

Design and Implementation of a Motor-Based Kinetic Morphing System

Lee Yesung

Hongik University

February 2026

Design and Implementation of a Motor-Based Kinetic Morphing System

Lee Yesung, B.E.

THESIS

Presented to the Faculty of the Graduate School of

Hongik University

In Partial Fulfillment of

the Requirements

for the Degree of

MASTER OF ENGINEERING IN SMART DESIGN

ENGINEERING

Hongik University

January 2026

Design and Implementation of a Motor-Based Kinetic Morphing System

Approved by
Thesis Committee

Supervising Professor, Heechang Moon

Kicheol Pak

Junghun Kim

ABSTRACT

The increasing demand for design variety in contemporary manufacturing, combined with a growing cultural interest in kinetic and computational media, challenges conventional assumptions about fixed physical form. This research presents the design and implementation of a motor-based kinetic morphing system—a reconfigurable pin-array surface capable of forming static height-based geometries from multiple types of digital input. The project does not seek to replicate industrial forming or continuous kinetic expression; instead, it demonstrates how an integrated hardware–software–AI pipeline can translate human intention into physical topography.

The system is built across three interconnected layers: hardware engineering, computational control, and human–machine interaction. Mechanically, a grid of 25 stepper-motor-driven pins forms the morphing surface, each actuated through a distributed network of five Arduino Mega microcontrollers. This architecture enables reliable, coordinated actuation without requiring high-end drivers, precision sensors, or centralized real-time computation. At the software level, a Python-based interface provides three modes of control—manual adjustment, data-driven heightmap generation, and AI-driven command synthesis—reflecting the project’s interest in evolving modes of human–machine interaction.

A key contribution of this work is the integration of a locally hosted large language model (LLM) as a mediator between natural-language prompts and machine actuation. The system employs a dual-channel structure: an action channel, which generates a strictly formatted static SET command controlling all 25 motors, and a reasoning channel, which provides optional text-based explanations of how the generated shape interprets the user’s request. This separation allows the LLM to participate in the design workflow—supporting interpretation, reflection, and explanation—while remaining bounded by the mechanical realities of the physical system.

By combining hardware design, computational pipelines, and AI-mediated interaction, this research demonstrates a fully operational kinetic morphing platform built from first principles. Rather than presenting a complete intelligent surface, the work establishes a factual and extensible foundation from which future developments—such as sensing, continuous motion, or machine-trained actuation models—may emerge. The project contributes to the broader discourse at the intersection of programmable matter, computational design, and human–AI collaboration.

Table of Contents

Abstract	1
Table of Contents	3
List of Tables	6
List of Figures	7

1. INTRODUCTION 10

1.1	Background	10
1.2	Research Problem and Objectives	13
1.3	Conceptual Framework	16
1.4	Technical Overview	18
1.5	AI as Design Mediator	20
1.6	Contribution and Significance	21
1.7	Structure of the Thesis	22

2. LITERATURE AND PRECEDENT REVIEW 23

2.1	Review Focus and Analytical Lens	23
2.2	Pin-Based Surfaces as Discretized Height-Field Systems	23
2.3	Reconfigurable Pin Tooling in Manufacturing: Forming, Fixturing, and Repeatability	24
2.4	Shape-Changing Surfaces in Kinetic Art and Interactive Media	25
2.5	Interaction and Control Paradigms for Morphing Surfaces	27
2.6	AI-Mediated Design Interfaces: Semantic Inputs Under Physical Constraints	28
2.7	Synthesis: Literature-Derived Tensions and Their Implications	29

3.	SYSTEM DESIGN AND DEVELOPMENT	30
3.1	System Overview	30
3.2	Mechanical Architecture	33
3.3	Embedded Electronics Architecture	38
3.3.1	Custom 5-Driver Shield for Arduino Mega (DRV8825 Carrier Board)	40
3.4	Software Framework	42
3.4.1	Arduino Firmware	43
3.4.2	Python GUI and Control Software	46
3.4.2.1	Manual Control Tab	52
3.4.2.2	Image-to-Heightmap Tab	54
3.4.2.3	AI-Assisted Tab	56
3.4.2.4	3D Visualization	58
3.5	AI Reasoning and Interaction Layer	59
3.5.1	Dual-Channel Structure	59
3.5.1.1	Action Channel: Strict Motor Command Generation	60
3.5.1.2	Reasoning Channel: Optional Explanatory Layer	62
3.5.2	AI as Mediation Layer	63
3.5.3	Limitations of AI Integration	64
3.6	System Integration	64
3.6.1	Data Flow Across Layer	65
3.6.2	Practical Integration Constraints	67
3.6.3	Integration Verification Through LLM-Control Tests	67
3.6.4	Summary	68
4.	IMPLEMENTATION AND EXPERIMENTS	69
4.1	Prototype Fabrication and Assembly	69
4.2	Graduation Exhibition Demonstration	71
4.3	Homing, Reference State, and Repeatability	72
4.4	Experimental Methodology	72

4.5	Image-to-Heightmap Trials (Physical Experiments)	73
4.6	LLM-Control Tests and Category-Based Prompt Framework	74
4.7	Results and Evaluation	76
4.7.1	Image-to-Heightmap Results	77
4.7.2	LLM Action Channel Results	78
4.7.3	LLM Reasoning Channel Results	87
4.7.4	Summary of Results	91
5.	DISCUSSION	92
5.1	Overview	92
5.2	Technical Implications of a Distributed Morphing Surface	92
5.3	Interaction Implications of Language-Mediated Control	93
5.4	Role of Explanation and Interface Transparency	94
5.5	Limitations and Future Directions	94
5.6	Summary	95
6.	CONCLUSION	96
6.1	Research Summary and Outcomes	96
6.2	Contributions	97
6.3	Limitations	98
6.4	Future Research Directions	99
6.5	Closing Statement	100
	Bibliography	101
		104
	Appendix (Listing R-1 to R-7)	105

List of Tables

Table 1-1.	RESEARCH AIMS AND IMPLEMENTATION FOCUS	14
Table 1-2.	OVERARCHING RESEARCH OBJECTIVE	15
Table 1-3.	SUMMARY OF THE THESIS CONTRIBUTIONS ACROSS DESIGN ENGINEERING, HUMAN-ROBOT INTERACTION, COMPUTATIONAL DESIGN, AND KINETIC ART	21
Table 2-1.	INTERACTION AND CONTROL PARADIGMS FOR PIN-BASED MORPHING SURFACES	27
Table 2-2.	LITERATURE-DERIVED TENSIONS AND RESEARCH IMPLICATIONS ACROSS DOMAINS	29
Table 3-1.	ELECTRONIC SYSTEM SUMMARY TABLE	42
Table 3-2.	OVERALL SYSTEM STACK	65
Table 4-1.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R1	88
Table 4-2.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R2	88
Table 4-3.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R3	89
Table 4-4.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R4	89
Table 4-5.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R5	90
Table 4-6.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R6	90
Table 4-7.	PROMPT AND SET COMMAND FOR REASONING EXAMPLE R7	91

List of Figures

Figure 1-1.	BMW Kinetic Sculpture (2008)	08
Figure 1-2.	MIT INFORM SHAPE DISPLAY	09
Figure 1-3.	THESIS POSITIONING IN A HYBRID DOMAIN ..	12
Figure 1-4.	MULTI-MODAL INPUT-TO-ACTUATION PIPELINE	13
Figure 1-5.	DETERMINISTIC WORKFLOW VS MEDIATED EMERGENCE	14
Figure 1-6.	DUAL-CHANNEL AI SEPARATION	14
Figure 1-7.	INPUT MODES BY ABSTRACTION LEVEL	14
Figure 2-1.	AI-MEDIATED DESIGN INTERFACES: SEMANTIC INPUTS UNDER PHYSICAL CONSTRAINTS	14
Figure 3-1.	OVERALL ARCHITECTURE OF THE MKMS, ILLUSTRATING THE RELATIONSHIP BETWEEN THE MECHANICAL SURFACE, EMBEDDED CONTROLLERS, HOST SOFTWARE, AND THE LLN INTERFACE	14
Figure 3-2.	SINGLE ACTUATOR UNIT	14
Figure 3-3.	INTERNAL STRUCTURE OF THE ACTUATOR UNIT	14
Figure 3-4.	CUTAWAY VIEW OF A SINGLE ACTUATOR UNIT	14
Figure 3-5.	ASSEMBLED 5X5 ACTUATOR ARRAY AND SUPPORT FRAME	14
Figure 3-6.	LOWER-END LIMIT SWITCH USED FOR ACTUATOR HOMING REFERENCE	14
Figure 3-7.	NEOPIXEL LED HEIGHT-CORRELATED COLOR FEEDBACK INTEGRATED IN SEVERAL ACTUATOR UNITS	14
Figure 3-8.	CUSTOM DRV8825 DRIVER CARRIER BOARD STACKED ON A SINGLE ARDUINO MEGA	14
Figure 3-9.	DISTRIBUTED CONTROL ARCHITECTURE FOR THE 5×5 ACTUATOR ARRAY	14
Figure 3-10.	SERIAL COMMAND PARSING AND SET VALUE-COUNT VALIDATION (ARDUINO FIRMWARE)	14

Figure 3-11.	HEIGHT BOUNDS ENFORCEMENT AND MILLIMETER-TO-STEP CONVERSION I EXECUTION (ARDUINO FIRMWARE)	14
Figure 3-12.	SYNCHRONIZED STEPPING USING A SHARED STEP INDEX ACROSS FIVE MOT (ARDUINO FIRMWARE)	14
Figure 3-13.	HOMING ROUTINE USING BOTTOM LIMIT SWITCH TO ESTABLISH THE ZERO (ARDUINO FIRMWARE)	14
Figure 3-14.	CONTROL SOFTWARE FLOWCHART FOR THE PYTHON HOST APPLICATION	14
Figure 3-15.	ARDUINO IDENTIFICATION BY SERIAL NUMBER AND MULTI-PORT CONNEC (EXCERPT)	14
Figure 3-16.	IMAGE DOWNSAMPLING AND INTENSITY-TO-HEIGHT MAPPING (EXCERPT)	14
Figure 3-17.	PARTITIONING 25 HEIGHTS INTO FIVE PER-BOARD SET COMMANDS (EXCERPT) .	14
Figure 3-18.	PARSING AND VALIDATING SET OUTPUT PRIOR TO EXECUTION (EXCERPT)	14
Figure 3-19.	CONTROL FLOW FOR THE SLIDE CONTROL MODE	14
Figure 3-20.	MANUAL CONTROL GUI WITH PER-PIN SLIDERS AND 3D HEIGHT PREVIEW	14
Figure 3-21.	CONTROL FLOW FOR THE IMAGE-TO-HEIGHTMAP CONTROL MODE	14
Figure 3-22.	IMAGE-TO-HEIGHTMAP CONTROL GUI WITH 3D HEIGHT PREVIEW	14
Figure 3-23.	CONTROL FLOW FOR FOR THE LLM ASSISTANT MODE	14
Figure 3-24.	AI-ASSISTED GUI TAB WITH DUAL-CHANNEL LLM INTERFACE FOR COMM GENERATION AND OPTIONAL REASONING OUTPUT	14
Figure 3-25.	ACTION-CHANNEL SYSTEM PROMPT ENFORCING A COMMAND-ONLY OUTP CONTRACT (EXCERPT)	14
Figure 3-26.	HOST-SIDE EXTRACTION AND VALIDATION PRIOR TO DISPATCH (EXCERPT)	14
Figure 3-27.	REASONING-CHANNEL PROMPT CONSTRAINTS AND CONTEXT PACKAGING	14

Figure 4-1.	AN FDM PRINTER (CREALITY K1C) USED FOR FABRICATION OF PROTOTYPE COMPONENTS	14
Figure 4-2.	3D-PRINTED STRUCTURAL AND ACTUATOR COMPONENTS PRIOR TO ASSEMBLY	14
Figure 4-3.	PARTIALLY ASSEMBLED 5×5 ACTUATOR FRAME DURING INSTALLATION OF ARDUINO MEGA BOARDS	14
Figure 4-4.	PARTIALLY ASSEMBLED 5×5 ACTUATOR FRAME WITH FIVE ARDUINO MEGA BOARDS INSTALLED PRIOR TO FINAL CABLE MANAGEMENT	14
Figure 4-5.	GRADUATION EXHIBITION DEMONSTRATION: IMAGE-TO-HEIGHTMAP CONTROL	14
Figure 4-6.	GRADUATION EXHIBITION DEMONSTRATION: LLM-ASSISTED CONTROL MODIFICATION	14
Figure 4-7.	LLM-CONTROL TESTS AND CATEGORY-BASED PROMPT FRAMEWORK	14
Figure 4-8.	IMAGE-TO-HEIGHTMAP VALIDATION	14
Figure 4-9.	CATEGORY 1: BASIC GEOMETRIC SHAPES	14
Figure 4-10.	CATEGORY 1: NON-PLANAR GEOMETRIC SHAPES	14
Figure 4-11.	CATEGORY 2: TOOLING PRIMITIVES	14
Figure 4-12.	CATEGORY 3: HOLDING AND SUPPORT SHAPES	14
Figure 4-13.	CATEGORY 4: SYMBOLIC LETTER PATTERNS: “HONGIK”	14
Figure 4-14.	CATEGORY 4: SYMBOLIC ICONS AND PATTERNS	14
Figure 4-15.	CATEGORY 5 (5-1): LOWERING ONLY THE HIGHEST PINS	14
Figure 4-16.	CATEGORY 5 (5-2): SMOOTHING NEIGHBOUR DIFFERENCES	14
Figure 4-17.	CATEGORY 5 (5-3): UNIFORM HEIGHT OFFSET	14
Figure 4-18.	CATEGORY 5 (5-4): LEFT–RIGHT MIRRORING	14
Figure 4-19.	CATEGORY 5 (5-5): SNAPPING HEIGHTS TO 10 MM GRID	14

1. Introduction

1.1 Background

The changing landscape of contemporary manufacturing, computational design, and interactive media has amplified the need for systems capable of physical transformation.¹

Traditional manufacturing tools, particularly those relying on fixed molds, dies, or rigid jigs—excel at high-volume production² but struggle with the increasing demand for variation and personalization. As industries move toward more diverse product lines and shorter development cycles³, the ability to generate new physical geometries without fabricating new tooling becomes increasingly attractive⁴.

Simultaneously, a parallel evolution is occurring in the fields of kinetic art and interactive design. Researchers and artists are experimenting with physical motion as a medium, using actuated surfaces to create breathing forms, volumetric animations, and dynamic installations. Landmark projects such as the BMW Kinetic Sculpture⁵ and MIT's inFORM⁶ illustrate how reconfigurable surfaces can serve as expressive spatial interfaces. These works demonstrate a speculative vision of programmable matter⁷ physical substrates that transform like pixels in three-dimensional space.

¹ Sá et al., "State of The Art on Functional Fabrication.",1-2

² Pelin et al., "The Use of Additive Manufacturing Techniques in the Development of Polymeric Molds.", 12,15

³ DeVOR et al., "Agile Manufacturing Research"; Olson and Sharma, "Beating the Commoditization Trend.",813, 817-818

⁴ Munro and Walczyk, "Reconfigurable Pin-Type Tooling," 2007., 551-552, 561

⁵ *Kinetic Sculpture — The Shapes of Things to Come.*

⁶ Follmer et al., "inFORM."

⁷ Goldstein et al., *Programmable Matter*; MacLennan, "Universally Programmable Intelligent Matter Summary.",99, 101

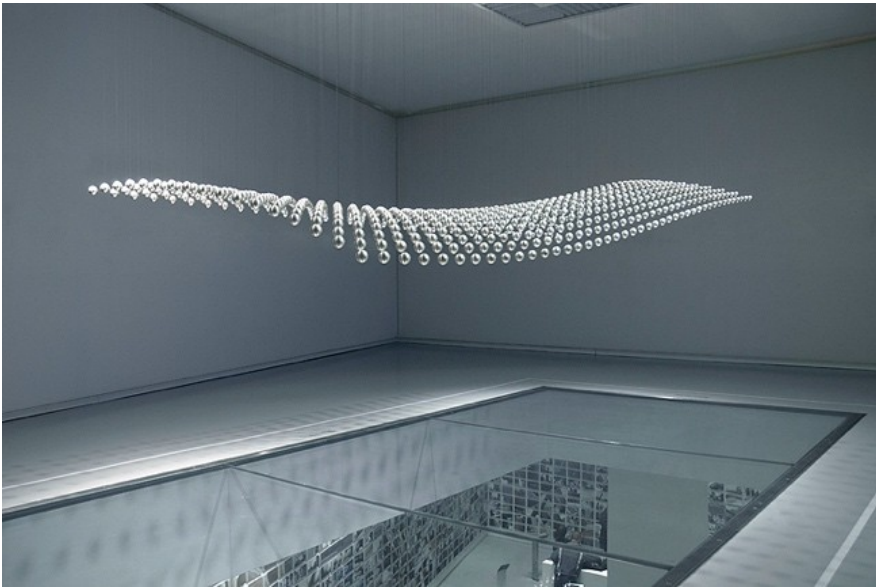


Figure 1-1. BMW Kinetic Sculpture (2008), a large-scale reconfigurable surface composed of suspended spheres.

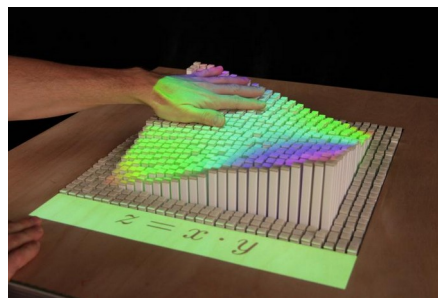
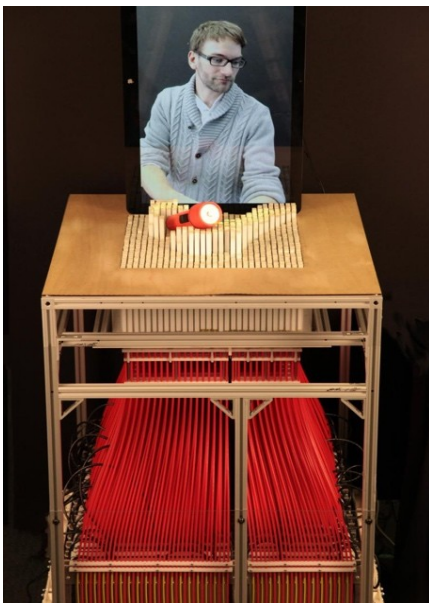


Figure 1-2. MIT Media Lab inFORM shape display, a pin-based interface for remote physical interaction and dynamic geometry.

Although these industrial and artistic trajectories share a common technical foundation: arrays of actuated pins capable of changing height, they differ drastically in their

priorities. Industrial systems focus on strength and repeatability, while artistic systems prioritize responsiveness, motion, and interaction. What remains largely unexplored is a research space that treats a reconfigurable surface not as a single-purpose tool or an artistic display but as an integrated ecosystem involving mechanical engineering, computational control, human-machine interaction, and AI-mediated expression.

