ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Design Document for Project: Design, Build and Control of a Jumping Robot

---

Team #36

XINYI YANG
(xinyiy19@illinois.edu)
SIYING YU
(siyingy3@illinois.edu)
HANJUN LUO
(hanjunl2@illinois.edu)
XUECHENG LIU
(xl125@illinois.edu)

TA: Leixin Chang

April 11, 2025

Contents

1 Introduction

In today's complicated and continually changing real-world application settings, robotics development is confronted with the challenge of navigating difficult terrains. Typical wheeled or legged robots generally do not perform well on rugged, non-uniform terrains, confined spaces, or environments with vertical objects, limiting their real-world value and versatility [1].
Our proposal intends to tackle this important problem by designing a bio-inspired jumping robot system in order to create a unique, low-cost and reliable jumping robot design. Biological research indicates that fleas can produce jumping forces in excess of 100 times their body weight. This extraordinary means of storing and releasing energy density has heavily inspired us and provided an important model for our research. Thus, our proposed jumping robot solution imitates the free mechanism of fleas and combines an efficient spring-based storage/energy system, a smart motor-driven control mechanism and a lightweight structural design inspired from biological systems. Compared to existing jumping robots [2]–[4], our design achieves higher energy density through an energy storage module that mimics the elastic protein pad structure of fleas. Additionally, an innovative trigger-release mechanism ensures explosive force release, enabling more controllable jumping performance while maintaining a compact structure. A pictorial representation is shown in Figure 1.

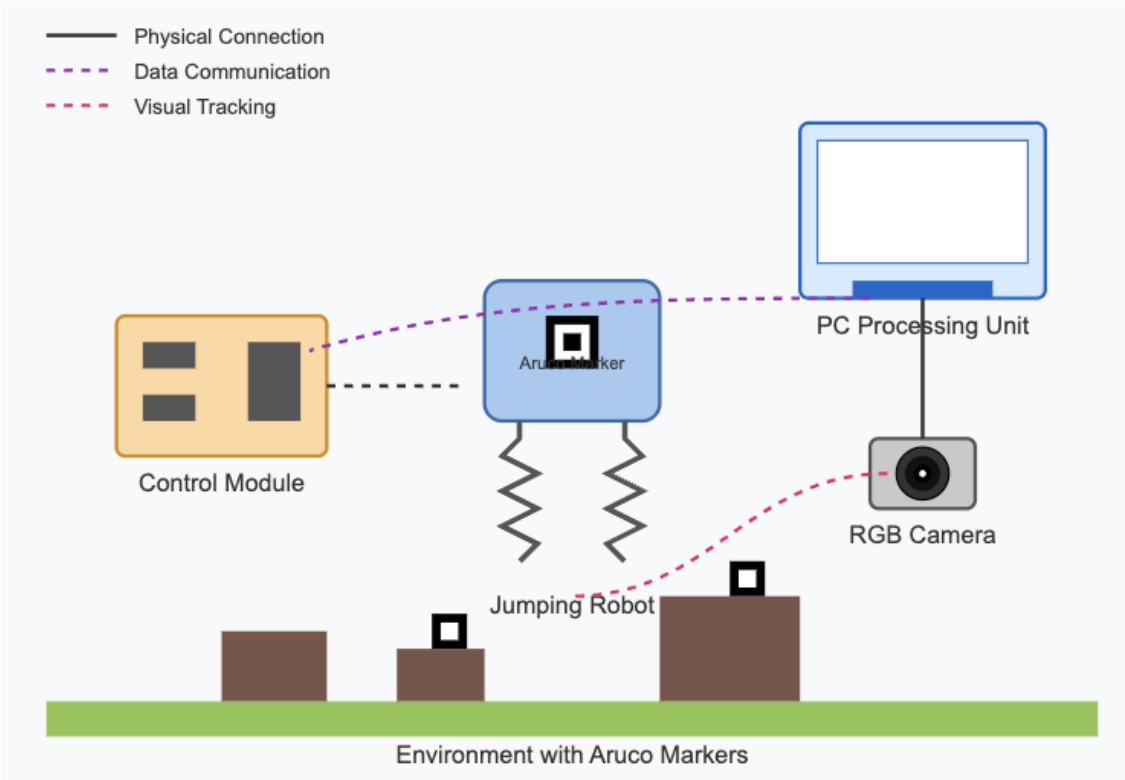

Figure 1: Our Jumping Robot includes mechanical module with Aruco markers, control module, and computer vision module.

Our jumping robot must meet the following key requirements:

- The robot must demonstrate multi-level jumping capability with at least three distinct and reproducible jump heights (low, medium, and high), allowing it to navigate various obstacle configurations.
- The energy storage and release system must achieve sufficient power density to generate vertical jumps of at least 20 cm in height, enabling the robot to overcome typical obstacles.
- The control module must provide precise regulation of jumping force with an accuracy of ±10%, ensuring predictable control through the collaboration of real-time visual feedback from the camera distance calculation system.
- The vision module must achieve 3D distance estimation between the robot and environmental features with an accuracy of ±1 cm and complete all computation tasks within 100 ms per frame, enabling real-time trajectory planning and adaptive jumping control in dynamic environments.

2 Design

2.1 Block Diagram

The system consists of four main subsystems: power, control, motion, and computer vision. The power subsystem ensures continuous operation by providing 5V and 3.3V to power the ESP32 microcontroller, stepper motor driver, and coreless motor. The control subsystem is managed by an ESP32 microcontroller, which communicates via Bluetooth for data transfer. The motion subsystem includes a coreless motor and a stepper motor, driving the robot's mechanical components including the torsional spring and detach module. The computer vision subsystem, connected to the control system via Bluetooth, processes visual input and sends the required jump parameters to the control system.

