ECE 445

Senior Design Laboratory

Final Report

JengaBot: A Robotic System for Playing Jenga with a Human

Team #11

Peiran Wei
(pw19@illinois.edu)
Wangyihan Guo
(wguo12@illinois.edu)
Jiacheng Ye
(ye32@illinois.edu)
Hengtie Zhu
(hengtie2@illinois.edu)

Supervisor: Pavel Loskot
TA: Yue Yu

May 2026

Abstract

JengaBot is an autonomous robotic system designed to play Jenga interactively against a human opponent. The system integrates a custom 3D-printed structural frame, a three-axis Cartesian motion platform, a hybrid push-and-grip end-effector, and a four-camera vision subsystem to perceive, decide, and act within a constrained workspace. A Raspberry Pi 5 performs high-level image processing and strategy computation, while an Arduino Uno R3 handles real-time motor control via TMC2209 stepper drivers. According to the Jenga rules, the top 2 layer of bricks shouldn't be moved at any time, so only 24 bricks are available to move. Each of the 24 Jenga blocks carries 10 AprilTags (tag36h11), enabling multi-view pose estimation and a MuJoCo-based digital twin for tower-state reconstruction. The system achieves repeatable positioning accuracy of $\pm 1.0\,\text{mm}$ over a $100\,\text{mm} \times 300\,\text{mm} \times 120\,\text{mm}$ workspace and completes a full manipulation cycle within $15\,\text{s}$. Over 10 test games, JengaBot achieves at least 3 successful block extractions per game with a $\geq 70\%$ game completion rate under standard tower conditions.

Contents

1 Introduction

This report presents JengaBot, an autonomous robotic system that plays the tabletop game Jenga against a human opponent. The project demonstrates precise object manipulation in a confined workspace by combining 3D-printed structural components, a multi-camera vision system, and a three-axis motion platform.

1.1 Problem Statement

While technologies such as 3D printing and embodied intelligence have progressed significantly, precise object manipulation in constrained spaces continues to be a complex problem. Conventional robotic arms are too bulky for tight workspaces, yet such spaces are well-suited for customized 3D-printed frameworks. This project explores the feasibility of combining compact 3D-printed structural frames with precision robotic mechanisms to perform delicate manipulation tasks within confined volumes.

1.2 Solution Overview

To address this spatial limitation, we propose JengaBot: a system that integrates a custom 3D-printed enclosure with a three-axis motion control platform and a specialized end-effector. Four OV9732 cameras mounted at the corners of the frame capture the real-time state of the Jenga tower from multiple angles. A Raspberry Pi 5 processes these images, constructs a digital model of the tower using a MuJoCo simulation, selects the optimal block to remove via a predefined algorithmic strategy, and commands the three-axis system to execute the move. An Arduino Uno R3 serves as the low-level real-time motion controller, receiving target coordinates from the Raspberry Pi and generating STEP/DIR signals for TMC2209 stepper motor drivers.

The system consists of five major subsystems: (1) the vision subsystem (cameras and AprilTag detection), (2) the three-axis motion control system, (3) the operation unit (end-effector), (4) the central controller (Raspberry Pi 5 and strategy algorithm), and (5) the power supply. Fig. 1 shows the system block diagram with data and power connections among these subsystems. Fig. 2 shows the CAD model of the integrated mechanical assembly.

Figure 1: System block diagram showing data and power connections between the vision, processing, motion, and power subsystems.

Figure 2: CAD model of the three-axis motion platform and end-effector arrangement within the 3D-printed frame.

1.3 High-Level Requirements

The system is designed to meet the following high-level performance requirements:

1. Gameplay Completeness: The system shall autonomously complete the core Jenga interaction cycle—detecting the tower state, selecting a target block, moving to the target, extracting the block, and placing it on the top layer—for at least 3 successful block operations per game without causing tower collapse due to robot-induced error. Over 10 test games, the system shall achieve a successful game completion rate of at least 70% under standard tower conditions.
2. Motion Precision: The three-axis motion platform shall achieve a repeatable positioning accuracy of $\pm 1.0\,\text{mm}$ within the full operating workspace of approximately $300\,\text{mm} \times 300\,\text{mm} \times 250\,\text{mm}$, which is sufficient for targeting individual Jenga blocks ($30\,\text{mm} \times 30\,\text{mm} \times 90\,\text{mm}$) and aligning the end-effector with the selected extraction face.
3. Response Time: The system shall complete one full manipulation cycle, including image capture, tower-state update, target selection, motion execution, block extraction, and top placement, within $15\,\text{s}$ under normal operating conditions.

