strong enough. The chain is fastened to the base/secondary frame with clamps and screws. The 04c chain just fits through the M3 screws, but we used half-tooth screws to increase the shear resistance a bit.

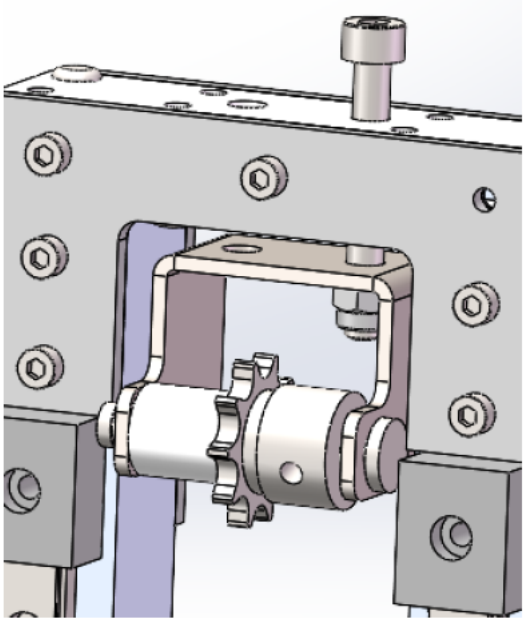


The fixed Chain



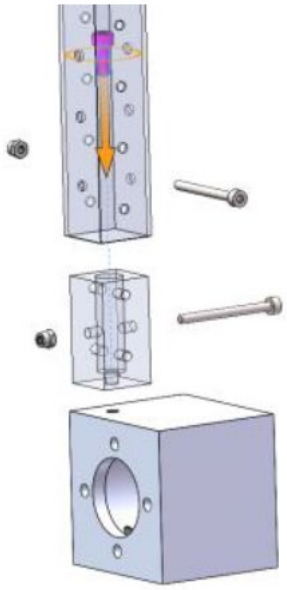
Two chain fixing positions

The first stage of the lifting frame is used to fix the sprocket and the motor. At the upper sprocket, there is a movable tensioning structure, where tensioning is carried out by changing the center distance of the sprocket. A compression spring is placed between the tapping screw and the upper plate, and tensioning can be achieved by adjusting the center distance with two M6 screws. A sheet metal part is produced for strength and is made of cold forged steel.



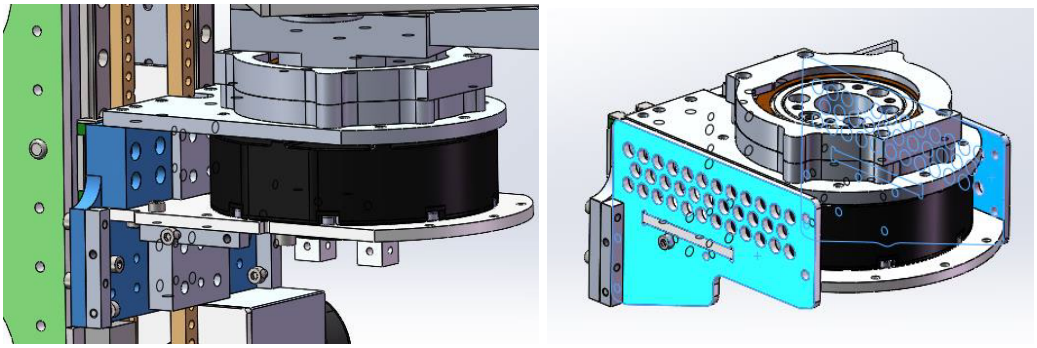
Tensioning structure

The lower part of the motor fixing, this place made a motor seat workpiece. An aluminum square tube insert is attached to the aluminum square tube above, so that the motor base is stably connected to the aluminum square tube that holds the slide.



Motor Block Connection

The first joint (lifting mechanism) and the second joint are joined by an aluminum workpiece. Two plates are clamped at the top to hold the motor in place, and the sides and the underside of the workpiece are connected together by side plates to ensure the overall flexural strength of the joint.

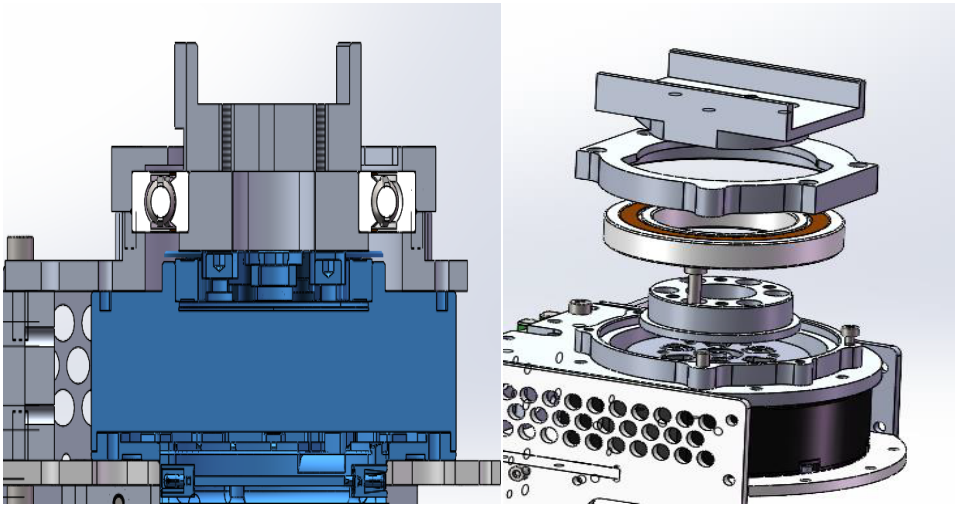


The Connection of first two joints

3.1.2.3 Structural design: Second/Third joint

The second and third joints are rotary joints with the same axis system and joint design, which are introduced here. The length of the connecting rods of joints two and three are both 215mm.

Shafting cross section and shafting exploded view are as follows:



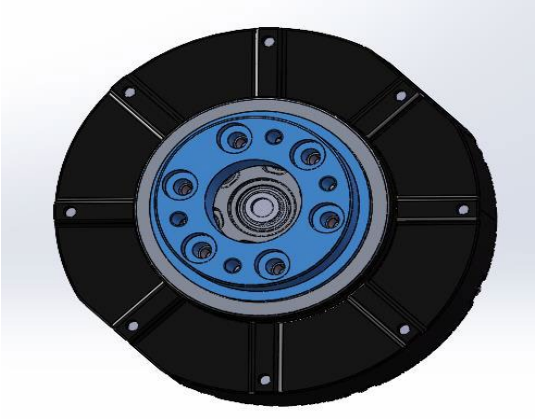
Shafting cross section and shafting exploded view

The motor is a Yushu GO-M8010-6 motor with 16009 deep groove ball bearings, which have been calibrated to meet safety requirements. The deep groove ball bearings are not really suitable for this application because of the large axial loads at rest and the radial and axial loads during movement. However, space is tight here, and bearings such as angular contact ball bearings are thicker. In the end,

the space was compromised and deep groove ball bearings with a smaller thickness were used, but a type with a larger bore diameter was selected to ensure that they would not fail.

Below the bearings is an aluminum workpiece to hold the motor and bearing outer ring in place, and to ensure the concentricity of the assembly by controlling tolerances.

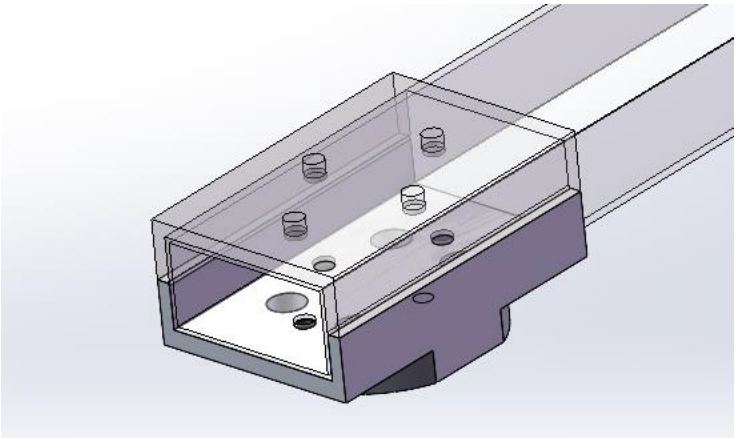
The motor output is flanged with six M4 screw holes. Considering the size of the bearing and the installation space, another aluminum workpiece is designed to extend the output flange. The 6 screws used to connect with the motor output flange will not be disassembled once it is installed, and the 4 threaded holes on the top are used to connect the connecting rods (connected to the next joint), which can be disassembled several times during debugging.



Motor with extended output shaft workpiece

The lower part of the joint is made of aluminum, which fits into the square tube by means of a square tube profile and a center hole press, while the upper part is a U-shaped POM material cover. The joint linkage is connected to the extended output shaft workpiece below by four screws, making it easy to dismantle separately.

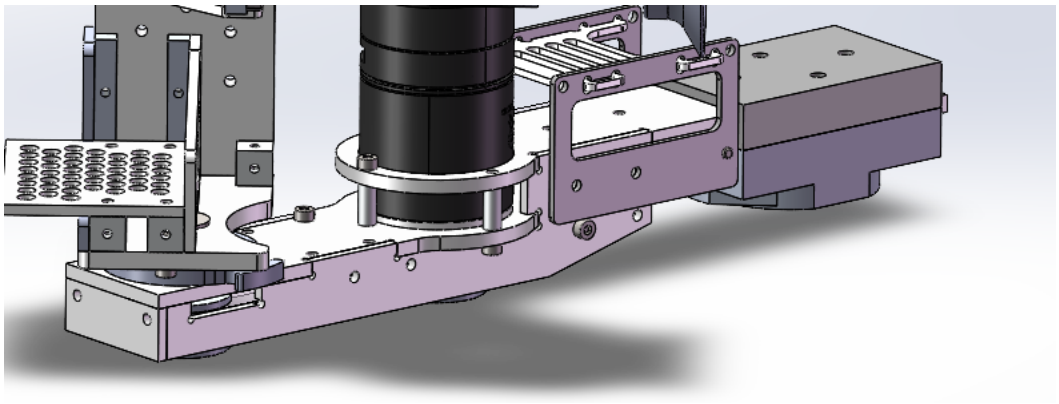
At the same time, this component is fitted with a bearing outer ring compression cap, which also serves as a limiting device.