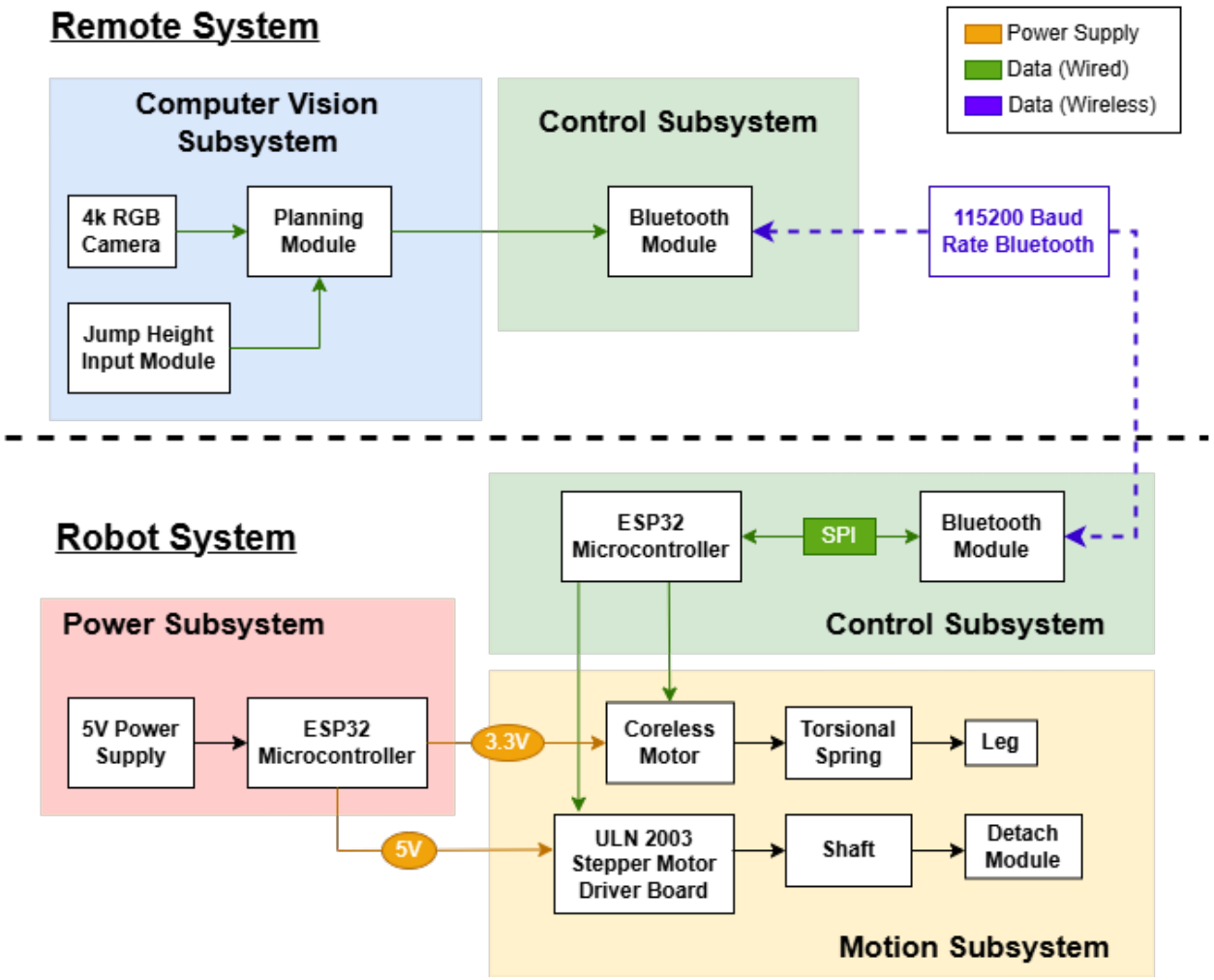

Figure 2: Block Diagram

2.2 Physical Design

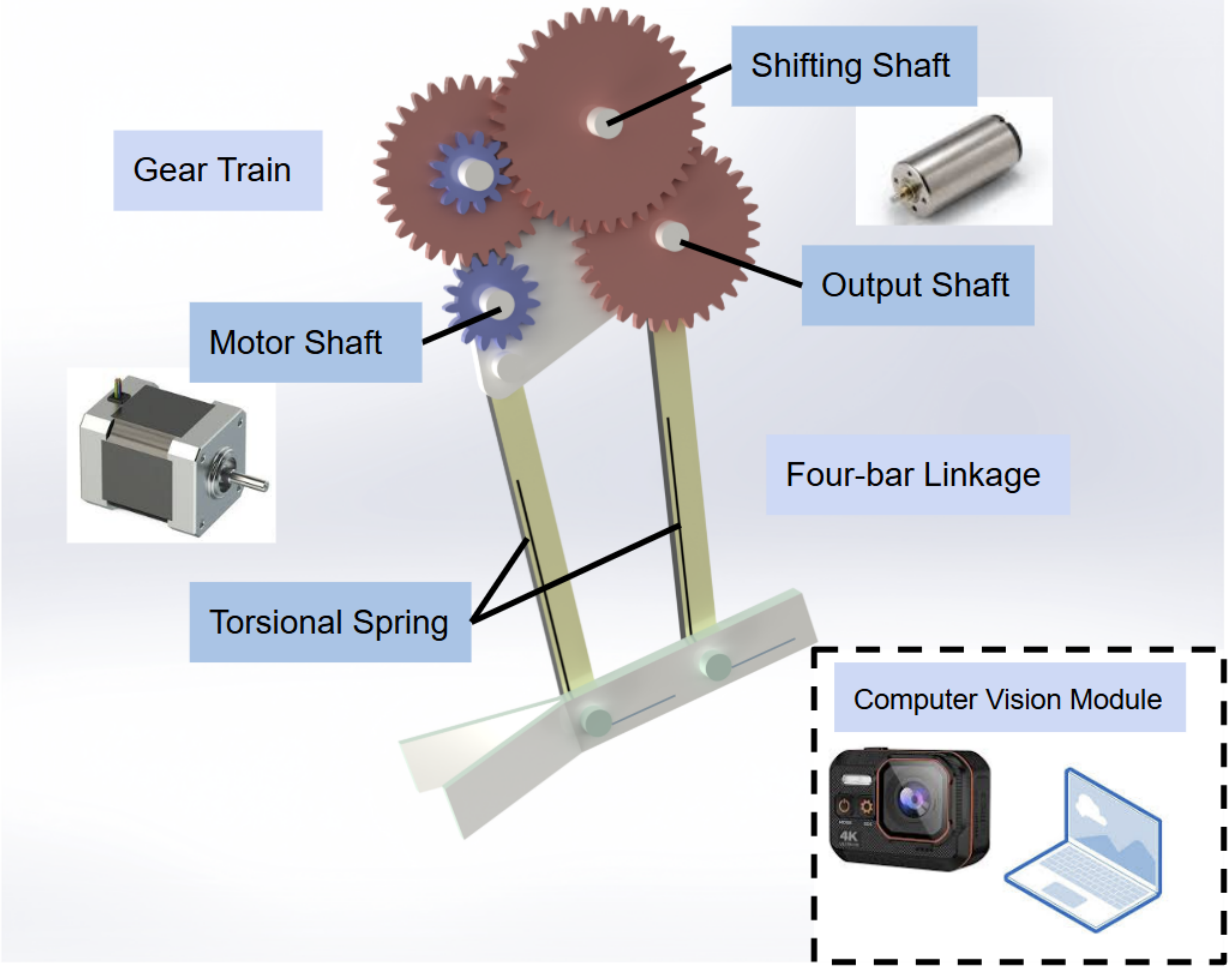

Figure 3: Physical Design of the Robot.

Figure 3 illustrates the updated physical layout of our robotic system. The structure consists of a 3D-printed main chassis that houses the mechanical transmission system, actuation components, and embedded electronics, along with a computer vision module for motion planning.

2.3 Subsystem

