ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

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

May 18, 2025

Abstract

Complex and uneven terrain presents significant challenges to the locomotion systems of traditional robots. Some researchers have proposed equipping robots with jumping mechanisms to address this issue; however, existing jumping robots often suffer from drawbacks such as high cost, difficulty in miniaturization, structural complexity, and maintenance challenges. To address these limitations, this project introduces a novel, low-cost, and reliable bio-inspired jumping robot designed to overcome these issues by mimicking the jumping mechanism of fleas. Our system integrates an efficient spring-based energy storage module, an intelligent actuation and control mechanism incorporating a computer vision module and motors, and an innovative trigger-release system, ensuring powerful yet precisely controllable jumps within a compact form factor. The integrated system, which includes mechanical, control, and computer vision modules, aims to achieve multi-level jumping capabilities, precise jump force regulation guided by real-time visual feedback, and accurate 3D environmental perception for adaptive trajectory planning. This research aims to provide a versatile robotic platform with enhanced agility and maneuverability for operation in challenging real-world application environments.

Contents

1 Introduction **1**

2 Design **3**
2.1 Design Procedure 3
2.1.1 Mechenical Design 3
2.1.2 Embedded Design 3
2.1.3 Computer Vision Design 5
2.2 Design Details 5
2.2.1 Motion Subsystem 5
2.2.2 Control Subsystem 8
2.2.3 Computer Vision Subsystem 11

3 Verification **16**
3.1 Spring and Detachment Module 16
3.2 Gear Module 16
3.3 Embedded Control Module 18
3.4 Computer Vision Module 19

4 Costs **20**

5 Conclusions **21**

References **22**

1 Introduction

This project aims to design and implement a spring-powered jumping robot to explore the feasibility of using mechanical energy for impulsive vertical motion. Unlike traditional wheeled or tracked robots, a jumping robot stores and rapidly releases elastic potential energy to achieve lift-off. This mechanism offers advantages such as compact structure and rapid response, making it suitable for scenarios with constrained space or sudden mobility demands.

The system is divided into three main subsystems: the CV (Computer Vision and Communication), Control, and Motion modules. The CV module allows the user to send a jump command via Bluetooth. The control module, based on an ESP32 microcontroller, receives the command and drives a servo motor to preload the spring mechanism. Upon reaching the trigger angle, a half-gear mechanism releases the stored energy to initiate the jump. The motion module consists of a four-bar linkage, gear transmission, and two parallel torsional springs responsible for energy storage and mechanical release. The overall system architecture is illustrated in Figure 1.

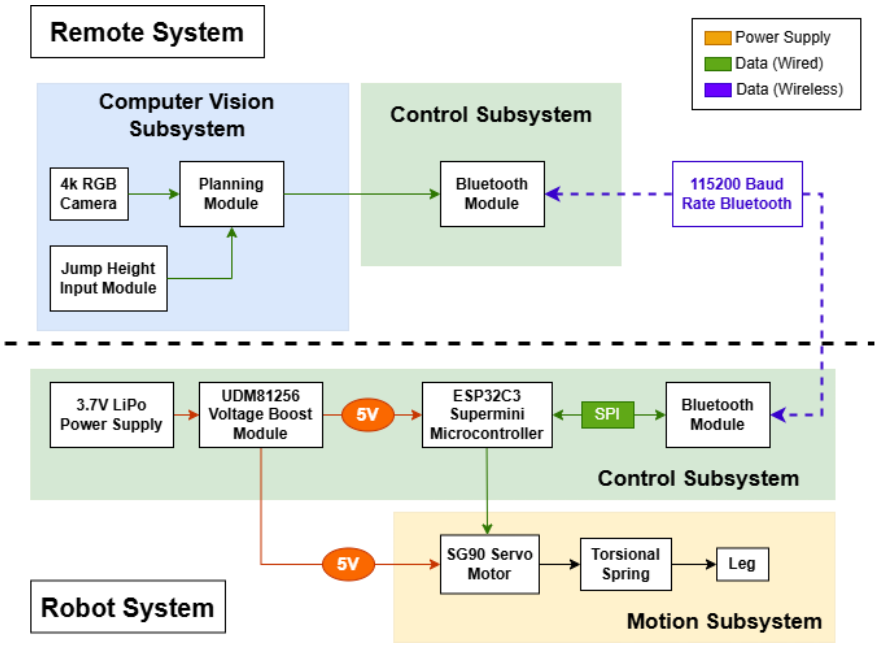

Figure 1: Block Diagram of our jumping robot.

The project is designed to meet the following performance requirements:

- The overall robot dimensions must not exceed $15 \times 15 \times 15\,\mathrm{cm}^3$.
- The system must support two levels of jump height, with at least one exceeding 10 cm.
- The jump must be triggered wirelessly via Bluetooth.

The significance of this project lies in its role as an experimental platform for validating the feasibility of spring-driven jumping mechanisms in compact robotic systems. While the current implementation remains a basic prototype with limited jump height and frequency, the design demonstrates the viability of performing mechanical jumps with minimal electronic control. In the future, this type of robot has potential applications in areas such as urban obstacle traversal, search-and-rescue, or even low-gravity mobility exploration. Our system establishes a practical foundation for further research on multi-jump stability, spring selection optimization, and integrated compliant mechanisms.

2 Design

2.1 Design Procedure

Discuss your design decisions for each block at the most general level: What alternative approaches to the design are possible, which was chosen, and why is it desirable?

Introduce the major design equations or other design tools used; show the general form of the circuits and describe their functions.

2.1.1 Mechenical Design

The mechanical structure of the jumping robot is based on a four-bar linkage mechanism. This configuration is chosen for its ability to convert rotational motion into linear displacement at the robot's foot, enabling vertical jumping.