This thesis situates itself in that hybrid domain. Rather than optimizing for force capacity or high-resolution choreography, the goal is to design and implement an end-to-end kinetic morphing system that integrates hardware engineering, computational pipelines, interface design, and AI-driven mediation. The result is a fully functioning proof-of-concept (POC) that physically manifests digital information through multiple layers of human control.

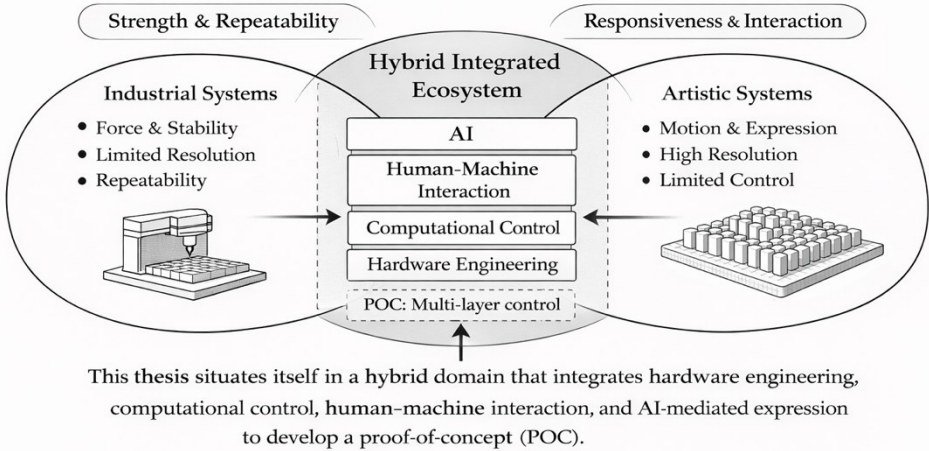


Figure 1-3. Thesis positioning in a hybrid domain.

1.2 Research Problem and Objectives

Reconfigurable pin-based systems have emerged in both industrial and artistic contexts, yet they often emphasize only a single dimension of capability. Industrial reconfigurable pin tooling (RPT) systems prioritize force capacity and mechanical stability⁸ but offer limited resolution or adaptability. Artistic and interactive installations explore motion and expression⁹ but typically lack the structural or computational rigor required for controlled digital manipulation. Across these domains, the integration of hardware engineering, computational control, human-machine interaction, and AI-mediated interpretation remains underexplored¹⁰

This research examines how a reconfigurable surface can be constructed as a complete system. The system integrates mechanical actuation, software control pipelines, and multi-level interfaces ranging from manual manipulation to AI-driven command generation. Rather than focusing on load-bearing forming or real-time responsive behavior, the project investigates how digital and linguistic inputs can be translated into static, reliable physical geometries through a fully implemented hardware-software-AI workflow.

⁸ Walczyk and Longtin, “Fixturing of Compliant Parts Using a Matrix of Reconfigurable Pins.”, 767, 771

⁹ Han et al., “PneuModule”; Nakagaki et al., “TRANS-DOCK.”

¹⁰ Yin et al., “Imaginal Thinking-Based Human-Machine Design Methodology for the Configuration of Reconfigurable Machine Tools.”

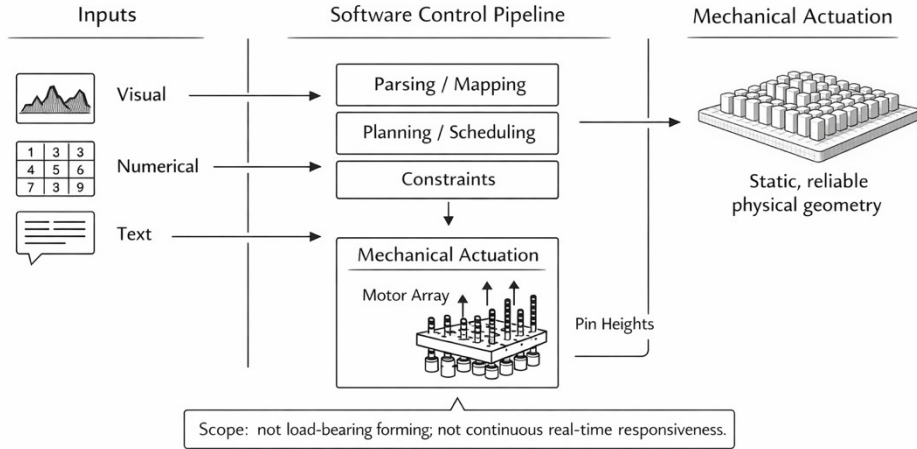


Figure 1-4. Multi-modal input-to-actuation pipeline.

Accordingly, the aims of this research are to develop a motor-based kinetic morphing system that is:

Aim / Requirement	Definition in this study	Implementation focus
Mechanical functionality	Static height-based shaping (not load-bearing forming)	25-pin surface, stepper actuation, repeatable homing, stable static output
Computational integration	Digital representations → physical configurations	Python pipeline for manual input, heightmaps, and structured commands
AI assistance	LLM generates shape commands + optional reasoning	Dual-channel LLM: action (SET) vs reasoning (non-controlling explanation)

Table 1-1. Research aims and implementation focus.

Objective area	Objective (what this research will do)	Implementation (system component)	Evaluation focus (what you will assess)
Hardware Engineering	Design and build a 25-pin morphing surface driven by stepper motors with distributed control across five boards, ensuring stable actuation and repeatable homing.	25-pin motor array; five independent microcontroller boards; homing logic and calibration routine	Homing reliability; actuation consistency; repeatability of static outputs
Computational Control	Develop a consolidated control pipeline that converts diverse inputs into deterministic pin-height configurations for static actuation.	Python control software; mapping/parsing layer; command formatting and dispatch	Correctness of translation (input → pin heights); robustness of pipeline execution
Human–Machine Interface (HMI)	Construct a multilevel interaction workflow enabling progression from manual manipulation to data-driven inputs and LLM-mediated control.	GUI for manual control; image/heightmap input; structured interaction flow	Usability; clarity of interaction stages; reduced operator ambiguity
AI Integration	Implement a dual-channel LLM system separating actionable outputs from explanations to preserve deterministic hardware behavior.	Action channel generating strictly formatted SET commands; reasoning channel generating explanatory text	Command validity and determinism; usefulness of reasoning without behavioral interference
System Evaluation	Evaluate the integrated system as an end-to-end workflow, not as an industrial forming or continuous-motion platform.	Test protocol + documentation (logs, repeat trials, user-self evaluation)	Reliability; usability; workflow coherence (not load/precision/continuous motion)

Table 1-2. Overarching Research Objective

1.3 Conceptual Framework

The proposed morphing surface operates as a hybrid platform that merges mechanical actuation, computational control, and design mediation. Each pin can be understood simultaneously as a unit of physical displacement and as a discrete element of representation—functioning analogously to a pixel in a spatial display. Through this dual identity, the system positions itself not only as a technical apparatus but also as a medium capable of embodying computational ideas through physical form.

In traditional industrial workflows, form generation is deterministic: geometry is predefined, and machines execute fixed toolpaths with little ambiguity. In contrast, this project investigates how form may emerge through a layered interaction between human intention, digital representation, and AI-mediated interpretation. Rather than attempting real-time responsive behavior or continuous motion, the research focuses on how static shapes can be generated through multiple modes of input—manual control, data-driven heightmaps, and natural-language prompts.

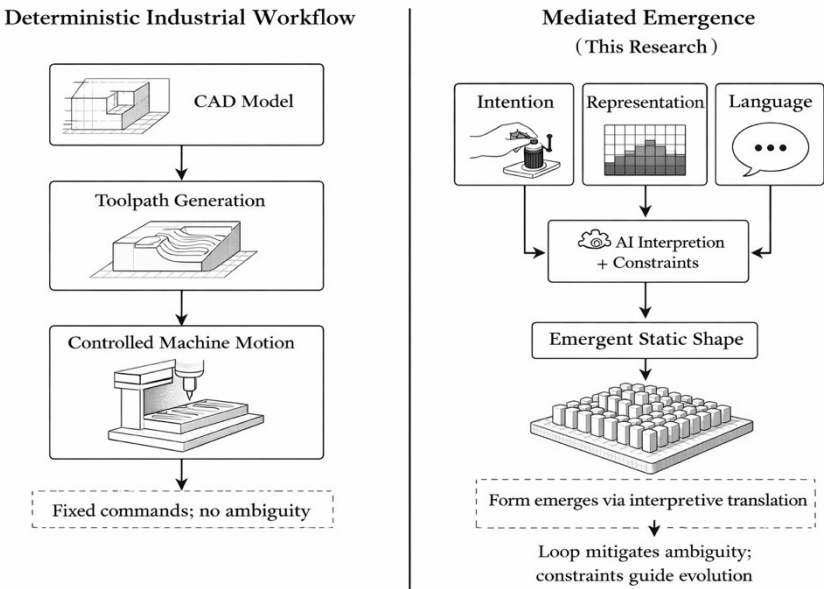


Figure 1-5. Deterministic workflow vs mediated emergence.

A key conceptual element is the dual-channel AI framework, where physical action and computational reasoning are intentionally separated. The action channel produces a strictly formatted SET command that defines the 25-pin height configuration. The reasoning channel, activated only when requested, articulates the logic behind the generated shape. This explanation is not used to adjust motion or modify the machine's behavior; instead, it serves to reveal how the AI interpreted the user's request within the constraints of the physical system.

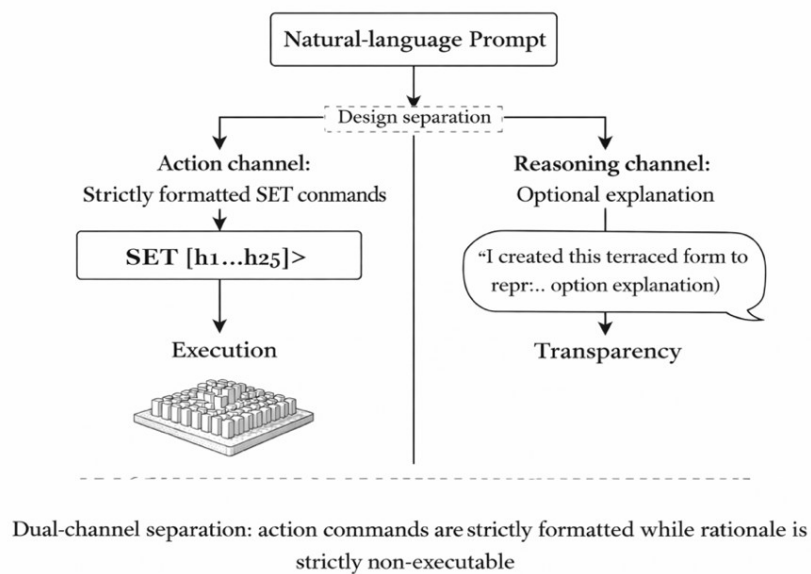


Figure 1-6. Dual-channel AI separation.

By making the AI's interpretive process explicit, the system creates a reflective relationship between human, AI, and machine. The surface does not simply execute commands but communicates the conceptual pathway that leads to a given configuration. This establishes a subtle form of dialogue—one centered not on real-time interaction but on computational transparency. The framework highlights how language-based models can participate in physical design workflows, offering insight into the translation of abstract descriptions into spatial form.

1.4 Technical Overview

The hardware configuration of the final proof-of-concept consists of a 5×5 grid of 25 stepper motors, each coupled to a lead screw and equipped with an individual bottom limit switch for homing. To meet the I/O and timing requirements of controlling 25 motors, the system is distributed across five Arduino Mega 2560 boards, with each board responsible for five actuators. This multi-microcontroller architecture ensures stable pulse timing, avoids electrical noise accumulation, and enables simultaneous homing across the entire grid. A host PC runs a Python-based control interface, communicating with all five microcontrollers over serial connections.

The software layer supports three forms of input that reflect the project’s interest in progressively higher levels of human–machine interaction:

1. **Manual Mode.** Users directly manipulate the height of each pin through sliders on the graphical interface. This mode provides low-level control and is used primarily for testing, calibration, and exploratory shaping.
2. **Data-Driven Mode (Image-to-Heightmap).** 2D images are converted into a 25-value heightmap in row-major order. This process produces a single static SET command, allowing the surface to display relief-like interpretations of digital imagery without continuous motion or animation.
3. **AI-Driven Mode.** A locally hosted large language model (LLM) interprets natural-language prompts and generates a strictly formatted SET command defining the desired pin heights. A separate reasoning channel can optionally provide a textual explanation of why the generated shape corresponds to the user’s request.

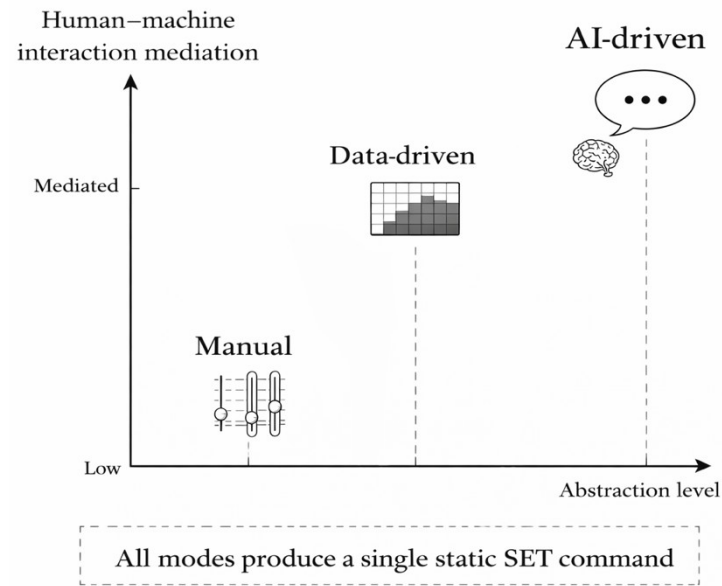


Figure 1-7. Input modes by abstraction level.

A small subset of motor units incorporates NeoPixel LEDs that provide simple visual feedback by mapping height values to color intensity. These LEDs do not encode system state or motion and serve only as representational indicators.

Together, these components form a cohesive technical stack that integrates mechanical actuation, distributed control infrastructure, a multi-mode software pipeline, and AI-mediated command generation within the constraints of a static kinetic surface.

1.5 AI as Design Mediator

In this system, artificial intelligence functions not as an automation tool but as a mediator between human intention and machine actuation. Traditional CAD/CAM workflows transform predefined geometry into deterministic toolpaths¹¹. In contrast, the integration of a large language model introduces a semantic layer in which descriptions, metaphors, and qualitative concepts can be translated into a physical configuration of pins. Although the system ultimately produces a single static SET command, the interpretive process behind that command reflects a different mode of design interaction, one grounded in language rather than explicit geometric specification.

This mediation is supported by the system's dual-channel AI framework. The action channel generates the constrained motor command, while the reasoning channel articulates the conceptual logic behind the chosen configuration. Because the reasoning does not influence the physical actuation, it serves as a reflective layer that makes the AI's interpretive process visible to the user. The designer is invited not only to command the machine but also to understand how the system internally negotiates meaning within the limits of its mechanical capabilities.

From a cultural and design perspective, the morphing surface becomes a site where computational interpretation acquires physical form. The static configuration of pins, though not performative or dynamic, acts as a tangible artifact of how the AI processed and transformed the user's language. In this way, the system frames AI as a partner in the design workflow: not a generator of autonomous actions, but a collaborator whose interpretations can be examined, questioned, and redirected. This relationship positions the surface as a modest but compelling example of AI embodiment, where algorithmic reasoning is expressed through material displacement.

¹¹ Xu * et al., "STEP-Compliant NC Research.", 25-26

1.6 Contribution and Significance

This research contributes to multiple domains, each aligned with the system’s integrated approach to hardware engineering, computational control, human–machine interaction, and AI-mediated interpretation.

Domain	Contribution
Design Engineering	Demonstrates a fully implemented 25-pin kinetic morphing surface built from the ground up, integrating distributed microcontroller architecture, homing logic, and coordinated actuation.
Human–AI Interaction	Introduces a dual-channel AI framework in which a large language model generates constrained SET commands and provides optional explanatory reasoning, enhancing interpretability without affecting actuation.
Computational Design / Digital Fabrication	Provides a complete pipeline for translating manual inputs, image-derived heightmaps, and natural-language prompts into static physical geometries through unified software control.
Kinetic Art and Media	Repositions the morphing surface as a medium for reflective expression, where static configurations become artifacts of AI interpretation rather than performative motion.

Table 1-3. Summary of the thesis contributions across design engineering, human-AI interaction, computational design, and kinetic art

Beyond technical implementation, this project contributes to an emerging view of design systems in which intelligence and materiality interact across multiple layers. Although the platform does not perform continuous motion or real-time responsiveness, it functions as a hybrid interface that reveals how language, computation, and physical form can be woven together into a coherent workflow. In this sense, the morphing surface becomes a reflective partner in the design process: a machine that embodies the translation of concepts into tangible spatial configurations.

1.7 Structure of the Thesis

- **Chapter 2. Literature and Precedent Review.** Reviews prior work in reconfigurable surfaces, adaptive manufacturing, kinetic art systems, and AI-mediated design workflows. This chapter positions the project within both engineering and design-research contexts.
- **Chapter 3. System Design and Development.** Describes the hardware, electronics, firmware, and software architecture of the system, including the distributed multi-microcontroller setup, homing logic, Python control interface, and the three input modes (manual, data-driven, and AI-driven).
- **Chapter 4. Implementation and Experiments.** Documents the fabrication of the 25-motor surface, the calibration process, and an evaluation focused on system reliability and command validity. The chapter does not include load-testing, continuous motion experiments, or user studies.
- **Chapter 5. Discussion.** Reflects on the system’s technical performance, the role of LLM-mediated interpretation in physical design workflows, and broader conceptual implications of treating AI as a participant in material expression.
- **Chapter 6. Conclusion.** Summarizes the research contributions, clarifies limitations, and outlines future directions such as continuous motion control, sensing integration, and machine-specific AI training in simulation environments.

2. Literature and Precedent Review

2.1 Review Focus and Analytical Lens

Research on reconfigurable surfaces spans manufacturing tooling, kinetic and interactive media, tangible interfaces, and AI-mediated design systems. Building on the problem framing established in Chapter 1, this chapter reviews these domains through a single analytical lens: pin-based surfaces as discretized height-field platforms. This framing makes comparable a set of recurring tensions in prior work, including resolution versus scalability, expressiveness versus mechanical rigor, and semantic intent versus deterministic control, and clarifies the unresolved challenges that arise when diverse inputs must be translated into stable, repeatable physical configurations.

2.2 Pin-Based Surfaces as Discretized Height-Field Systems

A pin matrix represents geometry through a grid of independently actuated vertical displacements. Unlike continuous fabrication processes that define form through uninterrupted toolpaths, pin-based systems approximate surfaces by sampling a target geometry into discrete height values. This discretized representation produces several persistent characteristics across both industrial and artistic precedents¹².