1.4 Block-Level Changes from Proposal

Since the original proposal, the mechanical subsystem has undergone a significant revision. Instead of fabricating the entire motion base from scratch, the team purchased a commercial two-dimensional linear motion platform to serve as the X-Y stage. A vertical linear module is mounted on the moving platform to provide Z-axis motion, forming a compact three-axis Cartesian gantry. This change reduces fabrication risk, improves rail alignment and mechanical stiffness, and allows the team to focus on end-effector integration, motor control, and system-level testing. All other subsystems remain consistent with the proposed architecture.

2 Design

The JengaBot system is organized into five subsystems: vision, motion control, end-effector operation, central controller, and power supply. This section describes the design rationale and detailed implementation of each.

2.1 Design Procedure

The system architecture follows a perception–decision–action pipeline common to autonomous robotic systems. Several alternative approaches were considered at the architectural level:

Centralized vs. distributed control: A fully centralized architecture with the Raspberry Pi handling both high-level computation and real-time motor pulse generation was considered. This was rejected because Linux-based systems introduce timing jitter in pulse trains, which degrades stepper motor performance. Instead, a split-controller architecture was adopted, with the Raspberry Pi 5 handling perception and strategy and an Arduino Uno R3 generating deterministic STEP/DIR signals.

End-effector design: A purely suction-based gripper was considered for block extraction but was rejected because the wooden Jenga blocks have porous surfaces that reduce vacuum effectiveness. A purely pushing mechanism was also considered but cannot transport blocks to the top layer. The hybrid push-and-grip design separates extraction (push) from transport (grip), improving mechanical reliability and reducing tower disturbance.

Vision approach: A depth-camera-based approach (e.g., Intel RealSense) was considered for tower-state reconstruction but was rejected in favor of AprilTag-based fiducial markers because tags provide unambiguous ID information that directly maps to individual blocks, eliminating the need for complex point-cloud segmentation.

2.2 Design Details

2.2.1 Vision Subsystem

Four OV9732 cameras (720p, 30 fps; operating at 1280×720) are mounted at the four corners of the 3D-printed frame and connected to a Raspberry Pi 5 via a USB hub. Each camera provides an independent viewpoint of the Jenga tower at a working distance of approximately 20 cm.

Each Jenga block carries 10 AprilTags (tag36h11 family), with a total of 300 tags coded from 1 to 300 across the 30-block set. Tag placement follows the convention defined in Table 1: 2 end-face tags and 8 side-face tags per block, distributed to maximize visibility from multiple camera angles. The tags are generated as PNG textures and mapped onto block faces in the MuJoCo simulation.

Each camera runs an independent AprilTag detection pipeline using the apriltag Python library (quad_decimate=2.0, nthreads=2), identifying each tag's ID, pixel center, and four corner coordinates. Detection results are served via a Flask HTTP API on each camera stream.

Table 1: AprilTag placement convention for each Jenga block.

Index	Suffix	Face	Relative Position
0	end_pz	+Z end	$(0, 0, +\text{END\_Z})$
1	end_nz	−Z end	$(0, 0, -\text{END\_Z})$
2–3	x_head, x_tail	+X side	$(\pm\text{SIDE\_XY}, 0, \pm 1.0)$
4–5	nx_head, nx_tail	−X side	$(\mp\text{SIDE\_XY}, 0, \pm 1.0)$
6–7	y_head, y_tail	+Y side	$(0, \pm\text{SIDE\_XY}, \pm 1.0)$
8–9	ny_head, ny_tail	−Y side	$(0, \mp\text{SIDE\_XY}, \pm 1.0)$

For calibration, four fixed AprilTags (IDs 301–304) are mounted on the bottom turntable surface. At system startup, each camera captures a fixed number of images of these reference tags and automatically computes its intrinsic matrix (K-matrix) and extrinsic pose, ensuring robustness against minor camera displacement.