The four-bar linkage consists of a body, two leg links, and a fixed base. The relative lengths were chosen to maximize vertical displacement while maintaining mechanical stability.

To power the jump, we used a torsional spring mounted at two base axes. Compared to conventional linear springs, torsional springs offer a more compact form factor and allow direct coupling with the rotating shaft.

For the release mechanism, a custom half-gear was designed to act as a passive detachment system. This gear has teeth only over a portion of its circumference. As the motor compresses the spring, it engages with a secondary gear. At a specific angular position, the teeth disappear, causing the gears to disengage automatically and releasing the stored energy. This eliminates the need for a separate actuator and significantly reduces system weight and detachment failure risks.

We also considered an adjustable jumping mechanism by implementing a worm gear and motor to change leg length dynamically. However, this approach introduced excessive mass and complexity. As a result, we opted for a simpler and lighter solution using manual sliding slots that allow for pre-jump leg length tuning to achieve different jump heights.

2.1.2 Embedded Design

Phase I: Stepper and Coreless Motor Prototype The first prototype was a two-motor design, utilizing a stepper motor to actuate the release and using a coreless brushed DC motor to compress the spring. The stepper motor used full-step drive mode, allowing for accurate position control and triggering the release part of the constraint. The coreless motor did not produce enough torque to compress the spring through the gear train, even when duty cycles were at the maximum. As a result, the cumulative weight and size of the two motors and supporting control electronics exceeded our limits for a lightweight mobile platform. In addition, in order to quickly test and iterate, we added an infrared control module to this generation of control system.

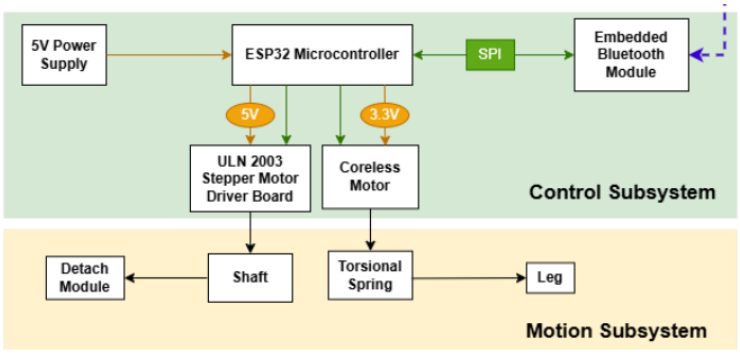

Figure 2: Block Diagram for Two Motor System

Phase II: Servo Motor and ESP32 Development Board To simplify the mechanical and control structures, the second iteration replaced the two motors with one continuous 360° SG90 servo. The servo offered bi-directional continuous rotation with ample torque to compress and release the spring and reduced the mechanical complexity significantly. To generate PWM signals and facilitate bluetooth communication with a host PC, an ESP32-WROOM development board was used. While this configuration worked properly and allowed for testing of control via bluetooth, the development board's large form factor and peripheral overhead were not acceptable for onboard use in the compact robot chassis.

Phase III: Servo Motor and ESP32-C3 SuperMini Integration The final design is still in progress. We used the ESP32-C3 SuperMini, a small and lightweight microcontroller module, to control the SG90 servo. This allowed us full integration of the embedded system onto the robot, while still maintaining full functionality of the BLE and PWM generation. The ESP32-C3 SuperMini was powered by a UDM81256 boost converter that stepped up the 3.7V output from a single cell (401119-100) LiPo battery to a steady 5V. This 5V rail powered both the servo and the ESP32-C3 board. We also designed a custom PCB, as shown in fig 3 to consolidate the boost converter, servo headers, battery connector, and ESP32-C3 interface. The EN pin of the UDM81256 was connected to a toggle switch so that power could be turned on and off conveniently.

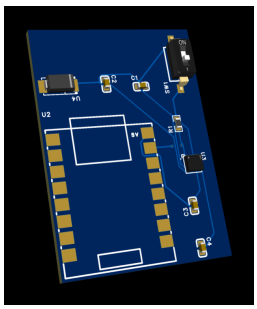

Figure 3: PCB Draft for Phase III

2.1.3 Computer Vision Design

The computer vision module of our jumping robot is based on a UHD (4k) RGB camera, a computing unit, and corresponding algorithm. Our choices were guided by the specific constraints of our small-scale jumping robot, the requirements for real-time performance, cost-effectiveness, and the nature of our controlled testing environment.

For the camera intended to measure distance between the robot and possible locations of jump destination, we chose an external high-resolution RGB camera connected to an off-board, high-powered computer that performs all vision processing. A sizable alternative we explored was a fully on-board system, with a miniaturized camera and processing unit onboard the robot. However, an on-board design would present significant problems. First, because our robot is lightweight for our intended purpose, an onboard camera and either a sufficiently powerful processor to process the camera feed or a wireless signal module with enough bandwidth for video signal would add considerable weight which could directly affect the jump. The off-board system keeps the robot un-encumbered. Secondly, integrating a high-performance vision system onto a small robot would significantly increase its complexity and cost, including challenges with power supply and heat dissipation. The external setup is simpler in these regards for the robot itself and can be more economical. Finally, real-time processing of high-resolution video streams demands significant computational resources, which would be difficult or costly to replicate in a miniaturized on-board form.

Regarding the method for 3D perception and robot pose estimation, we selected Aruco-based detection methods and a single RGB camera. Other possible approaches to 3D perception include stereo camera systems and LiDAR. Aruco markers [1] provide robust 3D pose information using just a single, standard RGB camera. This makes the system relatively inexpensive and straightforward to implement compared to alternatives, with sufficient accuracy for our navigation and jump control tasks.