¹² Everitt, *Digital Fabrication Approaches for the Design and Development of Shape-Changing Displays.*, 103-105

First, geometric fidelity is bounded by grid resolution: increased pin density improves approximation but amplifies actuator count, wiring and control complexity, calibration effort, and potential failure modes. Second, discretization yields visible “stepping” artifacts that are not merely errors but structural properties of the medium¹³. Existing systems either mitigate these effects using compliant skins and smoothing strategies or employ them deliberately as a volumetric pixel language. Third, the central engineering problem is often not geometry generation alone but repeatable state control, returning pins reliably to reference positions, maintaining consistency across repeated cycles, and managing tolerance accumulation¹⁴ in mechanical and sensing subsystems.

Taken together, these properties position pin arrays less as single-purpose mechanisms and more as platforms in which form emerges from the interaction between representation, control constraints, and physical repeatability.

2.3 Reconfigurable Pin Tooling in Manufacturing: Forming, Fixturing, and Repeatability

Industrial reconfigurable pin tooling (RPT) has been explored as an alternative to dedicated molds and fixtures^{15,16}, particularly for low-volume production and high-variation geometries. Foundational work established pin-based approaches for approximating complex surfaces in adaptable manufacturing contexts¹⁷, while later

¹³ Wang, *Rapid Manufacturing of Vacuum Forming Components Utilising Reconfigurable Screw-Pin Tooling.*, 27-29, 192, 202

¹⁴ Koc and Thangaswamy, “Design and Analysis of a Reconfigurable Discrete Pin Tooling System for Molding of Three-Dimensional Free-Form Objects.”, 342-343

¹⁵ Owodunni et al., “Development and Evaluation of a Low-Cost Computer Controlled Reconfigurable Rapid Tool.”, 102

¹⁶ Walczyk and Longtin, “Fixturing of Compliant Parts Using a Matrix of Reconfigurable Pins.”, 766-767

¹⁷ Munro and Walczyk, “Reconfigurable Pin-Type Tooling,” 2007., 551-552

studies expanded the scale and application range of pin arrays, including large-format systems developed for aerospace-related forming and reconfigurable support¹⁸.

Within this lineage, evaluation criteria commonly emphasize repeatability, configuration time, and the capacity to sustain process demands. At the same time, the literature consistently reports constraints relevant to pin-based systems: discretized geometry limits surface continuity; load-bearing capability is bounded by actuator and structure design; and many implementations assume workflows where reconfiguration is infrequent and operator-driven rather than continuously interactive. For the present research, industrial RPT provides essential precedent for calibration strategies and structural thinking, yet it often treats the pin field primarily as a tooling substrate rather than a user-facing interface supporting multi-modal input.

2.4 Shape-Changing Surfaces in Kinetic Art and Interactive Media

In kinetic art and interactive media, reconfigurable surfaces are commonly framed as spatial communication devices rather than manufacturing tools¹⁹. Within this context, discretized height-field mechanisms are valued for what they can render and evoke—silhouettes, gradients, and transitions that become legible through coordinated actuation, even when resolution is limited. A useful reading of this body of work separates (1) large-scale kinetic installations that treat discretized elements as “physical pixels,” and (2) research-driven shape displays developed as interaction platforms.

A reference point for the first category is the BMW Museum Kinetic Sculpture (in operation since 2008)²⁰, which choreographs a dense field of individually controlled suspended elements to visualize form emergence as a time-based sequence. Although not

¹⁸ Walczyk and Im, “A Hydraulically-Actuated Reconfigurable Tool for Flexible Fabrication.”, 562, 566

¹⁹ Leithinger et al., “Shape Displays.”

²⁰ *Kinetic Sculpture — The Shapes of Things to Come*.

a tabletop pin array, the work is conceptually aligned with discretized height-field logic in that three-dimensional form becomes perceptible through collective motion and staged stabilization. Design exhibition contexts further foreground the pin field as expressive terrain. TRANSFORM²¹, developed by MIT’s Tangible Media Group and exhibited at Lexus Design Amazing 2014, employs a high-density pin array to produce real-time landscape-like transformations where motion design and sensing serve performative clarity rather than precision reproduction.

In parallel, HCI research has positioned pin-actuated devices as general-purpose interaction substrates. Systems such as inFORM articulate a vocabulary of dynamic affordances and constraints, including mechanically rendered features that support co-located manipulation of passive objects²². Earlier work such as Relief similarly emphasizes modularity and demonstrability by combining pin actuation with projected graphics and compliant surface skins to support scenario-based interaction. Across these projects, evaluation tends to prioritize interactive capability and communicative legibility, while practical constraints of scale, cost, and deployment limit how such systems generalize beyond prototype settings.

Taken together, these precedents demonstrate how discretized height-field systems succeed as perceptual and interactive media: they can render form convincingly, communicate transformation through choreography, and enable hands-on interaction via physically instantiated interface features. At the same time, this literature often treats the surface primarily as a responsive display, leaving open a different problem central to this thesis: how heterogeneous inputs—including linguistic descriptions—can be translated into static, reliable configurations under enforceable mechanical constraints, and how the interpretation layer can remain transparent and debuggable when it becomes part of the control pipeline.

²¹ “TRANSFORM.” *MIT Media Lab Tangible Media Group*

²² Follmer et al., “inFORM.”

2.5 Interaction and Control Paradigms for Morphing Surfaces

The interaction contract in a morphing-surface system specifies valid input formats, translation rules to an internal representation, constraint enforcement (bounds, homing references, safe states), and feedback mechanisms (state visibility, warnings, and error recovery). Table 2-1 summarizes three dominant paradigms in prior work and their implications for a deterministic pin-based platform.

Paradigm	Typical input	Core benefit	Common limitation
Direct actuator specification	Per-pin heights, grouped heights, manual sliders	Maximum determinism and traceability	Low scalability and high user effort as pin count increases
Mapping-based control	Images, scalar fields, parametric surfaces	Scales well; repeatable when mapping is fixed	Sensitive to preprocessing choices (sampling, smoothing, normalization), which can reduce interpretability
Goal/intent-driven control	Natural language, qualitative constraints, feature goals	High-level usability and exploratory interaction	Ambiguity; requires an interpreter layer and rigorous validation

Table 2-1. Interaction and Control Paradigms for Pin-Based Morphing Surfaces.

Following this comparison, the present work formalizes the interaction contract as an enforceable pipeline: each input modality must resolve into a bounded height-field representation, validated against mechanical limits, and executed through a deterministic command format, with feedback that supports user correction and debugging.

2.6 AI-Mediated Design Interfaces: Semantic Inputs Under Physical Constraints

Generative AI, particularly large language models (LLMs), extends interaction from geometric specification to semantic interpretation. In “design by description” paradigms, language can be used to express desired characteristics that are subsequently translated into representational outputs. When embedded in physical systems, the problem becomes more constrained: ambiguous linguistic descriptions must be translated into deterministic actions that remain valid under mechanical limits.

A key challenge is that language models naturally produce open-ended outputs, while actuation requires strict formatting, bounded values, and predictable outcomes. This motivates a separation between (1) an actionable output representation that can be validated and executed, and (2) an explanatory layer that clarifies the basis of interpretation without altering the executable command logic. In embodied systems, interpretability is not only conceptual but operational: it supports debugging, iteration, and trust when interaction relies on descriptive rather than numerical input.

AI-Mediated Design Interfaces: Semantic Inputs Under Physical Constraints

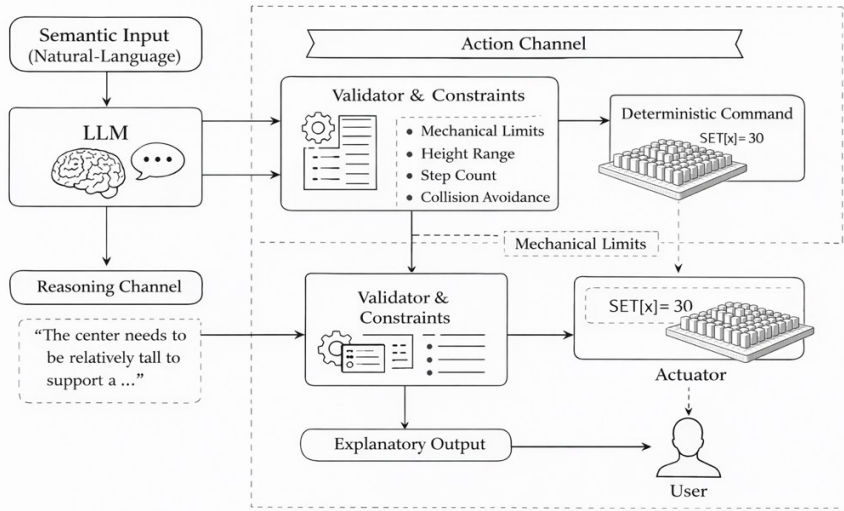


Figure 2-1. AI-Mediated Design Interfaces: Semantic Inputs Under Physical Constraints.

2.7 Synthesis: Literature-Derived Tensions and Their Implications

Taken together, prior work highlights recurring trade-offs between resolution and scalability, expressiveness and repeatability, and semantic flexibility and command validity. These findings imply that a practical pin-based morphing platform must treat the surface as a discretized height-field representation, enforce calibration and bounded motion as first-class requirements, and translate high-level inputs into deterministic, verifiable actuation commands. The next chapter operationalizes these implications as system requirements and implementation choices for the proposed platform.

Dimension	Conventional Focus	Gap Identified	Relevance to This Research
Manufacturing	Precision, force capacity, repeatability using fixed tooling	Limited adaptability; tooling changes require time and cost	Pin-based systems conceptually demonstrate how geometry can be reconfigured digitally, even if the present prototype is not intended for forming.
Kinetic Art	Expression, perception, interactivity	Limited structural strength; often lightweight and non-load-bearing	Kinetic precedents inspire the use of a pin field as a medium for spatial representation rather than force.
AI Design	Semantic	Lack of physical	The morphing surface materializes AI-

Tools	understanding, generative design, descriptive control	embodiment; outputs remain digital	generated descriptions, offering a tangible endpoint for linguistic prompts.
--------------	---	---------------------------------------	--

Table 2-2. Literature-derived tensions and research implications across domains.

3. System Design and Development

3.1 System Overview

The Motor-Based Kinetic Morphing System (MKMS) is an integrated prototype that combines mechanical actuation, distributed embedded control, and AI-mediated interpretation within a single workflow. The system is implemented as a 5×5 array of vertically actuated pins. Each pin is driven by an individual stepper motor and coordinated through a multi-controller architecture in which five Arduino Mega boards execute homing and motion for fixed subsets of actuators. The prototype is designed to produce static height-field configurations and does not target material forming, high-force operation, or continuous kinetic animation. Instead, it serves as a proof-of-concept platform for investigating how digital inputs, including language-driven prompts, can be translated into tangible physical geometry.

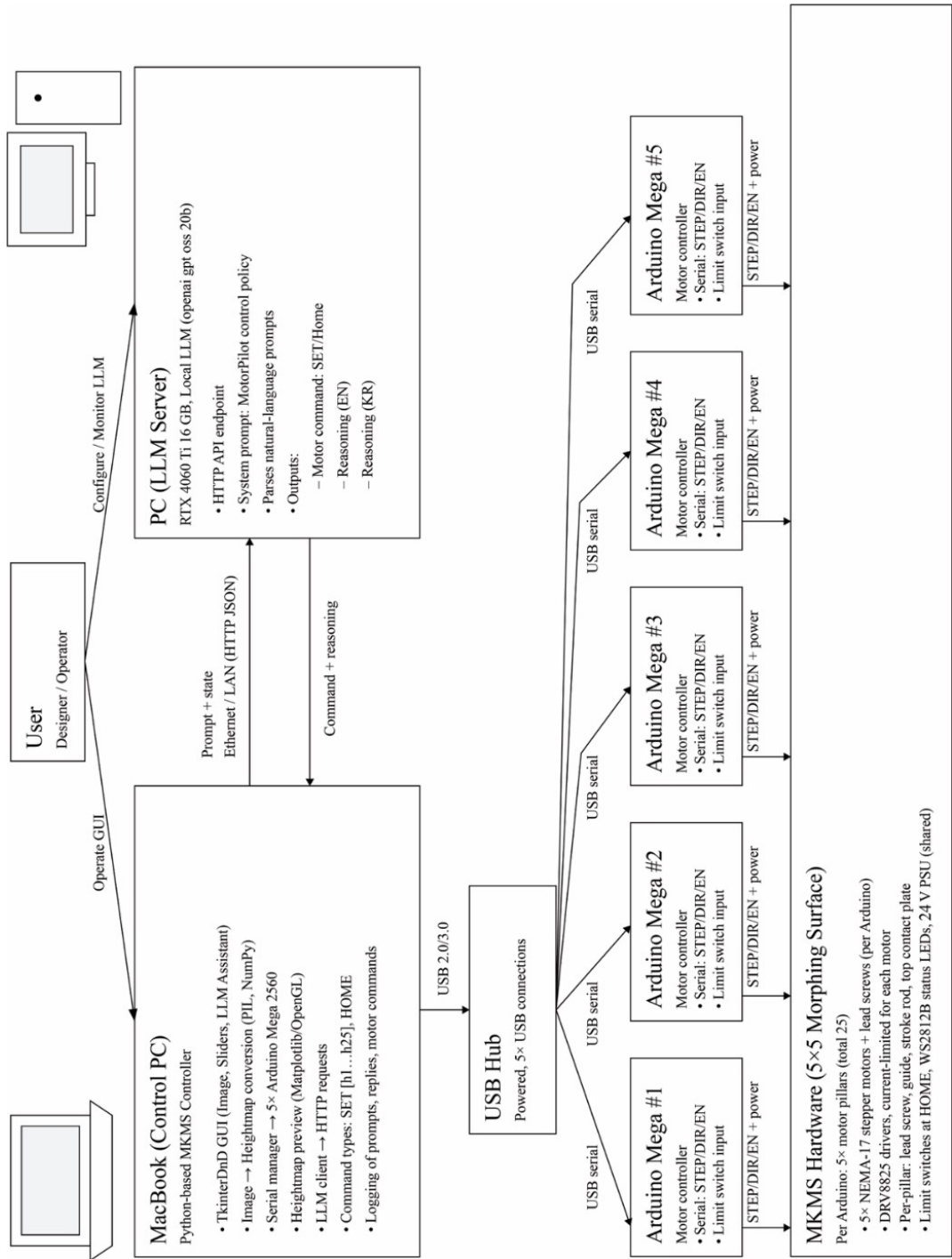


Figure 3-1. Overall architecture of the MKMS, illustrating the relationships between the mechanical surface, embedded controllers, host software, and the LLM interface.

Two design principles guided the development of the system.

(1) **Modularity.** The mechanical and electronic implementation is structured as repeatable motor and lead-screw actuation units integrated into a shared frame. Control responsibilities are distributed across five microcontrollers, each assigned a consistent subset of the 25 actuators. While the current prototype is limited to 25 pins, this organization establishes a scalable pattern in which larger arrays can be constructed by replicating actuator modules and extending the distributed control layer.

(2) **Layered interaction.** The system supports multiple input modalities through a shared internal representation of the surface as a 25-value height field. The host application enables (1) direct manual specification of heights, (2) data-driven heightmap generation from external inputs (e.g., grayscale images), and (3) LLM-assisted command generation in which natural-language prompts are converted into bounded SET commands. In addition to command generation, the interface provides an optional reasoning channel that presents a natural-language explanation of the executed configuration. This explanatory layer is intentionally decoupled from actuation and does not introduce sensing or closed-loop correction. Its role is limited to improving interpretability at the interaction level.

Overall, the MKMS integrates hardware, software, and language-mediated interpretation into a coherent demonstration pipeline. Although its physical output is restricted to static configurations, the system can be framed in three complementary roles: (1) a configurable tool for rapid redefinition of surface geometry through digital input, (2) a low-resolution display for materializing numeric and visual patterns as relief, and (3) an interpretive interface that translates linguistic descriptions into discrete height configurations and provides an accompanying explanation. Within the scope of this thesis, the MKMS is presented not as a finalized morphing device, but as a foundational prototype for studying the coupling of deterministic actuation and AI-mediated interpretation in a physical design system.

3.2 Mechanical Architecture

The mechanical structure of the Motor-Based Kinetic Morphing System (MKMS) is composed of 25 vertically actuated pin units arranged in a 5×5 grid. Each actuator is a self-contained linear motion module housed within a 3D-printed body and mounted to an aluminum-extrusion frame. Figure 3-2, 3-3, 3-4, and 3-5 illustrate the mechanical layout of a single actuator and the assembled 5×5 array.

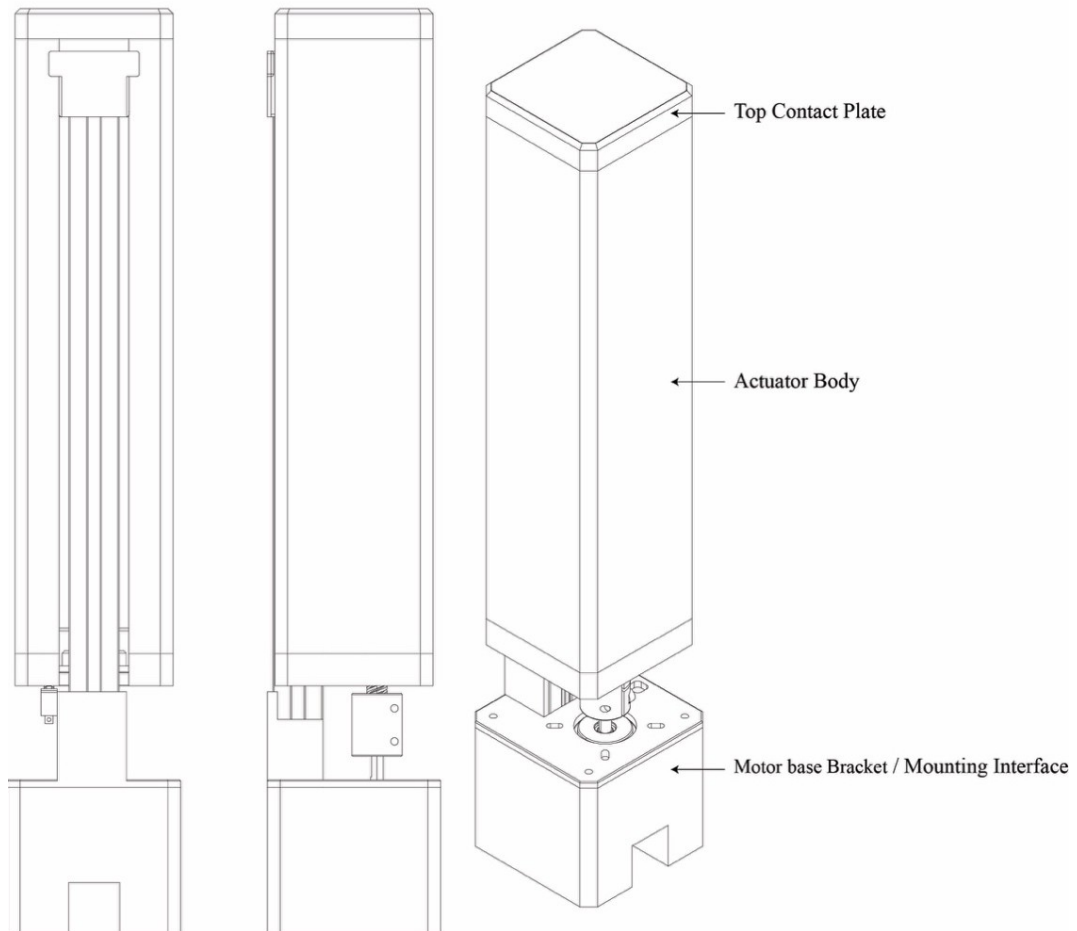


Figure3-2. Single actuator unit.

