



國立中央大學
National Central University

Capstone Project: Development of a Frisbee-Launching Mobile Robot

Department of Mechanical Engineering, Systems Division

Advisor: Prof. Tsai, Shyi-Jeng

Author: Chao Po-Ju Hsu Tang-Wei Lo-Ping

Date: 2025.01.08

I. Abstract

1. Project Concept

This project was inspired by the rules of the Tokyo Tech Robot Contest 2024. Based on the organizer's constraints regarding robot dimensions and frisbee specifications, a launching mechanism was designed and integrated into a mobile robot. The robot was developed to take advantage of the frisbee's aerodynamic characteristics, with the objective of enabling remote-controlled stable operation and continuous frisbee launches into designated target hoops. After the competition, the system was further reviewed, refined, and optimized for improved performance.

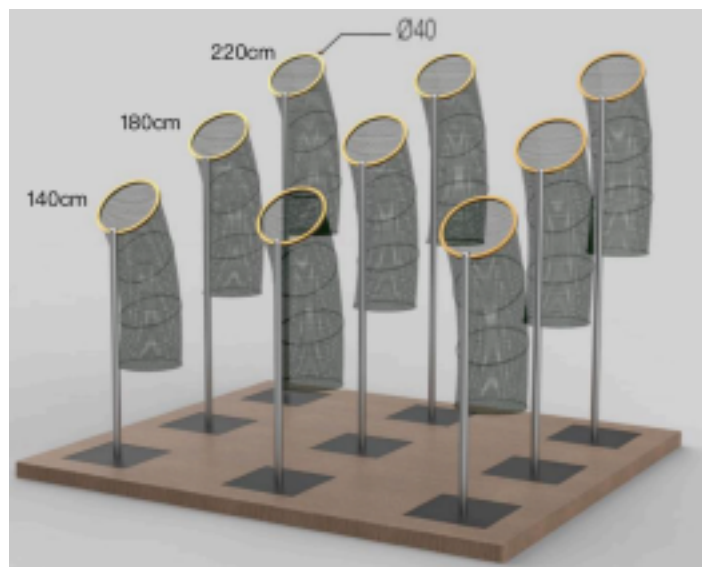


Figure 1. Competition Field

As illustrated in Figure 1, the competition field featured three target hoops at heights of 140 cm, 180 cm, and 220 cm, each with a diameter of 40 cm and spaced one meter apart in both horizontal and vertical directions. The frisbees provided by the organizer had a diameter of 27.5 cm, while the robot was constrained to a maximum size of 70 cm in both length and width, 90 cm in height, and a total weight of 60 kg.

2. Project Goal

Pre-Competition Expected Objectives	<ul style="list-style-type: none">• Omnidirectional movement• Remote-controlled launching• Automatic reloading and feeding• Ability to shoot frisbees into hoops at all target heights
Objectives Achieved Before the Competition	<ul style="list-style-type: none">• Continuous frisbee launching with stable entry into all target hoops
Post-Competition Review and Optimization	<ul style="list-style-type: none">• Control system• Vision-based recognition

II. Research Methodology and Design

1. Mechanical Design

a. Machine Specifications

- Dimensions (L × W × H): **70 × 70 × 80 cm**
- Weight: **45 kg**
- Supported frisbee diameter: **27.5 cm**
- Maximum frisbee capacity: **20 discs**

b. Mobile Chassis

The chassis was constructed using aluminum extrusion as the structural frame, equipped with Mecanum wheels to achieve omnidirectional movement. Bearing housings were added between the motor shafts and wheels to withstand the transmission of forces. Encoders were mounted on the outer side of the wheels to enable closed-loop control.

The driving motors used were **DC motor 07SGN-12V-1800R** paired with **4GN-3K gear reducers**.

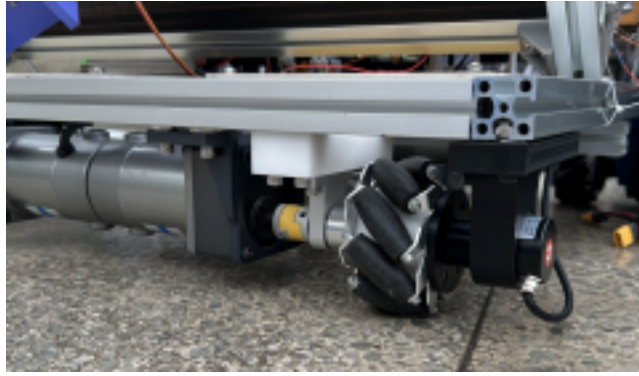


Figure 2. Physical Chassis

c. Launching Mechanism

To ensure stable flight, the frisbee must rotate while being launched. Therefore, the launching track was designed to provide both linear propulsion and rotational motion. Once the frisbee enters the track, it is pressed between the central roller and the track's sidewalls, generating friction and forward thrust. Rubber strips were attached to both the roller and the track to enhance friction. The roller drives the frisbee into rotation until it exits the track.

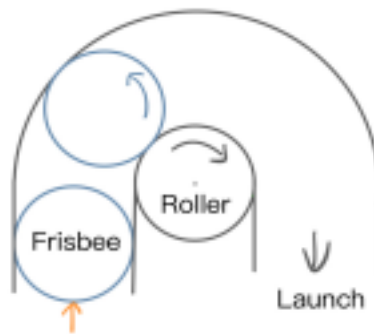


Figure 3. Launching Concept Diagram

The launching velocity was calculated using the planetary gear ratio formula. In this model, the roller acts as the sun gear, the outer track as the ring gear, and the frisbee as the planet gear. Due to robot size limitations, the actual track angle was set at 100° . After multiple tests, the selected roller diameter was **10 cm**, track width **26.5 cm**, rubber strip thickness **2 mm**, and the launching assembly was fixed at an elevation angle of **45°** .

The launching structure was assembled from aluminum extrusions and wooden boards (polyboard and MDF). The roller was powered by a **CIM motor**, while a 3D-printed feeder connected to a **TD-8120MG servo motor** guided frisbees into the track.



Figure 4. Physical Launching Mechanism

d. Storage and Reloading Mechanism

The frisbee storage design adopted a **dual-cylinder system**. The first cylinder was located at the track entrance, and the second was mounted above the track beneath a protective wooden cover. After the first set of frisbees was launched, a **V-shaped 3D-printed component** (Figure 4) swept frisbees from the second cylinder into the first.