Articulated connecting rod joints

3.1.2.4 Structural design: Fourth joint

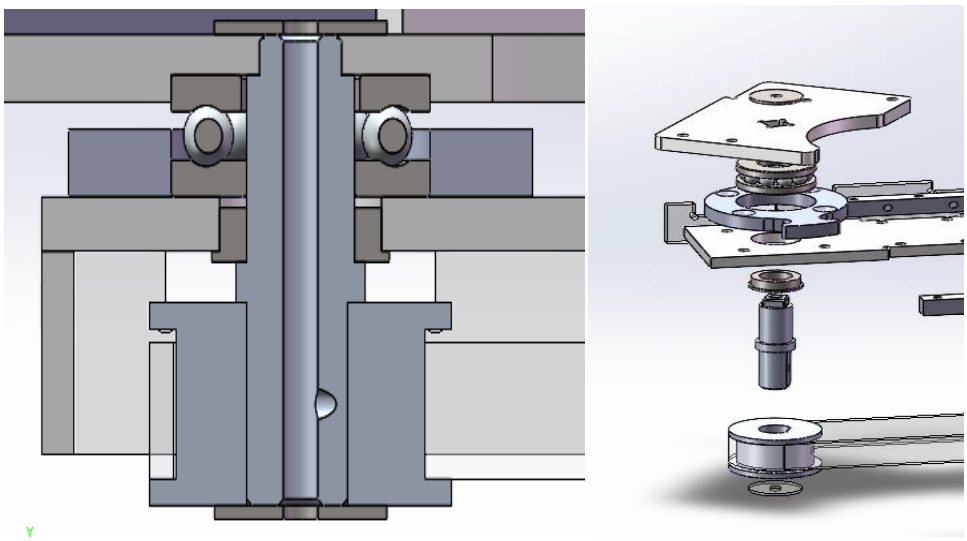
The connecting rod of the 3rd and 4th joints is made of 4 plates with a wall thickness of 3mm, the bending strength is calculated to be sufficient.



connecting rod

The fourth joint is equipped with an M3508 motor with an official gearbox, and a synchronous belt drive is used so that the motor of the joint is rearward-mounted and the longitudinal height of the fourth joint is as small as possible in terms of space. The synchronous belt is 3M series.

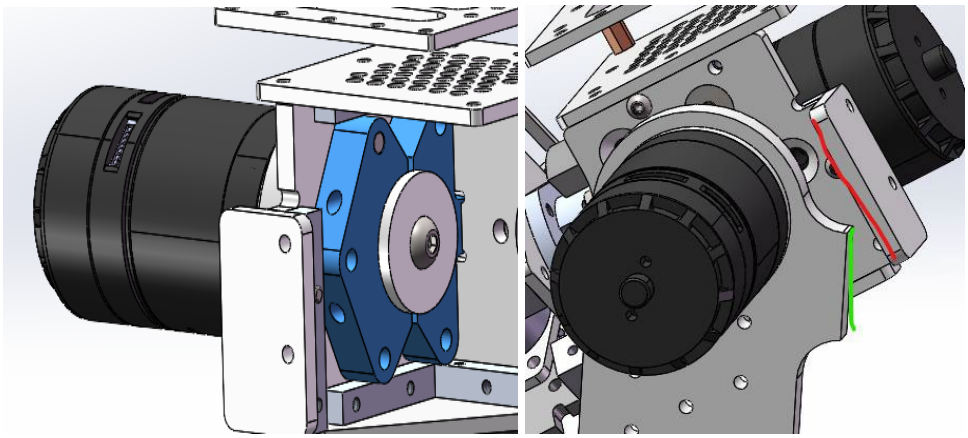
The total weight of the arm and the ore behind the 4th joint is about 4kg, the moment of inertia itself is not too big, but it may be hit by projectiles and impact with walls or cars. Here used a thrust ball bearings a deep groove ball bearings, thrust ball bearings will mainly bear the total static load, at the same time because the synchronous belt selection intentionally let it have a certain tension, so add a deep groove ball bearings to withstand the radial force. The shaft system section and exploded view are as follows:



shaft system section and exploded view

3.1.2.5 Structural design: Fifth joint

The fifth joint is a direct drive M3508 motor. The coupling and limiter is shown below:

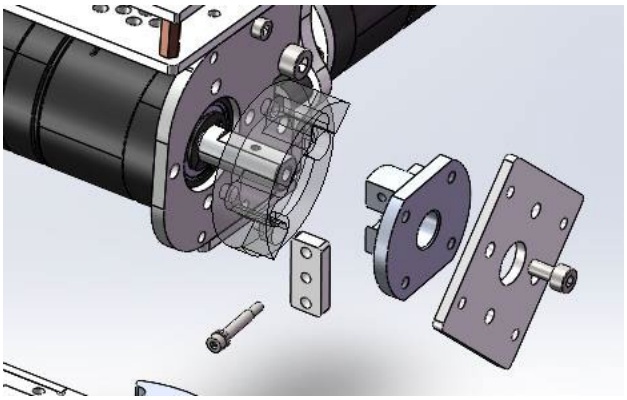


Coupling and limiters

3.1.2.6 Structural design: Sixth joint

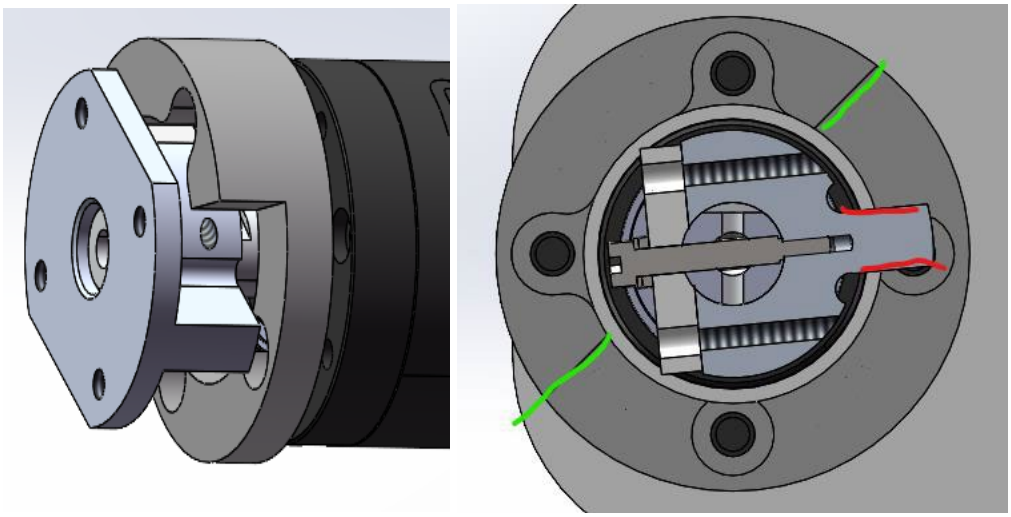
The sixth joint is a direct-drive M3508 motor with an official gearbox. In order to minimize the length to the end (end of the suction cup), it is necessary to compress the space for the motor output shaft coupling as much as possible. Here, a flange was designed to connect to the next joint on the one hand, and on the other hand, to integrate the limit directly into it.

The sixth joint uses a D-type output shaft that is clamped to the motor via two parts. Meanwhile, here the M3 threaded hole of the output shaft is punched through to a through hole, and a corkscrew is inserted in the center, which also serves as an axial fixing.



Exploded view

The limit is achieved by a form fit between a protrusion in the connecting shaft workpiece and a POM material workpiece at the motor stator, as shown in the figure below.



Physical limit

3.1.3 End-effector

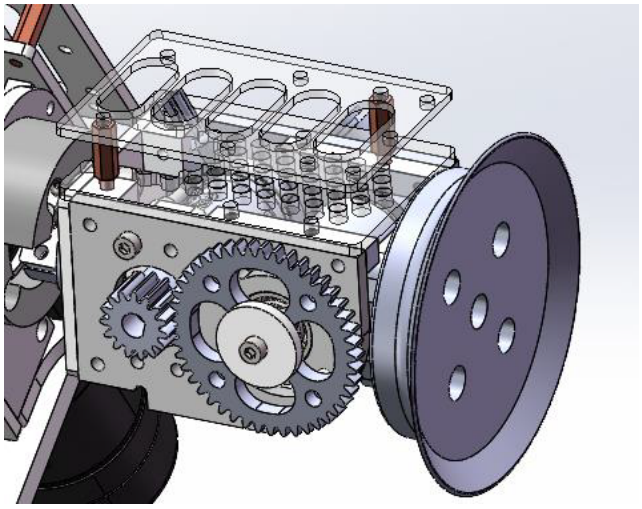
3.1.3.1 Requirements analysis

Considering the end-effector as a whole, the distance from the pivot axis of the fifth joint of the robot arm to the very end of the end-effector is the length of the linkage of the sixth joint. In order to minimize the change of z-axis height caused by the change of pitch angle during mining, we would like to see the length of the linkage of the sixth joint to be as short as possible, i.e., we would like to see the total length of this part of the end-effector to be as short as possible.

Of course, from the point of view of the motor load, we also hope that this part is as light as possible. Because the three axes of the end of the robot arm are orthogonal three axes, and the RPY rotational change will happen, we hope that the overall space of the end-effector is as small as possible, in order to try to avoid the spatial interference with the body of the robot arm in the process of the three-axis movement.

We add a rotational degree of freedom to the end-effector so that the suction cup can be vertically downward, so that we can get the ore from directly above. The success rate of fetching ore from directly above has been tested to be 100%, with a very high degree of fault tolerance.