2.2 Design Details

The overall layout and mechanical structure of the jumping robot are shown in Figure 4.

2.2.1 Motion Subsystem

Gear Transmission and Linkage Actuation To enable the jumping motion, we implemented a four-bar linkage system that transforms the motor's rotational motion into a vertical thrust.

The main purpose of using the four-bar linkage was to achieve a feasible and controllable jumping trajectory without relying on complex mechanisms. We used simulation tools to model the linkage motion and evaluate the resulting trajectory, ensuring that the output motion remained predominantly vertical. By adjusting the lengths and pivot positions of the links, we were able to control the initial launch angle, which in turn affected the

jumping height. This tunability allowed us to explore different configurations and select a geometry that met both the robot's size constraints and the desired performance.

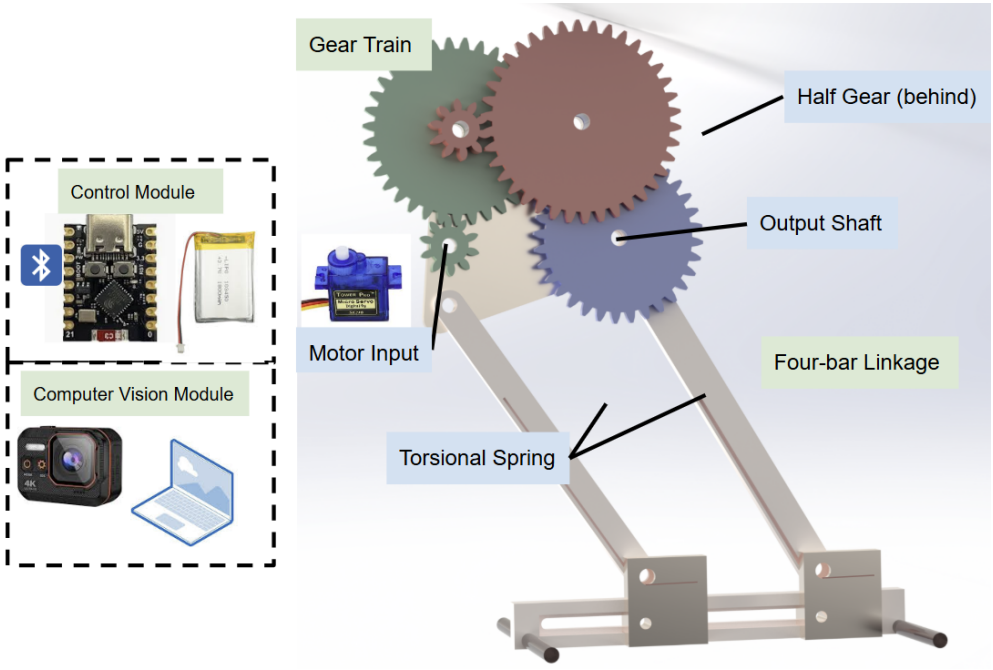

Figure 4: Physical design of the jumping robot.

The input to the linkage is driven by a motor connected to a compound gear train with a total reduction ratio of approximately $3.8 \times 3.8 \times 2 = 28.9$. This multi-stage reduction significantly lowers the motor's output speed while increasing the available torque, allowing sufficient force to preload the torsional spring before the jump. All linkage and gear components were 3D-printed and manually assembled.

Torsional Spring Selection The jumping mechanism relies on storing energy in torsional springs and releasing it rapidly. To estimate the minimum required energy for a successful jump, we apply the conservation of energy principle. The potential energy needed to lift a robot of mass $m = 30\ \mathrm{g} = 0.03\ \mathrm{kg}$ to a height of $h = 0.1\ \mathrm{m}$ is:

$$E = mgh = 0.03 \times 9.81 \times 0.1 = 0.0294\ \mathrm{J}$$

This energy must be provided by the torsional springs. The energy stored in a torsional spring is given by:

$$E = \frac{1}{2} k \theta^2$$

where k is the torsional stiffness (in $\mathrm{Nm/rad}$) and θ is the preload angle (in radians). Since we used two identical springs in parallel, the total energy becomes:

$$E_{\text{total}} = 2 \times \frac{1}{2} k\theta^2 = k\theta^2$$

Assuming a preload angle of $\theta = \frac{\pi}{2}$ rad, the required minimum spring stiffness becomes:

$$k \geq \frac{0.0294}{(\pi/2)^2} \approx 0.0119 \text{ Nm/rad}$$

In our prototype, we used two torsional springs made of spring steel, each with an outer diameter of 5.5 mm and a wire diameter of 1 mm. The mean coil diameter is approximately 4.5 mm. Each spring has 6 active turns. Using the standard torsional spring stiffness formula [2]:

$$k = \frac{d^4 G}{10.8 D N}$$

where $d = 1$ mm, $D = 5.5$ mm, $G = 79.3 \times 10^9$ Pa, and $N = 6$, we estimate:

$$k \approx \frac{(0.001)^4 \times 79.3 \times 10^9}{10.8 \times 0.0055 \times 6} \approx 0.2225 \text{ Nm/rad}$$

Each spring contributes approximately 0.2225 Nm/rad, giving a combined effective stiffness of about 0.45 Nm/rad, which is sufficiant for the 10cm jump.

Nonetheless, our physical testing showed that this spring configuration successfully launched a 30g robot to a height of approximately 10cm. This suggests that the theoretical estimation is conservative, and that real-world factors—such as rapid energy release, energy losses, and mechanical amplification—can compensate for lower spring stiffness.

The current design demonstrates that even relatively soft torsional springs can achieve functional jumping performance under appropriate mechanical conditions.