To prevent misalignment or collapse during transfer, the second cylinder was wrapped in a cardboard sleeve with Velcro strips, which adhered to Velcro attached to the aluminum extrusion of the first cylinder. This provided sufficient lateral constraint while ensuring that frisbees could still be properly guided into the track.

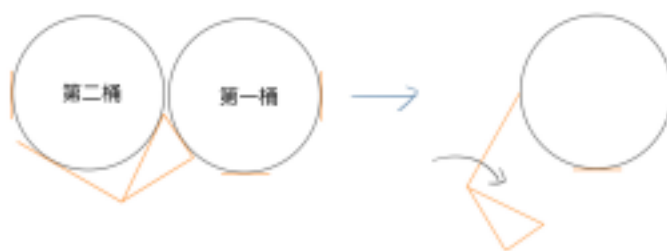


Figure 5. Top View of Frisbee Storage

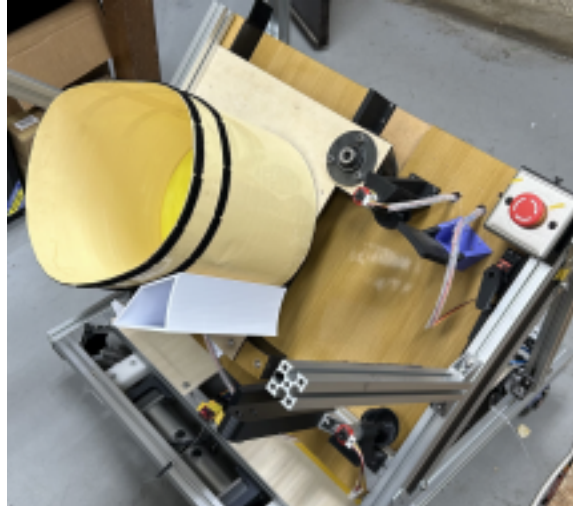


Figure 6. Physical Reloading Mechanism with Cardboard Sleeve

Due to the unique geometry of the frisbee, precise positional constraints were required to prevent jamming during reloading. Multiple 3D-printed components were added around the discs to maintain alignment. However, because the track entrance required free space, excessive constraints caused frequent jamming. To resolve this, **two helical 3D-printed screws** were added to feed the frisbees downward. Each screw had a lead equal to one frisbee thickness, holding a single frisbee in place and lowering it smoothly into the track without interference. The screw threads also supported the weight of the upper frisbees, preventing excessive downward force at the entrance.

The screw shafts were mounted vertically, so the motor shafts did not directly bear the frisbee load, significantly increasing storage capacity. Most parts were 3D-printed and fixed onto aluminum extrusion or wooden boards. The **V-shaped component** was driven by a **TD-8120MG servo motor**, while the **helical screw** was powered by a **JGA25-371 DC motor with encoder**, allowing precise 360° rotation to release one frisbee per cycle.

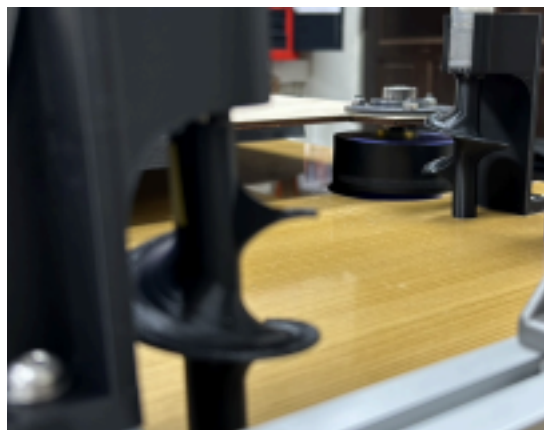


Figure 7. Helical 3D-Printed Component

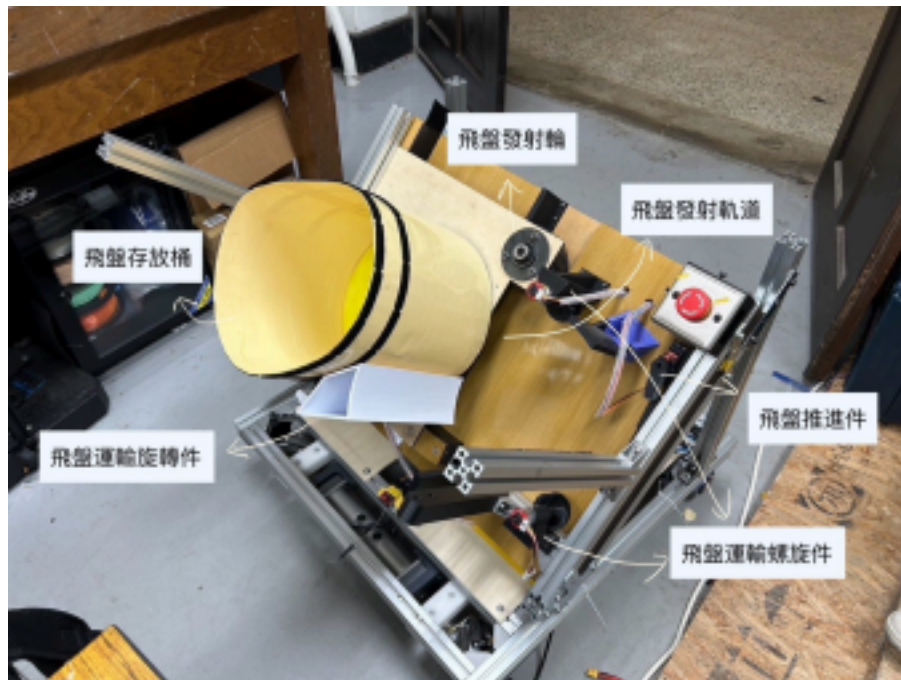


Figure 8. Overall Mechanical Assembly

2. System Structure

The system is designed to achieve automated operation and precise control of the frisbee robot. The overall architecture is illustrated in **Figure 8**. The system is divided into two main parts: the **Master Control Unit** and the **Robot Unit**. The Master Control Unit is responsible for high-level logic operations and data processing, while the Robot Unit executes specific operational tasks.