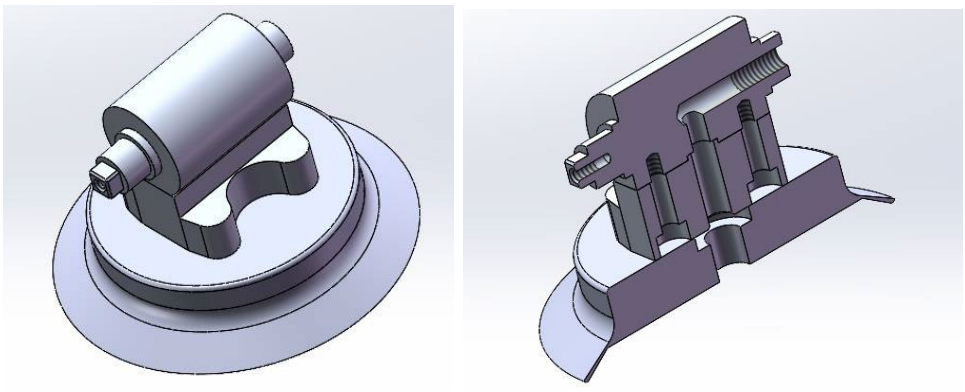
3.1.3.2 Structural design



Overall structure of end-effector

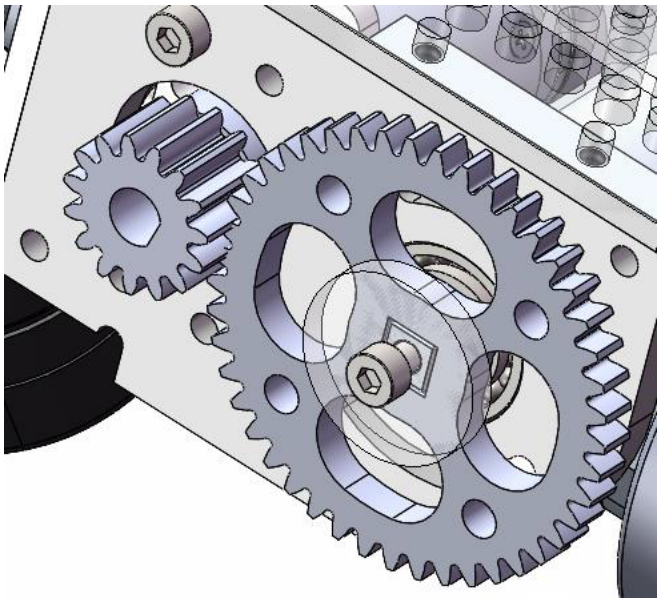
The end-effector consists of an end suction cup and air circuit elements and a motor and transmission structure for the drive.

Our suction cups are H8 black nitrile rubber suction cups, single layer without sponge, because the attitude of the silver ore is very certain and upright, single layer suction cups do not need any special treatment to get the silver ore from above stably. The air inlet fixture behind the suction cup, we did not use the original official fixture, because the volume is too large, it is difficult to reduce; here we designed our own two parts, a shaft and a flange connected to the suction cup, so that the air inlet path over the center of the suction cup axis of rotation, and outside the outside of the rotating quick release coupling, so as to achieve the minimization of space at the same time to elegantly go through the air.



Suction cups and self-finger intakes

Drive part, we use the original M2006 motor, its maximum torque is 1NM, the fifth joint original m3508 motor in the running 30 minutes after the phenomenon of serious heating, the original M3508 motor's maximum torque of 3NM, so here we add a gear drive, reduction ratio of 1:3. here in order to space as small as possible, with a 1-mode gear gears large gears are wire-cutting machining The center drive hole is a 6mm square hole. An M3 threaded hole was added to the shaft workpiece for axial fixation, but the square shaft was sheared off after impact during the test, and it was determined that the cross-sectional area here was too small, and the shaft diameter of the square shaft should be made larger.

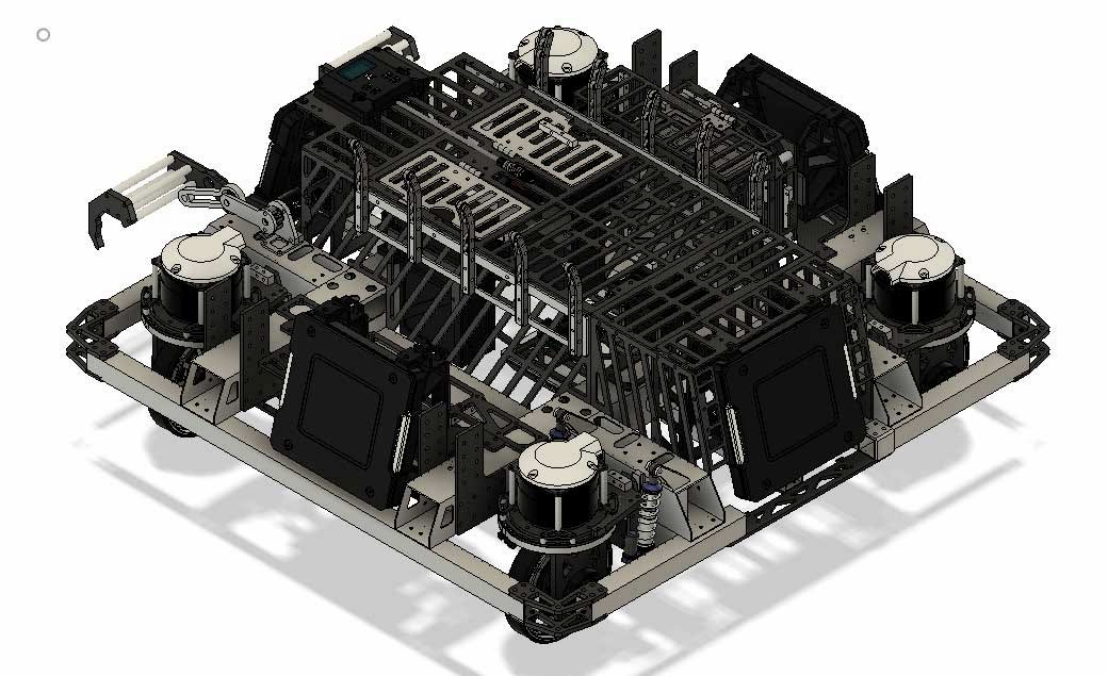


Gear Transmission Schematic

3.2 Omni-Chassis

3.2.1 Overview

The Omni-Chassis system is designed according to the tic-tac-toe structure, which mainly consists of four parts: the main frame, the rescue mechanism, the protection frame, the wheelset, and the ore bin.

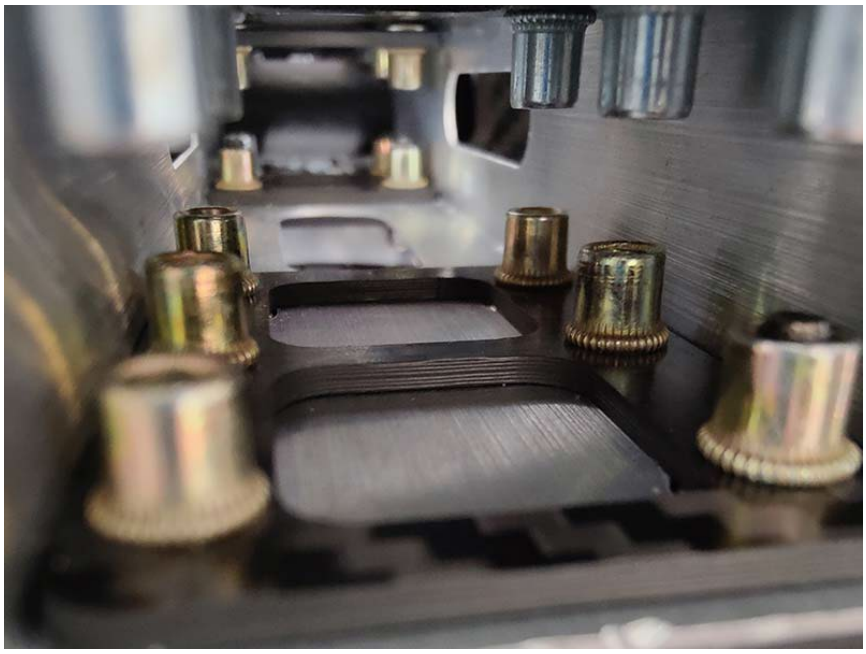


Omni-Chassis

3.2.2 Main Frame

The main frame is the main component of the chassis, which bears the role of carrying the ore bin and robotic arm. The main frame mainly includes the main beam, carbon plate, rescue mechanism and so on.

The main beam is made of thin-walled aluminum square tubes, but after testing, we found that although it is convenient to use the inserts with nuts to connect the square tubes in the chassis, if we dismantle the chassis frequently, the 3D-printed inserts will wear out and fail to hold the nuts, which will make the nuts slip inside the inserts and make it impossible to dismantle the chassis. Therefore, we use 2mmta carbon plate to insert into the main beam(as the figure), and use rivet nut to ensure the connection strength and rigidity of the joint, which has the advantages of light weight, low cost, and easy to disassemble.



Rivet nut in aluminum tube

3.2.3 Rescue Mechanism

The rescue mechanism is divided into two parts: the swipe card rescue mechanism and the towing rescue mechanism.

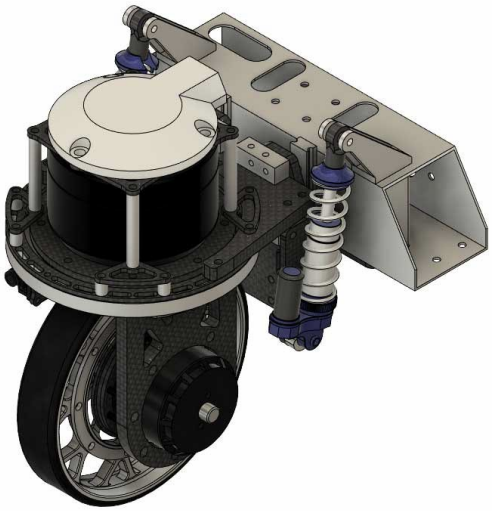
Rescue card rescue mechanism adopts rack and pinion drive, realizes long-distance output through wire rail, and uses M2006 motor for driving. The towing rescue drives the claw rotation by driving the 4-links through the M2006, and utilizes the principle of self-locking 4-links to make the towing process less susceptible to external influences.

The swipe card rescue mechanism

The towing rescue mechanism

3.2.4 Wheelset

The wheelset system is responsible for the normal movement of the robot as well as adapting to the terrain, and includes mechanisms such as wheel supports, wheels, and suspension.

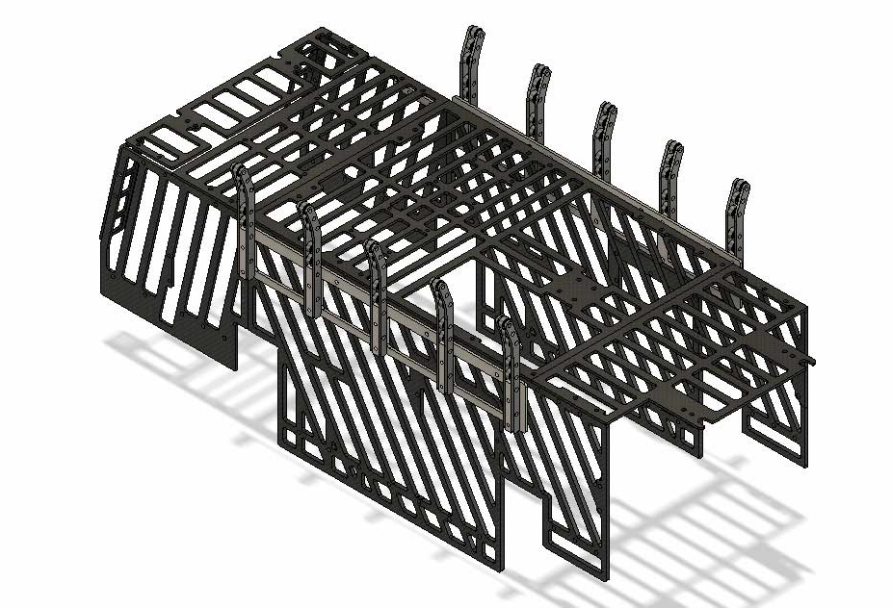


The Wheelset

3.2.5 Ore bin

In order to allow the robot to acquire multiple ores and redeem them at the same time if it has the ability to do so, the designer designed the ore bin in the middle of the chassis to store the ores.