Motor Transmission and Integration To determine whether the selected motor is capable of compressing the torsional springs to the desired preload angle, we estimated the minimum required torque based on experimental measurements. Specifically, we measured the tangential force required to twist a single torsional spring by $90°$.

The total torque needed at the output shaft (i.e., at the spring interface) is given by:

$$\tau_{\text{spring}} = F \cdot r \cdot N$$

where $F = 6\,\text{N}$ is the experimentally measured tangential force, $r = 0.04\,\text{m}$ is the moment arm, $N = 2$ is the number of torsional springs in parallel.

Substituting the values:

$$\tau_{\text{spring}} = 6\,\text{N} \cdot 0.04\,\text{m} \cdot 2 = 0.48\,\text{N m}$$

Note that this torque is be delivered after the gear train. To calculate the corresponding required motor torque, we divide by the gear reduction ratio $G = 28.88$ and account for transmission inefficiency using a safety factor of $\frac{1}{0.8} = 1.25$:

$$\tau_{\text{motor}} = \frac{\tau_{\text{spring}}}{G \cdot \eta} = \frac{0.48}{28.88 \times 0.8} \approx 0.0208\,\text{N m}$$

Therefore, the motor must supply at least 0.0208 Nm of torque at its shaft to overcome mechanical losses and preload the springs to the desired angle. This value serves as the baseline for selecting an appropriate actuator and ensuring reliable operation under real-world conditions.

The servo under consideration weighs only 9 grams and provides a stall torque of 1.6 kg·cm (approximately 0.157 Nm). Its built-in position feedback and angle control make it well suited for use with the partial gear design.

2.2.2 Control Subsystem

The Embedded Control System coordinates communications and motion for the robot. We used an ESP32-C3 development board soldered onto a custom PCB which is then powered by a small 401119 LiPo battery. The control commands are sent to the ESP32-C3 microcontroller wirelessly over Bluetooth Low Energy (BLE) from the host device. The microcontroller processes the received commands and generates the PWM signals to drive the servo motor. All connection routing and power management is handled on the custom PCB.

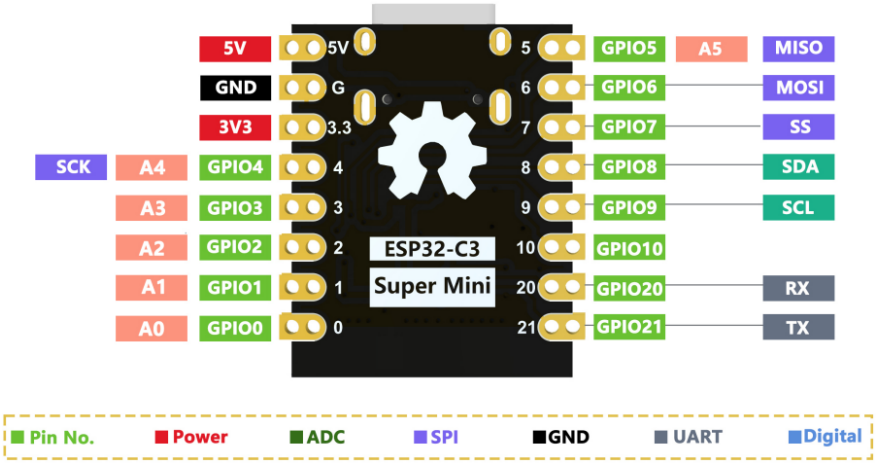

Figure 5: ESP32C3

Microcontroller The microcontroller, ESP32-C3 Supermini5 development board, 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.

To facilitate compact installation and autonomous operation, we designed a custom PCB 6 to connect the ESP32-C3 SuperMini to peripheral components. The PCB incorporates a boost converter module, a toggle switch, and connections to the servo, while providing a clean way to route power and deliver signals. It provides 3.7V regulated voltage from the 401119 LiPo battery to the servo through the boost module, while also consolidating all control and power paths into a single lightweight board.

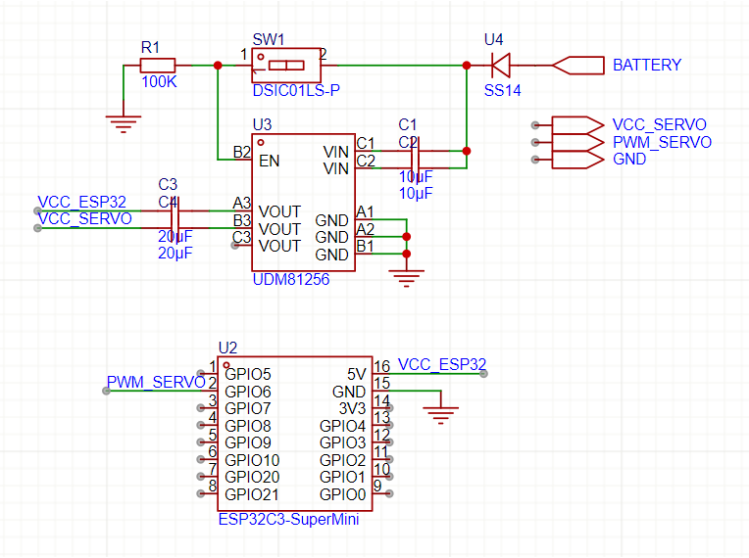

Figure 6: PCB Schematic

Bluetooth The Bluetooth communication channel serves as the wireless bridge between the host PC and the embedded control system. This module is logically divided into two parts: the embedded device side, implemented on the ESP32 microcontroller, and the host-side controller, which initiates and manages the communication session. Together, these subsystems form a low-latency control loop that supports command dispatch and feedback monitoring, as shown in Figure 7. The embedded Bluetooth Low Energy (BLE) module is implemented on the ESP32-C3 using the native BLE stack. The ESP32 acts as a BLE server that exposes a custom service with a writable characteristic. Upon initialization, the system configures a primary service and registers a characteristic that allows both read and write operations, serving as the main channel for remote command input.