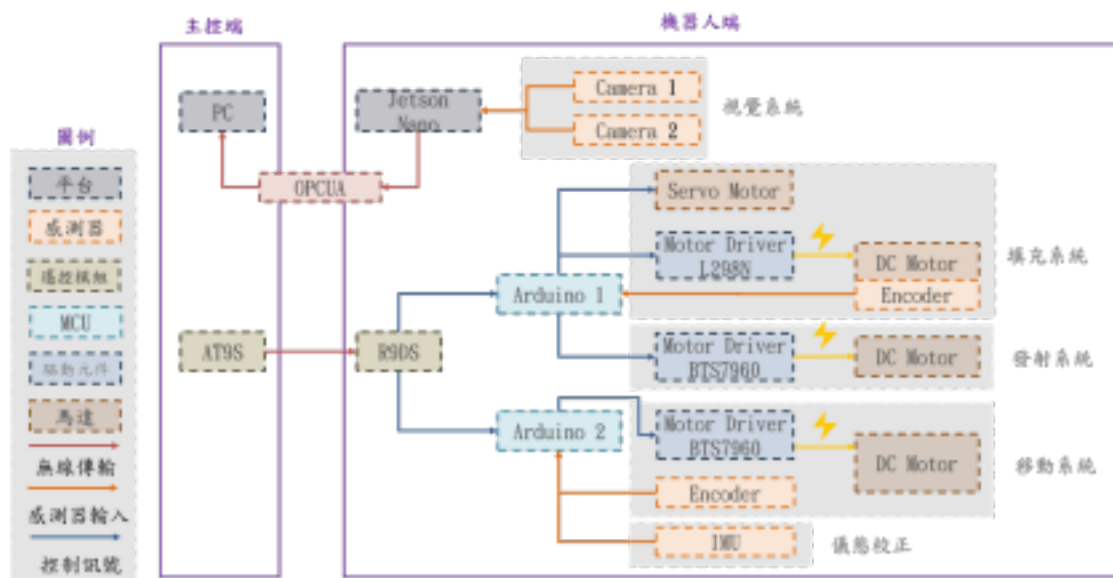


Figure 9. System Architecture

a. Master Control Unit

The Master Control Unit consists of a **PC** and a **remote controller**.

- The PC monitors the data transmitted from the Robot Unit.
- The remote controller issues control commands, including movement, launching, and reloading.

b. Robot Unit

The Robot Unit primarily includes the **Jetson Nano**, vision system, control modules, and actuators. These subsystems collaborate through signal exchange.

- **Jetson Nano**: Performs vision processing, including image recognition and deep learning inference, converting camera inputs into target coordinates.

Subsystems:

- **Vision System**:
Equipped with a dual-camera module for frisbee and hoop detection, as well as 3D coordinate measurement. These data provide the foundation for trajectory fitting, ensuring accurate launching.
- **Reloading System**:
Utilizes servo motors and DC motors in conjunction with an **L298N motor driver** to accomplish frisbee feeding and preparation. The DC motor is equipped with an encoder for real-time feedback, ensuring stability and precision during reloading.
- **Launching System**:
Employs the **BTS7960 motor driver module** to control the launching motor. By adjusting the PWM duty cycle, the system controls both launching distance and height.
- **Mobility System**:
Drives the chassis with Mecanum wheels, powered by BTS7960 drivers and encoders for speed and direction control. An IMU sensor provides orientation data, enabling correction and enhanced stability during movement.

c. Wireless Control and Data Exchange

The system uses **AT9S** and **R9DS** wireless modules for remote control, providing flexible manual operation. An **Arduino microcontroller (MCU)** serves as the central hub, converting wireless signals into executable commands and integrating sensor data for processing.

d. Overall Signal Flow

The data flow between the Master Control Unit and the Robot Unit is clearly defined. The PC and Jetson Nano handle high-level computation and transmit coordinate and monitoring data via **OPC UA**. The Arduino module receives commands to control drivers, while simultaneously collecting sensor feedback to achieve closed-loop control.

3. Communication System

a. Wireless Communication



Figure 10. Wireless Communication Architecture

For the communication system of the robot, we selected the **AT9S radio controller** instead of Wi-Fi. The main reasons are as follows:

- **Strong Anti-Interference Capability:**
Operates on a dedicated 2.4 GHz frequency band and employs **FHSS (Frequency Hopping Spread Spectrum)** technology, which significantly reduces interference from external devices or signals. In contrast, Wi-Fi is more prone to interference from nearby devices, resulting in less stability.
- **Extended Communication Range:**
Provides an effective range of **1–2 km**, making it suitable for long-distance control applications. Wi-Fi, by comparison, has a typical range of only 50–200 m, which is insufficient for such use cases.
- **Low Latency:**
Ensures minimal signal transmission delay, enabling real-time control responses. Wi-Fi suffers from higher latency, which can degrade performance in time-sensitive control scenarios.

Considering these factors, the AT9S radio controller offers superior stability, anti-interference capability, communication distance, and low latency, making it the optimal choice for the robot's wireless communication system.

b. RC Signal Processing

We experimented with two communication methods to establish data transmission between the **R9DS receiver** and the **MCU**:

- **PWM Mode**
 - **Concept:**
Reflects the remote controller status by parsing the high-level pulse width of the PWM signal sent from the R9DS to the Arduino.
 - **Processing:**
Uses external interrupts triggered on the rising edge to start timing, and stops timing on the falling edge.
 - **Characteristics:**
Simple to implement, but when multiple channels are used, it becomes highly time-consuming and significantly affects program execution.
- **SBUS Mode**
 - **Concept:**
SBUS transmits data in **25-byte frames**, each containing **16 standard channel signals** and **2 auxiliary status channels**.
 - **Processing:**
Uses **UART protocol** for data reception, where the MCU reads data directly via its serial port.
 - **Characteristics:**
Efficient transmission without interfering with MCU clock sources or other resources, requiring only **one communication line** for multi-channel data transfer.

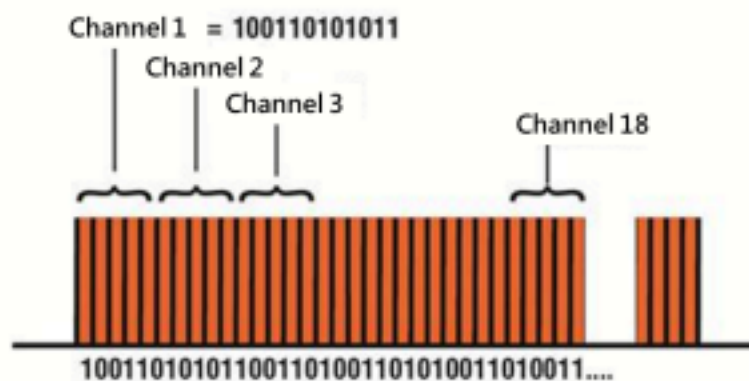


Figure 11. SBUS Concept Diagram

Based on these considerations, we selected **SBUS Mode** as the final communication method.

c. RC Signal Processing

Since SBUS Mode operates on inverted logic, the signal must be reversed before being processed by the Arduino. To achieve this, an **inverter circuit** was added.