2.3.1 Computer Vision Module

The computer vision module directly supports our high-level requirement of achieving distance estimation within ±1 cm accuracy and completing all computational tasks within 100 ms per frame. Our implementation utilizes an external 4k RGB camera, an external processing unit, and some aruco markers on the robot and environment. A pipeline is shown in Figure 4.

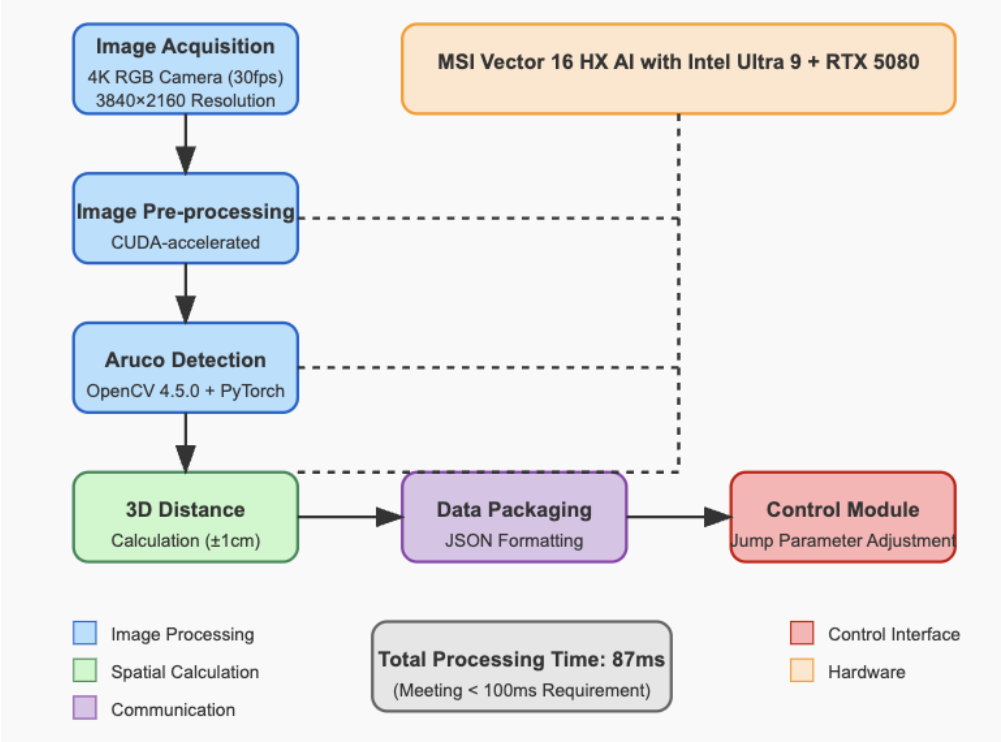

Figure 4: Computer Vision Module Processing Pipeline.