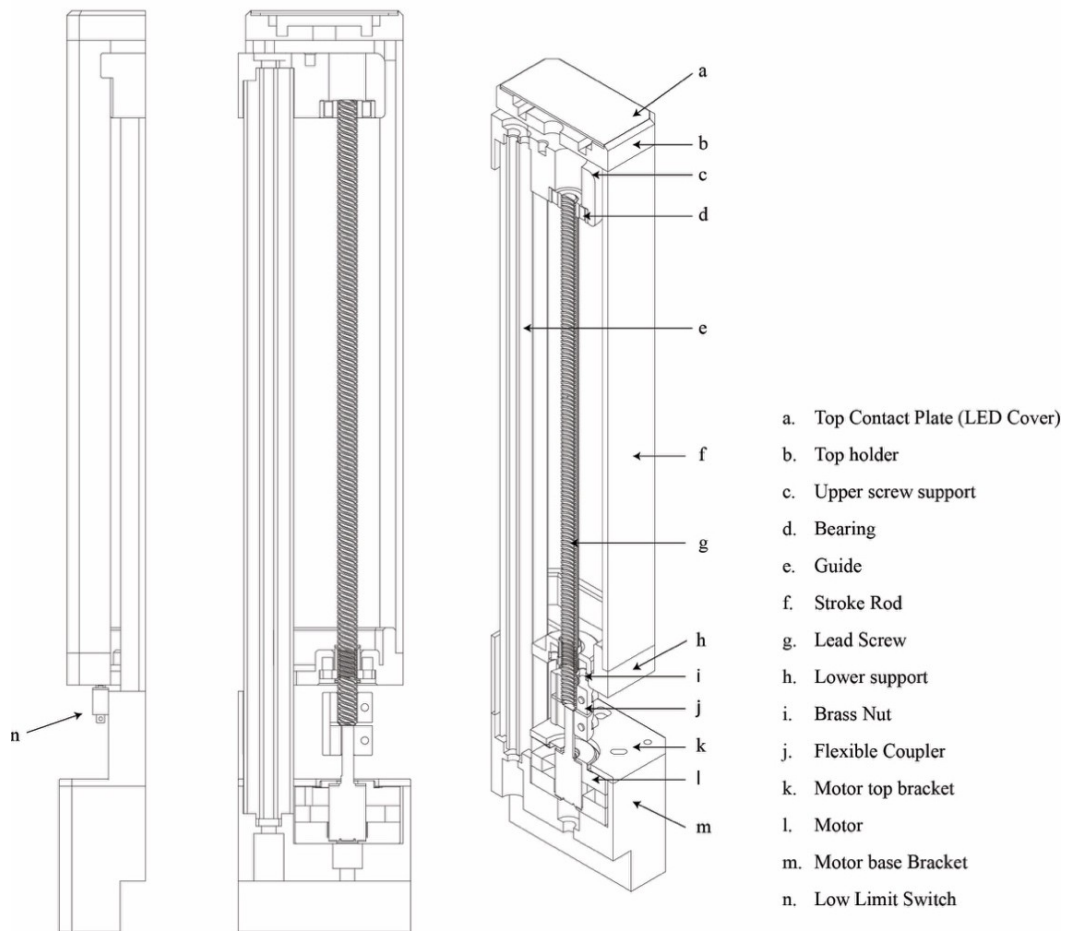


Figure 3-3. Internal structure of the actuator unit.

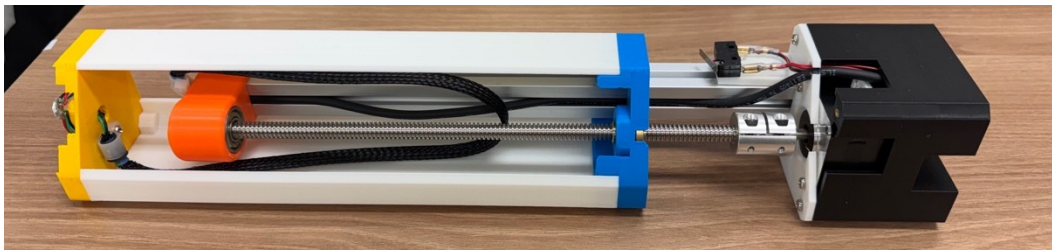


Figure 3-4. Cutaway view of a single actuator unit.

Actuator Structure. Each pin unit converts rotary motion from a NEMA-17 stepper motor into vertical displacement through a screw-driven mechanism. As shown in Figure 3-3, the actuator consists of:

- **a. Top Contact Plate (LED Cover)** – A flat plate forming the user-facing surface. In five selected units, this plate includes a NeoPixel LED beneath a translucent cover.
- **b. Top Holder** – A structural component that supports the upper end of the guide rods and houses the top interface for the stroke rod.
- **c. Upper Screw Support** – Maintains alignment of the lead screw and distributes load to the top of the housing.
- **d. Bearing** – Ensures smooth rotation of the lead screw while constraining axial movement.
- **e. Guide Rods** – Two vertical rods that constrain the motion of the stroke rod, preventing lateral wobble.
- **f. Stroke Rod** – The moving element connected to the top plate; its vertical position creates the visible height of the surface.
- **g. Lead Screw** – A threaded rod that converts motor rotation into linear motion via the brass nut.
- **i. Brass Nut** – Translates rotary motion of the lead screw into linear movement of the stroke rod.
- **j. Flexible Coupler** – Compensates for small misalignments between the motor shaft and the lead screw.
- **l. NEMA-17 Stepper Motor** – Mounted at the base, providing the rotational actuation.
- **m. Motor Base Bracket** – A rigid mounting interface that affixes the motor to the aluminum frame.
- **n. Low Limit Switch** – A mechanical switch at the bottom that defines the homing reference.

This stacked arrangement allows each actuator to achieve approximately 200 mm of usable stroke, corresponding to the full travel range of the stroke rod.

Motion Characteristics. Mechanical motion is produced through a lead-screw mechanism, offering controlled and repeatable vertical displacement. The system uses:

- NEMA-17 stepper motors with $1.8^\circ/\text{step}$,
- DRV8825 drivers operating at full step,
- a lead screw with typical pitch values ($\text{TR8} \times 2$).

Because the system is configured for full-step operation, its motion resolution is determined primarily by the step angle and lead screw pitch rather than microstepping. In practice, achievable positioning accuracy and repeatability are further constrained by 3D-printed tolerances, flexible couplers, brass nut backlash, and frame compliance. The system is therefore designed for coarse geometric shaping rather than micron-level precision.

The actuator achieves a usable stroke of approximately 200 mm. This extended vertical range enables significant height variation across the 70 mm $\times$ 70 mm grid spacing of the pins.

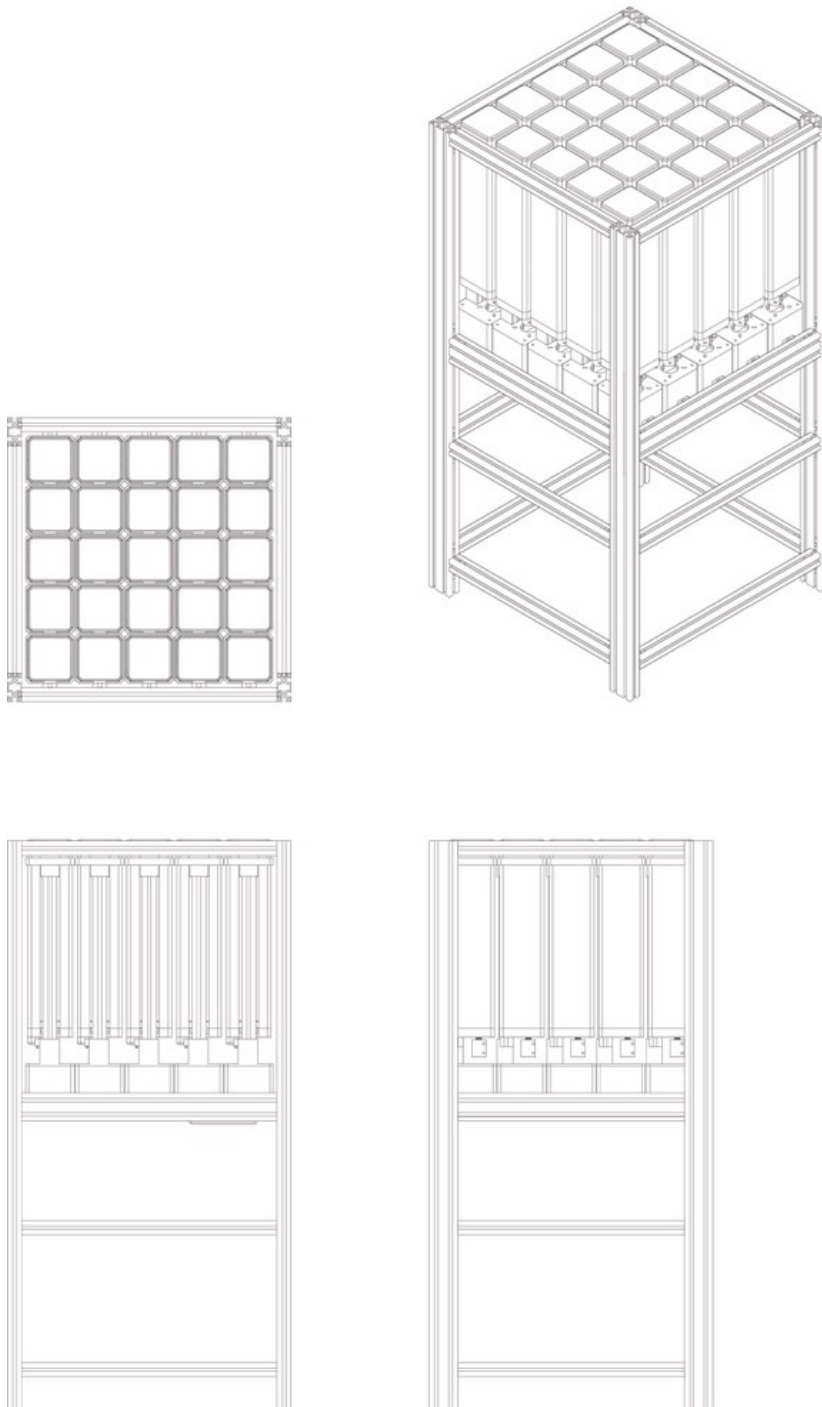


Figure 3-5. Assembled 5x5 actuator array and support frame.

Frame Structure. The full 5×5 actuator assembly is mounted to a rigid frame constructed from 2020 aluminum extrusions (Figure 3-5). The frame is organized into three functional layers: (1) an upper plane that aligns the 25 actuator heads to a common reference surface, (2) a mid-level support layer that anchors the stepper motors and constrains drive alignment, and (3) a lower bay that accommodates the electronic components, including the Arduino controllers, SMPS power supply, and USB hubs.

Maintainability. Although the system is not designed for hot-swapping, each actuator module can be removed independently using standard fasteners. This modular access supports maintenance and iterative refinement during prototyping. For example, during early testing, a misaligned actuator was disassembled and re-aligned without requiring frame-level rework, indicating that the structure provides sufficient serviceability for debugging and calibration.

3.3 Embedded Electronics Architecture

The MKMS uses a distributed embedded architecture comprising five Arduino Mega 2560 controllers, each responsible for a fixed subset of five actuators within the 5×5 array. This partitioning reduces per-controller I/O demand and helps maintain stable step-pulse timing during multi-actuator motion.

Each actuator is driven through a DRV8825 stepper driver. Motor drivers share a 24V supply for actuation power, while the microcontrollers are powered separately via a powered USB hub to maintain stable logic operation.

A normally-open limit switch is installed at the lower end of each actuator to provide a homing reference. During initialization, each controller performs a local homing routine for its assigned actuators and establishes the zero-height baseline in firmware.

To reduce idle heating and allow per-actuator control, each motor driver uses an independent enable line rather than a shared enable configuration.

For visual feedback, NeoPixel LEDs are integrated in five actuator units only. These LEDs provide a height-correlated color cue and are not used for sensing, fault reporting, or closed-loop control.

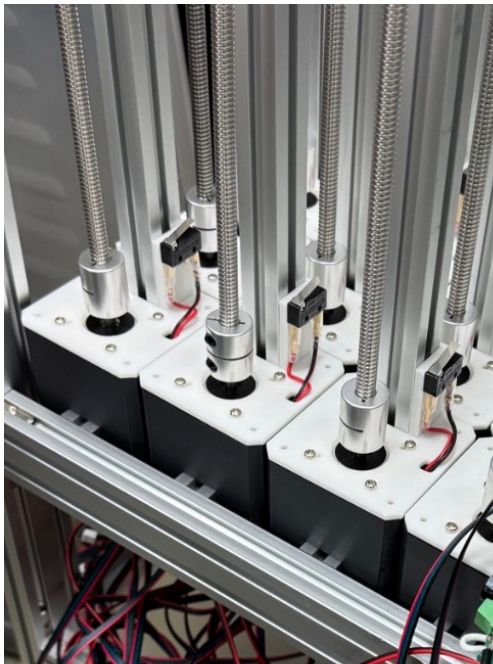


Figure 3-6. Lower-end limit switch used for actuator homing reference.

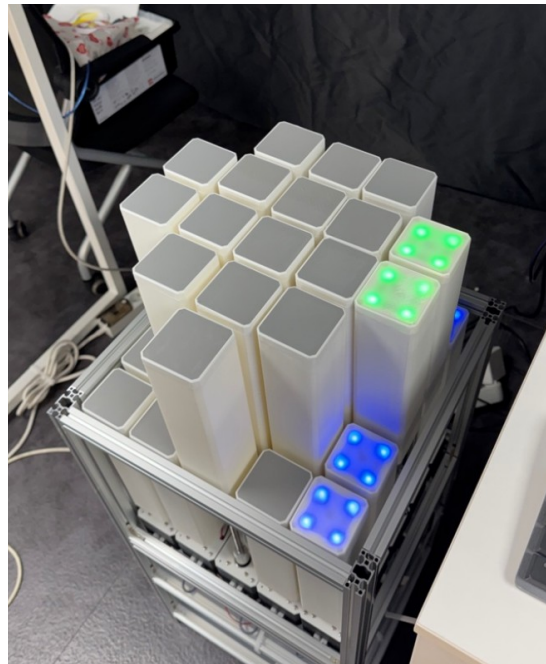


Figure 3-7. NeoPixel LED height-correlated color feedback integrated in selected actuator units.

3.3.1 Custom 5-Driver Shield for Arduino Mega (DRV8825 Carrier Board)

To improve packaging density and reduce wiring complexity, a custom driver carrier board was developed for each Arduino Mega. The board is mechanically and electrically configured as a shield that mounts directly on the Mega and integrates five DRV8825 stepper-driver modules on a single PCB. This arrangement shortens motor-driver wiring, consolidates power and signal routing at the controller, and reduces cable management overhead compared to distributing individual driver modules via jumper wiring. The carrier-board approach also supports modular maintenance, since each controller and its associated five actuators can be handled as a single replaceable unit (see Figure 7).

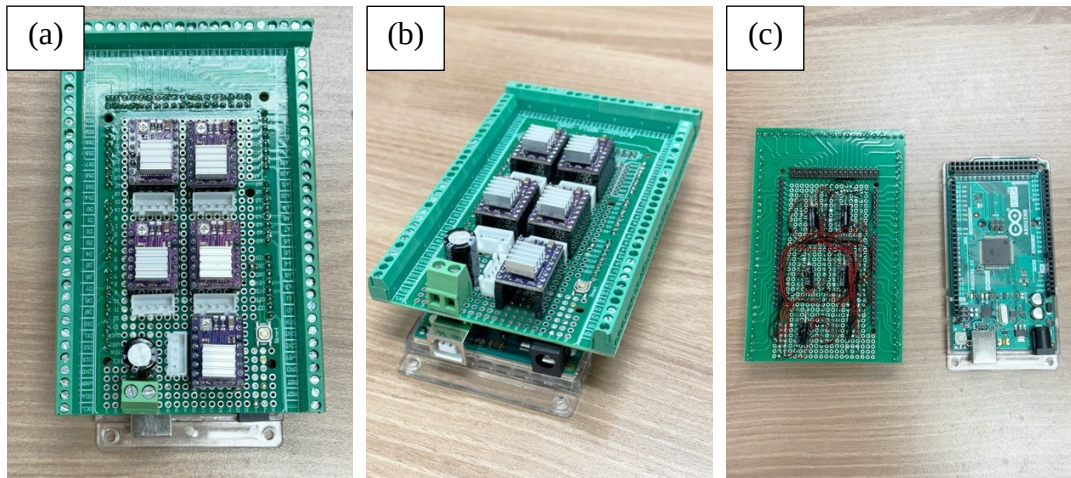


Figure 3-8. Custom DRV8825 driver carrier board for a single Arduino Mega: (a) top view showing five DRV8825 modules mounted on the PCB, (b) assembled stack with the PCB seated on the Arduino Mega, and (c) bottom view showing routing and solder-side wiring.

Motor motion is coordinated at the microcontroller level using a simple synchronous stepping routine. All five motors on a given Arduino move at the same step frequency, regardless of the distance they must travel. As a result, motors that require smaller moves complete their travel earlier and wait while others finish, but the overall movement remains reliable for the static final shapes the system is designed to produce. The system does not implement multi-axis interpolation, acceleration planning, or trajectory optimization commonly found in CNC control systems.

Communication between the host computer and the five Arduinos occurs through individual serial connections. The Python control software sends a single SET command to each board, specifying the five heights assigned to that microcontroller. Each board then executes the command independently according to its internal homing reference and movement logic.

Overall, the electronic architecture prioritizes robustness and clarity over complexity. Distributed control via multiple Arduino Mega boards, dedicated limit switches, and straightforward stepping logic ensures dependable operation of the 25-actuator array within the constraints of the prototype.