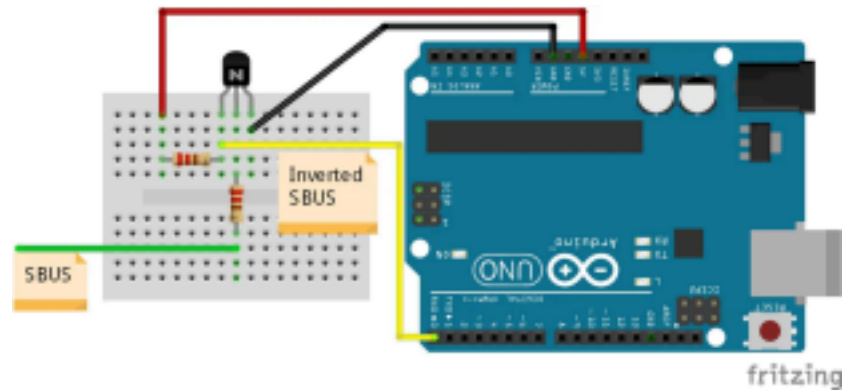


Figure 12. Inverter Circuit Diagram

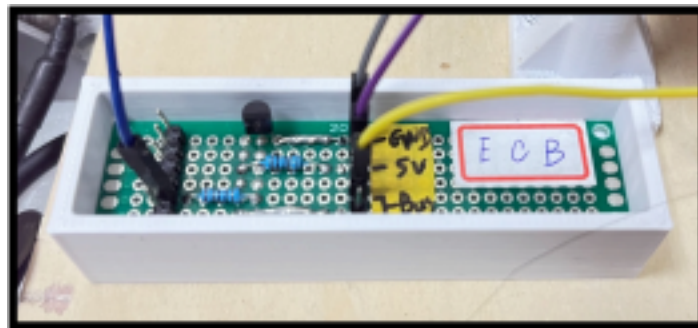


Figure 13. Self-Assembled Inverter Circuit

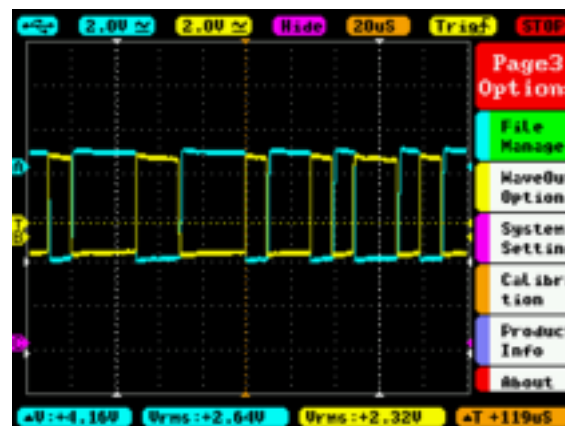


Figure 14. Oscilloscope Result of Inverted Signal

As shown in the figure, the yellow waveform represents the signal after inversion, while the blue waveform shows the original input signal.

4. Control System

a. System Overview

b. Chassis Motion Control

- **System Input:**
Error = Target Speed – Current Speed
- **Performance Evaluation:**

Rising Time	0.01 s
Settling Time	1.44 s
Overshoot	14.7%

- **Problems Encountered & Solutions:**
 - The main issue was **excessive steady-state error** during four-wheel speed control.
 - **First approach:** Increasing the integral gain (K_i) reduced steady-state error, but introduced strong oscillations at the target value. Even small errors caused excessive influence, leading to instability.
 - **Second approach:** Applying **feedforward control** successfully eliminated steady-state error. However, this did not fundamentally address the root cause, indicating that deeper analysis and improvement were still required.

$$\text{PWM_Output} = \text{feedForwardVal} + K_p \cdot e + K_i \cdot \int e \, dt + K_d \cdot \frac{de}{dt}$$

- **Optimization:**

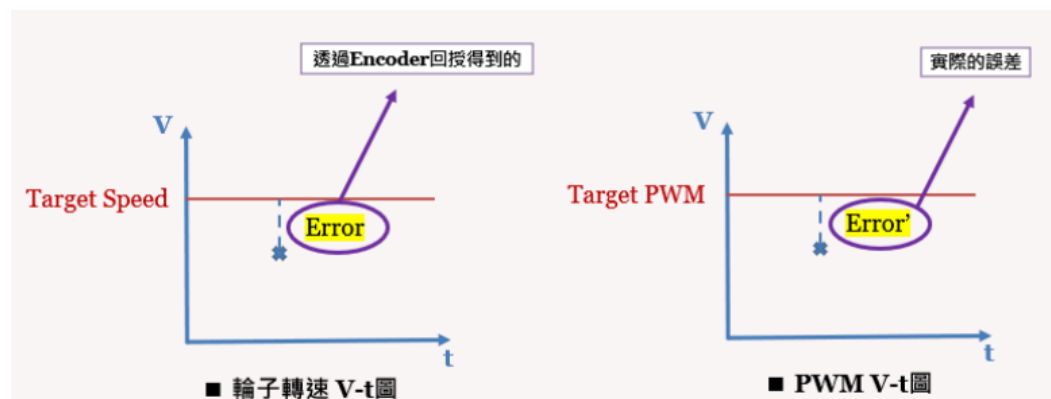


Figure 15. Speed Control Diagram


The control target is to compensate for the error between the desired speed and the encoder feedback. In practice, the motor is driven by the **PWM value**

output by the controller. Ideally, a fixed target speed should converge to a specific PWM value.