The camera captures frames at 30Hz with 3840*2160 resolution (UHD), providing sufficient visual information. Prior to deployment, we perform a rigorous camera calibration procedure using a standard 5*7 checkerboard to obtain precise intrinsic parameters, which are essential for accurate 3D reconstruction. We take 30 pictures for the checkerboard with the camera from different angles to complete calibration and utilize OpenCV for calculating intrinsic parameters. For marker tracking, we employ the 4×4 Aruco markers from the DICT_4X4_250 dictionary, which offer an optimal balance between detection reliability and computational efficiency. These 3cm×3cm markers are strategically placed on the robot body and throughout the test environment, enabling reliable detection at distances up to 3 meters with our UHD camera.

Our software processing pipeline utilizes OpenCV's ArUco module (version 4.5.0) to detect and decode markers with adaptive thresholding techniques that handle varying lighting conditions while maintaining a false positive rate below 0.1%. To ensure that the computation tasks could be completed at a speed that meets the requirements, we used a high-performance laptop equipped with an Intel Ultra 9 275HX CPU and an Nvidia RTX 5080 Laptop GPU as the computing unit [5]. During the code development process, we implemented specific optimizations by utilizing PyTorch and OpenCV libraries that are compatible with CUDA technology, ensuring that GPU acceleration could be leveraged. This enabled high-speed processing of 4K resolution high-definition images, successfully reducing the latency to 87ms.

The vision subsystem interfaces with the control module using lightweight JSON data structures to minimize latency. Each message contains the robot's position and orienta-

tion (x, y, z, roll, pitch, yaw), obstacle coordinates, target landing zone coordinates, and a timestamp for precise synchronization. This carefully engineered subsystem directly enables the robot to perceive and navigate complex environments through precise real-time spatial awareness, fulfilling its role as a critical component in achieving our high-level requirements for controlled and efficient jumping.

Requirement	**Verification**
The UHD camera must capture frames at 30 Hz with a resolution of 3840×2160 to ensure sufficient visual detail.	A. Connect the camera and collect 100 consecutive frames using OpenCV. B. Verify frame rate using timestamp differences and confirm resolution through pixel dimensions.
Camera intrinsic parameters must be precisely calibrated for accurate 3D reconstruction.	A. Capture 30 checkerboard images (5×7) from various angles. B. Use OpenCV calibration functions to compute reprojection error and ensure it is below 0.3 pixels.
ArUco markers (3cm×3cm) must be detected up to a distance of 3 meters with ¡0.1% false positive rate.	A. Place markers at distances ranging from 1 to 3 meters under different lighting conditions. B. Evaluate detection accuracy and false positive rate across 200 frames using OpenCV's ArUco module.
Vision processing must maintain latency below 100 ms per frame.	A. Run the full detection and decoding pipeline (OpenCV + PyTorch with CUDA) on the target laptop. B. Measure end-to-end latency using timestamps before and after image processing.
The system must transmit pose and environment data to the control module using lightweight JSON structures.	A. Log transmitted JSON messages and confirm each message includes position (x, y, z), orientation (roll, pitch, yaw), obstacles, targets, and timestamp. B. Measure parsing time and communication delay to ensure data arrives within 10 ms.

Table 1: Vision Subsystem Requirements and Verification

2.3.2 Spring Module

Torsioanl Spring We use two torsional springs to store rotational potential energy and release them during the jumping process. When the motor rotates, it winds the torsional spring, building up torque. Upon release, the spring drives the linkage mechanism to generate a sudden rotational motion for jumping.

Linear Spring Additionally, a linear (compression) spring is placed beneath a sliding shaft. During motor rotation, the shaft moves forward to engage the gear set. Once the spring is compressed and the latch is triggered, the shaft retracts, allowing the gear group to instantly detach. This detachment releases the stored energy from the torsional spring, enabling a rapid jump.

Requirement	**Verification**
Torsional spring must output sufficient torque ($\geq 0.35\,\mathrm{N}\cdot\mathrm{m}$) when twisted from 120° to 30° to actuate the linkage mechanism for jumping.	A. Mount the torsional spring on a shaft with a known arm length (e.g., 0.1 m). B. Rotate the spring to 90° and hang weights on the arm until static equilibrium is reached. C. Calculate torque as $T = F \cdot r$. Verify that $T \geq 0.35\,\mathrm{N}\cdot\mathrm{m}$. D. Release the spring and verify that it drives the jumping linkage through the full stroke.
Traditional (linear) spring must retract a shaft to detach gear set and apply $\geq 0.1\,\mathrm{N}$ force to overcome friction.	A. Compress the spring to the expected preload length. B. Observe gear disengagement visually or via encoder feedback to confirm successful detachment. C. Attach a force sensor to verify that the force is $\geq 0.1\,\mathrm{N}$.

Table 2: Spring Module Requirements and Verification

2.3.3 Gear Train

The gear train is designed to achieve an 18:1 speed reduction between the motor and the output shaft, enabling sufficient torque transmission to preload the torsional spring. It consists of multiple compound gears, all 3D-printed, and assembled with precise alignment. The smoothness and accuracy of this reduction are critical to ensure proper energy storage without slippage or jamming.

Requirement	Verification
The gear train must achieve a total gear reduction ratio of 18:1.	A. Count the teeth of each gear in the system and verify the reduction ratio mathematically. B. Rotate the input shaft 18 full turns and confirm the output shaft completes exactly 1 rotation.
Gears must mesh smoothly, without jamming, abnormal noise, or significant backlash.	A. Observe the transmission during operation and listen for noise. B. Check rotational consistency and measure backlash using a dial indicator if needed.

Table 3: Gear Train Requirements and Verification

2.3.4 Four-Bar Mechanism

The four-bar linkage is coupled with the torsional spring and is responsible for storing and releasing mechanical energy during the jump phase. It must rotate through 90° in sync with the torsional spring to achieve full preload. Smooth, reliable motion is essential to ensure consistent energy delivery and controlled jumping trajectories.