The sidewalls of the ore bin use bearings, and the sidewalls are 135 degrees to ensure that the ores can be stored smoothly in the bin, and the robot arm is able to quickly and continuously pick up and redeem the ores. The bottom of the ore bin is lubricated by a 200*300mm carbon plate with Teflon tape to ensure the tolerance of the robot arm when accessing the ore. The rear of the silo is connected by a 2mmta carbon plate to ensure the rigidity of the sidewalls and to avoid damage to the silo during the ore storage process.



Ore bin

3.3 Power supply

3.3.1 Overview

The Power Supply (PS) system is crucial for the operational efficacy of the movable robotic arm, featuring a simplistic yet highly efficient design composed of a battery and a power control board. The system's primary component, the battery, ensures a steady supply of 24V DC power. This pivotal system not only energizes the robotic arm but also seamlessly integrates with and powers the three other major subsystems: the Robotic Arm system, the Control System, and the Omni-Chassis system, guaranteeing a continuous 24V DC power supply to these systems.

3.3.2 Requirement

Central to the PS system are two key components: a 24V battery and a Main Power Control board. The Main Power Control board functions as the intermediary between the battery and the subsystems, facilitating the distribution of power necessary for their uninterrupted function.

3.3.3 Tolerance Analysis

The PS system is engineered to support a variety of critical components to ensure the robotic arm's high performance and reliability. It must efficiently power at least 3 STM32F04 chips located in the handle, each paired with an acceleration sensor for enhanced precision and control. Additionally, the system powers a main control chip (STM32F04) that orchestrates the operation of the arm, and three servos on the robotic arm side, crucial for the arm's movement and functionality. This requirement highlights the PS system's integral role in sustaining the arm's responsiveness and operational integrity across a range of tasks.

3.4 Control

3.4.1 Overview

The main function of the control system is to control the action of the robotic arm. The system recognizes the hand movement of the manipulator through the and then controls the robotic arm to simulate the hand movement.

The code for the control system is detailed in the Appendix section.

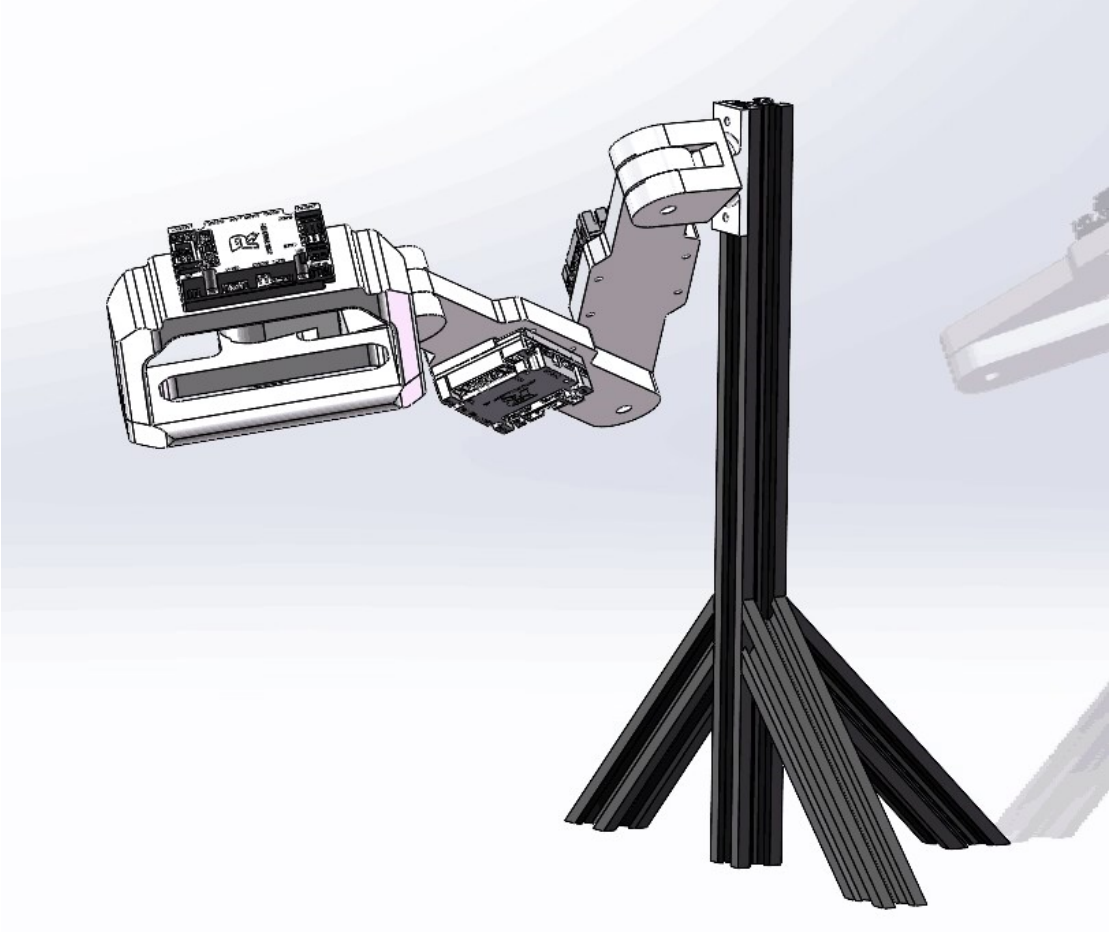
The control system consists of two subsystems: On-Chassis Control System and Remote System.

3.4.2 Remote System

The remote system is mainly responsible for recording and sending the commands we use to control the robot. The commands to control the robot are done by the customized controller.

While traditional controllers can only control each of the six degrees of freedom of the robotic arm through toggles, the use of customized controllers enables gesture tracking and ensures that the operator is able to control all six degrees of freedom at the same time in a comfortable position to complete the exchange operation.

In order to realize this function, the simplest way is to establish a six-axis positional relationship between the controller and the robot arm, and to connect the controller and the manipulator to the robot arm.



customized controller

3.4.3 On-Chassis Control System

On-Chassis Control is mainly responsible for receiving signals from the Remote system and exercising control over the control robot arm.

The system also has a subsystem computer vision component.

The computer vision system consists with a Logitech C270i HD webcam camera and a ThundeRobot NUC mini PC. The camera is mounted on the chassis and connected to the mini PC on the car. The camera captures images of the workspace and sends them to the mini PC. The mini PC, running Ubuntu 22.04, should processes the images and extract the location and orientation of the box, solve the inverse kinematics and pass the control signal to the robotic arm to move the box to the desired location.

The computer vision system should be able to: detect the box (location) in the image and recognized the orientation of the box in the image.

The computer vision system is implemented using the craves framework for robotic arm controlling and the YOLO model for object detection and recognition. For the orientation of the box, we plan to use 6Do-f pose estimation and PnP algorithm.

4. Cost and Schedule

4.1 Cost Analysis

4.1.1 Non-standard Parts and Equipment

Our project incorporates a blend of standard and non-standard components, the latter of which include:

- **Laptops**: Each team member is equipped with a personal laptop, pivotal for programming, design, and communication throughout the project. These laptops are pre-existing resources and, thus, not factored into our project costs.

- **Batteries and Power Lines**: For powering our prototypes, we utilize batteries and power lines graciously provided by our lab. Their cost is absorbed by the lab's resources, emphasizing the supportive infrastructure of our educational institution.

4.1.2 Labor Costs

Zhuohao Xu

Initial Setup and Configuration

Software Installation: Zhuohao Xu begins by installing necessary software tools such as STM32 CubeMX, which facilitates the creation of a programming framework specific to the STM32F407 microcontrollers. This software aids in configuring the microcontrollers' hardware features and in initializing the project with the appropriate middleware and libraries.

Development Environment Setup: Xu selects CLion as the Integrated Development Environment (IDE) for programming and flashing the STM32 microcontrollers. The choice of CLion is strategic, offering robust features for code development, debugging, and management, ensuring an efficient workflow.

Environment Configuration: The process involves meticulous setup and configuration of the development environment, including the installation of necessary drivers, toolchains, and utilities for microcontroller programming. Understanding the STM32 programming architecture and tailoring it to the specific needs of the robotic arm project forms the foundation of this phase.

Programming and Optimization

Control Input Handling: With the setup in place, Xu's primary responsibility is programming the STM32F407 microcontrollers to handle input from the control handle accurately. This involves writing code that efficiently processes input signals, translating them into movements by the robotic arm with minimal delay.

Responsive Movement: Implementing sophisticated sensing and control systems to detect and respond to handle movements promptly, ensuring the robotic arm accurately follows the movements of the handle without unnecessary delays or unintended movements.

Directional Accuracy: Developing complex algorithms that interpret the handle's movements and translate them into precise motor actions in the robotic arm. This ensures that the direction of the handle's movement is consistently mirrored by the robotic arm.

Calibration and Mapping

Proportional Scaling: Xu must ensure the movements from the input handle to the robotic arm are scaled proportionally. This involves calibrating the system to maintain a consistent ratio of movement between the handle and the robotic arm, accommodating different operation modes or payloads as necessary.

Minimized Delay: Employing optimization strategies to minimize system latency. This includes algorithmic optimizations, adjusting parameters for efficiency, selecting more efficient communication protocols, and distributing the computational load effectively across the system components.

Auxiliary Control Integration

Force Feedback: Integrating force feedback mechanisms, potentially informed by visual recognition systems, to provide tactile feedback to the user. This requires programming the microcontrollers to process feedback from the environment and adjust the control signals accordingly.

Calibration Mechanisms: Implementing and programming periodic calibration mechanisms to refine the system's understanding of the robotic arm's position and movements. This is crucial for maintaining ongoing accuracy and reliability in movement reflection and mapping.