Therefore, the actual compensation should be based on the **PWM error**, not the speed error. The compensation method is straightforward: multiply the feedback error by a gain factor (**G**) to achieve error correction and improve control stability.

c. Helical Reloading Control

- **Motor Selection:**

Design Requirements	Motor Specifications
<ul style="list-style-type: none">→ Compact size→ Precise one-rotation control→ Continuous operation capability→ High torque requirement→ Chosen motor: DC gear motor with encoder	 <p>DC Gear Motor with Encoder</p>

- **System Input:**

$$\text{Error} = \text{Target Pulses} - \text{Current Pulses}$$

- **Performance Evaluation:**

- Speed: 55 RPM (1.09 s/rev) → Achieved 1.5 s/rev
- Error integration: Supported at least 30 consecutive reload cycles
- Motors operated in synchronized rotation
- Introduced a **Boundary Layer** method to reduce vibration

d. Orientation Correction

In this design, a **BNO055 IMU** was used to calibrate the robot's orientation, primarily using gyroscope data. To address common **drift issues** in gyroscope measurements, an effective correction method was implemented:

- During fixed time intervals in static conditions, the IMU's angular acceleration was observed and recorded as the **bias**.
- During operation, this bias was periodically subtracted from the measured values, effectively minimizing drift.

```
01:28:32.620 -> Corrected Gyro Z: -0.001042, Yaw Angle: -0.130416
01:28:32.751 -> Corrected Gyro Z: 0.000049, Yaw Angle: -0.130411
01:28:32.829 -> Corrected Gyro Z: 0.000049, Yaw Angle: -0.130406
01:28:32.953 -> Corrected Gyro Z: 0.000049, Yaw Angle: -0.130400
```

Figure 16. Gyroscope Drift after 20 Minutes

Tests showed excellent stability: after **20 minutes of continuous operation**, the drift was only **0.1°**, demonstrating the reliability and accuracy of the method.

e. Power Regulation

A **voltage detection module** was used to provide feedback for stable power supply to the DC motors. The system employed a **4S lithium battery** as the power source, regulated down to **12V** to ensure consistent supply.



Figure 17. Voltage Regulation Diagram

The voltage detection module continuously monitored the supply voltage to the motors. If the detected voltage dropped below the required level, the control system automatically increased the PWM value to compensate. This method not only ensured motor stability but also improved system efficiency and reliability.

5. Vision System

a. System Functions and Objectives

The vision system is a key subsystem of the frisbee robot, responsible for **object detection** and **3D localization**, providing the foundation for trajectory fitting and aiming mechanisms. Its main functions and objectives include:

- **Object Detection and Recognition:**

- Real-time detection and classification of frisbees and target hoops using deep learning (YOLOv5 model).
 - Ensuring high detection accuracy and robust performance under varying angles and distances.
- **3D Spatial Localization:**
 - Applying stereo vision techniques to compute depth information from binocular disparity.
 - Calculating relative 3D coordinates (X, Y, Z) of frisbees and hoops as inputs for trajectory modeling and aiming.
- **Real-Time Processing and Accuracy:**
 - Supporting fast processing to meet the real-time demands of high-speed frisbee motion.
 - Maintaining precision in detection and localization to prevent cumulative errors that may degrade overall performance.
- **Understanding Launch Characteristics:**
 - Using deep learning models trained on frisbee trajectory data to identify the optimal height range for successful target entry.

b. Technical Architecture and Implementation

The vision system integrates **stereo vision** and **deep learning**, combining hardware setup with software processing to achieve accurate 3D localization.

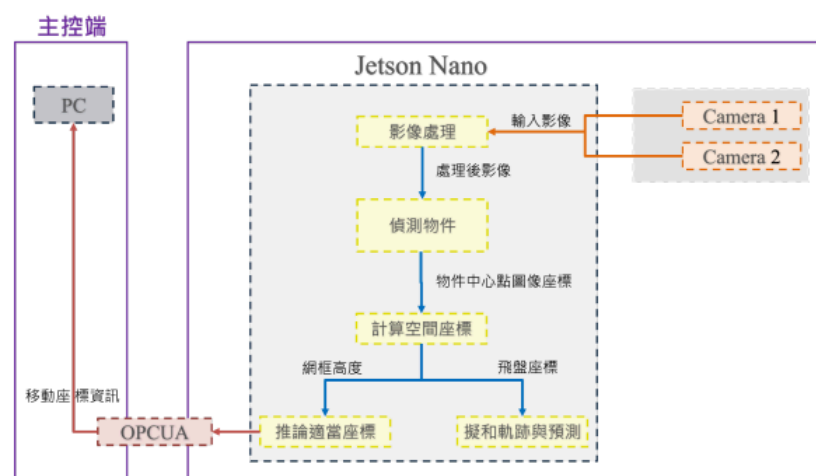


Figure 18. Vision System Architecture

- **Camera Hardware Setup:**

- **Stereo Camera Module:** Two cameras mounted on a horizontal aluminum extrusion frame with parallel optical axes and a fixed baseline of **52 cm**, ensuring stable and accurate disparity computation.
- **Stereo Calibration:** Intrinsic and extrinsic parameters calibrated using MATLAB Stereo Camera Calibrator App with chessboard patterns. Images were taken at distances from **80 cm to 250 cm**, and calibration results exported to OpenCV for rectification and disparity calculation.

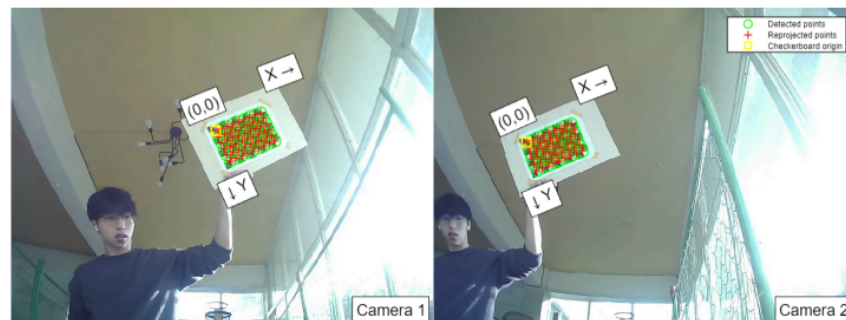


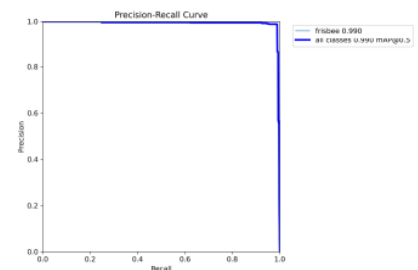
Figure 19. Stereo Calibration Result

- **Software Processing Pipeline:**

- **Object Detection:**

YOLOv5n (lightweight version) was selected to balance inference efficiency and accuracy. The dataset included annotated frisbee and hoop images, augmented with rotation, scaling, and brightness adjustments to improve generalization.