Requirement	Verification
Microcontroller must initialize and become responsive to Bluetooth input within 3 seconds of power-on.	A. Power on the ESP32 while monitoring status via serial log or onboard LED. B. Send a test command from the host computer via Bluetooth within 3 seconds. C. Confirm command is received and acknowledged by ESP32 through observable output (e.g., LED toggle or serial print).
Microcontroller must drive two motors through GPIO control signals in response to external commands.	A. Send two distinct test commands via Bluetooth. B. Observe motor driver inputs (e.g., logic analyzer or test LED). C. Confirm the correct motor activates for each command.
Microcontroller must successfully execute a full compress-release cycle when triggered by the host system.	A. Trigger a jump command via Bluetooth. B. Observe motion (spring compress + detach mechanism) or substitute with LED sequence. C. Confirm system returns to idle state after cycle completes.

Table 4: Microcontroller Requirements and Verification (Functional-Level)

2.3.5 Embedded Control System

The Embedded Control System functions as the central coordinator for both actuation and communication within the device. It receives Bluetooth commands from the host system (e.g., the computer responsible for the visual module), and generates corresponding signals to drive mechanical components via the motor driver. The microcontroller is physically connected to the host computer via a USB type-C interface, through which it is also powered.

Microcontroller The microcontroller, Espressif 32, handles data transmitting and motor driving. This microcontroller was chosen because of its low cost, small size, light weight, and built-in Bluetooth capability. It communicates with the developer's computer and flash the firmware through the type-C serial port.

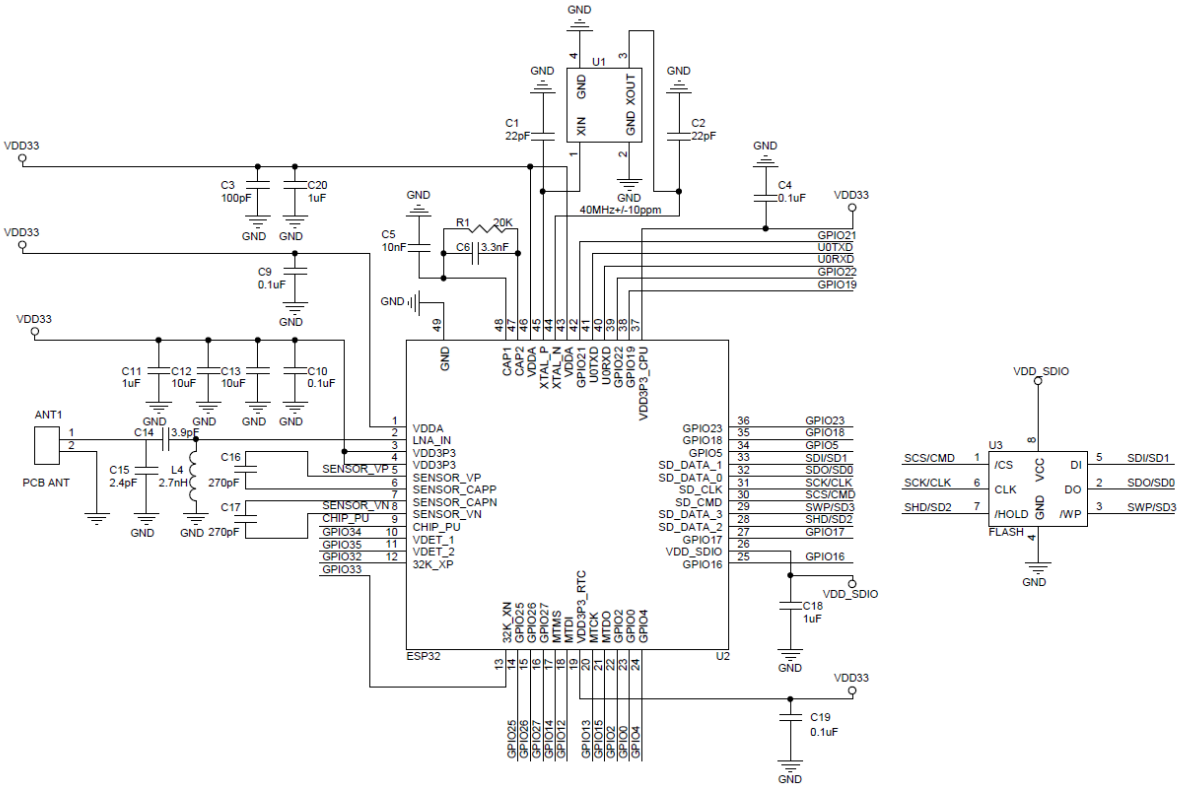

Figure 5: ESP32 Microcontroller Schematic

Requirement	Verification
Microcontroller must initialize and become responsive to Bluetooth input within 5 seconds of power-on.	A. Power on the ESP32 while monitoring status via serial log or onboard LED. B. Send a test command from the host computer via Bluetooth within 3 seconds. C. Confirm command is received and acknowledged by ESP32 through observable output (e.g., LED toggle or serial print).
Microcontroller must drive two motors through GPIO control signals in response to external commands.	A. Send two distinct test commands via Bluetooth. B. Observe motor driver inputs (e.g., logic analyzer or test LED). C. Confirm the correct motor activates for each command.
Microcontroller must successfully execute a full compress-release cycle when triggered by the host system.	A. Trigger a jump command via Bluetooth. B. Observe motion (spring compress + detach mechanism) or substitute with LED sequence. C. Confirm system returns to idle state after cycle completes.

Table 5: Microcontroller Requirements and Verification (Functional-Level)

Bluetooth Module The classic Bluetooth module of the microcontroller starts in slave mode, using a baud rate of 115200. The host computer that performs CV processing will search and actively connect to the ESP32 as a master, and communicate through the Bluetooth serial port.