The system uses tag ID presence/absence across all four camera views to determine which blocks remain in the tower and at which layer. This information is fed into a MuJoCo simulation that maintains a real-time digital twin of the tower state, enabling the strategy algorithm to evaluate structural stability and select the optimal block for removal.

2.2.2 Three-Axis Motion Control System

The motion subsystem uses a purchased two-dimensional linear motion platform as the X-Y stage, combined with a vertically mounted linear module to form the full three-axis Cartesian gantry. The X-Y platform provides horizontal positioning around the Jenga tower, while the Z-axis adjusts the height of the end-effector relative to the selected block layer.

Each axis is driven by a NEMA17 stepper motor controlled by a TMC2209 driver module operating at 16× microstepping. A custom PCB was designed around the TMC2209 module to simplify wiring and improve integration. The PCB provides a 24 V motor power input, motor terminal outputs, logic power input, and STEP/DIR control terminals. A local 220 µF capacitor is included on the motor supply side to improve supply stability during current transients.

2.2.3 Operation Unit (End-Effector)

The end-effector adopts a hybrid push-and-grip design mounted on the Z-axis carriage. It includes a slender pushing element for laterally displacing a selected Jenga block and a compact two-sided gripping mechanism for securing the partially extracted block and transporting it to the top of the tower.

A dedicated micro electric push rod drives the pushing motion, while a separate small actuator drives the gripping mechanism. This avoids the complexity of pneumatic systems.

During operation, the end-effector first aligns the push rod with the centerline of the target block and applies a controlled lateral push to partially extract the block. Once sufficient block length is exposed, the gripping mechanism closes on the block and secures it for transport.

The current design uses an open-loop force strategy based on calibrated displacement, limited motor current, and controlled speed, rather than active closed-loop force sensing. To reduce the risk of toppling the tower, the system employs a low-speed initial push, restricts the maximum commanded push distance per attempt, and avoids sudden acceleration. If resistance exceeds the expected range, the controller can stop the motion and reattempt or select another block.

2.2.4 Central Controller

The central controller is split between a Raspberry Pi 5 and an Arduino Uno R3. The Raspberry Pi receives multi-camera image data, runs the full perception-to-action pipeline (tag detection, tower-state reconstruction, strategy computation), and transmits motion commands to the Arduino via USB serial. The Arduino handles deterministic low-level motor control by generating STEP/DIR signals and reporting execution status back to the Raspberry Pi.

This split-controller architecture isolates high-level processing from low-level pulse generation, ensuring that motor execution remains functional during high CPU usage on the Raspberry Pi. The controller also manages game-flow logic: detecting the human opponent's move completion, handling rule violations, and signaling turn transitions.

2.2.5 Power Supply

The motor drive stage is powered by a single 24 V / 6 A switching power supply, which powers the TMC2209 driver boards and associated stepper motors. A dedicated bulk capacitor on each custom driver PCB suppresses transient voltage fluctuations caused by motor current changes.

2.3 Tolerance Analysis

Two critical mechanical questions were evaluated analytically.

Positioning Accuracy: Each Jenga block has nominal dimensions of 30 mm × 30 mm × 90 mm. For a standard NEMA17 stepper motor (1.8° step angle, 200 steps/rev) with 16× microstepping and an estimated 40 mm axis travel per revolution, the theoretical linear resolution is:

$$\text{Linear resolution} = \frac{40 \text{ mm/rev}}{200 \times 16 \text{ microsteps/rev}} = 0.0125 \text{ mm/microstep} \tag{1}$$

This is far finer than the required ±1.0 mm system-level tolerance. In practice, positioning error is dominated by mechanical factors (frame flexibility, backlash, rail alignment, print

tolerance), making the $\pm 1.0\,\text{mm}$ target appropriate. Relative to the 30 mm block width, this error is only 3.3%, leaving sufficient margin for aligning the pusher or gripper with the central 10–15 mm of the block face.

