Firmware for Gecko Testbed: Applications in Capturing Orbital Debris

Ateeb Ahmed, Roshan Kumar Gupta, Vassilios Papadopoulos

August 20, 2025

Abstract

Since the landmark space mission Apollo 11, there has been an exponential increase in man-made objects in space. Most of these objects orbit around Earth at varying altitudes and perform specific tasks like telecommunication, navigation (GPS), space-exploration and imaging (Hubble and James Webb Telescope) etc. Today there are 7,560 satellites orbiting Earth [1]. These satellites are prone to failure due to extreme physical conditions and contribute to ever growing Space Debris. There are active efforts underway to capture such debris. To this end, this project explores suitability of Gecko Materials in applications of capturing and removing space debris. Specifically, we provide a firmware which enables researchers to control and program a testbed to evaluate and compare different types of gecko materials and their performance in space-like conditions. In the first section, we talk about the problem in more detail, describe the testbed and its components and finally lay out the requirements that needs to be satisfied by this software package. Second section breaks down the software package into its components and describes the working of each individual component in detail, further we also provide a step-by-step guide on how to use the developed User Interface, lastly this section also provides a table which lists all the requirements and contrast them against the developed features. Section 3, provides insights and discussion on the scope and findings of this project and future directions.

1. Introduction (By Ateeb Ahmed)

In the last century mankind has made unprecedented technological advances and discoveries. Naturally, Space exploration, which has fascinated all civilizations that have existed, also had a breakthrough in the previous century when USA launched Apollo 11 on 26th July 1969 to Moon. While this project was mainly targeted to explore the lunar surface. Since then many other such objects are sent in space but only to perform specific tasks like communication, observation and scientific research. These objects usually orbit around Earth and are called Satellite. Currently, there are 7,560 satellites orbiting Earth [1]. But keeping these satellites operational require constant maintenance since they are exposed to extreme temperatures, radiation and micro-meteoroids. Unfortunately, not all satellites stay operational through out their mission and gracefully leave the orbit. Some gets destroyed during deployment, or lose contact and control from Earth due to communication issues. These dead satellites and other small objects created due to some form of damage to existing satellites are collectively called "Space Debris". There are currently 32,750 cataloged and tracked debris objects, these 50 cm or more in size and traveling at a speed of 3 km/s [2].

These objects also travel with a random trajectory and are not predictable. Collision with this debris can be fatal for not only existing operational satellites but also for future deployments. To this end, ClearSpace SA, a Swiss startup,

is developing a device resembling a four-armed "space claw" that would grip junk satellites and de-orbit them. The project is expected to go live in late 2029, and intends to de-orbit a special kind of space debris called VESPA (Vega Secondary Payload Adapter) [3].

Recently, Gecko materials have shown promise in catching space debris [link]. Gecko materials exhibits certain properties which are desirable in space debris collection, such as: Gecko materials only require smooth surfaces. Gecko Materials do not require adhesive substance and work purely through fundamental property of materials called Van Der Waals forces. Due to these properties Gecko materials offer multiple docking opportunities to any space debris and offer good re-usability.

Given promising results from early proof of concept studies on Gecko materials for space debris collection, further research on how gecko materials behave in space-like conditions such as a combination of extreme temperatures, radiation, vaccume and microgravity is needed.

In this regard, TU Berlin and Julius-Maximilian-Universität Würzburg as part of the **gEICko: GEcko based Innovative Capture Kit for uncooperative and unprepared Orbital assets** has initiated a joint investigation of Gecko materials. The team from TU Berlin has developed a Testbed. The goal of this Testbed is to compare different kinds of gecko materials against each other. The testbed is also suitable for simulated space-like environments since it is robust and vacuum sealed, a more detailed description of TestBed is provided in the following section.

This project, in particular, is concerned with developing a firmware to control the testbed and provide researchers with an intuitive but exhaustive user interface (UI) to be able to command and control this testbed.

1.1. The Gecko Testbed

Figure 1 shows an image of the testbed developed by the engineers from TU Berlin. This consists of 3 stepper motors. Each motor is responsible for movement in each direction i.e. x, y and z axes. In this center of the testbed is a smooth glass surface which simulates a smooth surface from that of a space debris and also serves as a rendezvous point aka point of contact. The small green piece directly on top of this surface represents the Gecko material being tested. The vertical motors (z-axis) moves the Gecko material up and down along its axis and press it against the smooth surface such that the gecko material stick to it.

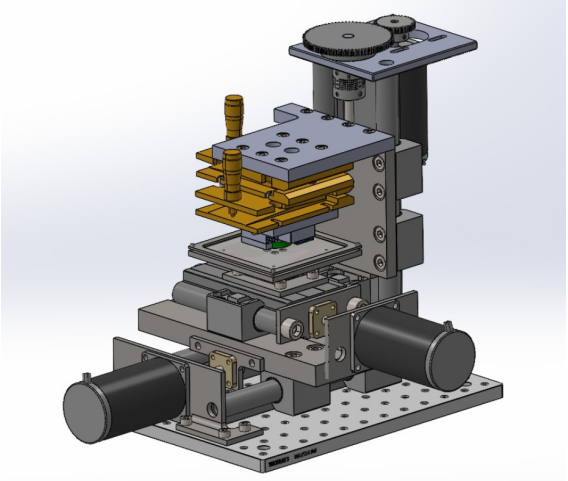


Figure 1: Testbed to test Gecko materials.

There is also a small 3 dimensional force sensor right underneath the smooth surface which can sense press/pull forces as well as shear and translatory forces.

The following are the exact specifications and details of each component of the Testbed:

- Stepper Motor: 3 stepper motors are used in this testbed. Each stepper motor is controlled using CL57T driver.
- Force Sensor: We have used a Strain Guage Force Sensor by ME Systeme model K3D40 (10N). We have further used GSV-4USB amplifier to control force sensor and also to serialize the output.
- Power Supply: We have used a generic powersupply easily available in all labs at 24V and 1.5A.
- Computing Unit: We have used a Raspberry Pi micro-controller as our main computing device.

All of the components of the testbed mentioned above are shown in figure 2.

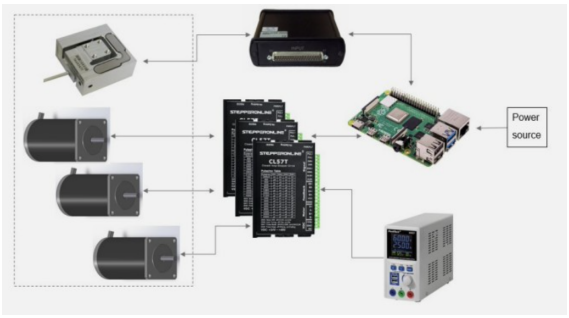


Figure 2: Individual Components of Gecko Testbed.

1.1.1. Uses Cases of Testbed

The main purpose for this testbed is to test and compare different gecko materials in space-like environments. The metric used to measure the effectiveness of a gecko material is the ratio between push and pull force generated, then an ideal Gecko material candidate would be one which requires minimum push forces against a smooth surface, sticks to it and then can pull the stuck surface with a much higher pull force.

This testbed accomplish this by pressing the gecko material against the smooth the surface until a target push force is reached (establishing docking), and then pull away the gecko material and measuring the pull force generated by gecko material. This simple experiment can have two further possibilities:

- Case 1: When smooth surface is static and only gecko material moves up and down.
- Case 2: When smooth surface is static until point of contact (docking), and while gecko material start pulling the smooth surface also moves along one or both horizontal axes (X and Y).
- Case 3: When the gecko materials establishes contact while smooth surface is moving along horizontal axes. Similarly, pull force is also measured with gecko and smooth surface moving. This case is the most complicated out of all above and accurately simulates the behavior of uncontrolled space debris.

Now that the use cases of the testbed are laid out, in the later sections we will describe how the developed firmware handles these situations.

2. System Architecture (Delivered by Roshan Kumar Gupta)

2.1. Overview

The developed testbed follows a three-layer architecture consisting of the frontend interface, the backend server, and the hardware integration layer. The goal of this design is to ensure modularity, real-time responsiveness, and safe interaction between user-defined sequences and the physical setup. Figure X illustrates the overall architecture of the system.