Requirement	Verification
Bluetooth connection must remain stable for at least 60 seconds under continuous transmission.	A. Establish Bluetooth connection and initiate repeated communication (e.g., ping or status messages). B. Monitor serial log or onboard indicator for disconnects. C. Verify uninterrupted connection for 60 seconds.
Bluetooth module must transmit valid control commands from the host to ESP32 with $\geq$ 90% reliability.	A. Send 100 predefined commands over Bluetooth. B. Log and count correctly executed responses. C. Confirm success rate $\geq$ 90%.

Table 6: Bluetooth Module Requirements and Verification

Motor The coreless motor is used for compressing the torsional spring and providing the force necessary for the robot's jumping motion. This motor was selected due to its high efficiency, compact size, and low inertia. The motor operates with a 3.3V power supply and is controlled via PWM signals from the ESP32 microcontroller, allowing for speed and torque adjustments. It is directly connected to the GPIO pins and is powered by ESP32 microcontroller.

Requirement	**Verification**
Coreless motor must begin rotating within 1s when PWM is enabled at 80% duty cycle.	A. Send command to activate the coreless motor. B. Use stopwatch, LED indicator, or camera to measure time to visible rotation. C. Confirm startup within 200 ms.
Coreless motor must maintain rotation for at least 5 seconds without overheating (surface temperature $\leq 80^{\circ}$C).	A. Run motor continuously for 5 seconds. B. Measure motor casing temperature using an IR thermometer. C. Verify temperature remains within safe limit.

Table 7: Coreless Motor Requirements and Verification

The stepper motor operates the detach switch mechanism, which, when activated, disengages the gears to release the torsional spring's stored energy, enabling the robot to jump. This motor is powered by a 5V supply and is driven by the ULN2003 stepper motor driver. Control signals are sent directly from the ESP32's GPIO pins to the ULN2003 driver, allowing precise manipulation of the detach switch.

Requirement	**Verification**
Stepper motor must activate the detach switch within 500 ms after receiving the release command.	A. Trigger detach command via Bluetooth. B. Observe gear release or LED indicator. C. Use stopwatch or video to verify switch actuates within 0.5 s.
Stepper motor must perform 10 continuous detach/reattach cycles without mechanical failure.	A. Automate repeated detach and reattach commands. B. Inspect gear interface after 10 cycles. C. Verify there are no cracks, jams, or misalignments.
Stepper motor must hold gear in detached position against spring preload force for at least 2 seconds.	A. Detach mechanism while spring is loaded. B. Observe gear stability for 3 seconds. C. Verify no re-engagement or slippage occurs.

Table 8: Stepper Motor Requirements and Verification

2.4 Tolerance Analysis

The most critical function of our jumping robot is to achieve a consistent jump such that its landing position is within a 5 cm error radius at a nominal jump distance of 30 cm. This performance depends on both the repeatability of the energy release from the spring module and the accurate determination of target parameters by the computer vision module.

2.4.1 Subsystem Tolerances and Key Parameters

Computer Vision Module:

- **Distance Estimation:** The vision system (using a 4K UHD camera calibrated with a 5×7 checkerboard) provides distance estimates with an accuracy of ± 1 cm.
- **Marker Tracking:** 3 cm×3 cm ArUco markers from the DICT_4X4_250 dictionary ensure reliable detection up to 3 m.
- **Processing Latency:** With processing completed in approximately 87 ms (within the 100 ms requirement), the vision subsystem delivers timely data.
- **Angular Accuracy:** The system is assumed to yield an angular uncertainty $\delta\psi$ (e.g., ≤ 0.03 rad) which contributes to lateral positioning errors.

Spring Module:

- **Torsional Spring:** Two torsional springs store energy for the jump. The stored energy is given by
$$E = \frac{1}{2} k \, \varphi^2,$$
where k is the spring constant and φ is the compression (wind-up) angle. The spring is designed to produce a minimum torque of 0.35 N·m when rotated between 120° and 30°. We assume a manufacturing tolerance of $\pm 5\%$ on k, with an additional uncertainty $\delta\varphi$ from the motor encoder resolution and mechanical backlash.
- **Linear Spring:** A compression spring retracts a sliding shaft to disengage the gear set, applying a release force of at least 3 N. Its function is primarily to trigger the mechanism.

Embedded Control System: The ESP32 microcontroller—coupled with a Bluetooth module and interfaced via USB Type-C—coordinates actuation. Their error contributions are minimal compared to mechanical and vision uncertainties.

2.4.2 Nominal Range and Energy Conversion

Assuming near-total conversion of stored energy into kinetic energy, the launch speed is:
$$v = \sqrt{\frac{2E}{m}}, \quad \text{with } E = \frac{1}{2} k \, \varphi^2,$$

where $m = 0.08\,\mathrm{kg}$ is the robot mass. Under projectile motion, the horizontal jump range is expressed as:

$$R = \frac{v^2 \sin 2\theta}{g} = \frac{k\,\varphi^2 \sin 2\theta}{m\,g},$$

with $g \approx 9.81\,\mathrm{m/s^2}$ and θ as the launch (release) angle. For a nominal jump distance $R_{\text{nominal}} = 0.30\,\mathrm{m}$, proper selection of k, φ, and θ is essential.