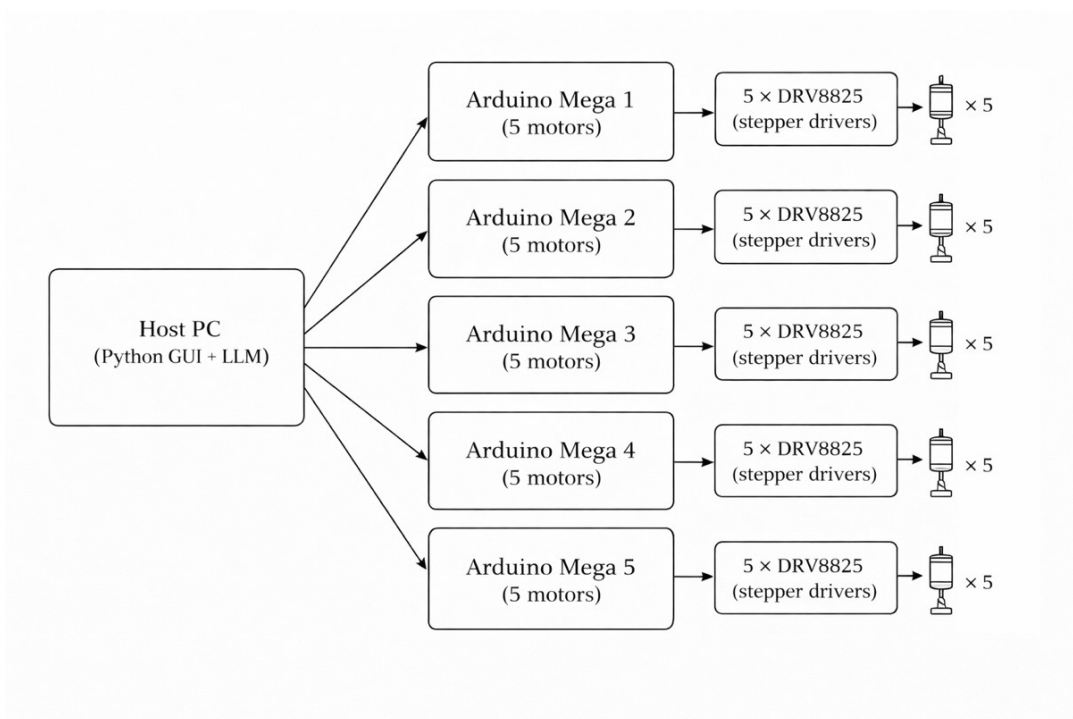


Figure 3-9. Distributed control architecture for the 5x5 actuator array.

Subsystem	Component / Specification	Notes
Microcontrollers	5× Arduino Mega 2560	Each controls 5 motors (one row)
Motor Drivers	25× DRV8825	One per actuator; operating at full step
Motors	NEMA-17 stepper motors	1.8°/step; directly coupled via flexible coupler
Power Supply (Motors)	24V SMPS	Drives all DRV8825 modules
Power Supply (Logic)	Powered USB Hub	Provides stable 5 V to all Arduinos
Limit Switches	25× mechanical microswitch	One per actuator; homing reference
Enable Pins	Independent EN pins	No shared emergency cutoff line
LED Feedback	5× WS2812B NeoPixel LEDs	Height-based color visualization
Wiring	Harness connectors + cables	Direct modular wiring
Communication	Multiple serial USB connections	Host PC sends SET per Arduino
Motion Control	Simple synchronous stepping	No interpolation; static final shapes

Table 3-1. Electronic system summary table

3.4 Software Framework

The control software for the Motor-Based Kinetic Morphing System (MKMS) is divided into two primary layers:

(1) the embedded firmware running on five Arduino Mega microcontrollers, which receives validated SET and HOME commands over serial, converts target heights into step counts relative to the homing reference, and executes synchronized open-loop motion for each board’s five actuators.

(2) a Python-based host application that handles user interaction, image processing, and LLM integration.

3.4.1 Arduino Firmware

Each Arduino Mega executes a lightweight C++ firmware that provides deterministic low-level control for a subset of five actuators. The firmware is designed to (1) receive structured commands from the host computer over serial communication, (2) validate and bound incoming targets to remain within mechanical limits, (3) convert target heights expressed in millimeters into step targets referenced to a homed zero position, and (4) execute coordinated open-loop stepping across the assigned motors. This implementation prioritizes repeatable convergence to static configurations rather than continuous trajectory generation or closed-loop control.

Command Interface and Validation. The firmware exposes a constrained command interface so that upstream modules, including LLM outputs, are forced into a deterministic and verifiable action representation. Shape formation is performed through the SET instruction, which requires exactly five numeric values.

Figure 3-10. Serial command parsing and SET value-count validation (Arduino firmware).

Height Bounding and Unit Conversion. To preserve safety and mechanical consistency, all target heights are clamped to the allowable stroke range before conversion to step counts. The system operates in full-step mode; resolution is therefore bounded by motor step angle, lead-screw pitch, and mechanical tolerances.

Figure 3-11. Height bounds enforcement and millimeter-to-step conversion prior to execution (Arduino firmware).

Coordinated Motion Execution. For multi-motor moves, the firmware performs synchronized stepping using a shared step index across the five motors. Motors that reach their target early stop stepping while others continue until completion. This strategy is reliable for static height-field settling and avoids CNC-style trajectory planning or interpolation. To improve stability at motion onset and reduce missed steps, the implementation applies a brief slow-start torque-assist phase, followed by an eased delay schedule during the main motion phase

Figure 3-12. Synchronized stepping using a shared step index across five motors (Arduino firmware).

Homing Routine and Reference State. Homing establishes the reference state for absolute positioning. Each motor is driven downward until the bottom limit switch triggers, after which the motor's internal position counter is reset to zero. The routine uses a conservative step timing profile during homing to improve stability and reduce missed steps near the limit condition. Repeatability is bounded by limit-switch hysteresis, lead-screw backlash, and structural compliance in the printed frame and couplers.

Figure 3-13. Homing routine using bottom limit switch to establish the zero reference (Arduino firmware).

Firmware Feedback and Scope. Firmware feedback is intentionally minimal to preserve predictable behavior across five independent serial channels. The system reports parsing errors (e.g., malformed SET messages) and basic status prints, but does not

stream sensor telemetry and does not implement closed-loop correction. The firmware therefore functions as a deterministic execution layer within the broader MKMS software framework, enabling upstream software modules to treat each Arduino board as a bounded actuator subsystem.

3.4.2 Python GUI and Control Software

The MKMS control software is implemented as a Python-based host application that functions as the central coordination layer between user input, data translation, and embedded actuation. The application provides three primary capabilities: (1) an interactive GUI for manual control and preview, (2) conversion of external image inputs into bounded 5×5 heightmaps, and (3) integration of a local LLM interface whose outputs are validated and translated into executable motor commands. Across all modes, the host application enforces a consistent internal representation of the surface as a 25-value height field and dispatches motion to five Arduino Mega boards over independent serial connections.

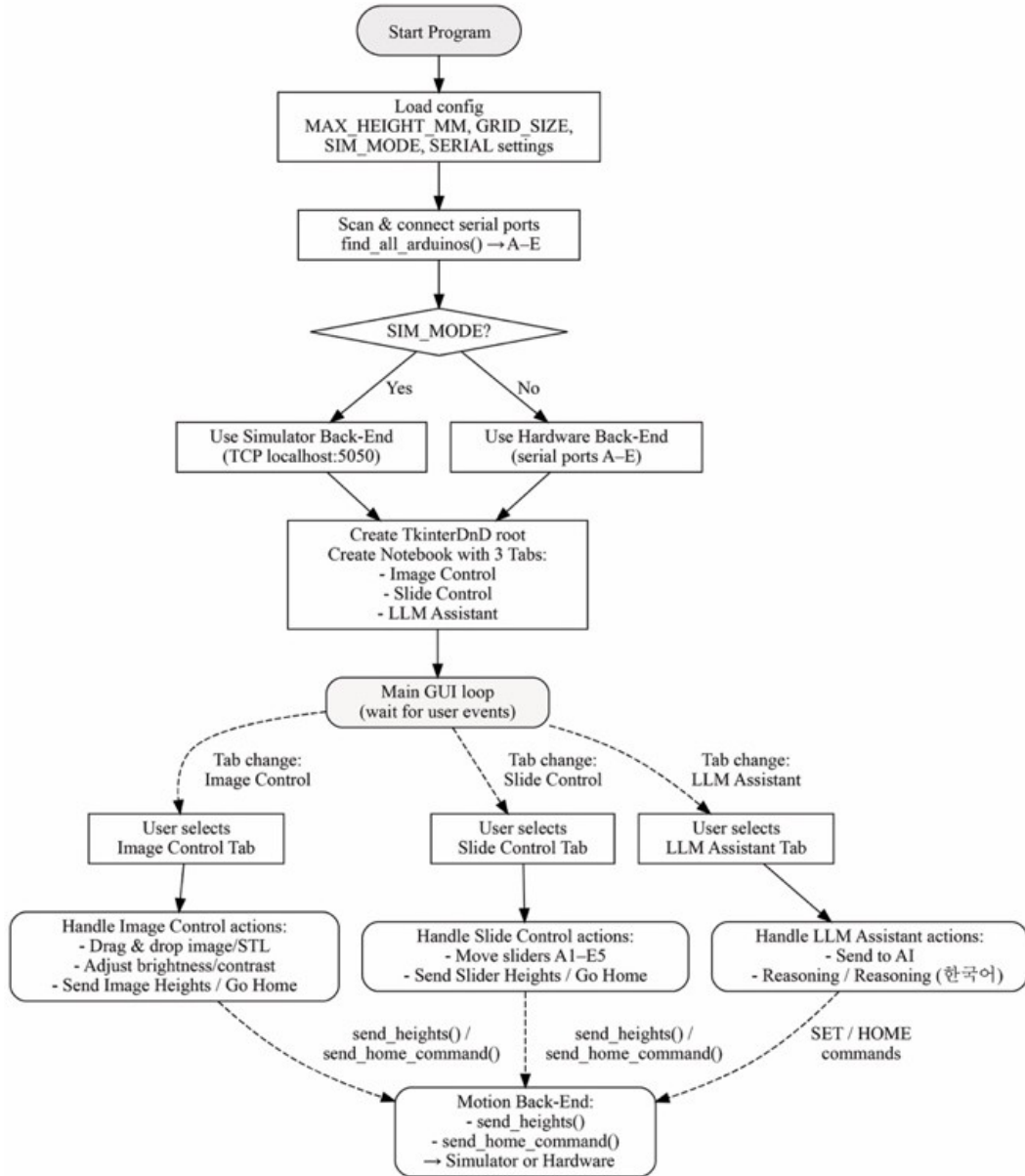


Figure 3-14. Control software flowchart for the Python host application.

Multi-Board Serial Discovery and Port Binding. Because the system uses five microcontrollers simultaneously, stable device identification is required to preserve deterministic mapping between software channels and physical actuator groups. The host program therefore binds each board using its USB serial number and assigns it a fixed label (A–E). On startup, all connected ports are scanned, matching serial numbers are identified, and a dedicated serial connection is opened per board. This strategy ensures that the same physical board is addressed consistently across sessions even if operating system port names change.

Figure 3-15. Arduino identification by serial number and multi-port connection (excerpt).

Heightmap Generation from Image Inputs. In data-driven operation, the host application converts external image files into the MKMS surface representation as a 5×5 height field. An input image is converted to grayscale and downsampled to a 5×5 pixel grid that matches the actuator layout. Pixel intensities are then linearly mapped to the permitted actuation range (0–200 mm), producing a deterministic height assignment at the current surface resolution.

Figure 3-16. Image downsampling and intensity-to-height mapping (excerpt).

Command Decomposition and Dispatch to Five Controllers. All actuation is represented at the host level as a 25-value height field for the 5×5 surface. Because the MKMS is driven by five Arduino Mega boards, each controlling five motors, the host decomposes this surface-level target into five 5-value SET commands and transmits one command per board over independent serial connections.

In the current implementation, the mapping is fixed and deterministic. The 25 values are interpreted in row-major order for the surface representation, but they are dispatched column-wise such that each Arduino receives one column of five heights (A–E). Commands are sent as ASCII strings of the form SET h1 h2 h3 h4 h5, and the host may optionally print a single response line when available; execution does not depend on a strict acknowledgement handshake.

Figure 3-17. Partitioning 25 heights into five per-board SET commands (excerpt).

This procedure defines the interface between surface-level shape specification and distributed firmware execution. The same dispatch routine is applied regardless of whether the target heights originate from manual input, image-based heightmap conversion, or LLM-generated output.

LLM Integration and Deterministic Execution Boundary. To support language-driven interaction, the host application integrates a locally hosted LLM through an HTTP API. The LLM interface is implemented as a dedicated GUI tab with two outputs: an action channel intended to generate an executable motor command, and a reasoning channel intended to generate human-readable explanation without commanding motion. The action channel is constrained by a system instruction that requires output in a strict SET <25 values> format aligned with the internal surface representation.

After an LLM response is returned, the host validates its structure before executing any motion. The program searches for a SET command, parses numeric values, and rejects responses that do not match the expected value count. Only validated 25-value commands are forwarded to the actuation dispatch routine.

Figure 3-18. Parsing and validating SET output prior to execution (excerpt).

This structure enforces a deterministic execution boundary: semantic interpretation may be flexible upstream, but the actuation layer accepts only bounded, explicitly formatted commands that are validated before dispatch.

Visualization and Feedback. The host application provides immediate visual feedback through an embedded 3D heightmap preview rendered with Matplotlib. This visualization is updated for manual adjustments and image-based heightmap conversions, enabling rapid verification of the intended surface configuration before and after motion is issued. Firmware feedback is treated as lightweight status output rather than telemetry: serial responses may be read and displayed for confirmation, but the system does not rely on continuous sensing or closed-loop correction.

3.4.2.1 Manual Control Tab

The Manual Control mode enables direct per-actuator manipulation through a set of vertical sliders corresponding to the 25 pins in the 5×5 array. Each interaction produces an explicit height vector that is translated into bounded target values and sent to the embedded layer as discrete positioning commands. This interface was primarily used during early prototyping for rapid functional verification, homing checks, and exploratory “by-hand” shaping of the surface.

Because the current system targets static configurations rather than continuous animation, each manual adjustment results in a single repositioning event, after which the actuators settle to the commanded heights.

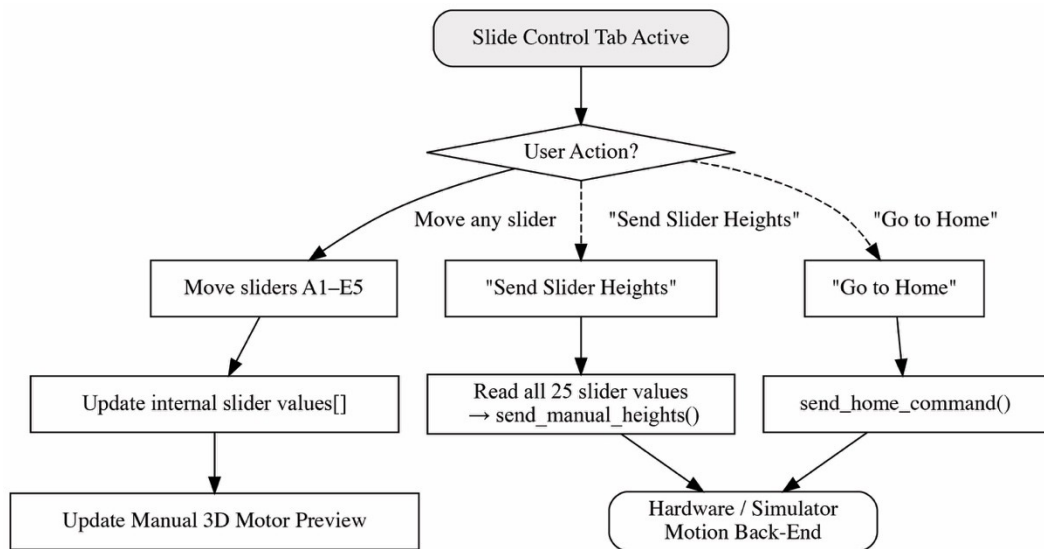


Figure 3-19. Control flow for the Slide Control mode.

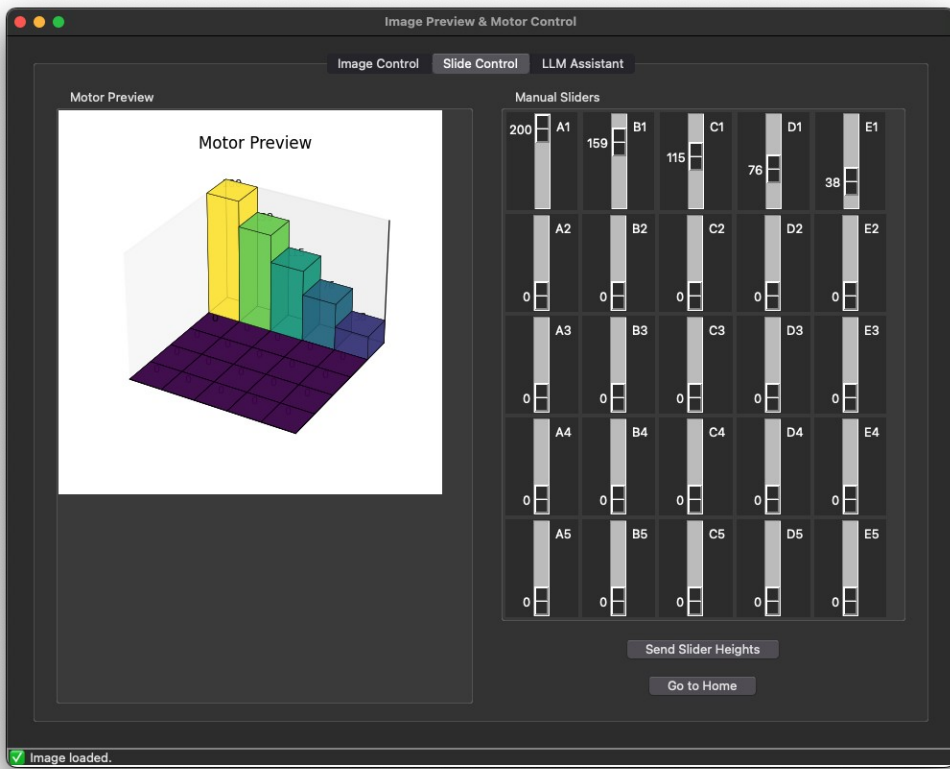


Figure 3-20. Manual Control GUI with per-pin sliders and 3D height preview

3.4.2.2 Image-to-Heightmap Tab

The Image-to-Heightmap workflow converts a 2D image into a 5×5 height field that can be physically rendered by the surface. The pipeline consists of: (1) loading an image (e.g., PNG/JPEG), (2) converting to grayscale intensity, (3) resampling to a 5×5 grid aligned with actuator resolution, and (4) mapping pixel values (0–255) to actuator heights (0–200 mm) through linear scaling. The resulting heightmap is shown to the user as a static 3D preview prior to execution.

When the user applies the configuration, the host application decomposes the 25 target heights into five groups of five and transmits one SET command to each Arduino Mega over independent serial connections. Since the system produces a single static configuration per interaction, this workflow remains deterministic and does not require continuous streaming or real-time trajectory updates.