The trained model achieved a mean Average Precision (mAP) of **0.9**, with PR curve results confirming robustness.



- **Coordinate Computation:**

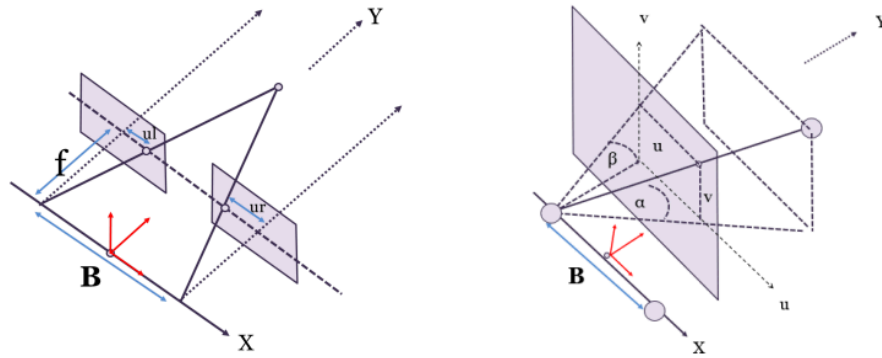


Figure 20. Stereo Disparity Calculation

Depth was calculated using binocular disparity:

$$Y = \frac{f \times B}{d}, \quad X = \frac{B}{2} \frac{\tan(\alpha_0) + \tan(\alpha_1)}{\tan(\alpha_0) - \tan(\alpha_1)}, \quad Z = y \tan(\beta_0)$$

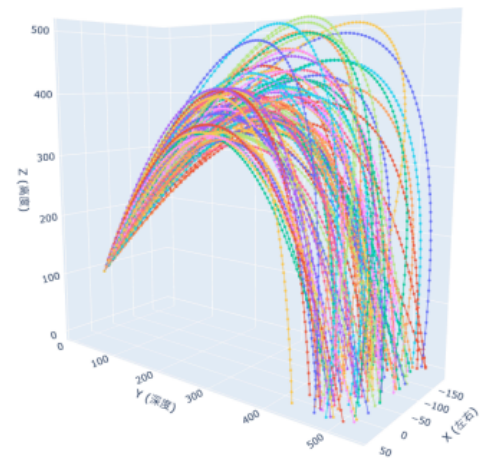
f is focal length, B is baseline, d is disparity, and α, β are horizontal/vertical viewing angles.



- **Trajectory Fitting and Model Inference:**

The stereo camera records frisbee 3D positions (X, Y, Z) during flight. Data is smoothed to reduce measurement noise, then fitted with polynomial regression to obtain a smooth parabolic trajectory.

For prediction, a **Long Short-Term Memory (LSTM)** model was trained using $Z(t)$ (height vs. time) as input, predicting corresponding (X, Y) coordinates over time. Predictions were considered accurate if within ± 10 cm of ground-truth.



c. Key Results and Data

- **Stereo Calibration Evaluation:**

Calibration using chessboard images across 80–250 cm resulted in a **mean reprojection error of 0.17 pixels**, providing reliable parameters for disparity computation.

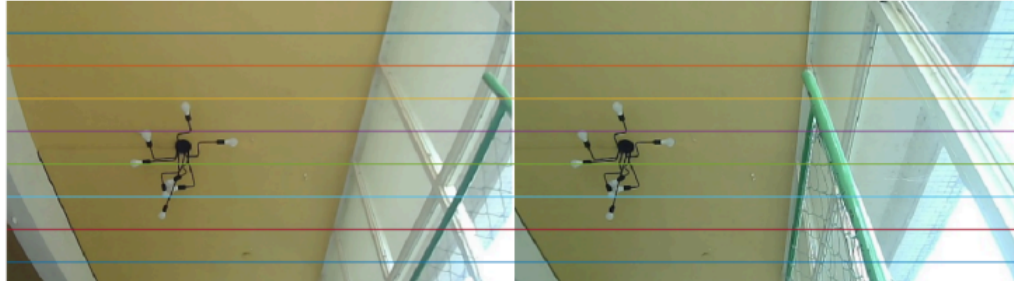


Figure 20. Stereo Disparity Calculation

- **Object Detection Performance:**

Inference speed reached **~16 fps**, sufficient for real-time frisbee tracking under dynamic conditions.

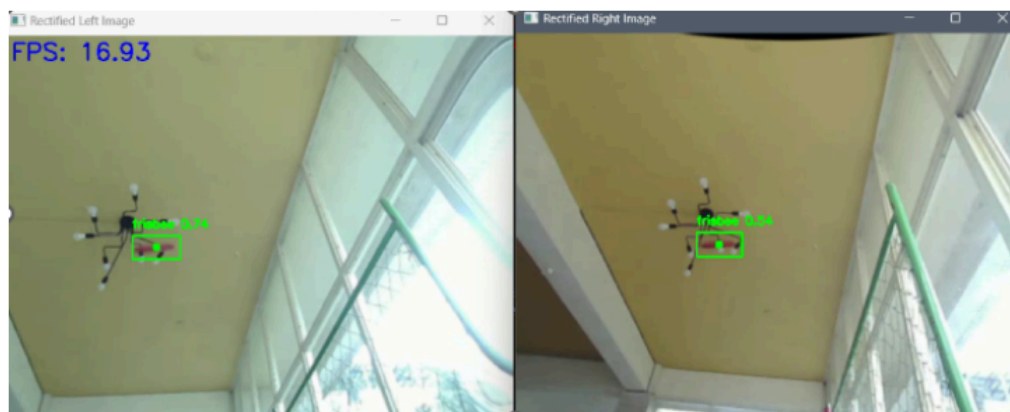


Figure 21. Object Detection Results

- **3D Localization Accuracy:**

Depth estimation tests (40–360 cm) showed average measurement error **< 10%**.

Actual (cm)		40	80	120	160	200	240	280	320	360
Measured (cm)	1	45.9	75.5	115.7	138.7	169.1	224.2	256.0	295.2	336.7
	2	45.9	75.1	109.1	141.8	176.2	224.2	258.3	292.2	333.5
	3	46.0	75.3	116.6	139.7	174.8	224.1	256.9	293.6	335.1
Avg		45.9	75.3	113.8	140.6	173.7	224.2	258.1	293.6	335.1
Error %		14.8	5.8	5.16	12.4	13.3	6.59	7.82	8.22	7.10

- **Trajectory Fitting & Model Inference:**

The primary objective of this section is to train a deep learning model capable of predicting the horizontal coordinates (X, Y) of the frisbee based on its vertical trajectory Z(t). A **Long Short-Term Memory (LSTM)** network was employed to capture the inherent temporal dependencies in frisbee flight trajectory data. The following summarizes the methodology and results.