At the top layer, the web frontend acts as the primary interface between the user and the testbed. Implemented using Three.js and Vanilla JavaScript, the frontend provides a sequence builder for defining multi-axis actions, a real-time force graph, motor control buttons, and logging/export options. It communicates with the backend through REST API calls for commands and WebSocket events for live updates.

The backend server, built with Flask and Flask-Socket.IO, forms the middle layer of the system. It exposes endpoints for sequence execution, emergency stop, calibration, motor checks, and data export. Additionally, the backend streams force sensor readings, motor step counts, and experiment logs to the frontend in real time. This ensures the user has continuous visibility and control during experiments.

At the core logic level, the backend handles sequence execution, trigger evaluation (force, duration, step-based), and logging. Concurrency is achieved using threads, where a dedicated force poller thread continuously reads sensor data,

while per-axis execution threads manage motor control. This modular approach allows the testbed to handle complex sequences while ensuring responsiveness to safety-critical events such as emergency stop signals.

The hardware interface layer provides low-level communication with the physical devices. GPIO control manages the stepper motors across three axes (X, Y, Z), while serial communication integrates the multi-axis force sensor. The Raspberry Pi 5 acts as the controller, bridging the backend software with the hardware. Feedback in the form of force measurements and step counts is continuously transmitted back to the backend and relayed to the user interface.

This layered architecture ensures a clean separation of concerns: the frontend focuses on visualization and usability, the backend manages logic and communication, and the hardware layer executes the actual physical tasks. Together, these modules form a reliable and extensible testbed for gecko adhesion experiments.

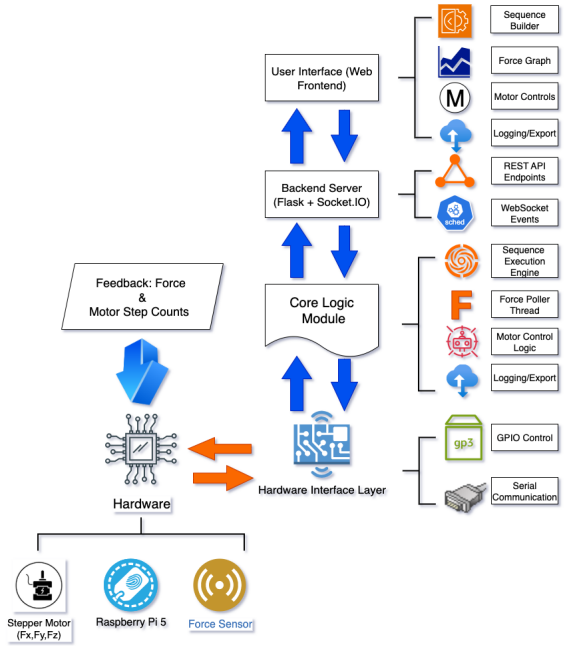


Figure 3: Overall system architecture showing the web frontend, backend (Flask + Socket.IO), and hardware layer (Raspberry Pi, motor drivers, force sensor) with data and control flows.

2.2. Components

Frontend (Three.js + Vanilla JS): Interactive StepBuilder, live telemetry (F_x, F_y, F_z, shear), per-axis step counts, 3D visualizer, and controls (Run, Stop, Emergency Stop (E-Stop), Motor Check, Zero Sensor, Repeat, Upload/Download State, Export Logs).
Backend (Flask + Socket.IO): HTTP actions for commands and JSON state, real-time event streaming, sequence execution threads, trigger evaluation, optional force-hold, logging, calibration/zeroing, and reset-to-origin.
Hardware layer (Raspberry Pi + drivers + force sensor): GPIO control for X/Y/Z stepper axes and a serial multi-axis force sensor. Feedback (forces, step counts) streams to the UI in real time.

2.3. Data & Control Flow

(1) *Authoring:* Define a sequence in the StepBuilder (movement, triggers, optional hold, Repeat) or upload a saved JSON.

(2) *Dispatch:* The frontend sends a validated experiment description to the backend.

(3) *Orchestration:* The backend runs a continuous force-poller thread and launches per-axis worker threads.

(4) *Execution:* Workers step motors and evaluate triggers ("Any" or "All"). If a hold is configured, micro-movements maintain the target force until release conditions.

(5) *Telemetry:* The backend emits `force`, `step_count`, and `log` events; the UI renders them live.

(6) *Lifecycle:* On completion—or Stop/E-Stop—the backend logs the outcome, resets axes to origin, and exposes a timestamped log for download. With Repeat, the full sequence executes N times with clearly delimited iterations.

2.4. Safety & Fault Handling

E-Stop immediately disables motion and raises a visible log event; a global force cap prevents out-of-range operation; Zero Sensor and calibration ensure stable force readings; reset-to-origin guarantees a known state between trials.

2.5. Extensibility & Constraints

The modular design allows adding distance-based triggers, richer analytics, or a framework UI (e.g., Svelte/React) without changing core motion/telemetry logic. Real-time responsiveness depends on sensible step sizes, trigger thresholds, and hardware health.

3. Frontend Implementation (Three.js + Vanilla JS) (Delivered by Roshan Kumar Gupta)

3.1. Role & Goals

The frontend is the *operator console*: simple for non-technical users, responsive for live experiments, and strict enough to prevent unsafe setups. Priorities: clarity, low-latency feedback, guardrails, and repeatability/sharing of configurations.

3.2. Three.js Scene

(1) *Scene graph:* world axes, camera (orbit controls), light rig, and minimal meshes to indicate stage/axes and motion direction.

(2) *Rendering loop:* RequestAnimationFrame drives lightweight renders; UI updates are decoupled from the physics/control loops to keep 60 FPS when possible.

(3) *Responsiveness:* layout adapts between wide desktop and narrow screens; critical controls remain above the fold.

(4) *Safety affordances:* emergency stop is persistent and visually distinct; destructive actions require a single explicit click (no hidden combos).

3.3. Interface Overview

Control Bar: Run, Stop, E-Stop, Motor Check, Zero Sensor, Repeat (count), Upload/Download State, Export Logs.
StepBuilder: Per-axis steps with direction and step size; movement-init triggers; break triggers (force/duration/steps) with *Any/All* logic; optional *Hold* with release triggers.
Telemetry Panel: Live F_x, F_y, F_z (and shear if shown) and per-axis step counts.
3D Visualizer: Three.js scene providing spatial context (axes, motion cues).
Log Console: Readable stream of step boundaries, trigger firings, hold state, stops, and errors.

3.4. Interaction & Workflows

Configure or Upload a sequence; set Repeat for multi-trial tests; Run; monitor forces, step counts, logs, and 3D view; Stop (graceful) or E-Stop (immediate); Zero Sensor before force-critical runs; Motor Check for quick health; Download/Upload State to reuse setups; Export Logs for audit.

3.5. Real-Time Behavior & Safeguards

WebSocket events are throttled to keep the UI responsive; forms validate unsafe combinations; connection status is visible; conflicting actions disable during active runs; E-Stop remains prominent.

4. Backend Implementation (Flask + Socket.IO) (Delivered by Roshan Kumar Gupta)

4.1. Role & Boundaries

The backend validates and executes sequences, manages concurrency and safety, streams telemetry, and persists an auditable log. It exposes clear actions: Run, Stop, E-Stop, Motor Check,

Zero Sensor, Calibration, Repeat, State Upload/Download, and Log Export.

4.2. Execution Model

A dedicated force-poller thread continuously converts sensor frames to F_x, F_y, F_z (plus F_{shear}) and broadcasts updates. Per-axis worker threads step motors and evaluate triggers concurrently. Serial I/O is lock-protected; shared state (forces, step counts) updates atomically. Long-running execution remains off the request thread so the API stays responsive.

4.3. Trigger Engine & Hold Control

Triggers support force thresholds (with a global safety cap), duration, and step count; each group can require *Any* or *All*. Motion proceeds until break triggers are satisfied. When configured, a hold loop maintains a target force within tolerance until release conditions. Significant events (initiation, trigger fired, hold start/complete) are logged and streamed.

4.4. Repeat & Lifecycle

A top-level Repeat count executes the full sequence N times. Iterations are clearly delimited in the log. At run end (or stop), the backend resets axes to origin using tracked step counts, writes a timestamped log file, and clears transient state.