Push Force Feasibility: The required extraction force depends on block-to-block friction. A simplified Coulomb model gives:

$$F = \mu N \tag{2}$$

where μ is the effective static friction coefficient between wooden blocks and N is the normal force from upper layers. The end-effector is designed to provide a controlled push force of up to 5 N, intentionally higher than typical extraction requirements while low enough to avoid aggressive impact loading. Conservative motion planning (low-speed push, limited stroke, reattempt logic) further reduces tower disturbance risk.

The tolerance analysis confirms that the motion subsystem has sufficient positioning resolution and that the push force capability is adequate for prototype-level Jenga manipulation.

3 Verification

This section discusses the testing of the completed JengaBot system and its major subsystems. The full Requirement and Verification Table is provided in Appendix B. The discussion below focuses on the main verification findings and any requirements that were not met or required special attention.

3.1 Vision Subsystem Verification

The four-camera vision subsystem was tested against its four requirements. Frame-rate testing over 1,000 consecutive frames confirmed an average frame interval of ≤ 30 ms, meeting the ≥ 30 fps target. Tag detection accuracy was evaluated by placing a fully tagged tower and comparing detected tag IDs against ground truth across 500 frames per camera; each camera achieved $\geq 90\%$ detection rate at the 20 cm working distance.

Multi-camera coverage testing was conducted on an intact 30-block tower over 100 cycles. The union of tag IDs detected across all four cameras included $\geq 99\%$ of the 300 tags. End-to-end latency measurements from image capture to MuJoCo tower-state update, taken over 500 cycles, showed a 95th-percentile latency of ≤ 50 ms.

3.2 Motion Control System Verification

The three-axis motion platform was tested for positioning accuracy, travel speed, and structural deflection. Repeatability testing at reference positions across the workspace confirmed positioning accuracy of ± 1.0 mm over 10 trials. Full-range traversal speed measurements over 5 trials verified an average speed of ≥ 80 mm/s on the X- and Y-axes without missed steps.

Payload deflection testing at 3 representative positions within the workspace, with the nominal 0.8 kg moving payload mounted, confirmed static deflection below 2.0 mm at the end-effector relative to the unloaded baseline.

3.3 End-Effector Verification

Bench-test force measurements over 5 trials confirmed that the push rod delivers a controlled lateral force of up to 5 N with repeatable operation within ± 1 N. Grip-and-transport testing over 10 cycles (pick-up, short-distance transport, release) showed no visible slippage or dropping. Timed functional testing over 5 trials confirmed an average push–extract–grip–place cycle time within 8 s after reaching the target position.

3.4 Central Controller Verification

USB serial communication testing over 30 minutes of continuous operation confirmed zero communication failures or unhandled timeouts between the Raspberry Pi 5 and Arduino Uno R3. The controller successfully generated valid command sequences for a range of

representative tower-state inputs. Isolation testing verified that the Arduino continues to execute and report motion correctly during high CPU usage on the Raspberry Pi.

3.5 Power Supply Verification

The motor power supply delivered a stable 24 V at the PCB input terminals during motor operation. The driver PCB correctly received STEP/DIR control signals and produced corresponding motor motion. Repeated start–stop motion tests confirmed no reset, missed motion, or abnormal driver shutdown occurred.

3.6 System-Level Verification

The three high-level requirements were assessed through integrated testing. Gameplay completeness was evaluated over 10 test games; the system achieved ≥ 3 successful block operations per game without robot-induced tower collapse, and a game completion rate of $\geq 70\%$. Motion precision of ± 1.0 mm was confirmed across the full 300 mm $\times$ 300 mm $\times$ 250 mm workspace. Full manipulation cycle time remained within 15 s under normal operating conditions. All quantitative results are documented in Appendix B.

4 Costs

This section presents cost estimates for the JengaBot prototype, including labor, parts, and a discussion of potential mass-production costs.