- 1. Problem Definition and Objectives**

- **Input:** Time-series data of frisbee height, Z(t).
- **Output:** Predicted horizontal coordinates (X(t), Y(t)) for each time step.
- **Design Concept:** By feeding continuous Z(t) sequences into the LSTM model, the network learns how vertical motion (height) relates to horizontal displacement (X) and depth (Y) over time.

- 2. Accuracy Evaluation Criteria**

A custom accuracy metric was designed:

- Predictions are considered correct if the Euclidean distance between predicted and true values satisfies:

$$\sqrt{(X_{\text{pred}} - X_{\text{true}})^2 + (Y_{\text{pred}} - Y_{\text{true}})^2} \leq 10 \text{ cm}$$

- This strict threshold ensures reliable trajectory prediction under real-world noise and variation.

- 3. Results and Visualization**

- 3.1 Time-Series Plots of Z, X, and Y**

- *t vs. Z (ground truth):* The vertical trajectory exhibits a parabolic curve, consistent with projectile motion.
- *t vs. X:* Blue points represent true X values; red triangles denote predicted values. Comparison shows the model captures the general trend, though deviations exist.

- t vs. Y : Blue points represent true Y values; red triangles denote predicted values. The model demonstrates alignment with the true trajectory shape, though some bias can be observed over time.

Explanation:

By inputting a continuous $Z(t)$ sequence (complete vertical trajectory), the LSTM processes the temporal sequence and predicts $(X(t), Y(t))$ at each time step. Comparison of predicted (red) and actual (blue) trajectories indicates that while discrepancies exist on the X-axis, the Y-axis predictions largely follow the true trajectory pattern.

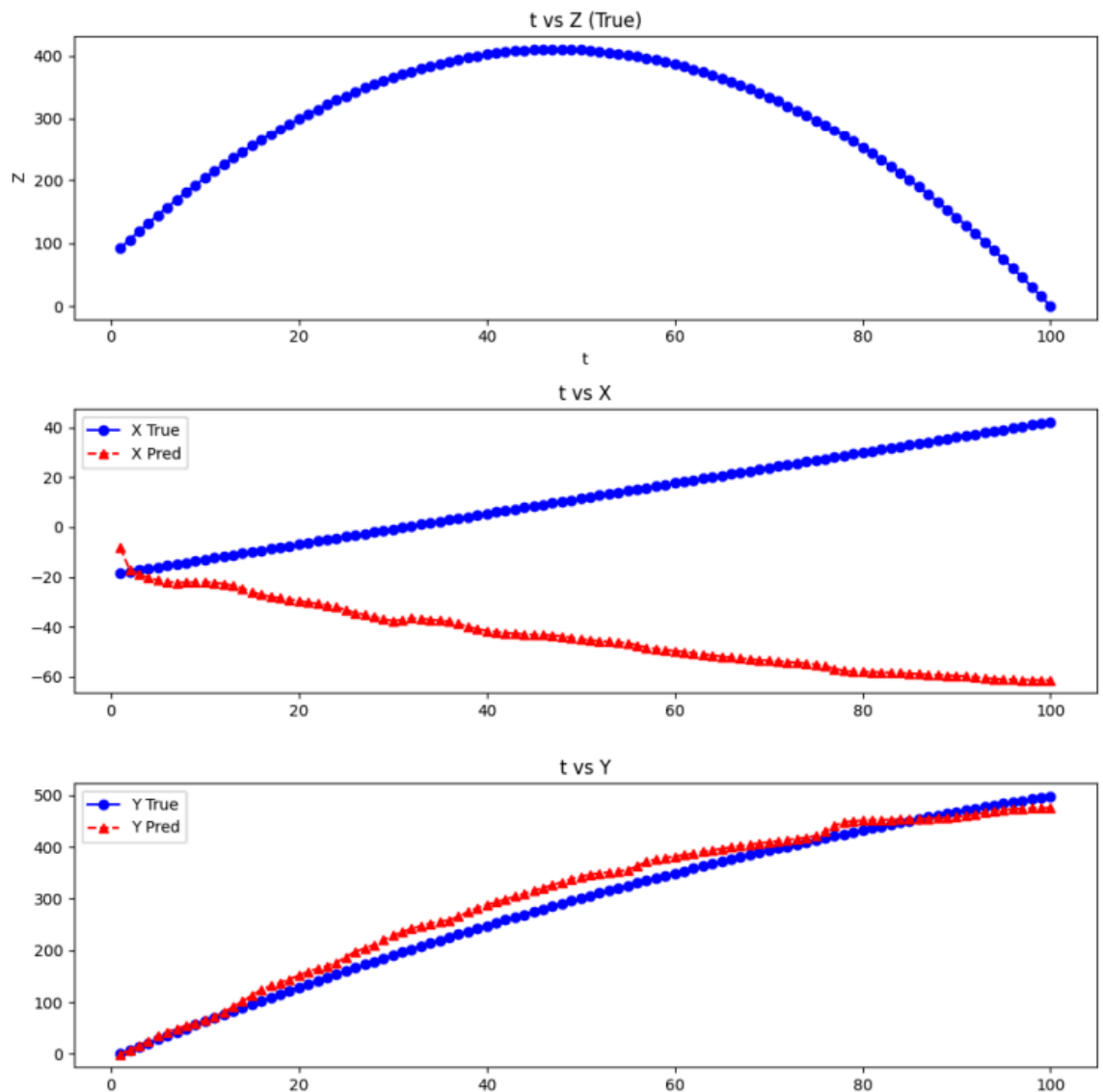


Figure 22. LSTM Prediction vs. Actual Trajectory

3.2 Training and Validation Loss

In the early stages of training, due to the large scale and variability of the

frisbee dataset, the Mean Squared Error (MSE) loss values were extremely high (on the order of tens of thousands). Effective normalization was difficult to achieve, resulting in large absolute error values.