The BLE interface is interrupt-driven. A subclass of `BLECharacteristicCallbacks` is defined to override the `onWrite()` function. Whenever the host sends a new command, the ESP32 immediately triggers this callback, which parses the received string

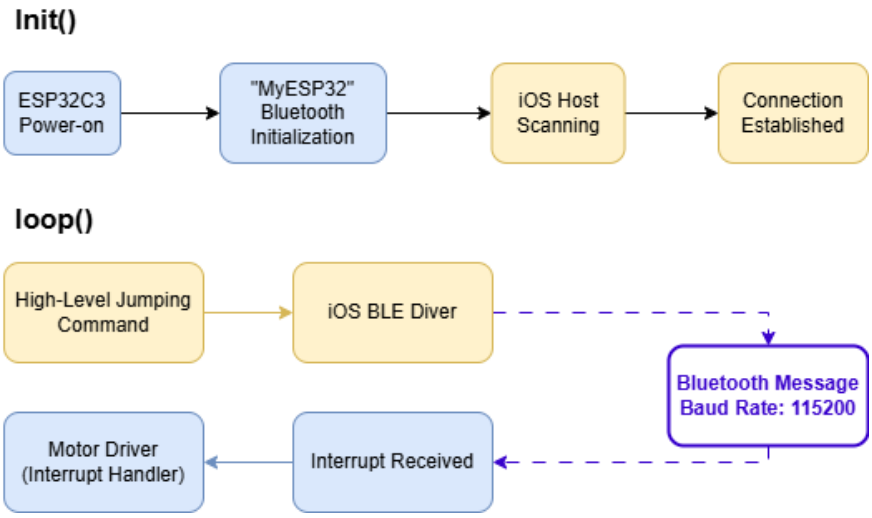

Figure 7: Bluetooth Module Processing Pipeline

(e.g., "+500" or "-800") and converts it into control actions for the servo motor, including rotation direction and duration.

To manage connection states, the ESP32 also registers a server callback to detect connect and disconnect events. When disconnection is detected, advertising is restarted automatically. In addition, a periodic "ping" string is sent to the host every 10 seconds using BLE notifications. This mechanism allows the host to verify that the connection remains active and facilitates lightweight keep-alive communication.

On the host side, a Bluetooth client (e.g., smartphone using *nRF Connect*) connects to the ESP32 device, discovers the BLE service, and writes UTF-8 encoded command strings to the characteristic. Commands such as "+500", "-400", or "0" represent forward, reverse, or stop signals for the servo motor respectively. The host can also receive periodic "ping" notifications, which provide feedback on connection health and device status.

Servo We selected the 360° version of the SG90 micro servo as the actuation component for the robot's jumping mechanism. This decision was motivated by multiple engineering trade-offs. The SG90 servo offers a better power-to-weight ratio than stepper motors in our application where weight is critical for the lightweight jumping robot design, even though they provide very good angular control, the stepper motoe is way too heavy. There was also consideration given to coreless DC motors, but they did not provide enough torque to reliably actuate the spring-loaded mechanical structure.

The 360° SG90 is a continuous rotation servo, which processes standard PWM signals as commands for speed and direction - not as commands for target angles. A PWM signal of 1.5ms pulse width results in zero speed (stop) and closer to 0.5ms pulse width maximum speed in one direction, and 2.5ms pulse width maximum speed in the opposite direction.

The ESP32-C3 microcontroller generates PWM actuators with a 50Hz signal using its

built-in LEDC (LED Controller) Peripheral. The duty cycle is modified in order to vary the pulse width. The pulse width can vary from a 0.5ms width to a 2.5ms width. The function rotate(speedPercentMaps) takes the appropriate input speed percentage (from -100% to +100%) mapping to the appropriate pulse width. In order to rotate the servo the necessary amount of time, the BLE handler will read the incoming commands (e.g. ”+500” or ”-800”) as direction and time. The sign signifies direction, and the magnitude signifies time in milliseconds.

This straightforward but functional control interface allows responsive and accurate actuation, while also reducing hardware complexity and software complexity. The servo is powered from the output of the UDM81256 boost converter, which provides a steady 5V output under dynamic load.

2.2.3 Computer Vision Subsystem

The Computer Vision Subsystem is responsible for providing real-time spatial awareness, enabling the robot to perceive its environment, determine its pose, and identify targets or obstacles. It comprises a high-resolution UHD RGB camera, a high-performance computing unit, and a specialized dynamics model and corresponding algorithm. The workflow of our module is shown in Figure 8.

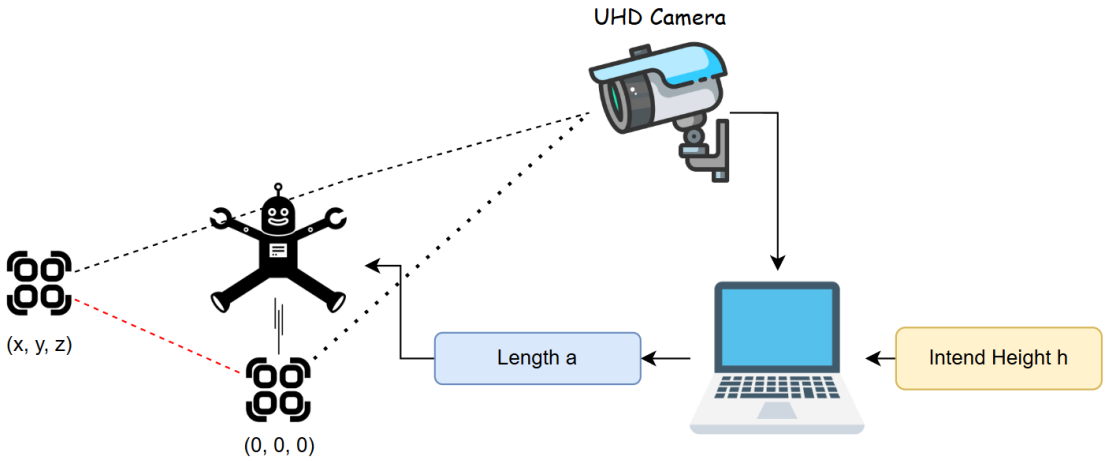

Figure 8: Workflow of our computer vision module.