4.1 Labor Costs

Labor cost estimates follow the ECE 445 formula:

$$\text{Labor Cost} = \text{Ideal Salary (hourly rate)} \times \text{Actual Hours Spent} \times 2.5 \tag{3}$$

Assuming an ideal hourly rate of \$50.00 for each team member and an estimated 150 hours per member over the semester, the total labor cost is:

$$\text{Total Labor} = 20.00/\text{h} \times 80\ \text{h} \times 2.5 = 4000 \tag{4}$$

4.2 Parts Costs

Table 2 lists all major components, their quantities, unit costs, and total costs. The overall parts total is approximately \$3,502.80.

Table 2: Parts and materials cost breakdown for the JengaBot prototype.

Component	Qty	Unit Cost (RMB)	Total (RMB)
Arduino Uno R3	1	15.00	15.00
TMC2209 Driver Module	4	8.00	32.00
Custom Driver PCB	4	5.00	20.00
24 V Switching Power Supply (6 A)	1	25.00	25.00
USB cables / interconnect wiring	1	10.00	10.00
Connectors / terminal blocks / passives	1	15.00	15.00
Micro electric push rod	1	145.80	145.80
2D motion platform (X-Y stage)	1	2,500.00	2,500.00
Custom Jenga blocks (30 pcs)	30	15.00	450.00
OV9732 cameras (720p, 30 fps)	4	60.00	240.00
Total			3,452.80

The largest single cost is the purchased 2D motion platform (2,500RMB), which was chosen over a custom 3D-printed alternative to reduce fabrication risk and improve mechanical

reliability. The micro electric push rod (145.80RMB) and custom Jenga blocks (450.00RMB) are the next largest expenses.

4.3 Mass-Production Cost Estimate

If the JengaBot were commercially viable, mass-production would significantly reduce per-unit costs. The 2D motion platform, currently the dominant expense at retail pricing, could be replaced by injection-molded components and bulk-purchased linear rails for an estimated \$300–\$500 per unit at scale. Custom driver PCBs at volume (1,000+ units) would cost approximately \$2.00 each. The Raspberry Pi 5 could be replaced by a lower-cost embedded processor. A rough estimate for mass-production parts cost is \$600–\$800 per unit, representing a 75–80% reduction from the prototype cost.

5 Conclusions

The JengaBot project successfully demonstrated that a compact, 3D-printed robotic system can autonomously play Jenga against a human opponent. This section summarizes accomplishments, identifies remaining uncertainties, and addresses ethical, safety, and broader-impact considerations.

5.1 Accomplishments

The completed JengaBot prototype integrates five subsystems—vision, motion control, end-effector, central controller, and power supply—into a functional autonomous game-playing robot. Key accomplishments include:

- A four-camera AprilTag-based vision system achieving $\geq 99\%$ tag coverage on an intact tower with end-to-end latency $\leq 50\,\text{ms}$.
- A three-axis Cartesian gantry with $\pm 1.0\,\text{mm}$ repeatable positioning accuracy over a $300\,\text{mm} \times 300\,\text{mm} \times 250\,\text{mm}$ workspace.
- A hybrid push-and-grip end-effector capable of $5\,\text{N}$ controlled push force and reliable block transport.
- A split Raspberry Pi / Arduino control architecture that isolates high-level computation from real-time motor control.
- System-level demonstration of ≥ 3 successful block operations per game with $\geq 70\%$ game completion rate and $\leq 15\,\text{s}$ cycle time.

5.2 Uncertainties and Future Work

Several uncertainties remain. The current open-loop force control strategy relies on calibrated displacement and current limiting rather than closed-loop force feedback. Under unusual friction conditions (e.g., humidity-swollen blocks or tightly wedged pieces), the system may fail to extract a block or may require multiple reattempts. Integrating a low-cost force sensor on the push rod would enable adaptive force control and improve robustness.

The vision system depends on consistent lighting conditions. Strong ambient light changes or shadows can reduce AprilTag detection rates. Future work could incorporate active illumination or a more robust detection pipeline.

The structural frame is currently 3D-printed PLA, which may creep under sustained load or elevated temperatures. For long-term reliability, aluminum extrusion framing or reinforced prints would be preferable.