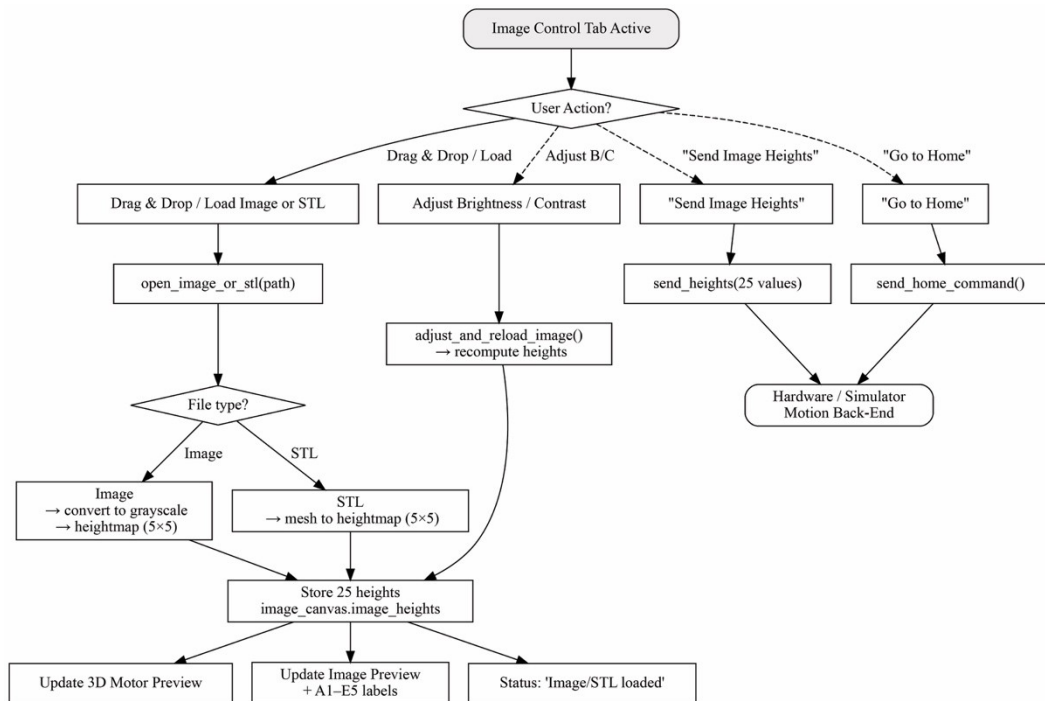


Figure 3-21. Control flow for the Image-to-Heightmap Control mode.

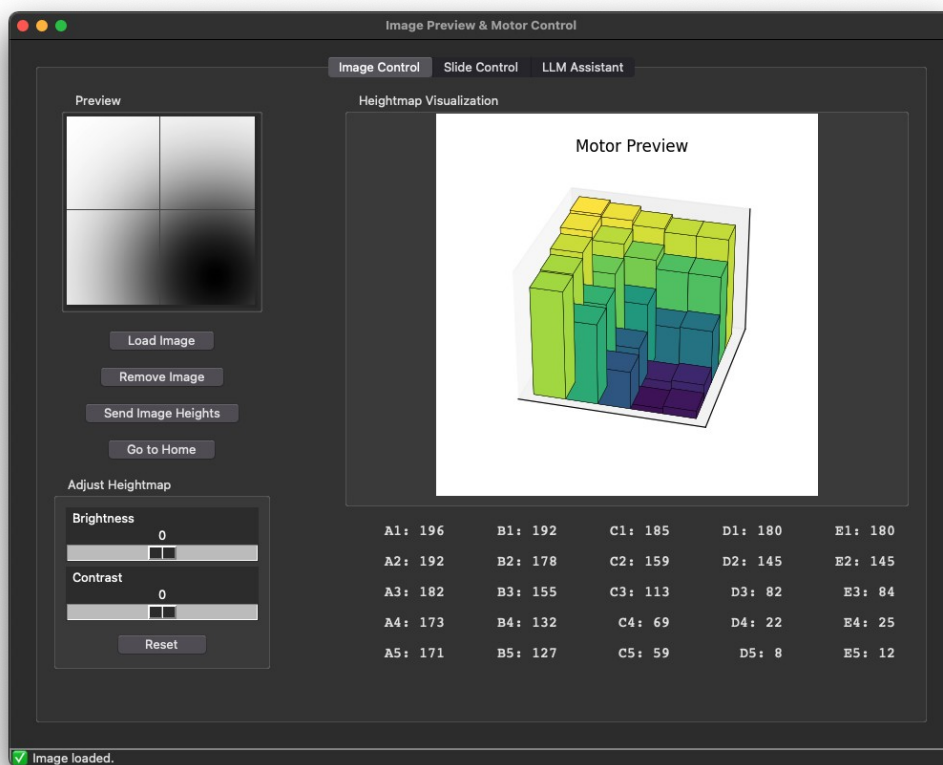


Figure 3-22. Image-to-Heightmap Control GUI with 3D height preview.

3.4.2.3 AI-Assisted Tab

The AI-Assisted mode integrates a locally hosted large language model (LLM) running offline on a desktop GPU platform (RTX 4060 Ti, 16 GB). The model is accessed via a local API and is not trained specifically for kinetic surfaces; instead, it is constrained through system-level instructions that define the 5×5 layout and the required output format. The objective of this mode is not autonomous control, but semantic translation: converting qualitative user language into a valid, bounded height command that can be executed deterministically.

Two-Channel Interaction Structure. To separate actuation from explanation, the AI interface is implemented as a dual-channel system:

A. Action Channel (Motor Command Generation). The user provides a natural-language prompt (e.g., “Create a small hill in the center”). The LLM is instructed to output exactly one executable command in a strict SET <25 values> format. The host program validates the response structure and value count before dispatch, then distributes the resulting 25 heights to the five Arduino boards. This channel is restricted to actionable commands and does not provide explanatory text.

B. Reasoning Channel (Textual Explanation). After a command is executed, the user may optionally request an explanation (English or Korean). The reasoning request is handled with a separate system prompt that prohibits motor commands and restricts output to descriptive rationale only. The reasoning channel receives the previous user prompt and the executed SET command as context, explains the mapping from intent to shape, and produces no actuation output.

This separation ensures that (1) actuation remains strictly formatted and safe, (2) explanation remains optional and non-operative, and (3) interpretive text does not interfere with deterministic execution. The GUI presents the reasoning output in a dedicated text area that is logically isolated from the motion dispatch pipeline.

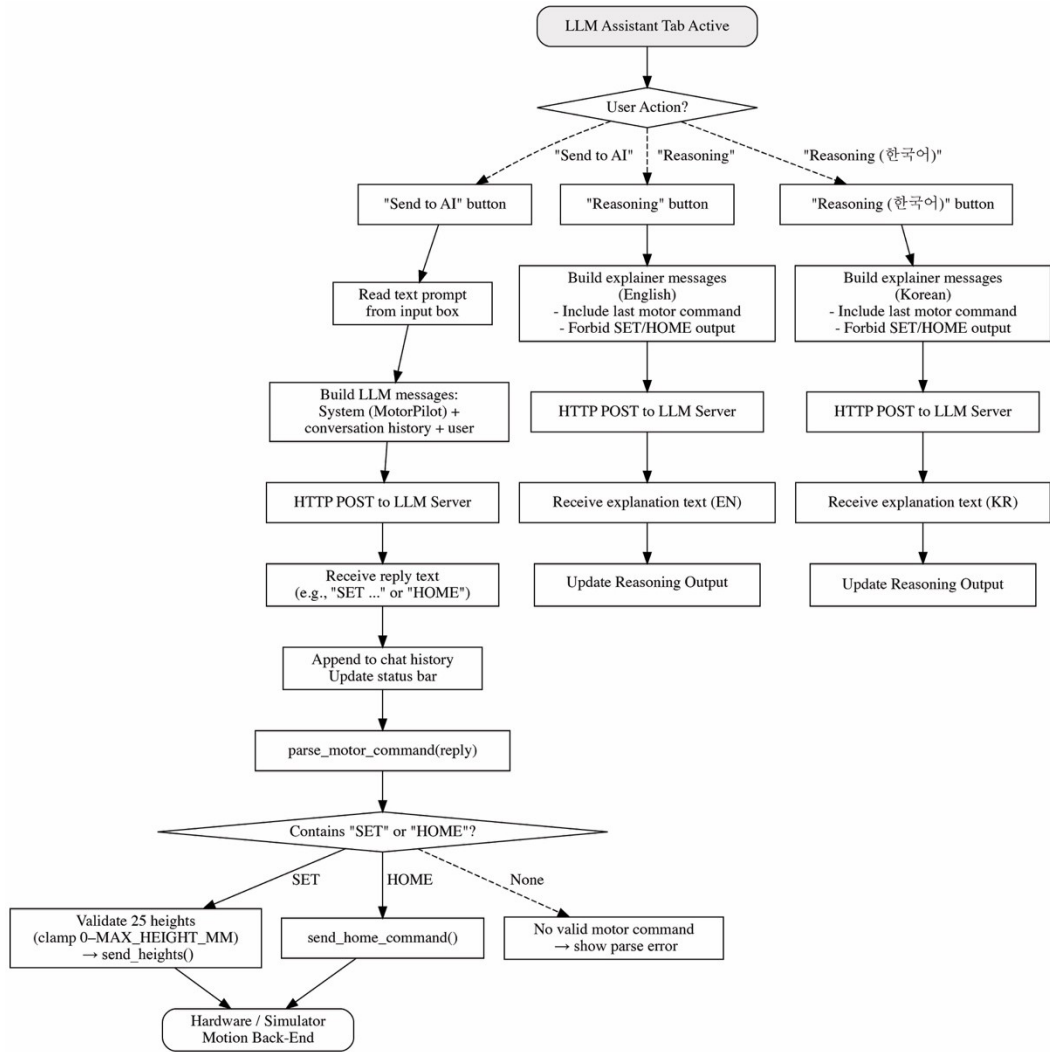


Figure 3-23. Control flow for the LLM Assistant mode.

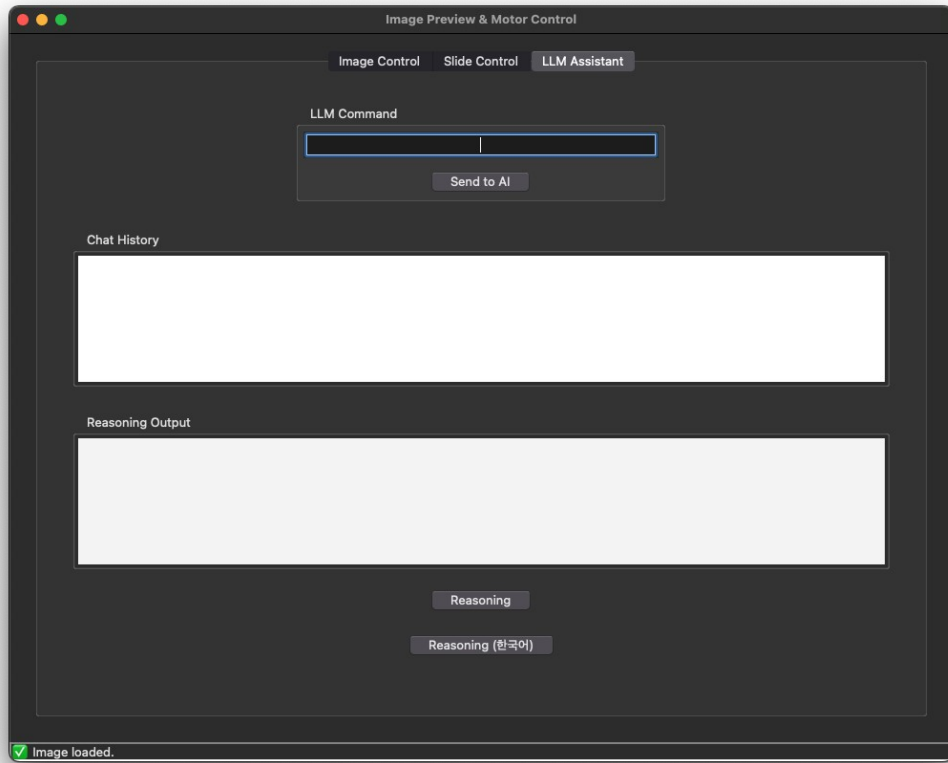


Figure 3-24. AI-Assisted GUI tab with dual-channel LLM interface for command generation and optional reasoning output.

3.4.2.4 3D Visualization

The GUI provides a 3D heightmap preview using a Matplotlib bar visualization of the 5×5 array. The preview represents the final static target configuration and updates when a new heightmap is generated or applied. Because the hardware control paradigm is static repositioning rather than continuous motion, the visualization is non-animated and is used primarily for verification prior to actuation.

Communication Protocol. Communication between the host application and the embedded controllers is implemented through five independent serial ports, one per

Arduino board. Commands are transmitted as ASCII text strings (e.g., SET commands containing five integers per board, and homing commands when required). In the current implementation, the host may read and display a single response line when available, but actuation does not depend on a strict acknowledgment handshake. The protocol is therefore lightweight and sufficient for low-frequency interactions that produce a single static configuration per user action

Logging. During execution, the application provides runtime status feedback through the GUI (e.g., load success, parse warnings, connection failures). A persistent timestamped log file is not currently implemented in the provided codebase. If long-term traceability is required for evaluation, the system can be extended to save prompts, executed SET commands, reasoning output, and serial errors to a structured log (e.g., CSV/JSONL) for later analysis.

3.5 AI Reasoning and Interaction Layer

3.5.1 Dual-Channel Structure

The AI subsystem is implemented as a dual-channel interface in which motor actuation and explanation are separated into two independent system roles within the Python GUI. Each channel is executed with a distinct system prompt and produces a different output type. This separation is intentional: it maintains deterministic hardware actuation while enabling optional interpretive transparency at the interface level.

3.5.1.1 Action Channel: Strict Motor Command Generation

The action channel is responsible exclusively for producing a single executable instruction that specifies the target surface configuration as a 25-value height field corresponding to the 5×5 actuator grid. The intended output contract is one valid SET command containing 25 integers.

In the current implementation, formatting discipline is primarily enforced through the system prompt, which instructs the model to output motor-control text only and to return a fixed non-actuating response when a request cannot be mapped to surface motion. On the host side, the program parses the returned text to extract a SET instruction and validates that exactly 25 values are provided before dispatch. Height bounds are treated as part of the system’s actuation constraints; values originating from manual sliders and image-derived heightmaps are generated within the allowed range, and the embedded firmware additionally clamps commanded heights to the physical stroke limits during execution.

Figure 3-25. Action-channel system prompt enforcing a command-only output contract (excerpt).

Figure 3-26. Host-side extraction and validation prior to dispatch (excerpt).

This design constrains the action channel to a format that can be mechanically interpreted at the software-to-firmware boundary. When the model produces non-command text (including the fixed “Command ignored...” response), the parser does not generate motion and instead reports a parse warning in the GUI, ensuring that hardware actuation is not triggered by ambiguous output.

3.5.1.2 Reasoning Channel: Optional Explanatory Layer

The reasoning channel is an optional explanatory layer that is invoked only after an action has been executed. It is triggered through dedicated GUI controls (English and Korean) and receives the previous user prompt and the previously executed motor command as context. Its role is to provide a natural-language explanation of why the selected height distribution plausibly corresponds to the prompt.

To prevent interference with actuation, the reasoning channel uses a separate system prompt that explicitly forbids the generation of new motor commands. The returned reasoning text is displayed in a dedicated GUI output area and is not forwarded to the motor-command parser.

3.5.2 Purpose of the Separation

Figure 3-27. Reasoning-channel prompt constraints and context packaging (excerpt).

The dual-channel architecture is a deliberate design strategy that separates two incompatible requirements within a single interface: deterministic actuation and interpretive explanation. Because the action channel directly triggers physical motion, its output must remain unambiguous and machine-parseable. Any additional conversational text increases the likelihood of malformed commands and introduces avoidable uncertainty at the actuation boundary.

By contrast, the reasoning channel is non-operative. Its output is displayed only within the GUI and therefore may use descriptive and reflective language without affecting safety or determinism. This separation reduces mixed-role failure cases, improves predictability at the software-to-firmware boundary, and provides a clearer interaction model for users. More broadly, it aligns with common human–AI workflow patterns in which producing an output and explaining an output serve different evaluative purposes.

3.5.2 AI as Mediation Layer

Within the MKMS software stack, the LLM is positioned as a mediation layer between natural-language intent and a discrete surface representation. In operation, the action channel interprets a user prompt and returns a single target configuration expressed as a 25-value height field for the 5×5 actuator grid. The reasoning channel then provides an optional explanation of how the produced configuration plausibly corresponds to the prompt, using the executed command as context.

This framing treats the AI output as an interpretable translation rather than an optimized solution. The primary function of the AI layer is to support language-to-geometry mapping under a constrained representation and to make that mapping legible to the user through a separate explanatory interface. Accordingly, evaluation emphasizes command validity, execution reliability, and interpretability at the interface level rather than performance optimization.

3.5.3 Limitations of AI Integration

The current AI integration is exploratory and bounded by practical and methodological constraints. First, the LLM used in the prototype is a general-purpose model (gpt-oss 20B) selected for local deployment feasibility rather than domain specialization. The model is not trained or fine-tuned on kinetic surfaces, height-field control, or shape-forming tasks. Consequently, its outputs reflect general semantic priors rather than learned knowledge of MKMS-specific mechanics.

Second, response latency under the local inference configuration can be substantial, which limits interaction to low-frequency, discrete configuration changes. Third, the system does not implement sensor feedback or closed-loop correction; the AI layer does not incorporate physical state estimation, and actuation remains open-loop aside from homing reference. Finally, the mechanical platform is restricted to static height-field configurations and does not support continuous motion, trajectory-based animation, or sensor-driven behaviors.

These constraints define the scope of the prototype as a proof-of-concept for language-to-geometry mediation under deterministic actuation constraints rather than an autonomous morphing system.

3.6 System Integration

Integrating the mechanical, electronic, software, and AI layers required an iterative development process in which each subsystem was validated independently before being combined into a single operational workflow. Although the prototype does not implement closed-loop sensing or continuous feedback, functional integration is achieved through a structured data pathway that links interaction, translation, execution, and physical realization. In the fully integrated configuration, system operation follows the sequence: user input → optional AI interpretation → host-side command generation → distributed Arduino actuation → static physical shape formation.

Table 3-2 summarizes the integrated system stack and the primary role of each layer.

Layer	Function	Technologies Used
Hardware	Motor actuation and homing reference	NEMA-17 steppers, DRV8825 drivers, 2020-frame actuator modules, 25 limit switches
Firmware	Parsing of SET commands, synchronous stepping, homing sequence	Arduino C++ on five Arduino Mega boards
Software	GUI interaction, image-to-heightmap conversion, visualization, serial communication	Python (Tkinter, PIL, Matplotlib, PySerial)
AI Layer	Natural-language interpretation and optional reasoning	Local offline LLM (gpt-oss 20B) served via local API on RTX 4060 Ti (16 GB)

Table 3-2. Overall system stack

3.6.1 Data Flow Across Layers

Because the prototype does not incorporate encoders, force sensors, or closed-loop control, hardware actuation is commanded in an open-loop manner. Communication from the host to the embedded controllers is therefore primarily unidirectional with respect to motion execution. Nevertheless, the system forms a coherent interaction loop at the level of design practice, in which the user observes the resulting physical configuration and iteratively refines subsequent inputs.