After approximately **1000 training epochs**, both the training loss (*train_loss*) and validation loss (*val_loss*) steadily converged toward near-zero values (relative to the large scale of the raw data). This indicates that the network successfully learned from the temporal sequence data without exhibiting significant overfitting, as *val_loss* decreased in synchronization with *train_loss*.

Although the final MSE remained relatively large in absolute magnitude, the consistent downward trend in validation loss strongly suggests that the method is feasible and that the model continually improved its predictive performance.

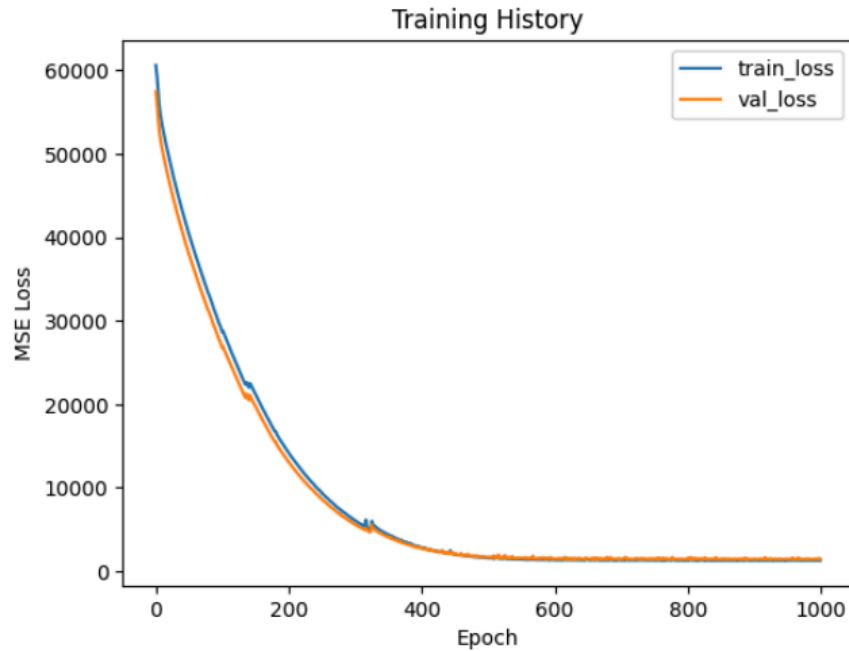


Figure 23. Training and Validation Loss Function

3.3 Validation Accuracy (± 10 cm Criterion)

A custom validation accuracy metric was defined to evaluate whether each predicted coordinate (X_{pred}, Y_{pred}) fell within ± 10 cm of the ground-truth value (X_{true}, Y_{true}).

As training progressed, the accuracy curve gradually increased, occasionally reaching **10–14%**. While this value appears modest, it is primarily constrained by the physical system: the frisbee launcher itself exhibited significant

variability, making precise prediction inherently challenging. Furthermore, the ± 10 cm threshold is a strict standard, amplifying the difficulty.

Despite these challenges, achieving accuracy above 10% under noisy real-world conditions demonstrates the model's ability to capture useful trajectory patterns and reflects its learning capability.

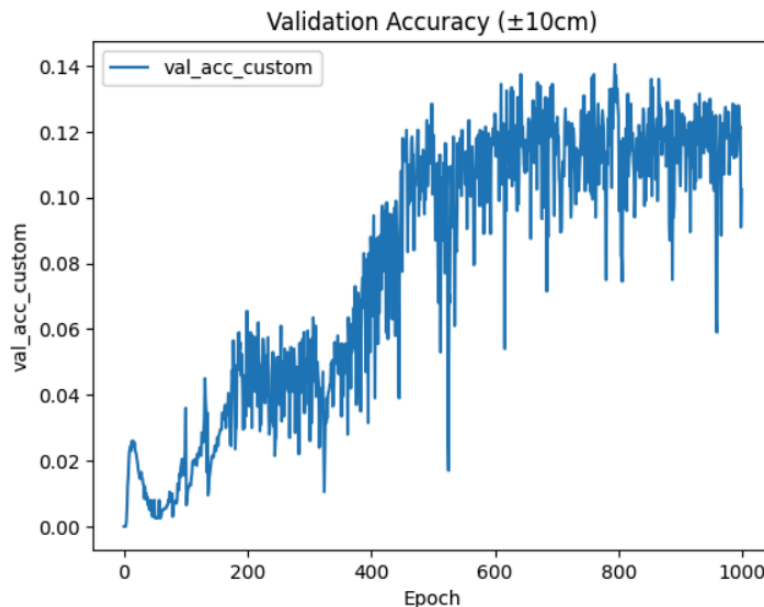


Figure 24. Custom Accuracy Distribution

III. Results and Discussion

1. Communication System

By adopting the **SBUS communication protocol**, the program structure and the number of required pins were greatly simplified, while system efficiency was optimized. Compared with traditional PWM signal processing, SBUS provided a more efficient and stable data transmission method. It ensured that signal delivery did not block or interfere with the execution of the main program, thereby enhancing overall system performance and stability.

2. Control System

The control system design enabled each wheel's speed to match more quickly and consistently. This was particularly evident during operations such as moving forward in a straight line, where the robot could operate more smoothly and reliably. Precise speed control significantly improved both accuracy and stability of motion, allowing the robot to adapt more effectively to complex environments and enhancing overall maneuverability.

3. Vision System

The vision system provided critical reference data for frisbee launching, eliminating the need for random manual adjustments during operation. Through the analysis of visual data, frisbee launch conditions could be determined more precisely, improving both stability and accuracy. Furthermore, the integration of the vision system made the robot more intelligent, strengthening both operational efficiency and overall effectiveness.

IV. Conclusion

This project progressed step by step from the ground up, covering mechanical design, control system implementation, and vision system integration. Through repeated trials and iterative improvements, the team successfully achieved the competition objectives while also gaining deeper insights into the importance of **self-directed learning, problem identification, method design, and solution development**.

In addition, the project demonstrated high **cost-effectiveness**, achieving results with relatively limited resources.

Special thanks are extended to the following professors for their invaluable guidance and support:

- **Mechanical Design:** Prof. **Hsi-Cheng Tsai**, for identifying design blind spots and providing design direction.
- **Control System:** Prof. **Hsien-Ting Chang**, for valuable advice on logic and circuit design.
- **Vision System:** Prof. **Hsiang-Chieh Chen**, for his support in enhancing computer vision applications.

Their professional guidance and constructive suggestions were instrumental to the successful completion of this project, for which we express our deepest gratitude.