4.5. Data, Logging & State I/O

During a run the backend buffers timestamps, force vectors, step counts, trigger events, step boundaries, and summaries; after a run it produces a uniquely named log file (Export Logs). Download State saves the current StepBuilder configuration as JSON; Upload State restores it for reproducible protocols.

4.6. Robustness & Safety

Emergency Stop immediately drops motor control outputs and emits an operator log event; graceful stop halts at the next safe opportunity. The backend validates inputs and runs long-lived work off the request thread to keep the server responsive. A quick Motor Check routine verifies motion on each axis before full experiments; Zero Sensor and Calibration actions maintain a clean baseline and consistent units.

5. Sensor Calibration & Safety Mechanisms
(Delivered by Ateeb Ahmed)

5.1. Calibration Method (Manufacturer Two-Point)

Software scaling uses manufacturer-provided two-point calibration (0 N, 10 N). The default factors applied in software are listed in Table 1. Raw frames are converted to physical forces per axis using these factors, and a *Zero Sensor* action establishes a clean baseline before runs. Updated factors (from certificates or lab recalibration) can be applied and then re-zeroed.

Table 1: Manufacturer-supplied two-point calibration factors (0 N, 10 N) used as software defaults.

Axis	Factor	Note
F_x	0.0612932	Provided by manufacturer
F_y	0.0603852	Provided by manufacturer
F_z	0.052317133779699	Provided by manufacturer

5.2. Operator Workflow (Pre-Run)

Zero Sensor with the fixture unloaded; verify factors; sanity-check idle F_x, F_y, F_z are near zero; (optional) Motor Check; ensure StepBuilder thresholds respect the safety limit.

5.3. Safety Envelope, Integrity & Notes

Global force cap (default 10 N), ever-present E-Stop, graceful Stop, and post-run reset-to-origin

ensure safe operation. The run log records timestamps, factors, zeroing actions, trigger events, step boundaries, and summaries; repeated runs are clearly delimited. Re-zero after fixture changes or temperature shifts; choose realistic setpoints.

6. StepBuilder — Operator Guide & Safety (Delivered by Ateeb Ahmed)

6.1. Purpose

"The StepBuilder is the control brain of the testbed (colloquially nicknamed 'Thanos' during development for its make-or-break impact). It governs what to move, when to stop, and whether to hold a target force — per axis — for reproducible and safe experiments."

6.2. Core Concepts

Step (per axis); movement-init triggers (preconditions); break triggers (force/duration/steps); optional Hold (maintain target force until release); *Any/All* logic; experiment-level Repeat; State import/export; Log export.

6.3. Recommended Workflow

Define scope (Repeat or Upload a saved setup); configure steps (direction, step size; init/break triggers; optional Hold with release); validate (limits, Any/All logic, safe thresholds); run & monitor (forces, steps, logs; Stop/E-Stop as needed); analyze & reuse (Export Logs; Download State).

6.4. Safety, Validation & Recovery

Keep thresholds within sensor range and below the safety cap; prefer force-based init triggers; E-Stop is always available; post-run reset to origin. Common mistakes: too many conditions with *All*; unreachable holds; misunderstanding that Repeat applies to the *entire* sequence; skipping Zero/Calibrate. Recovery: check init triggers/direction/step size; verify comparator and axis; re-zero for noisy holds; reconnect UI if needed.

7. Operator Walkthrough: Using the Sequence Builder (Screenshots) (Delivered by Roshan Kumar Gupta)

This section shows, step by step, how a new operator or tester should use the **Sequence Builder** to configure and run experiments safely. Each step references a UI screenshot.

Step 1 — Motor Check (unlock controls). Open the UI and click **Motor Check** first. This verifies basic motion and unlocks run-time actions (Run/Stop/E-Stop).

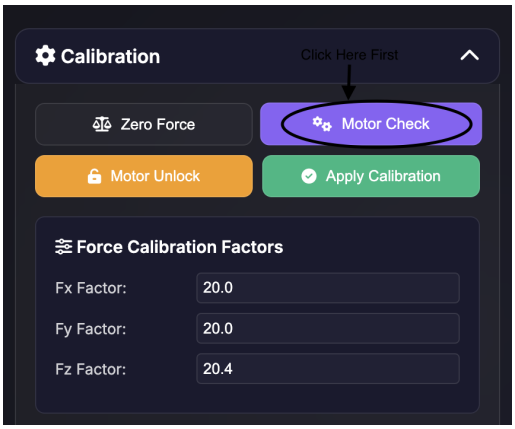


Figure 4: Run **Motor Check** to verify motion and unlock controls.

Step 2 — Add a step (choose axis & rate). In **Sequence Builder**, click **Add X Step**, **Add Y Step**, or **Add Z Step**. Select direction and set the **Pulse Frequency (ms)** to control stepping rate.

Step 3 — Add a movement-init trigger. Click **Add Trigger which starts movement**. Use this when the step must wait for a condition (typically a force threshold) before moving.

Step 4 — Configure the trigger. Choose the signal (e.g., **Fz (N)**), comparator ($\geq, \leq, ==, >, <$), and value (e.g., **1** = 1 N). **Fire All Triggers = True** requires all listed init triggers to be true; **False** means any one is enough. Checked

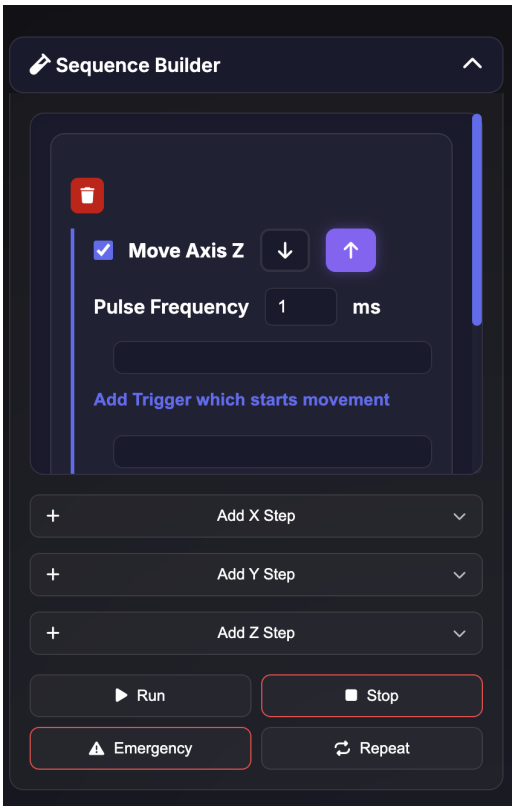


Figure 5: Add an axis step; set direction and **Pulse Frequency**.

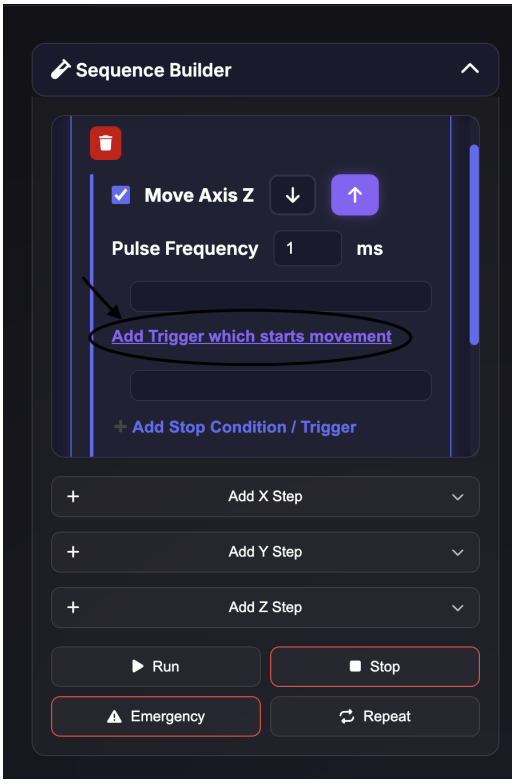


Figure 6: **Add Trigger which starts movement** to define preconditions before motion.

axes move together; unchecked axes stay idle.