Finally, the current strategy algorithm uses a deterministic rule-based approach. A machine-learning-based strategy trained in simulation could potentially discover more optimal block-selection policies.

5.3 Ethical Considerations

A suitable ethical framework for JengaBot is the ACM Code of Ethics and Professional Conduct [1], which states that computing professionals should contribute to society and human well-being, avoid harm, be fair, respect privacy, and give proper consideration to the public good.

Privacy and image handling. Since the system uses multiple cameras to observe the Jenga tower and nearby gameplay area, there is a possibility of capturing images of human participants. To reduce privacy concerns, the system only collects images needed for tower-state recognition, avoids unnecessary storage of personal images, and restricts recorded data to project development or demonstration use.

Responsible autonomous interaction. JengaBot moves mechanical parts near human users. The team prioritized conservative motion behavior, predictable operation, and safe testing procedures. The ethical goal is not simply to make the robot work, but to ensure it does not create unreasonable risk during operation.

Fairness and accessibility. The robot's actions are observable and the gameplay process does not rely on hidden intervention. Users can understand when the robot is sensing, moving, or waiting, which improves fairness in human–machine interaction.

5.4 Safety

The JengaBot prototype includes moving gantry axes, an end-effector, electrical power components, and a 3D-printed frame. Safety considerations address both mechanical and electrical hazards in accordance with OSHA machine-guarding requirements [2].

Pinch hazards. The linear axes, carriage interfaces, and end-effector create potential pinch points during motion. Moving regions are visually identified, human access to the active workspace is limited during operation, and the robot runs at conservative speeds during testing and demonstrations.

Electrical safety. The 24 V switching power supply, motor drivers, and control boards use insulated wiring, secure terminal connections, strain relief for cables, and enclosure of exposed conductors, consistent with the safety principles of IEC/UL 62368-1 [3].

Emergency stop. The robot includes an accessible emergency power-cutoff so that motion can be stopped immediately if unsafe behavior occurs.

Structural integrity. Printed parts are designed with sufficient wall thickness and reinforced mounting locations. Before full operation, the frame is checked for looseness, visible deformation, and alignment drift.

5.5 Broader Impacts

The JengaBot project addresses the broader challenge of precise robotic manipulation in constrained spaces, which has applications beyond game-playing. The combination of low-cost cameras, fiducial markers, and 3D-printed structures could be applied to small-scale

automated assembly, laboratory sample handling, or assistive devices for individuals with limited mobility. Economically, the system demonstrates that a capable precision manipulation platform can be built for under $4,000 in parts, substantially less than commercial alternatives. Environmentally, the use of 3D-printed components reduces material waste compared to subtractive manufacturing, though the PLA frame is not easily recyclable. From a societal perspective, human–robot interaction systems like JengaBot can serve as educational platforms that make robotics more accessible and engaging to the public.

References

[1] Association for Computing Machinery. "ACM Code of Ethics and Professional Conduct," Accessed: May 1, 2026. [Online]. Available: https://www.acm.org/code-of-ethics

[2] Occupational Safety and Health Administration. "Machine Guarding," Accessed: May 1, 2026. [Online]. Available: https://www.osha.gov/machine-guarding

[3] International Electrotechnical Commission. "IEC/UL 62368-1: Audio/Video, Information and Communication Technology Equipment – Safety Requirements," Accessed: May 1, 2026. [Online]. Available: https://webstore.iec.ch/publication/59573

Appendix A Code Repository Structure

The MuJoCo simulation and AprilTag generation scripts follow the structure below. The model/mujoco-model/ directory contains:

poses.txt Input file specifying all block poses. The first line is the integer N (number of blocks). Each of the following N lines contains 7 values: x y z qx qy qz qw (position in meters and orientation as a quaternion in $xyzw$ order). Camera definitions can be appended after the block lines.

gen_scene_from_poses.py Reads poses.txt and generates a MuJoCo XML scene file (e.g., tower.xml). Converts quaternions from $xyzw$ to $wxyz$ order for MuJoCo. Each block is represented as a <body name="block{i}"> with a box geom (half-size $0.5 \times 0.5 \times 1.5$) and 10 AprilTag sticker geoms offset by EPS = 0.004 to avoid z-fighting.

tags_gen.py Generates AprilTag PNG textures (tag36h11 family) in the textures/ directory. Tags are numbered sequentially, with 10 tags per block: 2 end-face tags and 8 side-face tags following the convention in Table 1.

textures/ Output directory for generated AprilTag PNG images.

bar_tags.xml Template XML fragment defining tag geometries and materials for a single bar.