Imaging Sensor (Camera) A UHD (3840x2160 resolution) RGB camera operating at 30 frames per second (Hz) serves as the imaging sensor. This camera provides detailed visual information necessary for detecting small features and markers from a distance. The high resolution and adequate frame rate are crucial for capturing clear images of the operational environment. We employ OpenCV to interface with this camera in real-time, capturing images for subsequent processing.

Camera Calibration To ensure accurate 3D distance detection, a rigorous camera calibration procedure is performed prior to deployment. We use a standard 5×7 checkerboard pattern, as exemplified in Figure 9. 40 images of this checkerboard are captured from various angles and distances relative to the camera. OpenCV library functions are then utilized to calculate the precise intrinsic camera parameters from these images. The calibration process solves for the camera's intrinsic matrix K and a set of distortion coefficients D. The intrinsic matrix typically takes the form:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where (f_x, f_y) are the focal lengths in pixels, and (c_x, c_y) is the optical center in pixels. The distortion coefficients $D = (k_1, k_2, p_1, p_2, k_3, \dots)$ model the lens's radial and tangential distortions. These parameters are subsequently used to undistort the captured images and are fundamental for accurately projecting 3D world points to 2D image coordinates and vice-versa, which is essential for 3D reconstruction tasks.

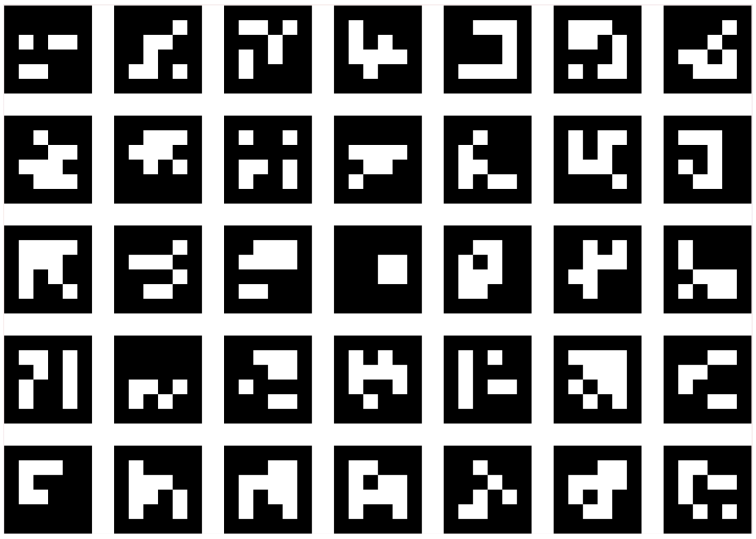

Figure 9: Our calibration board with 5x7 4×4 Aruco marker from the `DICT_4X4_250` dictionary. This figure also provides examples for the Aruco markers.

Aruco-based Distance Detection For robust and efficient tracking of the robot and environmental features, we employ 4×4 Aruco markers from the `DICT_4X4_250` dictionary provided by OpenCV [3]. These markers, with a physical size of 3cm×3cm, are strategically placed on the robot's body and at key locations within the test environment. The combination of the UHD camera and these distinct markers enables reliable detection at distances up to 3 meters. This marker system was chosen for its optimal balance between detection reliability, computational efficiency, and ease of implementation with standard vision libraries.

Image Processing and Computation The software processing utilizes OpenCV's ArUco module (version 4.5.0) to detect and decode markers. Adaptive thresholding techniques are employed within this module to robustly handle varying lighting conditions while maintaining a false positive detection rate below 0.1%. All computations are performed on a high-performance laptop equipped with an Intel Ultra 9 275HX CPU and an Nvidia RTX 5080 Laptop GPU [4], serving as the dedicated computing unit. To meet the real-time processing demands, specific optimizations were implemented by utilizing PyTorch and OpenCV libraries that are compatible with CUDA [5], thereby leveraging GPU acceleration for demanding tasks. This optimized pipeline successfully reduces the processing latency for each 4K resolution image to less than 100ms. The 3D distance D to a detected Aruco marker is then calculated using the pinhole camera model principles:

$$D = \frac{W_{real} \times f_{pixel}}{W_{pixel\_image}}$$

where W_{real} is the known physical width of the Aruco marker (e.g., 0.03m for the 3cm×3cm markers), f_{pixel} is the camera's focal length in pixels (obtained from calibration), and W_{pixel_image} is the marker's apparent width in the image in pixels.