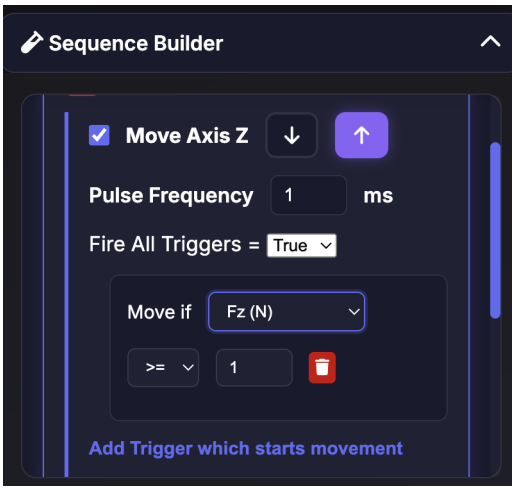


Figure 7: Configure a movement-init trigger: signal, comparator, and value.

Step 5 — Enable Hold Force. Scroll to **Hold Force** and enable it to add a force-controlled hold after movement stops.

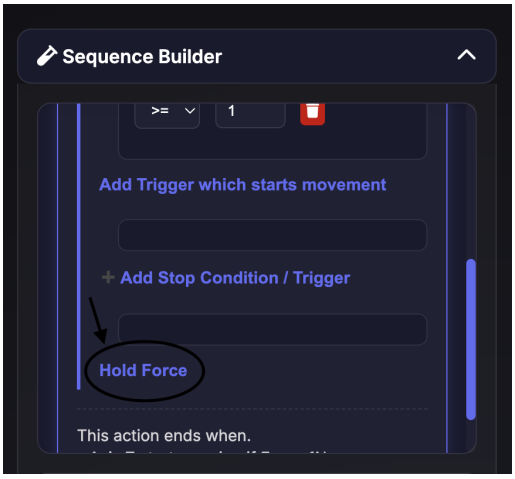


Figure 8: Enable **Hold Force** for a controlled force-maintenance phase.

Step 6 — Set Hold target and release conditions. Enter the hold setpoint (e.g., **Hold Fz = 5 N**) and add **Stop holding if** conditions (force/duration/step-count). *Example:* If the break trigger is **Fz** $\geq$ 10 **N** and Hold is **5 N**, the system presses until $\approx$ 10 N, then settles and maintains $\approx$ 5 N until the release condition fires.

Figure 9: Set the **Hold** setpoint and release conditions.

Step 7 — Run once or enable Repeat. Use **Run** to execute once, or enable **Repeat** to run multiple back-to-back trials for adhesion studies.

Step 8 — Manage data (logs & state). **Export**: download the **timestamped** run log. **Upload**: load a saved **State (JSON)** to rehydrate the builder. **Download**: save the current builder state as JSON for reuse or sharing.

Step 9 — Monitor in real time & use Manual Control. Observe **X/Y/Z positions** and **Force (N)** during a run. The **Manual Control** panel uses **click-and-release** for safety—movement occurs only while the button is actively pressed (no continuous joystick).

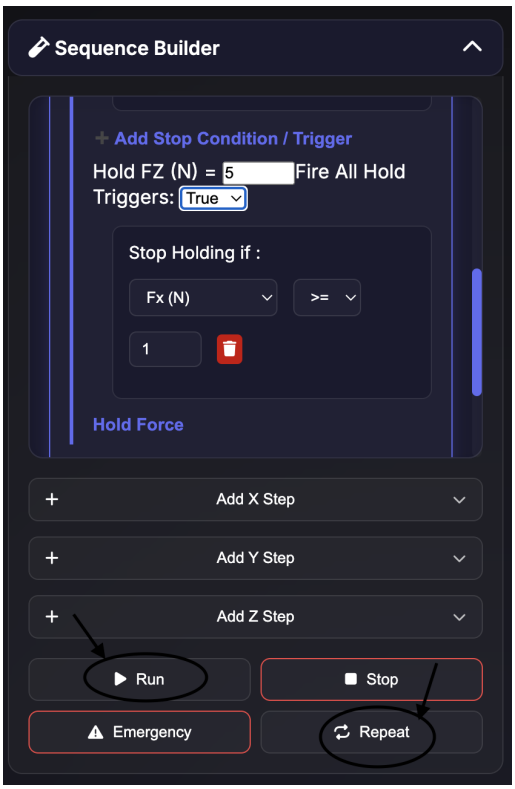


Figure 10: **Run** the sequence or enable **Repeat** for multiple trials.

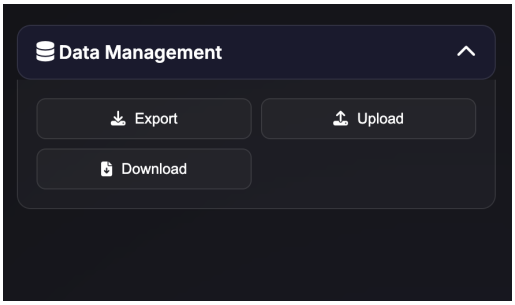


Figure 11: Data Management: **Export** logs; **Upload/Download State** JSON.

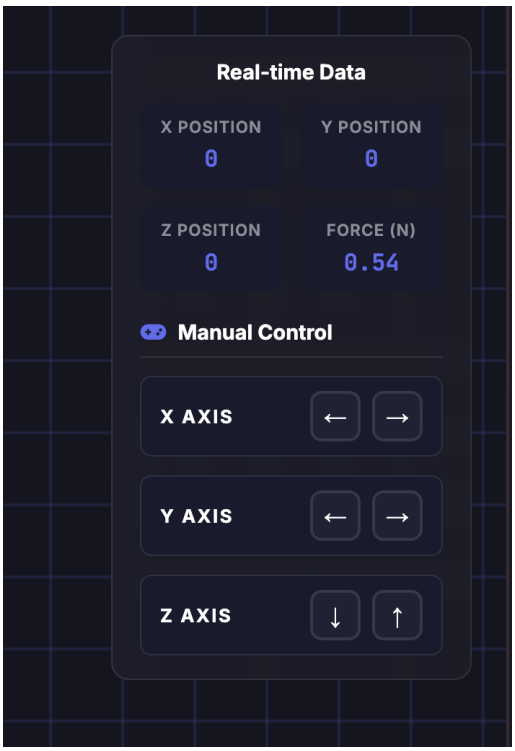


Figure 12: Real-time positions/force and **Manual Control** (click-and-release for safety).

Full UI overview (optional). A wide view helps new users orient themselves: Sequence Builder (left), 3D scene (center), telemetry and logs (right/bottom).

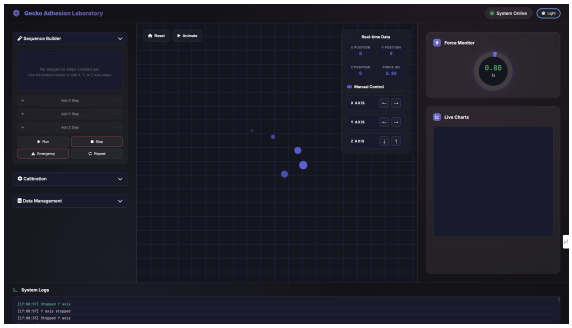


Figure 13: Full UI overview of the Gecko Adhesion Laboratory dashboard.

Get UI experience by clicking here *(some functionality won't work as it's not connected to the server).*
Link: https://roshangupta00750.github.io/geckotestbed/

Notes & tips.

- Step-count triggers are available under **Add Stop Condition / Trigger** (e.g., stop after 2000 micro-steps).
- Keep thresholds within the sensor's working range and below the global safety cap.
- **Emergency Stop (E-Stop)** is always available and halts motion immediately.

8. Testing & Results (Delivered by Roshan Kumar Gupta)

8.1. Objectives

Confirm safe, reproducible execution of user-defined sequences with real-time telemetry, Repeat trials, and post-run analysis via logs and state export.

8.2. Test Plan

(1) *Functional:* Run/Stop/E-Stop, Motor Check, Zero/Calibrate, JSON Import/Export, Log Export.

(2) *Execution:* Per-axis steps with movement-init, break, and optional hold.

(3) *Repeatability:* Run the same sequence with Repeat to compare adhesion behavior.

(4) *Telemetry:* Continuous F_x, F_y, F_z (and shear where shown), step counts, and logs.