Collaboration and Coordination

Throughout these tasks, Zhuohao Xu must coordinate closely with the rest of the project team, ensuring that the control input handle's programming and optimization are aligned with the overall goals of the "Movement Reflection" project. This includes collaborating on selecting and adjusting communication protocols to minimize latency and optimizing integrators based on real-world performance and feedback.

In summary, Zhuohao Xu's role is multifaceted, requiring a deep understanding of microcontroller programming, control theory, and system optimization. Through meticulous programming, calibration, and optimization of the STM32F407 microcontrollers, Xu plays a critical role in achieving a high degree of fidelity between the movement of the input handle and the corresponding action of the robotic arm, ensuring responsive, accurate, and intuitive control for a wide range of precision tasks.

Zhizhan Li

Overall Design and System Architecture

Zhizhan Li is tasked with creating the overall flowchart and design of the project. As the only mechanical engineering major, his profound understanding of mechanical frameworks and structures enables him to oversee the project from a high-level perspective, ensuring that all mechanical components work in harmony to achieve the project's goals. His responsibilities include:

Designing the Mechanical Architecture: Crafting a comprehensive blueprint that outlines the robotic arm's mechanical structure, ensuring it aligns with the project's objectives of responsive and accurate movement.

Integration with Control Systems: Collaborating closely with the team responsible for the arm's sensing and control systems to ensure seamless integration. His role is crucial in ensuring that the mechanical design supports the sophisticated algorithms required for directional accuracy and responsive movement.

Component Selection for Real-world Application

Given the project's emphasis on responsive movement, directional accuracy, and mapping accuracy, Zhizhan Li must meticulously select components that meet the demands of real-world application. This involves:

Choosing Appropriate Materials and Parts: Selecting materials that offer the necessary strength and flexibility, while also considering weight, as it impacts the system's responsiveness and accuracy.

Custom Component Design: Designing custom parts where off-the-shelf components do not meet the specific needs of the project, such as specific actuators or joints that provide the precise degree of movement and force feedback required.

Mechanical Stability and Force Analysis

To ensure the robotic arm performs with minimized delay and maximum fidelity, Zhizhan Li must conduct a thorough analysis of the system's mechanical stability and force dynamics. This includes:

Developing Mechanical Models: Creating models that simulate the robotic arm's behavior under various loads and movements to predict and enhance its stability and performance.

Force Feedback Integration: Designing mechanisms that can accurately convey tactile feedback through the control handle, allowing for nuanced control and interaction with different environments. This involves understanding the mechanical implications of integrating such systems and ensuring they do not compromise the arm's responsiveness or accuracy.

Optimization for Real-time Performance

In alignment with the project's goal to minimize delay, Zhizhan Li's expertise will also be leveraged to:

Material and Design Optimization: Selecting materials and designing components that minimize latency and inertia, thereby enhancing the system's real-time performance.

Collaboration on Optimization Strategies: Working closely with the team to implement hardware optimizations that reduce system delay, ensuring the robotic arm's movements are as close to real-time as possible.

Support for Auxiliary Controls

Zhizhan Li's role extends to integrating auxiliary controls into the mechanical design, facilitating enhanced control and functionality:

Designing for Calibration: Incorporating mechanisms that allow for easy calibration of the system, ensuring long-term accuracy and reliability of the robotic arm's movements.

Accommodating Force Feedback Mechanisms: Ensuring the design supports the integration of force feedback, enabling users to receive tactile feedback that enhances control precision and safety.

In summary, Zhizhan Li is a cornerstone of the "Movement Reflection" project, responsible for the core framework's construction and verification. His role involves a deep collaboration with other team members to ensure that the robotic arm not only meets but exceeds the project's stringent requirements for movement fidelity, responsiveness, and user interaction.

- ## Chenxi Wang

Decoding and Control Precision

My work begins with decoding the movements and positions relayed by the input handle. This involves sophisticated algorithms and computational strategies that interpret the handle's posture and trajectory, translating these into exact instructions for the robotic arm's movements. This translation process is crucial for maintaining the integrity of the movement reflection, ensuring that every nuance of the handle's motion is mirrored by the robotic arm with utmost precision.

Responsive Movement and Directional Accuracy

In line with the project's emphasis on responsive movement and directional accuracy, my role demands the implementation of real-time processing capabilities. This ensures that movements of the handle are followed by the robotic arm without unnecessary delay, maintaining a seamless and intuitive control experience. By fine-tuning the control algorithms, I ensure that the directional intent of the handle's movements is accurately reflected in the robotic arm's actions, thus achieving a high degree of movement synchronization and directional fidelity.

Mapping Accuracy and Minimized Delay

Achieving mapping accuracy involves complex calculations to maintain a consistent ratio between the movements of the input handle and the robotic arm. My responsibility includes adjusting the scaling factor to ensure proportional movement replication, which may require frequent calibration to adapt to different operation modes or payloads. Moreover, minimizing delay is a critical aspect of my role, necessitating ongoing optimization of the system's computational and communication protocols to ensure real-time responsiveness.

Integration of Auxiliary Controls

Another aspect of my responsibility is the integration of auxiliary controls to enhance the system's functionality. This includes the incorporation of force feedback mechanisms to provide tactile feedback to the user, enhancing control precision and safety. Additionally, I am involved in implementing calibration mechanisms that refine the system's accuracy over time, addressing any potential drift in sensor data to maintain reliable movement tracking.

In summary, my role in the "Movement Reflection" project is multifaceted and integral to its success. It encompasses the precise decoding of input movements, ensuring responsive and accurate output, and integrating advanced control features to enhance user interaction. Through my contributions, the project aims to achieve a level of control and interaction that sets new standards for robotic arm technology, ensuring its applicability in tasks that demand precision, real-time control, and intuitive operation.

- ## Shihua Zeng

Personal Work (Shihua Zeng) ## Camera Calibration Due to the forward installation position of the camera, the working distance is approximately 500mm. To meet the visual recognition requirements of all five levels of exchange, lenses with a focal length of 2.8mm or even shorter are needed to obtain a larger field of view. However, a problem arises: the lens's fisheye effect is particularly severe (distortion), but the camera calibration method based on chessboard pattern detection is not very effective. An inaccurate distortion matrix not only affects the results of SolvePNP calculations but can even impact contour recognition. Therefore, an accurate distortion matrix and intrinsic matrix are very important.

First, estimate an approximate value of f/dx and f/dy by dividing the focal length by the camera's image sensor size, then set u0, v0 to half of the screen width and height. This is just a rough adjustment and will be modified later.

Given that the distortion matrix only contains five values, it might be easier to create five sliders within the frame to manually adjust the distortion matrix. The callback function of the slider modifies the distortion matrix, and then two matrices, map1 and map2, are initialized. During the while loop, use `initUndistortRectifyMap(camera matrix, distortion matrix, Mat(), new camera matrix, image size, CV_16SC2, map1, map2)`. The `Mat()` is input directly as is. Since the correction of the image will change

the size of the image, the new camera matrix should ideally be calculated using the `getOptimalNewCameraMatrix()` function, but tests show that using the original camera matrix can also achieve high-accuracy pose estimation results (average error of 2mm under a working distance of 500mm). The `initUndistortRectifyMap` function assigns values to map1 and map2, and the image is adjusted using `remap(original image, new image, map1, map2, INTER_LINEAR)`. After setting camera parameters such as exposure and gain, place a calibration plate in front of the camera for visual adjustment. Due to the principle of projection, the image will appear larger near and smaller far away, but this can be ignored.

The adjustment only needs to ensure that any straight lines on the calibration plate are also straight in the image. The adjustment of the distortion matrix should now be completed. ## Slot Pose Estimation (Traditional Vision, Recognizing the Front Four Corners)

1. **Preprocessing** Correct image distortion, separate channels, and extract red and blue channels to overlay and then binarize. This binary image includes both red and blue, making both red and blue exchange slots recognizable. Due to the unique nature of engineering exchange vision, there won't be situations where both red and blue appear simultaneously during operation, avoiding potential bugs similar to autodescribe red and blue switch, and making debugging more convenient for testing the algorithm's robustness.

2. **Corner Contour Filtering** In the initial screening, I used the following criteria: a. The ratio of contour width to height and height to width is less than 4.5 (height/width < 4.5 && width/height < 4.5) b. Contour area is larger than 400 but smaller than 12000 pixels (area > 400 && area < 12000) c. The number of sides from polygon fitting is more than 5 but less than 9 (edges > 5 && edges < 9) Contours that meet the above conditions are placed into a vector for secondary selection, with the selection criteria being: a. The vector element count equals 4 b. The largest contour area is less than 10 times the smallest contour area (max.size() < 10 * min.size())

3. **Corner Vertex Positioning** a. Perform triangular fitting and find the minimum enclosing circle for contours in the vector. Use the centers of the four circles to determine the midpoint of the front surface of the exchange slot. Note: Due to the image characteristic of appearing larger near and smaller far, the largest angle from triangular fitting may not necessarily be the contour vertex. Similarly, the triangle vertex farthest from the midpoint may not always be the contour vertex. Observation shows that incorrect largest angle degrees generally do not exceed 100 degrees, and vertices only appear closer to the midpoint than the other two angles when exceeding 120 degrees. Based on these results, the following judgment criteria are formulated. b. Calculate the largest angle degrees of the four triangles. If this angle is greater than 130 degrees, it is considered as the contour vertex. c. If there are no angles greater than 130 degrees in the triangle, calculate the distance of each vertex to the midpoint and choose the farthest as the contour vertex. At this point, all four vertices have been identified.

4. **Corner Sorting** When the exchange slot is directly facing, assign the top-right corner as point 0, and sort in a counterclockwise direction: top-left, bottom-left, bottom-right as points 1, 2, 3, respectively. It has been observed that the order of corners in the vector is always counterclockwise (possibly due to characteristics of the OpenCV findContour function), meaning the vector corner has four possible sequences: 0123, 1230, 2301, 3012. The sequence in the vector can be determined by finding point 0. Due to the image characteristic of appearing larger near and smaller far, the contour area corresponding to point 0 may not be the smallest. Point 0's position is determined by finding two small squares beside it, using the following criteria: a. Area is less than 200 but greater than 50 (area < 200 && area > 50) b. The area of the rectangle fitted from the contour is less than 1.2 times the contour area (rectangle.shape.area() < 1.2 * contour.area()) c. Both the width to height and height to width ratios are less than 2 (height/width < 2 && width/height < 2) Now having found two small squares (or possibly only one), find the midpoint between them, and the nearest vertex to this midpoint is point 0. However, if no small square is recognized, then point 0 is definitely away from the camera, meaning the contour with the smallest area corresponds to point 0. An area judgment easily identifies point 0, after which the corners are reordered.