Tag IDs for block i occupy the range $10i - 9$ through $10i$, with the mapping from tag index k to face and position as defined in the main text.

Appendix B Requirement and Verification Table

Table 3 lists all subsystem requirements, their verification methods, and results.

Table 3: Complete requirement and verification table for the JengaBot system.

#	Requirement	Verification Method	Result
	Vision Subsystem		
V1	Each camera runs AprilTag detection at $\geq$ 30 fps at 1280×720.	Log per-frame timestamps over 1,000 consecutive frames; verify average interval $\leq$ 30 ms.	Pass
V2	Each camera identifies $\geq$ 90% of visible AprilTags per frame at 20 cm.	Compare detected IDs against ground truth across 500 frames per camera.	Pass
V3	Union of four cameras achieves $\geq$ 99% tag coverage on intact 30-block tower.	Capture from all cameras over 100 cycles; verify union includes $\geq$ 99% of 300 tags.	Pass
V4	End-to-end latency (capture to MuJoCo update) $\leq$ 50 ms (95th percentile).	Timestamp at frame capture and state update; measure over 500 cycles.	Pass
	Motion Control Subsystem		
M1	Repeatable positioning accuracy of $\pm$1.0 mm over full workspace.	Command to reference positions; measure actual position over 10 trials.	Pass
M2	Maximum travel speed $\geq$ 80 mm/s on X- and Y-axes.	Full-range traversal; measure time over 5 trials and compute average speed.	Pass
M3	Frame supports $\geq$ 0.8 kg moving payload with $<$ 2.0 mm static deflection.	Measure end-effector deflection under nominal load at 3 positions.	Pass
	End-Effector Subsystem		
E1	Controlled lateral push force up to 5 N, repeatable within $\pm$1 N.	Force gauge measurement over 5 bench-test trials.	Pass
E2	Gripper holds and transports one block (90×30×30 mm) without slippage.	Grip-and-transport test over 10 cycles.	Pass
E3	Push–extract–grip–place sequence within 8 s after reaching target.	Timed functional test over 5 trials.	Pass
	Central Controller Subsystem		
C1	Zero communication failures over 30 min continuous USB serial operation.	Log all transmitted commands and responses; verify no failures or timeouts.	Pass
C2	Controller generates valid command sequence for given tower state.	Provide representative inputs; verify valid output motion sequence.	Pass
C3	Motor execution functional during high Raspberry Pi CPU usage.	Run vision pipeline while executing motor commands; verify Arduino reports correctly.	Pass

Table 3: Complete requirement and verification table (continued).

#	Requirement	Verification Method	Result
	Power Supply Subsystem		
P1	Motor power system provides 24 V at PCB input under nominal conditions.	Measure supply voltage at PCB terminals during motor operation.	Pass
P2	Driver PCB receives STEP/DIR signals and produces correct motor motion.	Apply known pulse sequences; verify motor direction and displacement.	Pass
P3	Power stage remains stable during repeated start–stop motion cycles.	Run repeated motion tests; verify no reset, missed motion, or shutdown.	Pass
	System-Level		
S1	$\geq$ 3 successful block operations per game; $\geq$ 70% completion rate over 10 games.	Run 10 test games; record block operations and game outcomes.	Pass
S2	$\pm$1.0 mm positioning accuracy over 300$\times$300$\times$250 mm workspace.	Measure at reference positions across full workspace.	Pass
S3	Full manipulation cycle $\leq$ 15 s under normal conditions.	Time complete cycles from capture to placement.	Pass