(5) *Safety:* Force cap respected; E-Stop immediate; post-run reset to origin.

8.3. Scenarios & Expected Outcomes

Approach-until-contact ("$F_z \geq$ threshold") halts at the threshold with a trigger-fired log; force-controlled hold maintains setpoint until release; Repeat produces delimited iterations with reset between runs; mid-motion E-Stop halts immediately and logs the event; state I/O round-trip rehydrates and exports as expected.

8.4. Evidence (Artifacts)

Screenshots: dashboard during run; Repeat setting + per-iteration headers; excerpt of exported log (step starts/stops, triggers, holds); Upload/Download State flow confirming round-trip.

9. Conclusion (Delivered by Vassilios Papadopoulos)

9.1. Validation of the Testbed on Earth

The developed gecko-inspired adhesion testbed has first been validated under terrestrial laboratory conditions. Initial experiments demonstrated that the combination of a high-precision force sensor and automated software control

loops allows for reproducible adhesion and detachment measurements across a large number of cycles. Unlike manual test procedures, which are limited by operator accuracy and fatigue, the automated system can execute precisely defined loading–unloading sequences with high repeatability. This capability is critical for assessing long-term performance of micropatterned dry adhesives (MDA), where subtle changes in force profiles over hundreds or thousands of repetitions may indicate degradation of surface structures or material fatigue.

The integration of software-based closed-loop control further enhances the stability of the experimental setup. The implemented routines manage motor speed, preload forces, dwell times, and detachment trajectories in a way that minimizes variability between test runs. This not only improves confidence in the measurement data, but also reduces human-induced bias. Particularly for adhesives whose performance depends strongly on contact geometry and loading history, automation is indispensable to generate meaningful datasets for later comparison with results obtained in microgravity environments.

A key feature of the testbed is its ability to maintain constant normal forces during adhesion testing. Force feedback from the sensor is processed in real time by the control software, enabling rapid adjustment of the actuator position to compensate for surface irregularities or environmental disturbances. This ensures that the desired preload values are applied consistently, a prerequisite for studying the sensitivity of gecko adhesives to initial contact conditions.

Moreover, the validation phase confirmed the robustness of the hardware components. The linear actuator, load cell, and mounting frame remained stable under extended operation, allowing tests of several thousand adhesion cycles without measurable drift in calibration. This establishes the platform as a reliable tool for systematic material studies, paving the way for subsequent adaptation to reduced-gravity scenarios.

In summary, the terrestrial validation confirms that the testbed provides accurate, repeatable, and scalable measurement of gecko-inspired adhesives. This baseline is essential before transferring the system to constrained environments such as parabolic flights, drop tower experiments, or even orbital missions.

9.2. Scalability to Microgravity Platforms

While the testbed has proven effective under terrestrial conditions, its true potential lies in adapting the hardware and software for operation in reduced-gravity environments. Several experimental platforms are available within Europe that provide access to microgravity conditions, each with unique constraints in terms of duration, size, power availability, and environmental robustness. The following subsections outline how the testbed can be scaled and adjusted to these platforms, highlighting both the opportunities and limitations of each environment.

9.2.1. ESA Fly Your Thesis

Parabolic flights provide repeated intervals of approximately 20 seconds of microgravity per parabola. This offers the advantage of a relatively high number of test repetitions within a single campaign, allowing different adhesive samples or loading regimes to be investigated under similar flight conditions. However, the short duration of each microgravity phase requires highly efficient and fast experimental routines, forcing a trade-off between motor speed and precision of the applied motion profiles.

To meet the constraints of parabolic flight, significant modifications to the current testbed are necessary. These include a substantial reduction in system mass, integration into the standardized aircraft rack format (approx. $50\times50\times80\,\mathrm{cm}$), and autonomous power supply. Moreover, the software must be capable of reliably executing complete adhesion–detachment cycles within the short time window, with automatic synchronization to the onset of microgravity phases.

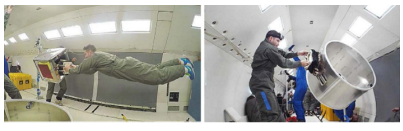


Figure 14: 2014 MDA Tests in-flight by ESA

9.2.2. ESA Orbit Your Thesis

The Orbit Your Thesis program extends the parabolic flight concept to long-duration orbital platforms. Here, the same fundamental constraints apply with respect to size, weight, and autonomy, but the experimental time is vastly increased compared to parabolic flights. This enables more systematic testing, including long-term degradation studies, radiation exposure experiments, and automated sequences across weeks or months of operation.

The adaptation requirements for this platform are similar to those of Fly Your Thesis—miniaturization, rack compatibility, and robust autonomous operation. However, the longer time frame provides opportunities to integrate advanced diagnostic routines and self-calibration features within the software. The combination of repeatable orbital cycles and extended exposure to the space environment makes this a promising platform for bridging ground validation with future operational use in Active Debris Removal missions.

Figure 15: 2020: MDA Tests on the ISS

9.2.3. Bremen Drop Tower

The Bremen Drop Tower offers highly controlled microgravity conditions for durations between 4.74 and 9.3 seconds, depending on whether the catapult mode is used. In principle, this facility could be used to test the gecko-inspired adhesion testbed, as it provides true free-fall conditions without vibrations or turbulence.

However, the extremely short time window places significant restrictions on the types of experiments that can be performed. Even with optimized control software, a complete adhesion cycle involving approach, contact formation, force measurement, and detachment is difficult to achieve within only a few seconds. At best, the system could perform a very rapid approach and a preliminary contact test, but there would be little opportunity to record high-quality force data or to repeat the experiment for statistical evaluation.

From an economic perspective, the drop tower is also less attractive compared to parabolic flight campaigns. The cost of preparing and executing a drop experiment is high relative to the small amount of usable data that can be collected, making it inefficient as a platform for systematic adhesive testing.

Therefore, while technically possible, the Bremen Drop Tower is not considered a suitable environment for comprehensive testing of the gecko adhesion testbed. Its role may be limited to quick proof-of-concept demonstrations or rapid-deployment tests, but meaningful measurements of adhesion dynamics would require longer-duration platforms such as parabolic flights or orbital missions.

Figure 16: Bremen Drop Tower facility with conceptual experiment capsule.

In summary, the testbed is inherently scalable across different microgravity platforms, with each environment offering specific advantages and limitations. Parabolic flights allow repeated testing, orbital missions enable long-term exposure, and drop tower experiments deliver clean free-fall conditions. Together, these platforms form a progressive pathway for validating gecko-inspired adhesives under increasingly realistic space conditions.

9.3. Scientific Basis of Gecko-Inspired Adhesion in Space

The central principle underlying gecko-inspired adhesion is the utilization of van der Waals (vdW) forces, which arise from intermolecular interactions at the contact interface between an adhesive surface and its counterpart. These forces are quantum-mechanical in nature and act independently of the gravitational field, making them fundamentally suitable for application in microgravity environments. Unlike suction-based adhesion systems, which depend on ambient pressure differences and are largely ineffective in vacuum, vdW adhesion is equally present in space as on Earth.

Experimental and theoretical studies consistently show that the contact mechanics of micropatterned dry adhesives (MDA) are dominated by these van der Waals interactions. The magnitude of the adhesive force depends strongly on the effective contact area between the adhesive microstructures and the target surface. This insight has driven the design of synthetic structures with optimized geometries, such as mushroom-shaped terminal features, which maximize real contact area while maintaining the ability to detach cleanly. These geometries not only enhance adhesion on smooth surfaces but also improve tolerance to surface irregularities, which is essential when considering the heterogeneous surface properties of space debris.

Another key aspect is that gecko-inspired adhesives are inherently reversible and self-cleaning to some extent. Their attachment relies on weak intermolecular interactions rather than chemical bonding or surface damage, allowing repeated use without contamination of the substrate. This reversibility is highly advantageous for on-orbit servicing and debris removal, where multiple docking and detachment operations may be required under varying conditions.

It is worth noting that earlier debates on potential suction contributions to gecko adhesion have largely been resolved in the context of space. Under vacuum conditions, suction plays at most a negligible role, and adhesion is overwhelmingly governed by van der Waals forces. This understanding reinforces confidence that the observed adhesive performance on Earth can be translated to the orbital environment without loss of the underlying physical mechanism.