Jump Parameter Calculation Based on the acquired 3D distance information, we calculates the optimal configuration for the robot's variable four-bar linkage to achieve the desired trajectory. The primary output is the length 'a' of the variable link (specifically, the bottom-most bar of the linkage), which must be set within a predefined operational range. This calculation relies on known robot parameters: the kinetic energy E_k available at launch (assumed constant per jump), the mass m of the robot (and thus its weight $W = mg$), and the lengths of the three fixed links of the four-bar mechanism (l_1, l_2, l_3). The initial launch velocity magnitude v_0 is determined from the kinetic energy:

$$v_0 = \sqrt{\frac{2E_k}{m}}$$

This v_0 forms the basis for subsequent trajectory calculations, assuming negligible air resistance. The relationship between the variable link length a and the resulting launch angle θ_{launch}, denoted as $\theta_{launch} = f_{linkage}(a, l_1, l_2, l_3)$, is crucial and is derived from a detailed kinematic model of the robot's specific four-bar linkage.

Scenario 1: Achieving a Target Jump Height In this scenario, the objective is to reach a specific jump height h, representing the apex of the parabolic trajectory. The target height h is provided as an input.

1. **Calculate Required Launch Angle (θ_0):** Given v_0 and the target height h, the required launch angle θ_0 (measured from the horizontal) is found using the projectile motion equation for maximum height:

$$h = \frac{v_0^2 \sin^2(\theta_0)}{2g}$$

Solving for θ_0:

$$\sin(\theta_0) = \frac{\sqrt{2gh}}{v_0}$$

$$\theta_0 = \arcsin\left(\frac{\sqrt{2gh}}{v_0}\right)$$

This calculation is valid if $v_0^2 \geq 2gh$; otherwise, the target height is unreachable with the given E_k.

2. **Determine Horizontal Range (x):** Once θ_0 is known, the corresponding horizontal range x for this jump can be calculated:

$$x = \frac{v_0^2 \sin(2\theta_0)}{g}$$

3. **Determine Variable Link Length (a):** The calculated θ_0 is the target launch angle. The appropriate variable link length a is then determined by finding an a such that $f_{linkage}(a, l_1, l_2, l_3) \approx \theta_0$. This may involve solving the inverse kinematics for the linkage or iterating through the permissible range of a to find the value that produces the launch angle closest to θ_0.

Scenario 2: Achieving a Target Horizontal Distance (x_{target}) over Two Jumps This scenario involves executing two distinct jumps to cover a specified horizontal distance. For each jump $j \in \{1, 2\}$, a target horizontal distance x_j is provided (e.g., from the CV system identifying segments of a path). The goal is to calculate a different variable link length a_j for each jump. The initial launch velocity v_0 (derived from E_k) is assumed to be the same for both jumps.

For each jump j:

1. **Calculate Required Launch Angle ($\theta_{0,j}$):** Given v_0 and the target horizontal distance for this jump x_j, the required launch angle $\theta_{0,j}$ is found using the projectile motion equation for range:

$$x_j = \frac{v_0^2 \sin(2\theta_{0,j})}{g}$$

Solving for $\theta_{0,j}$:

$$\sin(2\theta_{0,j}) = \frac{gx_j}{v_0^2}$$

$$\theta_{0,j} = \frac{1}{2}\arcsin\left(\frac{gx_j}{v_0^2}\right)$$

This calculation is valid if $v_0^2 \geq gx_j$. For a given $x_j < v_0^2/g$ (maximum range), two solutions for $\theta_{0,j}$ exist (a low and a high trajectory). Typically, one is chosen based on criteria like obstacle clearance or energy efficiency (often the lower angle, $\theta_{0,j} < 45°$).

2. **Determine Corresponding Jump Height (h_j):** The peak height h_j for this jump trajectory is:
$$h_j = \frac{v_0^2 \sin^2(\theta_{0,j})}{2g}$$

3. **Iterative Determination of Variable Link Length (a_j):** To find the optimal variable link length a_j that produces the target launch angle $\theta_{0,j}$, an iterative approach is employed. The length a is varied within its allowed operational range (e.g., from a_{min} to a_{max}) in discrete steps (e.g., $\Delta a = 0.001\,\text{m}$ or 0.1 cm). For each candidate length a_k in this iteration:
   - The resulting launch angle $\theta_{launch}(a_k) = f_{linkage}(a_k, l_1, l_2, l_3)$ is calculated using the linkage's kinematic model.
   - The difference $|\theta_{launch}(a_k) - \theta_{0,j}|$ is evaluated.

   The value a_k that minimizes this difference (i.e., produces a launch angle closest to the required $\theta_{0,j}$) is selected as the optimal a_j for that jump. This process is repeated to find a_1 and a_2 for the two jumps.

This systematic calculation ensures that the robot's jumps are tailored to the specific requirements of the terrain and obstacles identified by the vision system.

3 Verification

3.1 Spring and Detachment Module

To enable jumping, the torsional spring must be capable of storing and releasing sufficient torque to actuate the four-bar linkage. Additionally, a passive detachment mechanism ensures the spring disengages cleanly at the correct moment, enabling rapid energy release.

Requirement	**Verification**
The torsional springs must output sufficient torque ($\geq 0.35\,\mathrm{N\,m}$) when twisted from 120° to 30° to actuate the linkage mechanism for jumping.	A. The torsional spring was mounted to a shaft with a known moment arm ($0.04\,\mathrm{m}$). B. The spring was rotated to 90°, and weights were added until equilibrium. C. Measured force was $6\,\mathrm{N}$, resulting in torque: $$T = F \cdot r = 6\,\mathrm{N} \cdot 0.04\,\mathrm{m} = 0.24\,\mathrm{N\,m}$$ D. The spring was then released and successfully actuated the four-bar linkage through its full stroke. **Result: Requirement met.**
The detachment mechanism must disengage automatically at full compression to release stored energy instantly.	A. A custom half-gear was used as a passive release mechanism. B. Through testing, the optimal tooth coverage was found to be $\frac{7}{16}$ of the full gear circumference. C. This profile keeps the gear engaged from 30° to 120°, and ensures disengagement occurs precisely at the maximum preload angle. D. Experimental trials showed consistent detachment at the desired angular position with no failure or delay. **Result: Requirement met.**

Table 1: Spring and Detachment Module Requirements and Verification

3.2 Gear Module

The gear train connects the motor to the leg of the robot which compress the torsional spring. It amplifies the motor torque to the required level for spring actuation. It must ensure torque transmission efficiency and smooth operation without mechanical issues

such as jamming or excessive backlash.