The integrated data flow proceeds as follows:

1. **Human** → **Software**. The user specifies a target configuration through one of three input modalities: manual slider adjustment, file-based image input, or a natural-language prompt.
2. **Software** → **AI (optional)**. In AI-assisted mode, the natural-language prompt is passed to the action channel of the local LLM, which returns a single SET command specifying 25 target heights.
3. **Software** → **Firmware**. The host application parses and validates the target height field, partitions the 25 values into five groups, and transmits one per-board SET command to each Arduino over separate serial ports.
4. **Firmware** → **Hardware**. Each Arduino executes the homing or positioning routine for its five actuators and drives the corresponding stepper motors to the commanded heights.
5. **Hardware** → **Human**. The static surface configuration becomes physically visible on the morphing platform. In the current prototype, visual inspection of the resulting shape is the primary feedback mechanism.
6. **Human** → **Reasoning (optional)**. If explanation is desired, the user requests an interpretive account via the reasoning channel (English or Korean). The reasoning prompt is provided with the previous request and executed command as context and generates descriptive rationale without issuing any actuation instruction.

Although this sequence does not constitute a real-time cyber-physical feedback loop in the control-theoretic sense, it functions as an interaction loop in which observation and interpretation guide subsequent design inputs.

3.6.2 Practical Integration Constraints

Several practical constraints shaped the integration architecture. First, the prototype does not include force sensors, encoders, or other feedback instrumentation. As a result, the hardware cannot detect stalled or overloaded pins, and the firmware does not perform autonomous correction. In addition, motor enable lines are not safety-shared. Each actuator is controlled by an independent EN pin, and the system therefore does not provide a single global emergency shutdown mechanism in the current implementation.

The platform is further constrained to static actuation. It produces only single settled height configurations rather than continuous waves or real-time responsive behaviors. Because the five Arduino Mega boards do not communicate with one another, inter-board coordination is delegated entirely to the host computer, which must correctly partition and assign target heights.

Finally, the AI layer introduces practical latency that is typically on the order of tens of seconds per command, given local inference constraints on the available GPU. The model is also general-purpose rather than trained specifically for height-field shape generation or kinetic actuation tasks. Collectively, these constraints define the operational boundaries of the current prototype and clarify the scope of the research contribution.

3.6.3 Integration Verification Through LLM-Control Tests

To verify reliable end-to-end integration of the system stack, the prototype was evaluated using a structured set of LLM-driven shape tests, which are detailed in Chapter 4. These tests were designed to validate the complete execution chain from semantic input to physical output. Specifically, they assessed (1) the consistency of mapping between natural-language prompts and AI-generated SET commands, (2) the host software’s ability to parse commands and distribute target heights across five independent serial connections, (3) correct firmware execution of the received height values, and (4) physical formation of the intended static surface configurations. Where requested, the

evaluation also recorded the reasoning channel outputs to assess explanatory consistency relative to the executed commands.

The test set included representative geometric and representational patterns such as ramps, pyramids, letter-like glyphs, radial peaks, checkerboard configurations, smoothing and mirroring operations, and constraint-handling cases. Collectively, these experiments demonstrated that the integrated workflow from user prompt to hardware output operated coherently within the system’s static actuation constraints.

3.6.4 Summary

Despite the absence of sensing, real-time dynamics, and domain-trained AI models, the prototype achieves functional integration across the primary system layers. The integrated platform combines distributed motor actuation through five microcontroller boards, a Python-based GUI that supports multiple input modalities, and a locally hosted LLM that enables natural-language interpretation with optional bilingual explanation. The system further includes integrated 3D visualization for configuration preview and a deterministic firmware layer that executes bounded SET commands reliably.

Taken together, these components constitute an early-stage morphing interface in which language, computation, and physical form are linked through a unified workflow. While the system is not closed-loop in a control-theoretic sense, it supports a human-mediated feedback loop in which users observe outcomes, refine prompts or inputs, and iteratively explore the space of discretized physical configurations.

4. Implementation and Experiments

4.1 Prototype Fabrication and Assembly

The full-scale prototype was implemented as a 5×5 motor array (25 actuated pins), intended as a modular unit that can be conceptually extended to larger morphing surfaces. This scale was selected as a practical balance between expressiveness and feasibility, enabling meaningful height-field forms while remaining manageable for mechanical alignment, wiring, and debugging.



Figure 4-1. An FDM printer (Creality K1C) used for fabrication of prototype components.

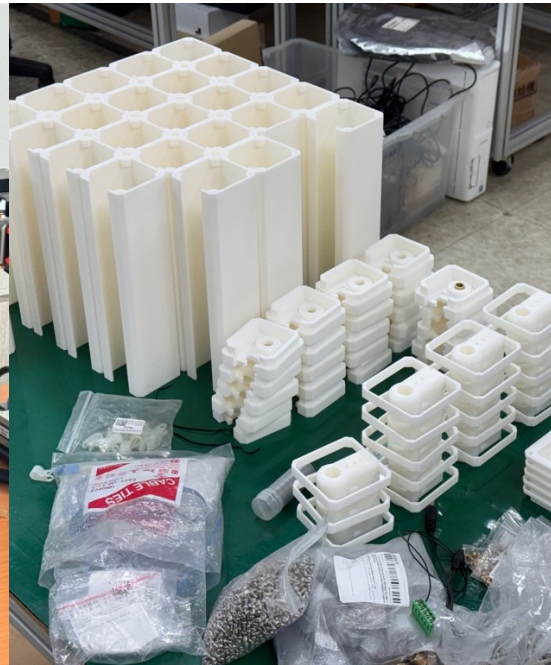


Figure 4-2. 3D-printed structural and actuator components prior to assembly.

The base structure was assembled using 20×20 mm aluminum extrusion to ensure rigidity and reconfigurability during iteration. Each actuator module consists of a NEMA-17 stepper motor, a lead screw mechanism, dual guide rods, and a 3D-printed carriage and housing. Most mechanical interfaces, including motor mounts and pin guides, were fabricated in PLA to support rapid modification when tolerances or alignment issues were identified. Actuators were arranged on a 70 mm grid, and each actuator provides approximately 200 mm of vertical travel. A bottom-position limit switch was installed for each actuator to establish a consistent homing reference.



Figure 4-3. Partially assembled 5×5 actuator frame during installation of five Arduino Mega boards.

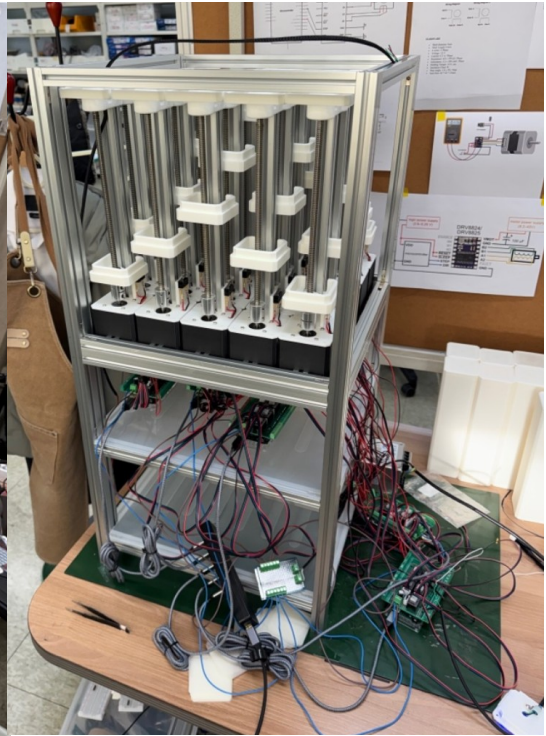


Figure 4-4. Partially assembled 5×5 actuator frame with five Arduino Mega boards installed prior to final cable management.

4.2 Graduation Exhibition Demonstration

The completed MKMS prototype was deployed as an interactive demonstration during the graduation exhibition. The system operated for approximately three days at roughly six hours per day, using the full integrated workflow, including the Python GUI, serial dispatch to five Arduino Mega controllers, and the optional LLM-based interaction functions. Across the exhibition operating period, no major failures were observed, and routine operation remained stable under repeated use. This deployment is reported as a practical confirmation of integration robustness in a public setting rather than as a formal user evaluation; no structured usability study, survey, or quantitative perception metrics were conducted during the exhibition.



Figure 4-5. Graduation exhibition demonstration: image-to-heightmap control mode.

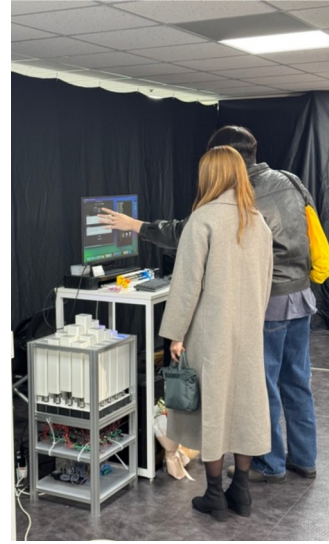


Figure 4-6. Graduation exhibition demonstration: LLM-assisted control mode.

4.3 Homing, Reference State, and Repeatability

A bottom limit-switch routine was used to define the reference state for positioning. During homing, each actuator is driven downward until the corresponding limit switch triggers, after which the internal step counter is reset to zero. This method provides a consistent and implementable baseline without requiring encoders or additional sensing.

Repeatability in the present prototype is bounded by mechanical and electromechanical factors, including switch hysteresis, backlash in the screw–nut mechanism, and structural compliance. For this reason, the evaluation in this chapter emphasizes stable execution and observable geometric plausibility rather than precision metrology.

4.4 Experimental Methodology

This chapter evaluates the MKMS as an end-to-end workflow that converts user input into a single settled static configuration of the 5×5 surface. Because the system does not include encoders, force sensing, or closed-loop correction, experiments are framed in terms of operational validity and workflow reliability rather than high-precision performance metrics. Accordingly, the evaluation focuses on (1) command validity and bounded actuation within the mechanically safe range, (2) robustness of the software-to-firmware dispatch pipeline across five distributed controllers, and (3) qualitative correspondence between intended and realized static geometry under researcher inspection of the physical surface and the 3D previews.

The control application, described in Chapter 3, supports three input modalities: manual slider control, image-to-heightmap conversion, and LLM-mediated command generation. Manual slider mode was not treated as an independent experimental condition, since it directly specifies target heights without an interpretation layer. Instead, it served as supporting infrastructure for calibration, homing verification, and debugging, and its functional correctness was confirmed repeatedly during the setup and execution of image-based and LLM-based trials.

The following sections apply this methodological framing to the two primary workflows of interest: the deterministic image control pipeline and the LLM-controlled pipeline, which is evaluated using the structured, category-based prompt set introduced earlier in Chapter 4.

4.5 Image-to-Heightmap Trials (Physical Experiments)

Image-to-heightmap trials were conducted to verify deterministic translation from 2D input data to a static physical height-field. Each trial followed a consistent procedure: (1) load a raster image (PNG/JPEG), (2) convert the image to grayscale, (3) downsample to a 5×5 grid aligned to the actuator layout, (4) map pixel intensity (0–255) linearly to actuator height (0–200 mm), and (5) execute the resulting configuration as a single static move.

For each input, the GUI generated a Matplotlib-based 3D preview of the computed heightmap. The corresponding physical output was then observed to confirm whether the relative height relationships and overall structure matched the preview. This experiment group validates the deterministic portion of the pipeline, including preprocessing, numeric mapping, segmentation of 25 values into five controller messages, and stable settling into a final static configuration.

4.6 LLM-Control Tests and Category-Based Prompt Framework

LLM-based evaluation was conducted using a structured prompt set designed to assess both semantic shape synthesis and iterative modification under strict actuation constraints. Rather than relying on ad hoc prompts, test cases were authored with five categories in mind. These categories functioned as a testing standard to ensure coverage of baseline geometry generation, constraint compliance, and editing operations applied to existing surfaces.

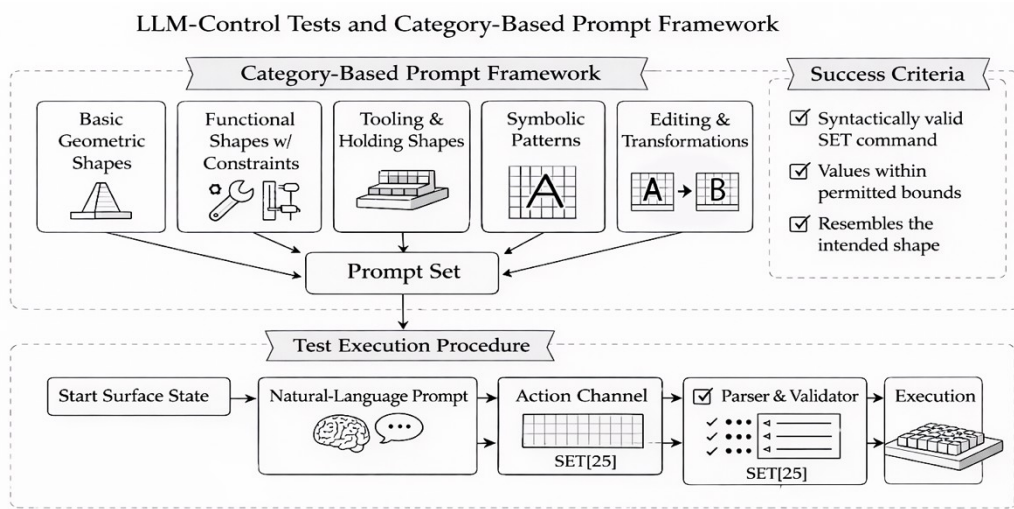


Figure 4-7. LLM-control tests and category-based prompt framework.

The five categories were defined as follows:

1. **Basic Geometric Shapes:** canonical height-field forms used to evaluate baseline mapping from descriptive language to coherent static geometry, including ramps, pyramids, bowls, peaks, and planar offsets.
2. **Functional Shapes with Constraints:** cases designed to stress the command contract and mechanical bounds, including strict height limits (0–200 mm), locally flat regions, monotonic gradients, and safe handling of invalid or out-of-scope requests.
3. **Tooling and Holding Shapes:** configurations analogous to seating, pockets, supports, or holding regions, used to test whether the model can generate spatially interpretable functional zones within a coarse 5×5 discretization.
4. **Symbolic Patterns:** letter-like or icon-like surfaces used to test recognizability at low resolution, where evaluation prioritizes legibility over geometric fidelity.
5. **Editing and Transformation of Existing Surfaces:** follow-up prompts requiring modification of a previously generated height-field, such as mirroring, smoothing, global offsets, selective reduction of peaks, or quantization-like adjustments.

Test execution procedure. Each test began from a defined surface state, typically a flat configuration established after homing. A natural-language prompt was submitted through the GUI and forwarded to the action channel. The action channel was required to output a single SET command containing 25 numeric values. The host application parsed the returned text, verified that exactly 25 values were present, and checked that values respected the permitted range before distributing the command across five serial ports for execution. For editing cases (Category 5), prompts were issued sequentially and interpreted as transformations of the most recent surface state.

Success criteria and interpretation limits. A test was considered successful when (1) the action channel produced a syntactically valid SET command, (2) values were within permitted bounds, and (3) the executed configuration exhibited the intended category-level structure under researcher inspection. No user study or perceptual survey was conducted; therefore, statements about prompt-to-shape correspondence are reported as researcher-observed plausibility rather than validated perception metrics. A structured perceptual evaluation is deferred to future work.

4.7 Results and Evaluation

The evaluation focused on (1) the physical performance of the image-to-heightmap workflow and (2) the expressive and interpretive behavior of the LLM-driven control interface. Because the prototype does not implement continuous sensing or closed-loop correction (beyond homing via limit switches) and does not support dynamic motion, results are reported primarily in terms of workflow reliability and researcher-observed qualitative correspondence between intended and realized static configurations.

4.7.1 Image-to-Heightmap Results

Across the image trials, the pipeline successfully produced physical topographies consistent with the processed 5×5 heightmaps and their Matplotlib previews. Gradients and ramp-like structures manifested clearly as monotonic surfaces, while peaks and Gaussian-like bumps remained distinguishable despite the low spatial resolution. More complex images lost fine detail, but coarse structure such as separation of light and dark regions remained readable. Collectively, these observations support the claim that the system reliably executes the full pipeline from image input to physical form under static actuation.

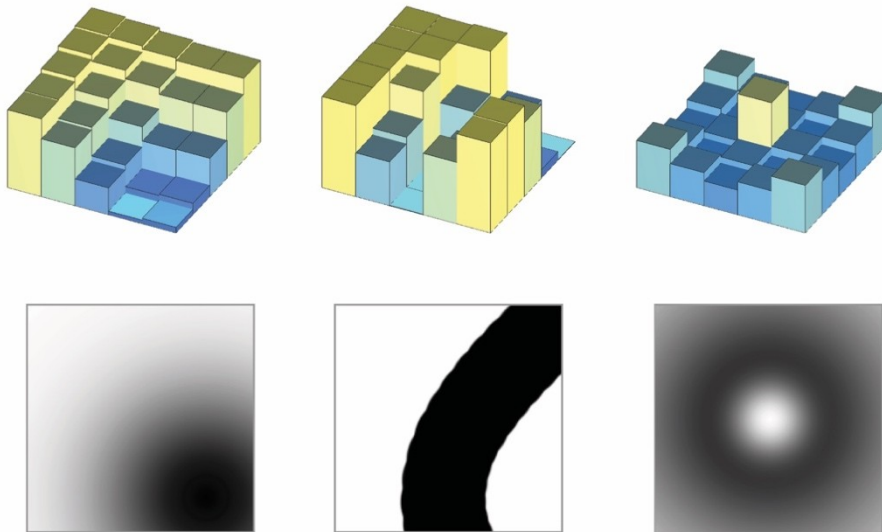


Figure 4-8. Image-to-heightmap validation. Example grayscale source images (bottom) and corresponding 5×5 reliefs (top) produced by the image control pipeline, showing gradient, edge, and radial profiles.

4.7.2 LLM Action Channel Results

This subsection reports the observed behavior of the LLM-driven control interface using the five-category test framework (Category 1 to Category 5). As defined in Section 4.4, “success” is interpreted at an operational and category level: the action channel must output a syntactically valid SET command within bounds, and the resulting 5×5 relief must exhibit the intended structural characteristics of the test category. Because no perceptual survey or recognition scoring was conducted, the interpretations below should be read as researcher-observed plausibility checks rather than validated measures of how third-party observers would perceive semantic match.

Category 1: Basic Geometric Shapes. Prompts in Category 1 targeted canonical height-field primitives, including flat planes, linear ramps, pyramids, and radial forms. The system produced structurally coherent configurations for uniform planes at multiple height levels and for monotonic ramps along both X and Y directions. More complex primitives, such as a pyramid, inverted pyramid, saddle-like relief, and radial mound/crater forms, were also realized as discrete approximations consistent with expectations for a low-resolution grid. Overall, Category 1 demonstrated that the LLM can reliably map common geometric descriptors into stable height distributions that preserve the intended global structure when sampled at 5×5 resolution.