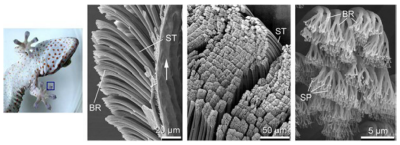


Figure 17: Adhesive feet of a gecko with a surface, made up of hundreds of thousands of hairs (setae)

Taken together, these findings establish a strong scientific foundation for the use of gecko-inspired adhesives in space. By relying on van der Waals forces that are unaffected by gravity or atmospheric pressure, MDA systems offer a robust and scalable approach for gripping and manipulating objects in orbit. The next step lies in evaluating how environmental stressors specific to space, such as radiation and atomic oxygen, affect the long-term performance of these materials.

9.4. Environmental Influences and Open Questions

While the scientific basis of van der Waals adhesion ensures that the fundamental adhesive forces remain effective in microgravity and vacuum, the performance of micropatterned dry adhesives (MDA) is still strongly influenced by environmental factors present in space. These conditions can alter material properties, surface chemistry, or mechanical stability, and therefore must be critically evaluated in the context of orbital applications.

Radiation represents one of the most significant challenges. High-energy particles and ultraviolet radiation are known to induce degradation in many polymeric materials, leading to embrittlement, loss of elasticity, and surface cracking. For MDA fabricated from polyurethane or similar polymers, long-term exposure can reduce adhesion strength by altering the flexibility of the microstructures. Preliminary irradiation tests have confirmed a measurable decline in performance after extended radiation doses, underscoring the need for either protective coatings

or the development of radiation-resistant polymers.

Thermal cycling in low Earth orbit exposes materials to temperature swings from approximately $-100\,^{\circ}\mathrm{C}$ to $+100\,^{\circ}\mathrm{C}$ over the course of each orbit.

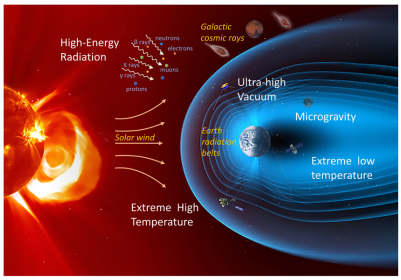


Figure 18: Harsh space environment, affecting the adhesion force of a MDA in the long term.

Interestingly, adhesion tests conducted under laboratory-controlled thermal variations suggest that polyurethane-based adhesives retain their functionality across this range. No significant mechanical failures were observed, indicating that thermal effects may be less critical than initially assumed. Nonetheless, prolonged cycling could induce fatigue in the adhesive backing materials or supporting structures, requiring additional investigation.

Vacuum exposure itself does not diminish van der Waals forces, but it can lead to desorption of absorbed water and volatile contaminants from the adhesive surfaces. Controlled experiments in vacuum chambers have shown that adhesion values remain largely stable, suggesting that this effect is negligible for short-term operations. However, multi-month exposure in orbit could still produce subtle changes in surface chemistry that merit attention.

A particularly critical factor is the presence of atomic oxygen in low Earth orbit. Reactive oxygen species are highly erosive and can etch polymer surfaces on a microscopic scale, directly degrading the adhesive microstructures that are essential for contact formation. Unlike thermal or vacuum effects, atomic oxygen erosion poses a direct threat to the long-term viability of gecko-inspired adhesives. Protective surface treatments, such as thin oxide coatings, may be required to mitigate this effect.

In summary, although van der Waals adhesion is fundamentally robust in space, the durability of micropatterned dry adhesives depends on the resilience of the underlying material system to radiation, atomic oxygen, and cumulative thermal cycling. Future research should prioritize long-duration in-situ testing and material optimization to ensure that adhesive performance is maintained throughout operational lifetimes in orbit. These open questions must be resolved before MDA technology can transition from laboratory demonstrations to real-world space missions.

9.5. Required Adjustments in Hardware and Software

Moving the testbed from a laboratory setup to a flight-ready system means that both hardware and software need to be adapted. The current version works well on Earth, but it is too large and too dependent on external power and control to be used directly in a microgravity campaign.

On the hardware side, weight and size are the biggest limitations. For parabolic flights, all equipment has to fit into the standardized aircraft racks (about $50 \times 50 \times 80\,\mathrm{cm}$). This means that actuators, mounting structures, and electronics will need to be redesigned with lighter materials and in a more compact layout. Battery power is also essential, since continuous lab power is not available. At the same time, the system has to remain mechanically stable and precise enough to run the same adhesion loops that were demonstrated on the ground.

For the software, the main challenge is autonomy. During a parabolic flight there is very little time for manual control, so the testbed must be able to recognize the start of the microgravity phase, run a full adhesion cycle on its own, and save all data without errors. The routines also need to be simple and reliable, because failed cycles mean lost opportunities.

Finally, the experiments have to be focused. In the limited time available it is not possible

to test everything, so it makes sense to concentrate on patches with different pre-exposures (for example radiation or thermal cycling) to compare how these conditions influence adhesion.

In short, the next step is to make the system smaller, lighter, and fully automatic. If these adjustments are achieved, the testbed will be ready to provide useful results not only in the lab but also in parabolic flights and, later on, in orbital missions.

9.6. Outlook

The gecko-inspired adhesion testbed represents an important step toward advancing bio-inspired gripping technologies for space applications. Terrestrial experiments have demonstrated that automated, repeatable adhesion testing is feasible with high precision, providing a reliable foundation for further development.

Since gecko-inspired adhesives have already been demonstrated successfully in both parabolic flights and orbital experiments, their fundamental suitability for space environments is established. The remaining challenge is not proof of principle, but rather the preparation of compact, flight-ready test systems that can operate autonomously under strict mass and volume constraints. Miniaturization is therefore the key requirement—both for parabolic flight campaigns, where size and weight are limited by aircraft rack standards, and for orbital missions, where payload capacity and resource efficiency are critical.

With these adaptations, the testbed could serve as a versatile platform for targeted investigations, including long-term material degradation, radiation effects, and repeated adhesion–detachment cycles in orbit. Such studies would generate the data needed to qualify micropatterned dry adhesives for integration into operational space systems.

In summary, the immediate focus must shift from demonstrating feasibility—already proven in earlier campaigns— towards engineering refinement. By achieving lightweight, miniaturized, and autonomous designs, the gecko-inspired testbed can transition from a laboratory instrument into a practical payload for future parabolic and orbital missions, supporting the broader goal of enabling sustainable Active Debris Removal and on-orbit servicing technologies.

References

[1] U. of Concerned Scientists. "Ucs satellite database." (2005), [Online]. Available: https://www.ucs.org/resources/satellite-database#:~:text=Total%20number%20of%20operating%20satellites,Military:%20246 (visited on 12/08/2005) (cit. on p. 1).

[2] A. Cacioni. "Space explained: How much space junk is there?" (2022), [Online]. Available: https://www.inmarsat.com/en/insights/corporate/2022/how-much-space-junk-is-there.html#:~:text=There%20are%20currently%20about%2032%2C750,remain%20a%20work%20of%20fiction. (visited on 12/19/2022) (cit. on p. 1).