Requirement	**Verification**
The gear train must provide sufficient torque amplification for the motor to preload the torsional spring. The designed gear ratio is 28.88:1.	A. The gear train was assembled with a total reduction ratio of 28.88:1 to amplify motor torque. B. Under test, the motor was able to rotate the spring from 30° to 120°, confirming sufficient torque transfer. C. No additional gearing or motor stall was observed, indicating the current configuration is adequate for spring loading. **Result: Requirement met.**
The gear train must mesh smoothly without jamming, abnormal noise, or significant backlash.	A. During continuous actuation cycles, gear motion was observed visually and acoustically. B. No instances of gear jamming or abnormal gear noise were detected. C. Backlash was minimal and did not affect system performance. D. The system completed multiple preload and release cycles with consistent behavior. **Result: Requirement met.**

Table 2: Gear Module Requirements and Verification

3.3 Embedded Control Module

Requirement	Verification
The embedded system must reliably generate PWM signals to control SG90, with direction and duration specified via bluetooth commands.	A. The system successfully generated 50 Hz PWM signals with pulse widths ranging from 1.0 ms to 2.0 ms using microcontroller's internal timer. B. The servo is enabled to rotate continuously in both clockwise and counterclockwise directions for arbitrary durations as specified by the input commands. C. Rotation duration matched command inputs within $\pm$500 ms tolerance over 5 trials. **Result: Requirement met.**
The system must establish and maintain BLE connection with the host device for at least 30 seconds under normal operating conditions.	A. BLE module initialized successfully within 3s after power-up. "MyESP32" shoud be observable from the host side. B. Connection with smartphone was sustained continuously for over 3 minutes in a static environment. **Result: Requirement met.**
The servo motor must provide sufficient torque to preload the torsional spring used in the leg actuation.	A. The SG90 servo was mounted with mechanical linkage connected to the spring preload mechanism. B. When commanded to rotate, the servo was able to compress the torsional spring from rest to its full preload position. **Result: Requirement met.**

Table 3: Control Subsystem Requirements and Verification

3.4 Computer Vision Module

Requirement	**Verification**
The UHD camera must capture frames at 30 Hz with a resolution of 3840×2160 to ensure sufficient visual detail.	A. The system successfully connected the camera. B. The system is able to collect consecutive frames using OpenCV at 30 fps, 3840x2160 resolution each frame.
Camera intrinsic parameters must be precisely calibrated for accurate 3D reconstruction.	A. The system successfully uses OpenCV calibration functions to compute reprojection error. B. The reprojection error is under 3.
ArUco markers (3cm×3cm) must be detected up to a distance of 3 meters with a false positive rate less than 0.1%.	A. The false positive rate is 0.057%.
Vision processing must maintain latency below 100 ms per frame.	A. The end-to-end latency is 87ms.
The system must successfully calculate the length a with a error in 10%.	A. We prove it by comparing the landing or highest location of the robot with the input height or distance successfuly for three times.

Table 4: Vision Subsystem Requirements and Verification

4 Costs

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 5.

Table 5: 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 6.

Table 6: Parts & Materials Cost

Component	**Manufacturer**	**Description**	**Qty**	**Unit Price ($)**	**Total ($)**
ESP32C3-Supermini	Espressif	WiFi/BLE Development Board	2	2.15	4.30
SG90 Servo Motor	-	360° Mini Servo (9g)	2	0.82	1.64
401119-100 LiPo Battery	-	3.7V Mini Battery	2	0.66	1.32
Torsional Spring	Misumi	Torsion Spring (1.0 mm wire)	10	0.30	3.00
4K Camera	Hengqiaotong	USB 4K Webcam with Autofocus	1	32.44	32.44
				Total Parts	**42.70**

The grand total cost is summarized as $ 8447.70 in total.

5 Conclusions

This project addressed some challenge of robotic movement in complex terrains by designing and conceptually validating a novel, flea-inspired jumping robot aimed at providing a low-cost, reliable, and highly agile solution. Althought due to cost and time constriants, The robot didn't completely meets its core objectives, it still demonstrates multi-level jumping capabilities, real-time 3D distance estimation, and self-adaptive parameters calculation. While this work establishes a strong foundation, future efforts may focus on extending modification to achieve the original objectives.

For ethical consideration, our camera system continues to comply strictly with privacy-preserving protocols and all testing takes place in a controlled environment with clear notifications. Thus these methods, we keep our robot safe and ethical, following IEEE Code: paramount importance must be given to the safety, health, and welfare of the public.

The wider impact of this project and similar progress in agile robotics will be multifaceted. On a global scale, this type of technology can improve capabilities for disaster response, remote inspection of infrastructure, and environmental monitoring in areas never considered before. This is valuable in regards to saving lives and resources. Also, there is the potential for economic impacts through innovation in specialized robotics types, which can create new markets and application spaces. We feel that what we have done here can stimulate ideas for future work.

References

[1] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[2] J. J. Uicker, J. J. Uicker Jr, G. R. Pennock, and J. E. Shigley, *Theory of machines and mechanisms*. Cambridge University Press, 2023.

[3] G. Bradski, A. Kaehler, *et al.*, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, no. 2, 2000.

[4] 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).

[5] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.