5. **Recognition Point Filtering** Since the rotation vector from PNP solution and the quaternion for final interfacing with the electronic control have certain dependencies, designing such a filter can be complex. Therefore, it's simpler to filter the recognized Point2f points, i.e., filter before SolvePnp. The principle is simple, set the recognition data of the first frame as (0,0), and the current frame data as the previous frame data multiplied by 0.9 plus the current frame data. X(0) = (0,0) X(t) = X(t-1) * 0.9 + X(t) * 0.1

6. **PNP** Use the SOLVEPNP_IPPE_SQUARE method, sorting the corners in the order required for square calculation. Testing showed that SOLVEPNP_IPPE_SQUARE's accuracy and speed are far superior to the common IPPE with 12 calculation points. Even with slight recognition jitter and errors, SOLVEPNP_IPPE_SQUARE can still solve correctly (with minor errors).

7. **Post-Processing (Correctness Judgment of Recognition)** No misrecognition occurred in the field exchange station environment, but home tests showed a chance of misrecognition due to exchange station light leakage, thus additional judgment criteria were added. The recognized quadrilateral, due to incomplete filtering fitting or misrecognition, can cause the quadrilateral to appear concave or its area to momentarily be smaller than the correct rectangular frame. Hence, it's necessary to judge the shape and area of the quadrilateral. Furthermore, since the exchange slot is angled upwards, the calculated quaternion x should be negative. Additionally, due to the filtering principle, the current frame data, if present, should not be identical to the previous frame's data, and the edge intensity of corner contours being weak can cause recognition points to have adjacent pixel jumps, leading to data jittering within a very small range (0.1 degrees, 0.5mm), so different data between adjacent frames is expected. Combining the above, the following judgment conditions are given: a. Any internal angle of the recognized quadrilateral is not less than 40 degrees and not more than 130 degrees (angle < 130 && angle > 40) b. The area of the recognition frame is greater than 30000 pixels (rectangle.shape.area > 30000) c. The calculated quaternion x value does not exceed 0.05

(quaternion.x < 0.05) d. The calculation value of the current frame does not match the previous frame (quaternionNow != quaternionLater)

With an estimated 200 hours of contribution from each team member and an hourly wage of $10—reflective of a typical TA salary at UIUC—our labor calculation per partner is: $10/hour x 2.5 x 200 hours = $5,000. Thus, the total labor cost for our team amounts to $20,000.

4.1.3 Parts

Key components for our project include:

STM32F407 Chips

For the core operations of our project, three STM32F407 microcontroller units are essential. These high-performance chips are at the heart of our system, enabling sophisticated control algorithms and ensuring real-time processing of sensory data and control commands. The STM32F407, with its advanced architecture and features like high-speed memory interfaces, multiple communication interfaces, and an extensive set of peripherals, is particularly suited for our application. It ensures a seamless interface between user inputs and mechanical responses, facilitating the precise control required for the robotic arm's operations.

IBM088 Sensors

Coupled with the STM32F407 chips are IBM088 sensors, which play a critical role in detecting handle movement. These sensors are pivotal for interpreting the user's manual inputs accurately. They provide high-precision measurements of orientation and movement, enabling the system to translate the user's intentions into precise movements of the robotic arm. This ensures a highly responsive and intuitive control experience, essential for tasks requiring fine manipulation and control.

Power Management

The utilization of batteries and power lines provided by our laboratory underscores our project's integration with existing resources. This approach not only fosters innovation but also minimizes additional costs. By effectively managing power through both batteries and direct power lines, we ensure that the system is versatile and adaptable to various operational contexts, whether it requires portability or continuous operation over extended periods.

Chassis

A key addition to our project is a remotely controlled omnidirectional gear system for the chassis. This innovative feature allows our robotic arm to move its base flexibly, enabling it to adapt and respond to complex environments in real life. The ability to reposition the entire unit effortlessly enhances the

system's overall versatility and efficiency, making it suitable for a wide range of applications, from precision tasks to operations in challenging or constrained spaces.

Control System Input Handle

The control system incorporates a remote-operable handle, which, in conjunction with the three STM32F407 chips, works to capture and process user hand input. This setup is crucial for achieving a high degree of control over the robotic arm, allowing for precise manipulation based on the user's manual inputs. The handle's design and functionality are tailored to ensure that the control inputs are intuitively translated into accurate movements by the main control unit, enhancing the user experience and the system's effectiveness in performing delicate tasks.

Visual Recognition Main Control

The visual recognition system is powered by a Thor MIX 1362H000 mini PC equipped with an Intel i7-13620H CPU. This high-performance computing unit is dedicated to processing the visual data captured by the cameras. It analyzes the information to adjust the fine angles of the robotic arm's end effector. The use of advanced image processing algorithms and the powerful computational capabilities of the i7-13620H CPU ensure that the system can recognize and interpret visual data with high precision, facilitating sophisticated manipulation and interaction with the environment.

Cameras

For capturing high-definition images, we utilize the Logitech C270i HD webcam. This camera is selected for its ability to deliver clear 720p high-definition images, essential for precise position recognition. The high-quality visual input from the Logitech C270i is vital for the visual recognition system, enabling accurate and responsive control of the robotic arm based on visual cues. This capability is fundamental for tasks that require a high degree of precision and adaptability, further enhancing the system's utility and performance.

These components collectively form a sophisticated system designed to achieve precise control and high adaptability, meeting the demands of various applications and environments.

4.2 Schedule

4.2.1 Project Milestones

Movement Reflection

The goal of "Movement Reflection" in our project is to ensure a high degree of fidelity between the movement of the input handle and the corresponding action of the robotic arm. This encompasses several key aspects:

Responsive Movement: The robotic arm must accurately follow the movements of the handle. When the handle is moved, the arm should replicate this motion without unnecessary delay. Conversely, when

the handle is stationary, the arm should maintain its position with sufficient damping to prevent unintended movements. This requires sophisticated sensing and control systems to detect and respond to handle movements promptly.

Directional Accuracy: The robotic arm's movement must not only be synchronized with the handle's movement but also accurately reflect its direction. For instance, if the handle is moved to the left, the robotic arm should also move to the left. Achieving this involves complex algorithms that interpret the handle's movements and translate them into precise motor actions in the robotic arm, ensuring that the movement's direction is consistently mirrored.

Mapping Accuracy

"Mapping Accuracy" refers to the precise and proportional translation of movements from the input handle to the robotic arm. This involves scaling the movement in a way that maintains a consistent ratio:

Proportional Scaling: If the handle moves by a certain distance (e.g., 1cm), the robotic arm should move by a proportional distance (e.g., 1cm * k), where k is a scaling factor. It's crucial that this scaling factor is kept within a tight range to ensure the movement's scale is accurately replicated, which might require calibration to accommodate different operation modes or payloads.

Minimized Delay

Reducing latency in the system to a minimum is critical for ensuring that movements of the handle are reflected by the robotic arm in real-time. This involves:

Optimization Strategies: Implementing algorithmic optimizations, adjusting parameters for efficiency, changing communication protocols to more efficient ones, and distributing computational load more effectively across the system components. The aim is to achieve a seamless interaction where the delay between input (handle movement) and output (robotic arm movement) is imperceptible to the user.

Auxiliary Controls

Integrating auxiliary controls enhances the system's functionality and user control precision:

Force Feedback: Incorporating force feedback mechanisms, possibly informed by visual recognition systems, to provide the user with tactile feedback regarding the environment the robotic arm is interacting with. This can aid in fine-tuning control and enhancing the realism and safety of remote operations.

Calibration Mechanisms: Given that position tracking might rely on the integration of acceleration sensor data, which can accumulate error over time, implementing periodic calibration mechanisms is essential. These mechanisms would adjust and refine the system's understanding of the robotic arm's

position and movements, ensuring ongoing accuracy and reliability in movement reflection and mapping.

Each of these milestones is designed to create a responsive, accurate, and intuitive control system for the robotic arm, enhancing its applicability across a wide range of tasks requiring precision and real-time control.

4.2.2 Time-table

Week of 3.25: Finalize the Foundational Action Mapping

During this week, our primary focus will be on establishing a robust foundational action mapping between the input handle and the robotic arm. This involves implementing sophisticated sensing and control systems that enable the robotic arm to accurately follow the movements of the handle with high responsiveness. Key activities will include:

Development and Integration: Finalizing the integration of sensors on the handle to detect movements accurately and translating these movements into commands for the robotic arm. This will ensure that actions performed with the handle, such as tilting, rotating, or shifting, are immediately and accurately reflected in the robotic arm's movements.

Initial Testing and Feedback Loop: Conducting initial tests to assess the responsiveness of the robotic arm to handle movements. This will involve simple directional tests to ensure that when the handle moves in a specific direction, the robotic arm follows suit without any unintended movements or delays.

Damping Mechanisms: Implementing damping mechanisms to prevent the robotic arm from making unintended movements when the handle is stationary. This requires careful calibration of the system's response to ensure stability and accuracy in reflecting the operator's intended movements.

Week of 4.1: Address and Rectify Any Emergent Bugs Affecting Handle Movement

This week is dedicated to identifying and rectifying any emergent bugs that affect the accuracy and responsiveness of handle movements, with a focus on:

Bug Detection and Analysis: Systematically identifying bugs that were introduced in the foundational action mapping phase, especially those affecting directional accuracy and responsiveness.

Corrective Actions: Implementing fixes for detected bugs, ensuring that the handle's movements are accurately and consistently mirrored by the robotic arm. This includes adjusting algorithms and control

systems to prevent any discrepancies, such as the robotic arm moving in the opposite direction of the handle or moving when it should remain stationary.

Enhanced Testing for Directional Accuracy: Conducting more rigorous testing to validate the fixes, focusing on directional accuracy to ensure that the system accurately translates the handle's movements into appropriate actions by the robotic arm.

Week of 4.8: Refine the Action Mapping for Greater Precision

The objective for this week is to enhance the precision of the action mapping between the handle and the robotic arm through proportional scaling and minimized delay, by:

Proportional Scaling Implementation: Fine-tuning the scaling factor (k) to ensure that movements of the handle are translated to the robotic arm in a consistent and proportional manner. This includes adjustments for different operation modes or payloads that the arm might encounter.

Latency Reduction Strategies: Implementing optimization strategies to reduce system latency, ensuring real-time reflection of the handle's movements by the robotic arm. This involves algorithmic optimizations and possibly adjusting communication protocols for efficiency.