2.4.3 Error Propagation in the Jumping Mechanism

Errors in the jump distance arise from variations in:

- **Spring Parameters:** Tolerance in k (relative error $\delta k/k$) and in the compression angle φ (relative error $\delta\varphi/\varphi$).
- **Energy Losses:** Unmodeled friction or incomplete energy transfer is represented by a fractional error δ_{loss}.
- **Launch Angle:** Imperfections in the cam and linkage yield an angular error $\delta\theta$.
- **Mass Variation:** Variations in mass, represented by $\delta m/m$, affect the conversion of energy to launch speed.

The relative error in the stored energy is:

$$\frac{\delta E}{E} = \frac{\delta k}{k} + 2\,\frac{\delta\varphi}{\varphi} + \delta_{\text{loss}}.$$

Differentiating

$$R = \frac{k\,\varphi^2 \sin 2\theta}{m\,g},$$

with respect to E, θ, and m, yields a linearized form for the relative range error due to jumping:

$$\left(\frac{\delta R}{R}\right)_{\text{jumping}} = \frac{\delta E}{E} + 2\cot(2\theta)\,\delta\theta - \frac{\delta m}{m}.$$

Thus, the absolute error from the jumping mechanism is:

$$\Delta R_{\text{jumping}} = R\left[\frac{\delta k}{k} + 2\,\frac{\delta\varphi}{\varphi} + \delta_{\text{loss}} + 2\cot(2\theta)\,\delta\theta - \frac{\delta m}{m}\right].$$

2.4.4 Vision and Alignment Errors

The vision system further contributes errors:

- **Range Error:** Denoted by ΔR_{sensor}, with an accuracy of ± 1 cm.
- **Azimuth Error:** An angular error $\delta\psi$ causes a lateral displacement:

$$\Delta a = R\,\delta\psi.$$

2.4.5 Total Error Budget

Since the errors in range (from both the jumping mechanism and vision system) and the lateral error (from azimuth uncertainty) are orthogonal, the total landing error is given by:

$$\Delta_{\text{total}} = \sqrt{(\Delta R_{\text{jumping}} + \Delta R_{\text{sensor}})^2 + (R\,\delta\psi)^2}.$$

To meet the design requirement, we must have:

$$\Delta_{\text{total}} \leq 0.05\,\text{m}.$$

2.4.6 Required Numerical Values

To ensure that the overall landing error remains below 5 cm at a nominal jump distance of 30 cm, the following target tolerances are recommended. (These values are somewhat relaxed; worst-case estimates may slightly exceed 5 cm, but average errors should be lower due to statistical cancellation.)

- **Torsional Spring:** The relative tolerance on the spring constant should satisfy

$$\frac{\delta k}{k} \leq 5\%.$$

- **Compression Angle Accuracy:** The relative error in the compression angle should be kept within

$$2\left(\frac{\delta\varphi}{\varphi}\right) \leq 6\%,$$

  i.e. $\delta\varphi/\varphi \leq 3\%$.

- **Launch Angle Error:** The error in the release (launch) angle is recommended to be

$$\delta\theta \leq 3^\circ \quad (\approx 0.052\,\text{rad}).$$

  (Note that for a nominal launch angle of 45°, the sensitivity to $\delta\theta$ is minimal.)

- **Mass Variation:** The variation in the overall mass should not exceed

$$\frac{\delta m}{m} \leq 2\%.$$

- **Vision System:**
  - The range error must be maintained at

$$\Delta R_{\text{sensor}} \leq 1\,\text{cm}.$$

  - The angular (azimuth) error should be restricted to

$$\delta\psi \leq 0.05\,\text{rad}.$$

Under the assumption that energy losses can be compensated by adjusting the compression angle, the nominal jump distance is modeled by

$$R = \frac{k\,\varphi^2\,\sin 2\theta}{m\,g},$$

with $R = 0.30\,\mathrm{m}$, $m = 0.08\,\mathrm{kg}$, and $g \approx 9.81\,\mathrm{m/s^2}$.

Assuming a release angle of $\theta = 45^\circ$ (minimizing the sensitivity to launch angle errors as $\cot(90^\circ) = 0$), the error contribution from the firing mechanism becomes:

$$\Delta R_{\text{firing}} = R\left[\frac{\delta k}{k} + 2\frac{\delta\varphi}{\varphi} - \frac{\delta m}{m}\right].$$

Substituting the target values:

$$\Delta R_{\text{firing}} = 0.30\ (0.05 + 0.06 - 0.02) = 0.30 \times 0.09 = 0.027\,\mathrm{m} \quad (2.7\,\mathrm{cm}).$$

Including the vision module's range error, the total range error becomes:

$$\Delta R_{\text{total}} = \Delta R_{\text{firing}} + \Delta R_{\text{sensor}} = 2.7\,\mathrm{cm} + 1\,\mathrm{cm} = 3.7\,\mathrm{cm}.$$

The vision subsystem's angular uncertainty contributes a lateral (azimuthal) error of:

$$\Delta a = R\,\delta\psi = 0.30 \times 0.05 = 0.015\,\mathrm{m} \quad (1.5\,\mathrm{cm}).$$

Thus, the total landing error is estimated by:

$$\Delta_{\text{total}} = \sqrt{(\Delta R_{\text{total}})^2 + (\Delta a)^2} = \sqrt{(3.7\,\mathrm{cm})^2 + (1.5\,\mathrm{cm})^2} \approx \sqrt{13.69 + 2.25} \approx \sqrt{15.94} \approx 4.0\,\mathrm{cm}.$$

This worst-case estimation (approximately 4.0 cm) meets the design requirement of a landing error below 5 cm.