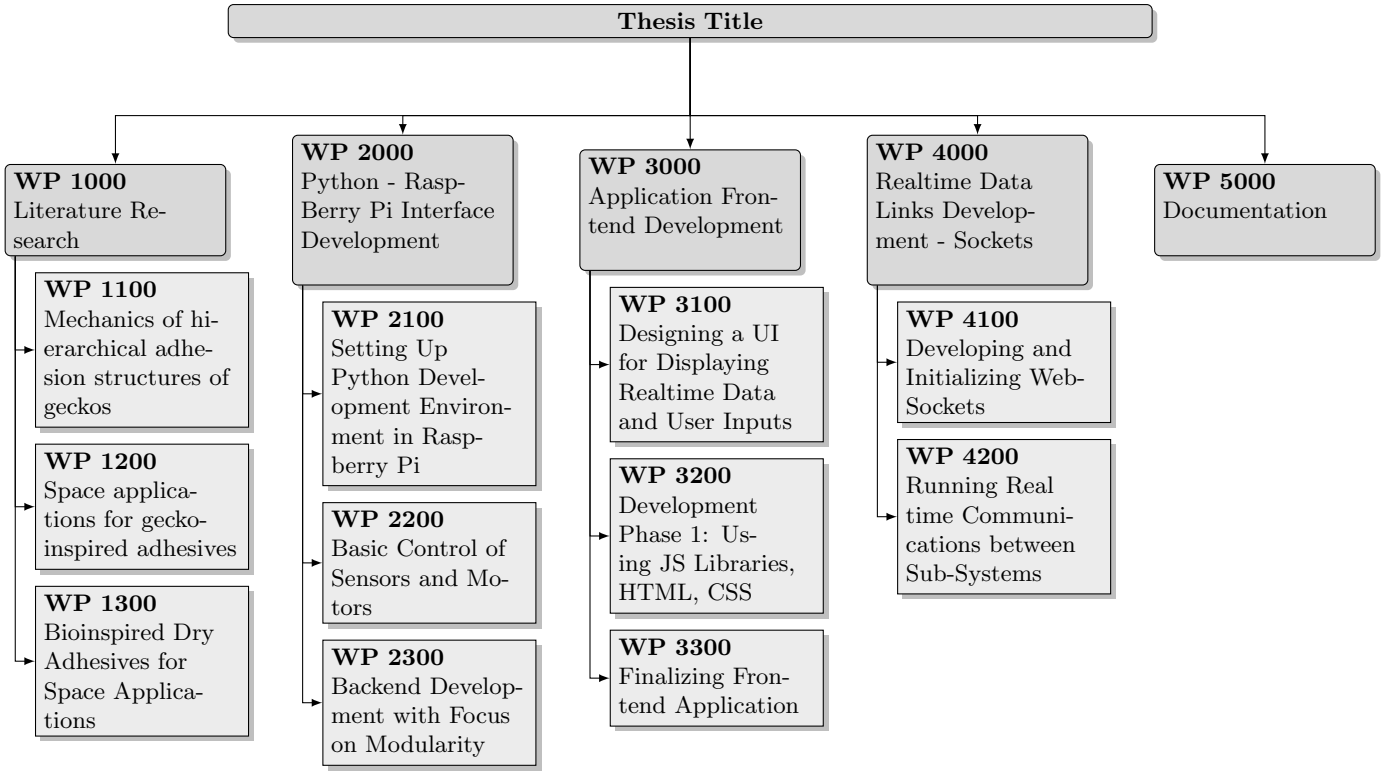
[3] T. E. S. Agency. "Clearspace 1." (2020), [Online]. Available: https://www.esa.int/Space_Safety/ClearSpace-1 (cit. on p. 2).

Table 2: Requirement → Evidence Traceability (Section 8.4)

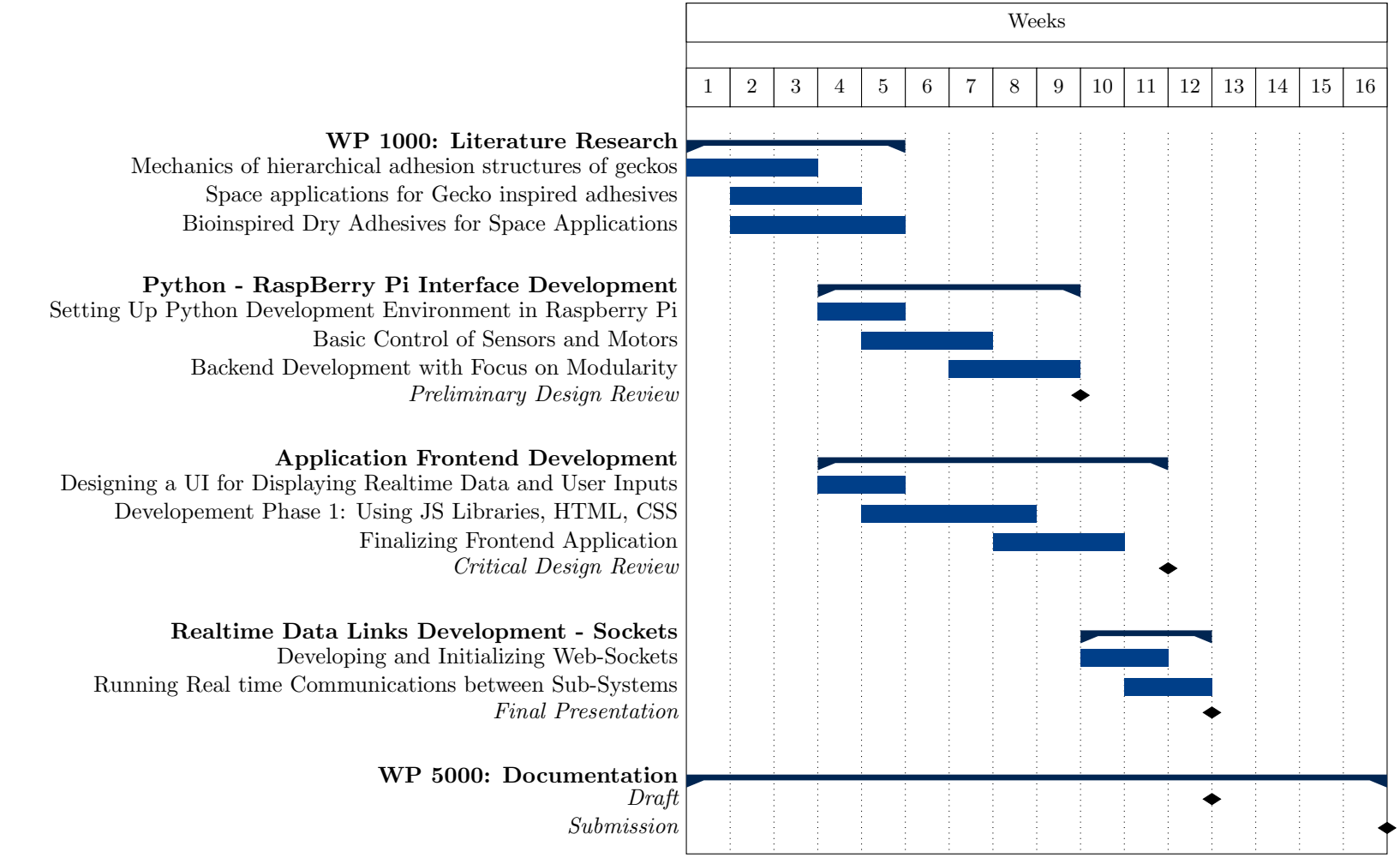
Requirement (what)	**Implementation (where)**	**Report Section**	**Evidence (figure/file)**
End-to-end run of a user-defined sequence	Backend per-axis worker threads execute steps; StepBuilder defines steps/triggers	§§ 3, 4, 6–7	Fig. 10 "Run/Repeat"; `LOG-1` exported run log
Movement-init triggers supported	Trigger evaluation before stepping (force preconditions)	§§ 4.3, 6.2, 7 (Steps 3–4)	Fig. 6 "Add Trigger which starts movement"; Fig. 7 configured trigger
Break triggers: **force / duration / steps**	Trigger engine with comparators ($\geq, \leq, ==, >, <$)	§§ 4.3, 6.2, 7 (Steps 5–6)	Fig. 8 "Add Stop Condition / Trigger"; Fig. 9 release conditions
Optional **Hold** (maintain target force) with release conditions	Hold control loop with corrective micro-moves	§§ 4.3, 6.2, 7	Fig. 8 enable Hold; Fig. 9 Hold = 5 N + release
Any/All trigger logic	Config flag for trigger groups	§ 6.3; 7 (Step 4)	Fig. 7 "Fire All Triggers = True"
Repeat full experiment N times	Repeat lifecycle wraps entire sequence; iterations logged	§ 4.4; 7 (Step 7)	Fig. 10 "Repeat = N"; `LOG-1` shows "Execution 0/1/..."
Emergency Stop immediate halt	E-Stop drops outputs and broadcasts event	§§ 3.4, 4.6, 6.4	Fig. 10 "Emergency" button; `LOG-1`: "Emergency stop: All motors halted."
Graceful Stop	Stop at the next safe point	§§ 3.4, 4.6	Fig. 10 "Stop"; `LOG-1`: "Stopping ..." then reset
Motor Check health routine	Backend quick forward/back per axis + logs	§§ 3.3, 4.6	Fig. 4 motor check
Zero Force baseline & **Calibration** factors	Zeroing + Apply Calibration in UI; software scaling	§§ 3.3, 5.1–5.2	Fig. 4 "Zero Force" / "Apply Calibration"; Table 1
Global force safety cap respected	Safety bound in trigger evaluation	§§ 5.3, 6.4	`LOG-1`: no overshoot beyond cap during runs
Live **Fx/Fy/Fz** (and shear) telemetry	Socket.IO streams; UI telemetry panel	§§ 3.4, 4.2	Fig. 12 real-time panel; Fig. 13
Live **step counts** per axis	Backend emits counts; UI status	§§ 3.4, 4.2	Fig. 12
3D visualizer (spatial context)	Three.js scene with axes/motion cues	§ 3.2	Fig. 13 full UI
State Export/Import (JSON)	Download current setup; Upload to rehydrate	§§ 3.3, 4.5, 7 (Step 8)	Fig. 11 "Export / Upload / Download"; `STATE-1` JSON file
Log export for analysis	Timestamped human-readable log file	§§ 3.3, 4.5, 9	`LOG-1` exported file (attach)
Post-run reset to origin	Walk-back using tracked steps	§§ 4.4, 6.4, 9	`LOG-1`: "Resetting motors to initial state."
Manual Control safety mode	Click-and-release (no continuous joystick)	§§ 3.2, 7 (Step 9)	Fig. 12 manual controls
Interactive UI preview (web)	Static demo of the interface	Artifacts / Appendix	Get UI experience here — https://roshangupta00750.github.io/geckotestbed/