Precision Testing: Conducting tests to assess the accuracy of proportional scaling and the effectiveness of latency reduction strategies, making further adjustments as needed to achieve high mapping accuracy and minimized delay.

Week of 4.15: Focus on Optimizing Overall System Performance

This week will be centered around optimizing the system's overall performance and stability, incorporating auxiliary controls for enhanced functionality:

System Optimization: Conducting comprehensive tests to identify any remaining inefficiencies in the system and implementing optimization strategies to improve overall performance and stability.

Force Feedback Integration: Integrating force feedback mechanisms to provide tactile feedback to the user, enhancing control precision and safety in remote operations.

Calibration Mechanisms: Implementing and testing calibration mechanisms to adjust and refine the system's understanding of the robotic arm's position and movements, ensuring ongoing accuracy and reliability.

Subsequent Weeks: To be Determined, Allowing Flexibility for Additional Refinements and Testing

In the following weeks, the project will remain flexible to accommodate additional refinements and testing, focusing on:

Continuous Improvement: Based on user feedback and ongoing testing, making iterative improvements to the system, focusing on enhancing responsiveness, accuracy, and user experience.

Additional Features: Exploring the integration of auxiliary controls and additional features that could further enhance the system's functionality, such as advanced visual recognition systems for improved environmental interaction.

Final Testing and Documentation: Conducting final comprehensive testing of the entire system to ensure all components work seamlessly together. Finalizing project documentation, including user manuals and technical guides, to facilitate deployment and usage.

4.2.3 Task Allocation

Our team's collaborative effort is strategically divided to leverage each member's expertise:

- **Programming and Testing Microcontrollers**: Xu's responsibility, focusing on the intricacies of control input handling.
- **Constructing the Robotic Arm**: Li's domain, where mechanical ingenuity comes to life.
- **Control Optimization**: Wang's forte, enhancing the robotic arm's responsiveness and efficiency.
- **Computer Vision Implementation**: Zeng's realm, integrating advanced recognition capabilities for precise object manipulation.

5. Ethics and Safety

5.1 Ethical Considerations

The development and deployment of our robotic arm system, designed to replicate human hand movements and operate in hazardous environments, raise significant ethical considerations. In alignment with the IEEE Code of Ethics and ACM Code of Ethics, our project commits to prioritizing the safety, health, and welfare of the public, ensuring that our technology enhances the quality of life, fairness, and dignity of all stakeholders involved.

Privacy and Surveillance: Given the system's reliance on camera technology to capture human movements, there is a potential risk of privacy invasion. To mitigate this, our project will implement

strict data handling protocols, ensuring that all captured data are anonymized and securely stored, with access restricted to authorized personnel only.

Autonomy and Misuse: The potential for the robotic arm to be repurposed for unintended or harmful activities, such as surveillance or in military applications, necessitates a robust ethical framework. We will incorporate failsafe mechanisms and operational limits to prevent misuse, ensuring the system's use remains within the scope of humanitarian and industrial assistance.

Accessibility and Inclusivity: Aligning with the principles of fairness and avoiding discrimination, our project aims to ensure that the robotic arm system is accessible and adaptable to a wide range of users, regardless of their physical abilities. This commitment extends to providing equitable access to the benefits of our technology, fostering inclusivity in its application.

5.2 Safety Concerns and Regulatory Compliance

Our project's design and operational framework will be developed in strict compliance with relevant safety and regulatory standards to mitigate potential risks associated with its functionality and deployment in various environments.

Operational Safety: The robotic arm will be engineered with built-in safety features, including emergency stop functions, collision avoidance systems, and adaptive response mechanisms to unexpected obstacles or failures. This approach ensures the protection of both the operator and bystanders during its operation.

Regulatory Standards: Compliance with state and federal regulations, industry standards, and campus policies will be rigorously pursued. This includes adhering to the Occupational Safety and Health Administration (OSHA) guidelines for robotic equipment and the American National Standards Institute (ANSI) safety standards for industrial robots and robot systems.

Risk of Injury: To address the potential for injury from mechanical failure or operational error, our system will undergo extensive testing and validation to meet high reliability and safety metrics. Training protocols will be developed to educate operators on safe handling and operation procedures, minimizing the risk of accidents.

Environmental Impact: Consideration will be given to the environmental impact of our system, ensuring that materials and processes used in the construction and operation of the robotic arm are sustainable and minimize ecological footprints. This includes evaluating the lifecycle of the system and implementing recycling or disposal protocols for end-of-life units.

5.3 Conclusion

In conclusion, our project team is committed to upholding the highest ethical standards and safety protocols, ensuring that the development and deployment of our robotic arm system contribute positively to society. By adhering to the IEEE and ACM Code of Ethics, incorporating safety and regulatory standards, and addressing potential ethical and safety issues proactively, we aim to develop a technology that is not only innovative but also responsible and beneficial to humanity.