2.4.7 Conclusion

The overall landing error is expressed as:

$$\Delta_{\text{total}} = \sqrt{(\Delta R_{\text{jumping}} + \Delta R_{\text{sensor}})^2 + (R\,\delta\psi)^2} \leq 0.05\,\mathrm{m}.$$

This analysis shows that with carefully selected components and rigorous calibration—especially in the spring mechanism and the vision module—the robot can reliably achieve a 30 cm jump with a maximum error of 5 cm. Moreover, this simulation-based tolerance analysis provides a systematic approach to refining the design and identifying the key factors limiting performance, thereby guiding further improvements.

3 Cost and Schedule

3.1 Cost Analysis

The calculation of Labor Costs use rates based on UIUC ECE graduate salaries ($85k/year = $41/hr, 2080 hours per year). We include 25% buffer for iterations. The information is provided in Table 9.

Table 9: Labor Cost Calculation

Role	**Hours**	**Rate ($/hr)**	**Total**
Mechanical (2 persons)	100	41	4,100
Computer Vision	50	41	2,050
Control Systems	55	41	2,255
		Total Labor	**8,405**

All materials are bought from Taobao/Tmall. We convert the units from RMB to USD. 3D Printing materials and equipments are provided by the university for free. The information is provided in Table 10.

Table 10: Parts & Materials Cost

Component	**Manufacturer**	**Description**	**Qty**	**Unit Price ($)**	**Total ($)**
ESP32-WROOM	Espressif	WiFi/BLE Development Board	2	3.40	6.80
28BYJ-48 Motor	-	5V Stepper Motor with Driver	2	1.15	2.30
0.4mm Spring	Misumi	Torsion Spring (0.4mm wire)	10	0.60	6.00
0.5mm Spring	Misumi	Torsion Spring (0.5mm wire)	10	0.70	7.00
4K Camera	Hengqiaotong	USB 4K Webcam with Autofocus	1	32.44	32.44
				Total Parts	**54.54**

The grand total cost is summarized in Table 11.

Table 11: Grand Total Cost

Category	**Amount ($)**
Labor	8,405.00
Parts & Materials	54.54
Total Project Cost	**8,459.54**

3.2 Schedule

Table 12: Project Schedule

Time Period	**Hanjun**	**Xinyi & Xuecheng**	**Siying**	**All**
Completed	Aruco tag generation, Camera calibration	Spring mechanism CAD design, Component ordering, First prototype assembly	Control system architecture	Design review
4/7-4/13	Distance calculation algorithm	Reinforced mounting solutions	Motor driver configuration	Prototype evaluation
4/14-4/20	Vision-system API development	Improved fastener implementation	Jump height control algorithm	Failure mode analysis
4/21-4/27	Integration with control system	Motor retention enhancements	Motor stall prevention	Component reliability testing
4/28-5/4	Testing	Structural optimization	System integration with vision	Second prototype test
5/5-5/11	Final optimization	Final mechanical validation	Feedback loop tuning	Final demonstration

4 Discussion of Ethics And Safety

The ethical considerations presented in the original proposal are thorough, and the project scope has not extended into areas requiring any additional ethical review. The camera system continues to comply strictly with privacy-preserving protocols, capturing only the minimum visual data necessary for the purposes of jump calibration, and all testing takes place in a controlled environment with clear notifications to bystanders.

Within safety, there was the unrecognized risk of parts loosening or separating after a sustained number of impact loads leading to unpredictable jump paths or possible total mechanism failure. To avoid this risk, reinforced mounting brackets and high-strength fasteners with locking compounds were provided throughout the assembly. The leg components are secured with redundant attachment points so that even under the most extreme stress conditions, it should remain stable. To mitigate the possibility of motor separation, made custom mounting plates with strain relief that distribute the load forces evenly across the motor housing and connection points.

According to our safety procedures, all fasteners are required to be checked for appropriate torque prior to use as well as be inspected visually with an inspection checklist to ensure that no early indications of damage or misalignment are present in the component. Motors track both temperature and current draw to prevent a motor from stalling, which can happen under too high of a load. Provisions in the control system allow for programmed automatic shut downs if temperature or current exceed user thresholds. Testing to establish a baseline for using the robot can be confined to a unique 3 meter perimeter (and tethered restraints will also be used judiciously). The safety protocols are efficient, while still strenuous - i.e. we are attempting to use predictable hardware and use safe standard operational protocols regularly across the industrial robotics safety standard [6]. Every safety measure can be verified in the field without specialized tools, demonstrating our objective of having a solid yet manageable risk reduction framework.

References

[1] C. Zhang, W. Zou, L. Ma, and Z. Wang, "Biologically inspired jumping robots: A comprehensive review," *Robotics and Autonomous Systems*, vol. 124, p. 103 362, 2020.

[2] M. Kovac, M. Fuchs, A. Guignard, J.-C. Zufferey, and D. Floreano, "A miniature 7g jumping robot," in *2008 IEEE international conference on robotics and automation*, IEEE, 2008, pp. 373–378.

[3] J. Zhang, G. Song, Y. Li, G. Qiao, A. Song, and A. Wang, "A bio-inspired jumping robot: Modeling, simulation, design, and experimental results," *Mechatronics*, vol. 23, no. 8, pp. 1123–1140, 2013.

[4] V. Klemm, A. Morra, C. Salzmann, *et al.*, "Ascento: A two-wheeled jumping robot," in *2019 International conference on robotics and automation (ICRA)*, IEEE, 2019, pp. 7515–7521.

[5] MSI. "Vector 16 HX AI A2XWX - born for performance," Micro-Star International Co., Ltd. (2025), [Online]. Available: https://www.msi.com/Laptop/Vector-16-HX-AI-A2XWX/Specification (visited on 04/14/2025).

[6] E. DIN, "12100: Safety of machinery—general principles for design—risk assessment and risk reduction (iso 12100: 2010)," *German version EN ISO*, vol. 12100, 2011.