A. Project Management

A.1. Work Breakdown Structure



A.2. Gantt Chart



A.3. Work Package Description

		WP 1100
Title	**Mechanics of Hierarchical Adhesion Structures of Geckos**	**Page:** 1 of 1
Responsible	Roshan, Ateeb, Vassili	**Version:** 1.0
		Date: 21.04.2025
Start	Week 1	
End	Week 3	**Duration**: 1 Weeks
Processed by	Roshan Kumar Gupta	

Goals:
- Investigate bioinspired dry adhesives for space use
- Optimize testbed for hierarchical adhesion analysis

Input:
- Journal papers on bioinspired dry adhesives
- Conference papers on gecko adhesion structures

Connections to other WPs:
- Collaboration with TU Berlin and Fraunhofer team

Tasks:
- Literature research on mechanics of hierarchical adhesion structures of geckos

Results:
- Basic knowledge about mechanics of hierarchical adhesion structures of geckos
- Deeper knowledge about mechanics of hierarchical adhesion structures of geckos

		WP 1200
Title	**Space Applications for Gecko-Inspired Adhesives**	**Page:** 1 of 2
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date:28.04.2025
Start	Week 2	
End	Week 4	**Duration**: 1 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:

• Investigate space applications of gecko-inspired adhesives

• Optimize testbed for space-relevant adhesion analysis

Input:

• Journal papers on gecko-inspired adhesives

• Conference papers on space applications of adhesives

Connections to other WPs:

bullet Collaboration with TU Berlin and Fraunhofer team

Tasks:

• Literature research on space applications for gecko-inspired adhesives

Results:

• Basic knowledge about space applications for gecko-inspired adhesives

• Deeper knowledge about space applications for gecko-inspired adhesives

		WP 1300
Title	**Bioinspired Dry Adhesives for Space Applications**	**Page:** 1 of 3
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 05.05.2025
Start	Week 2	
End	Week 5	**Duration**: 4 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
- Investigate bioinspired dry adhesives for space use
-

Input:
- Documentation of literature topic 3

Connections to other WPs:
- **WP 3200** Research for subsystems/units/parts/components
- **WP 3200** Gather technical solutions for novel design

Tasks:
- Literature research on literature topic 3

Results:
- Select technical solutions of literature topic 3 as proposal for thingy to be developed

		WP 2100
Title	**Setting Up Python Development Environment in Raspberry Pi**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 19.05.2025
Start	Week 4	
End	Week 5	**Duration**: 1 Week
Processed by	Ateeb, Roshan, Vassili	

Goals:

• Setup development environment including Github, Visual Studio IDE and SSH for remote access.

Input:

• None

Connections to other WPs:

• This is a base WP, other WPs are dependent on this.

Tasks:

• Same as in goals

Results:

• An environment is setup which enables contributors to start working.

		WP 2200
Title	**Basic Control of Sensors and Motors**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 26.05.2025
Start	Week 5	
End	Week 7	**Duration**: 3 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
• Motors and Sensors are controlled through Python Interface

Input:
• Results from WP 2100

Connections to other WPs:
• Future WPs depend on this. Since we are working serially.

Tasks:
• Produce Code which can control motors and sensors of testbed when executed.

Results:
• Yet to be worked on.

		WP 2300
Title	**Backend Development with Focus on Modularity**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 10.06.2025
Start	Week 7	
End	Week 9	**Duration**: 2 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:

• A fully fledged, modular Backend framework for controlling all components of TestBed

Input:

• Results from WP2200 are extended here.

Connections to other WPs:

Tasks:

• Write a framework-like which controls all sensors and motors of testbed.

• Make sure the application is modular and testbed modules can be swapped with no effect on working of application

Results:

• An executable file for testbed control.

		WP 3100
Title	**Designing a UI for Displaying Realtime Data and User Inputs**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 19.05.2025
Start	Week 4	
End	Week 5	**Duration**: 2 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:

• To have an approved UI design, which can be implemented in the next work package.

Input:

• Team input to decide if a UI deisgn has all the needed elements to display testbed state and capture user inputs.

Connections to other WPs:

• This WP is base for all UI related sub-tasks.

Tasks:

• To create a UI design using Figma or other design prototyping software.

Results:

• An approved UI specification which is ready to be developed.

		WP 3200
Title	**Developement Phase 1: Using JS Libraries, HTML, CSS**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 26.05.2025
Start	Week 5	
End	Week 8	**Duration**: 4 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
• A fully developed and deployment-ready UI application

Input:
• Design from the previous WP.
• Input-B for Sub Task 2.2

Connections to other WPs:
• Depends on **WP 3100**

Tasks:
• Create a structure of the application using HTML.
• Use a suitable JS library for two data-bindings.
• Use CSS for sufficent styling.

Results:
• A deployable JS bundle, which can be used to display data from python backend.

		WP 3300
Title	**Finalizing Frontend Application**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 17.07.2025
Start	Week 8	
End	Week 9	**Duration**: 1 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
• Finalize the frontend code and iron out any bugs.

Input:
• Input from WP 3200

Connections to other WPs:
• Dependent on **WP 3200**

Tasks:
• Test the frontend application and iron out any bugs if found.

Results:
• A well tested and optimized frontend application made using HTML, JS and CSS

		WP 4100
Title	**Developing and Initializing Web-Sockets**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 31.07.2025
Start	Week 10	
End	Week 11	**Duration**: 2 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
• Learn about Web-Sockets and use them as a communication channel.

Input:
• Input from **WP 2300** and **WP 3300**

Connections to other WPs:
• Dependent on **WP 2300** and **WP 3300**

Tasks:
• Learn about Web sockets and use the relevent libraries in both frontend and backend to start communication channel.

Results:
• Successful data transmission between frontend and backend.

		WP 4200
Title	**Running Real time Communications between Sub-Systems**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 07.08.2025
Start	Week 11	
End	Week 12	**Duration**: 1 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
• Successful Real-time data transmission between frontend and backend.

Input:
• None

Connections to other WPs:
• Depends on previous **WP 4200**

Tasks:
• As in title above.

Results:
• Final seamless communication between sub-systems.

		WP 5000
Title	**Documentation**	**Page:** 1 of 1
Responsible	Ateeb, Roshan, Vassili	**Version:** 1.0
		Date: 21.04.2025
Start	Week 1	
End	Week 16	**Duration**: 16 Weeks
Processed by	Ateeb, Roshan, Vassili	

Goals:
• Documentation of work on all WPs

Input:
• WPs 1000-4000

Connections to other WPs:

Tasks:
• Produce draft
• Revise draft

Results:
• Final documentation