6. Citation

7. Appendix

```
#include "imu.h"
#include "spi.h"
#include "tim.h"
#include <math.h>
#define PI (3.1415927F)
/*经验误差*/
const float YAW_GYRO_COMPENSATION = -0.29;
const float X_ACCEL_COMPENSATION = +0.01124;
const float Y_ACCEL_COMPENSATION = -0.00063;
const float _pi_over_180_ = PI/180.0f;

/* ------------------------------  Variables  ------------------------------ */
static imu_raw_data_t imu_raw_data = {0};  // IMU 原始数据结构体
IMU_Type imu = {0,0};

/* ------------------------------ Fuction Declaration --------------------------
--- */
void IMU_Get_Data(void);
float invSqrt(float x);
void MahonyAHRSupdateIMU(float q[4], float gx, float gy, float gz, float ax,
float ay, float az);
//陀螺仪
void BMI088_Write_Gyro_Single_Reg(uint8_t const reg, uint8_t const data);
uint8_t BMI088_Read_Gyro_Single_Reg(uint8_t const reg);
void BMI088_Read_Gyro_Multi_Reg(uint8_t *bmi_rx_data);
uint8_t BMI088_Gyro_Init(void);
//加速度计
void BMI088_Write_Acc_Single_Reg(uint8_t const reg, uint8_t const data);
uint8_t BMI088_Read_Acc_Single_Reg(uint8_t const reg);
void BMI088_Read_Acc_Multi_Reg(uint8_t *bmi_rx_data);
uint8_t BMI088_Acc_Init(void);
//微秒延时
void bmi_delay_us(uint16_t us);

uint8_t BMI088_Init(void)
{
    uint8_t state = BMI088_NO_ERROR;
```

```
    state |= BMI088_Gyro_Init();
    state |= BMI088_Acc_Init();
    if(state==BMI088_NO_ERROR){
        HAL_TIM_Base_Start_IT(&htim5);//1ms 一次中断用于数据处理
    }else{
        HAL_GPIO_WritePin(Red_GPIO_Port,Red_Pin,GPIO_PIN_SET);
    }
    return state;
}


void IMU_Get_Data(void)
{
    uint8_t gyro_buff[6];
    uint8_t acc_buff[6];
    float tmp;
    /* 陀螺仪数据读取 */
    BMI088_Read_Gyro_Multi_Reg(gyro_buff);
    tmp = (int16_t)((gyro_buff[1] << 8) | gyro_buff[0]);
    imu_raw_data.gx = (tmp / GYRO_SENSITIVITY_1000) * _pi_over_180_ ; //统一采用
弧度制
    tmp = (int16_t)((gyro_buff[3] << 8) | gyro_buff[2]);
    imu_raw_data.gy = (tmp / GYRO_SENSITIVITY_1000) * _pi_over_180_ ;
    tmp = (int16_t)((gyro_buff[5] << 8) | gyro_buff[4]);
    imu_raw_data.gz = (tmp / GYRO_SENSITIVITY_1000 + YAW_GYRO_COMPENSATION) *
_pi_over_180_ ; //误差补偿
    /* 加速度计数据读取 */
    BMI088_Read_Acc_Multi_Reg(acc_buff);
    tmp = (int16_t)((acc_buff[1] << 8) | acc_buff[0]);
    imu_raw_data.ax = (tmp / ACCEL_SENSITIVITY_3)   +   X_ACCEL_COMPENSATION; //
误差补偿
    tmp = (int16_t)((acc_buff[3] << 8) | acc_buff[2]);
    imu_raw_data.ay = (tmp / ACCEL_SENSITIVITY_3)   +   Y_ACCEL_COMPENSATION; //
误差补偿
    tmp = (int16_t)((acc_buff[5] << 8) | acc_buff[4]);
    imu_raw_data.az = (tmp / ACCEL_SENSITIVITY_3);
}


void BMI088_Write_Gyro_Single_Reg(uint8_t const reg, uint8_t const data)
{
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_RESET);  // 拉低
片选信号，开始传输 NSS_Low
```

```
    //数据交换
    bmi_tx_byte = reg & 0x7f;//首位是 0，表示此操作为写入，reg 后七位即为所写地址
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    bmi_tx_byte = data;
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    //结束
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_SET);  // 拉高片
选信号，结束传输 NSS_High
}

uint8_t BMI088_Read_Gyro_Single_Reg(uint8_t const reg)
{
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_RESET);  // 拉低
片选信号，开始传输 NSS_Low
    //数据交换
    bmi_tx_byte = reg | 0x80;  // 第一位为 1 表示读操作
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    //结束
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_SET);  // 拉高片
选信号，结束传输 NSS_High
    return bmi_rx_byte;
}

void BMI088_Read_Gyro_Multi_Reg(uint8_t *bmi_rx_data)
{
    uint8_t bmi_rx_byte, bmi_tx_byte, len = 6;
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_RESET);  // 拉低
片选信号，开始传输 NSS_Low
    bmi_tx_byte = BMI088_GYRO_X_L | 0x80;  // 第一位为 1 表示读操作
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    while (len != 0) {
        HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_data[6 - len], 1,
BMI088_SPI_TIME_THRES);
        len--;
    }
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_SET);  // 拉高片
选信号，结束传输 NSS_High
```

```
}

uint8_t BMI088_Gyro_Init(void)
{
    uint8_t BMI088_Gyro_Init_Config[3][3] = {
        {BMI088_GYRO_RANGE, BMI088_GYRO_1000, BMI088_GYRO_RANGE_ERROR},
        {BMI088_GYRO_BANDWIDTH, BMI088_GYRO_1000_116_HZ |
BMI088_GYRO_BANDWIDTH_MUST_Set, BMI088_GYRO_BANDWIDTH_ERROR},
        {BMI088_GYRO_LPM1, BMI088_GYRO_NORMAL_MODE, BMI088_GYRO_LPM1_ERROR}
    };
    static uint8_t read_value;
    BMI088_Write_Gyro_Single_Reg(BMI088_GYRO_SOFTRESET,
BMI088_GYRO_SOFTRESET_VALUE);
    HAL_Delay(BMI088_LONG_DELAY_TIME);
    // check commiunication is normal after reset
    read_value = BMI088_Read_Gyro_Single_Reg(BMI088_GYRO_CHIP_ID);
    bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
    // check the "who am I"
    if (read_value != BMI088_GYRO_CHIP_ID_VALUE) {
        return BMI088_NO_SENSOR;
    }
    // 根据 BMI088_Gyro_Init_Config 的配置，写入相应寄存器
    for (uint8_t i = 0; i < 3; i++) {
        BMI088_Write_Gyro_Single_Reg(BMI088_Gyro_Init_Config[i][0],
BMI088_Gyro_Init_Config[i][1]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        read_value = BMI088_Read_Gyro_Single_Reg(BMI088_Gyro_Init_Config[i][0]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        if (read_value != BMI088_Gyro_Init_Config[i][1]) {
        return BMI088_Gyro_Init_Config[i][2];  // 若有错误立即返回，不会进行后续配置
        }
    }
    return BMI088_NO_ERROR;
}

void BMI088_Write_Acc_Single_Reg(uint8_t const reg, uint8_t const data)
{
    // TODO(Hello World):
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_RESET);  // 拉
低片选信号，开始传输
    //数据交换
    bmi_tx_byte = reg & 0x7f;
```

```
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    bmi_tx_byte = data;
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    //结束
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_SET);  // 拉高
片选信号，结束传输
}


uint8_t BMI088_Read_Acc_Single_Reg(uint8_t const reg)
{
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_RESET);  // 拉
低片选信号，开始传输
    //数据交换
    bmi_tx_byte = reg | 0x80;  // 第一位为 1 表示读操作
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);//bit0 读，bit1-7 确定读取地址
    bmi_tx_byte = 0x55; //芝士 magic number，反正官方例程是这么写的
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);//bit8-15 无效值
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);//bit16-23 以及之后为有效值
    //结束
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_SET);  // 拉高
片选信号，结束传输
    return bmi_rx_byte;
}


void BMI088_Read_Acc_Multi_Reg(uint8_t *bmi_rx_data)
{
    uint8_t bmi_rx_byte, bmi_tx_byte, len = 6;
    //开始
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_RESET);  // 拉
低片选信号，开始传输
    //数据交换
    bmi_tx_byte = BMI088_ACCEL_XOUT_L | 0x80;  // 第一位为 1 表示读操作，寄存器地址
为加速度计 0x12，x 轴方向低八位
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
```

```
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);//额外的读取操作，筛选掉无效读数
    while (len != 0) {
        HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_data[6 - len], 1,
BMI088_SPI_TIME_THRES);
        len--;
    }
    //结束
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_SET);  // 拉高
片选信号，结束传输
}


uint8_t BMI088_Acc_Init(void)
{
    /* 加速度计配置 */
    // 配置一些寄存器，并定义对应的错误
    uint8_t BMI088_Acc_Init_Config[4][3] = {
        {BMI088_ACC_RANGE, BMI088_ACC_RANGE_3G, BMI088_ACC_RANGE_ERROR},
        {BMI088_ACC_CONF, BMI088_ACC_800_HZ | BMI088_ACC_CONF_MUST_Set,
BMI088_ACC_CONF_ERROR},
        {BMI088_ACC_PWR_CTRL, BMI088_ACC_ENABLE_ACC_ON,
BMI088_ACC_PWR_CTRL_ERROR},
        {BMI088_ACC_PWR_CONF, BMI088_ACC_PWR_ACTIVE_MODE,
BMI088_ACC_PWR_CONF_ERROR}
    };
    static uint8_t read_value;  // 写入上述寄存器后再读取的值，用来检查是否正确配置
    //软件复位，复位后必须开启，byd 不然用不了
    BMI088_Write_Acc_Single_Reg(BMI088_ACC_SOFTRESET,
BMI088_ACC_SOFTRESET_VALUE);
    HAL_Delay(BMI088_LONG_DELAY_TIME);
    BMI088_Write_Acc_Single_Reg(BMI088_ACC_PWR_CONF,BMI088_ACC_PWR_ACTIVE_MODE);
    bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
    // check commiunication is normal after reset
    read_value = BMI088_Read_Acc_Single_Reg(BMI088_ACC_CHIP_ID);
    bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
    // check the "who am I"
    if (read_value != BMI088_ACC_CHIP_ID_VALUE) {
        return BMI088_NO_SENSOR;//**?
    }
    // 写入相应寄存器
    for (uint8_t i = 0; i < 4; i++) {
        BMI088_Write_Acc_Single_Reg(BMI088_Acc_Init_Config[i][0],
BMI088_Acc_Init_Config[i][1]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
```

```
        read_value = BMI088_Read_Acc_Single_Reg(BMI088_Acc_Init_Config[i][0]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        if (read_value != BMI088_Acc_Init_Config[i][1]) {
        return BMI088_Acc_Init_Config[i][2];  // 若有错误立即返回，不会进行后续配置
        }
    }
    return BMI088_NO_ERROR;
}

void bmi_delay_us(uint16_t us)
{
    uint32_t ticks = 0;
    uint32_t told = 0;
    uint32_t tnow = 0;
    uint32_t tcnt = 0;
    uint32_t reload = 0;

    reload = SysTick->LOAD;
    ticks = us * 168;
    told = SysTick->VAL;

    while (1) {
        tnow = SysTick->VAL;
        if (tnow != told) {
        if (tnow < told) {
            tcnt += told - tnow;
        } else {
            tcnt += reload - tnow + told;
        }
        told = tnow;
        if (tcnt >= ticks) {
            break;
        }
        }
    }
}

/* ---------------------------- 姿态解算 ---------------------------- */
//变量
const float sampleFreq = 1000.0f;       // sample frequency in Hz
const float twoKp = (2.0f * 0.3f);  // 2 * proportional gain (Kp)
const float twoKi = (2.0f * 0.0f);  // 2 * integral gain (Ki)
volatile float integralFBx = 0.0f, integralFBy = 0.0f, integralFBz = 0.0f;  //
integral error terms scaled by Ki
```

```
float invSqrt(float x)
{
    float halfx = 0.5f * x;
    float y = x;
    long i = *(long *)&y;
    i = 0x5f3759df - (i >> 1);
    y = *(float *)&i;
    y = y * (1.5f - (halfx * y * y));
    return y;
}

void MahonyAHRSupdateIMU(float q[4], float gx, float gy, float gz, float ax,
float ay, float az)
{
    float recipNorm;
    float halfvx, halfvy, halfvz;
    float halfex, halfey, halfez;
    float qa, qb, qc;

    // Compute feedback only if accelerometer measurement valid (avoids NaN in
accelerometer normalisation)
    if (!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {
        // Normalise accelerometer measurement
        recipNorm = invSqrt(ax * ax + ay * ay + az * az);
        ax *= recipNorm;
        ay *= recipNorm;
        az *= recipNorm;

        // Estimated direction of gravity and vector perpendicular to magnetic
flux
        halfvx = q[1] * q[3] - q[0] * q[2];
        halfvy = q[0] * q[1] + q[2] * q[3];
        halfvz = q[0] * q[0] - 0.5f + q[3] * q[3];

        // Error is sum of cross product between estimated and measured direction
of gravity
        halfex = (ay * halfvz - az * halfvy);
        halfey = (az * halfvx - ax * halfvz);
        halfez = (ax * halfvy - ay * halfvx);

        // Compute and apply integral feedback if enabled
        if (twoKi > 0.0f) {
        integralFBx += twoKi * halfex * (1.0f / sampleFreq);  // integral error
scaled by Ki
        integralFBy += twoKi * halfey * (1.0f / sampleFreq);
```

```
        integralFBz += twoKi * halfez * (1.0f / sampleFreq);
        gx += integralFBx;  // apply integral feedback
        gy += integralFBy;
        gz += integralFBz;
        } else {
        integralFBx = 0.0f;  // prevent integral windup
        integralFBy = 0.0f;
        integralFBz = 0.0f;
        }

        // Apply proportional feedback
        gx += twoKp * halfex;
        gy += twoKp * halfey;
        gz += twoKp * halfez;
    }

    // Integrate rate of change of quaternion
    gx *= (0.5f * (1.0f / sampleFreq));  // pre-multiply common factors
    gy *= (0.5f * (1.0f / sampleFreq));
    gz *= (0.5f * (1.0f / sampleFreq));
    qa = q[0];
    qb = q[1];
    qc = q[2];
    q[0] += (-qb * gx - qc * gy - q[3] * gz);
    q[1] += (qa * gx + qc * gz - q[3] * gy);
    q[2] += (qa * gy - qb * gz + q[3] * gx);
    q[3] += (qa * gz + qb * gy - qc * gx);

    // Normalise quaternion
    recipNorm = invSqrt(q[0] * q[0] + q[1] * q[1] + q[2] * q[2] + q[3] * q[3]);
    q[0] *= recipNorm;
    q[1] *= recipNorm;
    q[2] *= recipNorm;
    q[3] *= recipNorm;
}

void Get_IMU_Yaw(void)
{
    static float q[4]={1.0f , 0.0f , 0.0f , 0.0f};  // 四元数
    static float Old_Yaw = 0;
    static int32_t Circle = 0;
    float newYaw;
    IMU_Get_Data();
    /* 由于板子是躺着放的，所以必须对部分数据取反 */
    MahonyAHRSupdateIMU(q,
```

```
    imu_raw_data.gx,
    -imu_raw_data.gy,
    -imu_raw_data.gz,
    imu_raw_data.ax,
    -imu_raw_data.ay,
    -imu_raw_data.az);
    newYaw = atan2f(2.0f * (q[0] * q[3] + q[1] * q[2]), 2.0f * (q[0] * q[0] +
q[1] * q[1]) - 1.0f);
    //*pitch = asinf(-2.0f * (q[1] * q[3] - q[0] * q[2])); //暂时用不到 pitch 角度
    if(newYaw-Old_Yaw<-PI){Circle+=1;}
    if(newYaw-Old_Yaw> PI){Circle-=1;}
    Old_Yaw = newYaw;//更新旧角度
    //累计的弧度广播到全局
    imu.Yaw_Angle = 2.0f*PI*Circle + newYaw;
    //云台角速度广播到全局，单位：rad/s
    imu.Yaw_Velocity = -imu_raw_data.gz; //板子躺着放，取反。
}
```