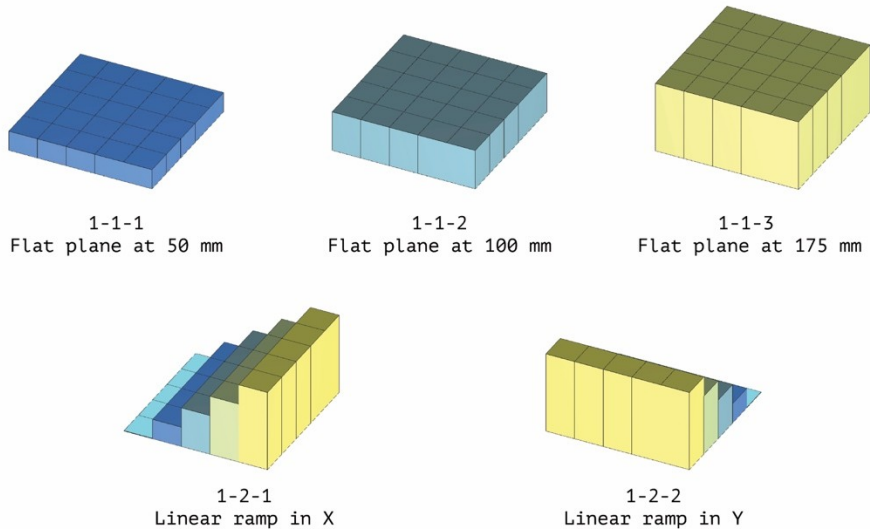


Figure 4-9. Basic geometric primitives (Category 1). LLM-generated flat planes at absolute heights of 50, 100, and 175 mm (1-1-1 to 1-1-3) and linear ramps in the X and Y directions (1-2-1, 1-2-2), used as baseline tests of monotonic height control on the 5×5 pin array.

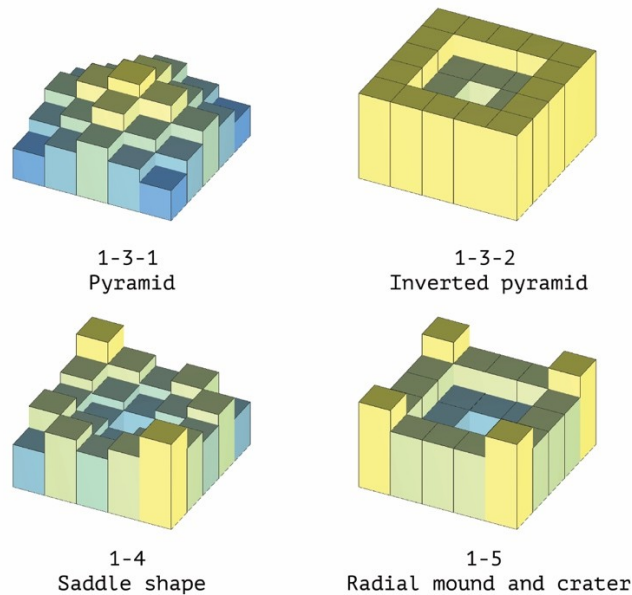


Figure 4-10. Basic non-planar geometric shapes (Category 1). LLM-generated non-planar surfaces—pyramid, inverted pyramid, saddle shape, and radial mound with central crater (1-3-1 to 1-5)—used to evaluate the system’s ability to approximate simple curved and stepped geometries.

Category 2: Functional Shapes with Constraints. Category 2 prompts required explicit boundary conditions or constraint satisfaction, such as endpoints, footprint limits, stepwise quantization, bounded maxima, and “moat” style separation between regions. The system generated configurations that reflected the requested constraints at the category level, including a ramp spanning a specified height range, a bounded box-like plateau within a prescribed footprint, and a staircase with discretized step heights. Constraint-oriented prompts that combine multiple objectives (for example, bounding all pins below a threshold while maximizing central volume) were handled as plausible compromises that preserved the prompt intent in overall structure rather than as provably optimal solutions. These results suggest that the LLM can operationalize constraint language into feasible surfaces, but that constraint satisfaction is best interpreted as qualitative compliance within a coarse grid, not as an optimization result.

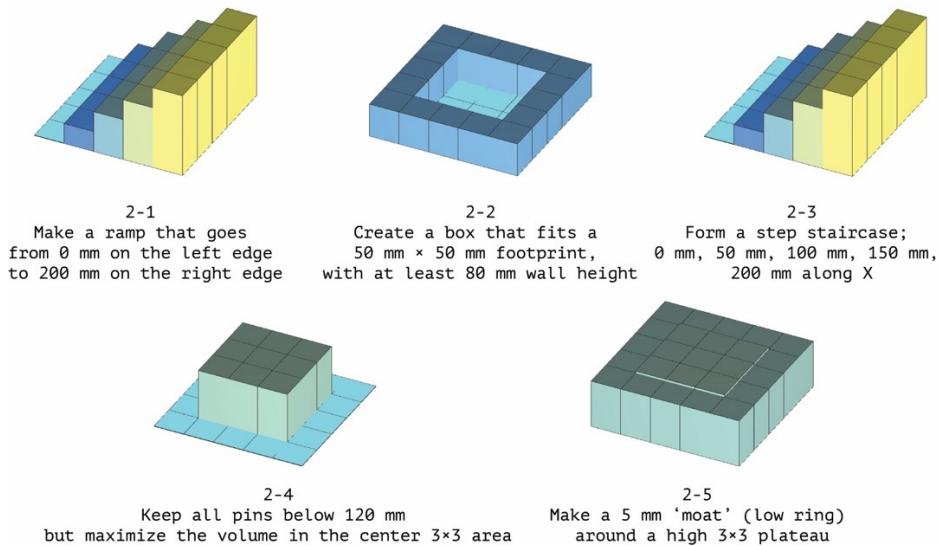


Figure 4-11. Tooling-oriented primitives generated by the LLM (Category 2). Ramp, boxed cavity, stepped staircase, constrained central plateau, and moat-like ring (2-1 to 2-5) produced from functional prompts, used to test whether the system can satisfy simple geometric and height constraints relevant to tooling.

Category 3: Tooling and Holding Shapes. Category 3 tested whether the LLM could generate forms that resemble practical support or guiding geometries, including gutters, parallel ridges, shallow cradles, concave pressing surfaces, and friction-enhancing row patterns. The produced configurations generally expressed the intended functional topology: channel-like depressions for guidance, raised rails for support, and alternating height rows to increase surface roughness under a sheet. While physical performance was not measured with sensors or force instrumentation, the surfaces were geometrically consistent with the intended holding or guiding logic at the level of discrete relief features. This category supports the claim that the interface can translate functional language into recognizable structural arrangements, even though it cannot validate mechanical efficacy in a quantitative manner.

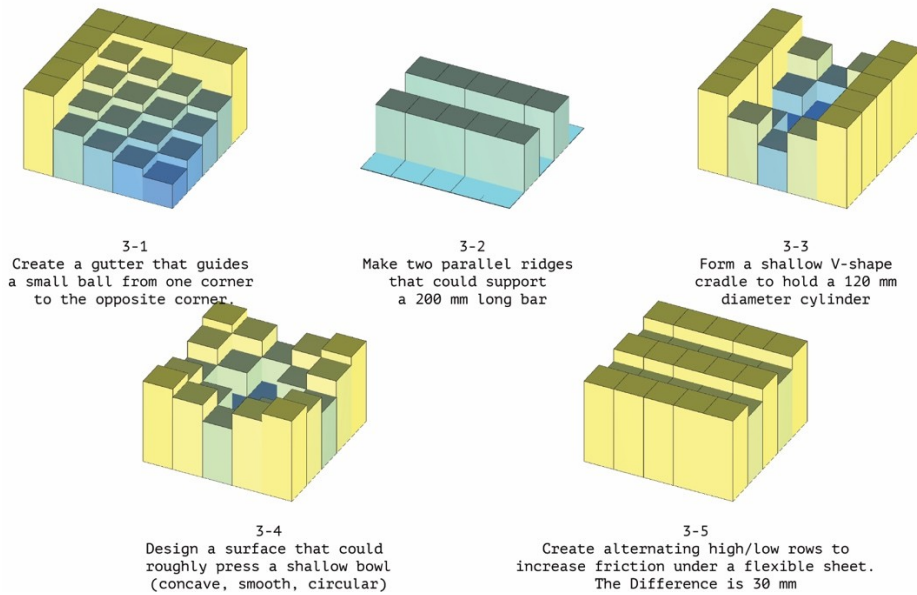


Figure 4-12. Object support and holding shapes generated by the LLM (Category 3). Gutter, dual ridges, shallow V-shaped cradle, bowl-like concave surface, and alternating high/low friction rows (3-1 to 3-5), demonstrating the system's ability to form holding and support geometries for guiding or constraining physical objects.

Category 4: Symbolic Patterns. Category 4 prompts evaluated symbolic legibility, including letters and simple icons. The system produced recognizable single-character outputs within the 5×5 constraint, with letter-like structures relying on strong strokes, simplified diagonals, and block-based approximations. However, multi-character composition revealed the representational limits of the array. Attempting to render a full word exceeded the practical bandwidth of a 5×5 grid and resulted in partial or fragmented symbolic forms. In this sense, the category demonstrates both capability and limitation: symbolic patterns are feasible when the symbol is compressible into a small set of high-contrast strokes, but longer strings and fine typography are not reliably representable at the current resolution.

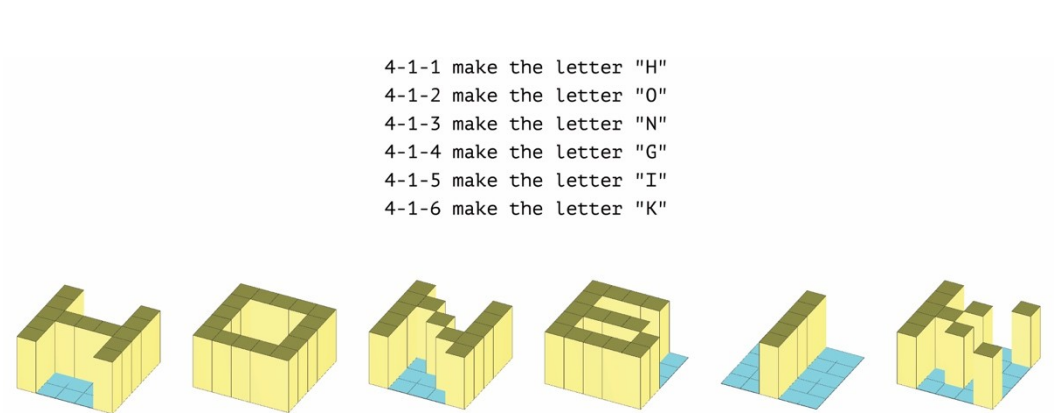
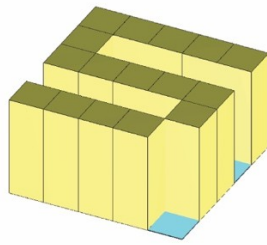
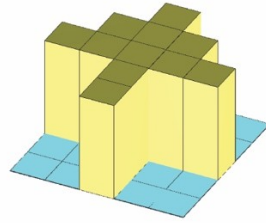


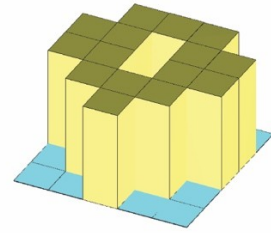
Figure 4-13. Symbolic letter patterns generated by the LLM (Category 4). LLM-generated 5×5 surfaces for the letters H, O, N, G, I, and K (4-1-1 to 4-1-6), collectively attempting to spell “HONGIK” on the low-resolution pin array.



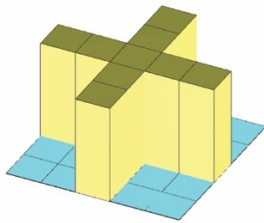
4-2
make the number "5"
as clearly as possible



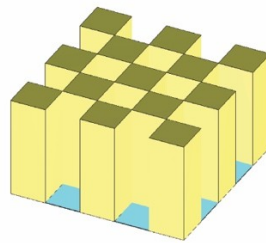
4-3-1
make an arrow icon



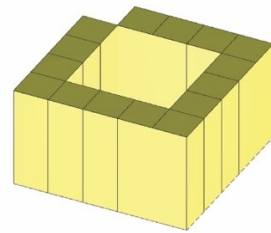
4-3-2
make a 2d heart icon



4-3-3
make a plus sign



4-4
make Checkerboard pattern
(alternating high/low squares)



4-5
make a maze-like ring
(high outer frame,
low inside, one opening)

Figure 4-14. Iconic and symbolic patterns generated by the LLM (Category 4). LLM-generated symbolic shapes including the digit "5," an arrow icon, a 2D heart, a plus sign, a checkerboard pattern, and a maze-like ring (4-2 to 4-5), used to evaluate recognizable symbolic forms on the morphing surface.

Category 5: Editing and Transformation of Existing Surfaces. Category 5 tested prompts that require operating on an existing configuration rather than generating a shape from scratch. The results indicate that the LLM can perform plausible transformation behaviors that resemble matrix-level edits, including smoothing under a neighbor-difference constraint, global reduction of heights by a fixed offset, left-right mirroring, and selective modification (for example, lowering only the highest pins). These outcomes are significant for interaction design because they demonstrate a mode of iterative refinement, where the surface can be treated as an editable state and adjusted through successive language commands. As with the other categories, the transformations are best interpreted as qualitative correctness and operational feasibility rather than as guaranteed exact numeric equivalence to an ideal transformation operator.

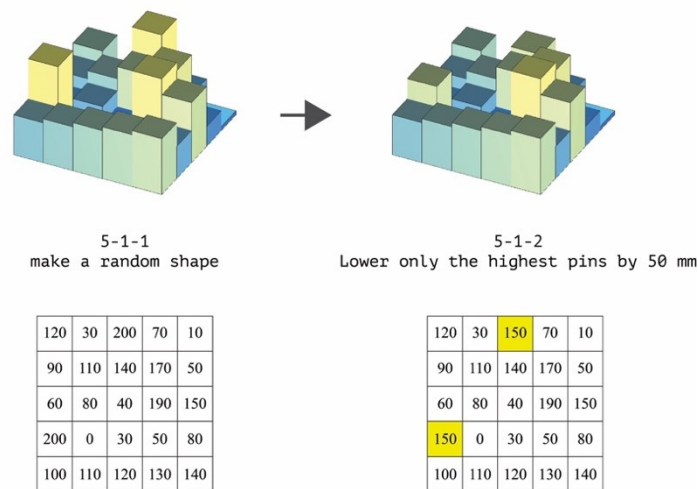


Figure 4-15. Category 5 – Editing and Transformation of Existing Surfaces (5-1). The LLM first generates a random surface (5-1-1), then applies the instruction “lower only the highest pins by 50 mm” (5-1-2). The tables show the original and modified heightmaps, with changed cells highlighted.

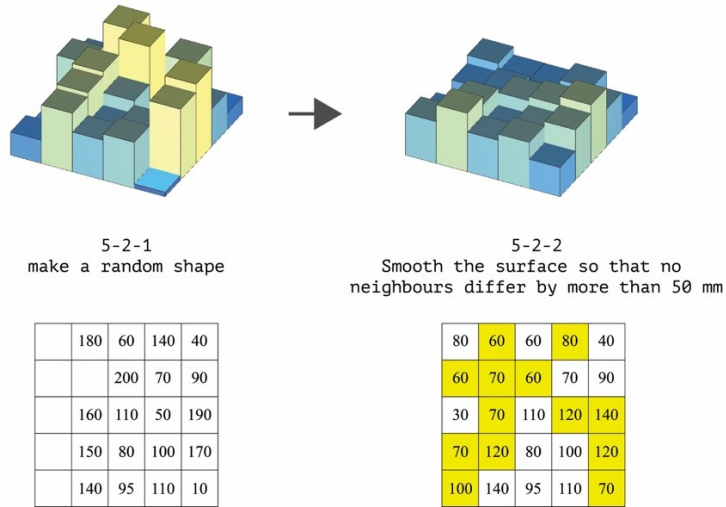


Figure 4-16. Category 5 – Editing and Transformation of Existing Surfaces (5-2). From an initial random surface (5-2-1), the LLM produces an edited surface (5-2-2) that “smooths the surface so that no neighbours differ by more than 50 mm,” with adjusted values highlighted in the heightmaps.

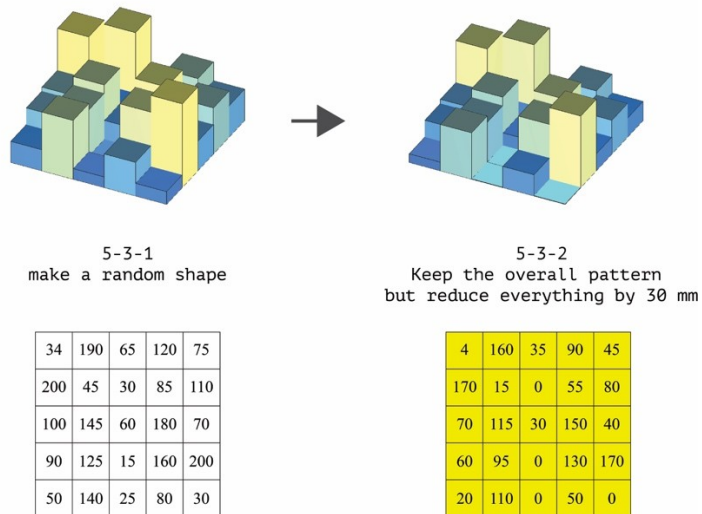
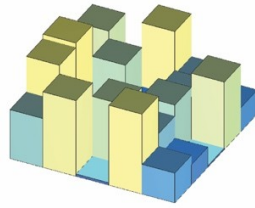
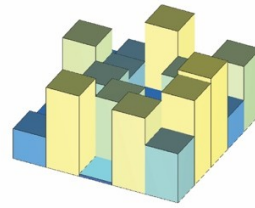


Figure 4-17. Category 5 – Editing and Transformation of Existing Surfaces (5-3). Starting from a random surface (5-3-1), the LLM “keeps the overall pattern but reduces everything by 30 mm” (5-3-2); the tables show original and modified heightmaps, with all heights uniformly offset and clamped at zero where necessary.



5-4-1
make a random shape

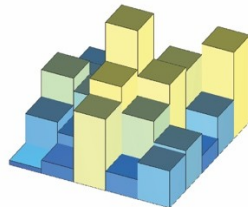
147	23	186	89	54
71	130	42	98	152
200	15	66	120	3
190	80	140	95	45
110	170	5	180	67



5-4-2
Mirror the current shape left-right

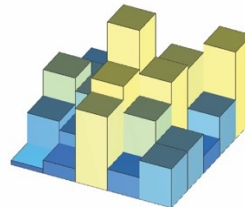
54	89	186	23	147
152	98	42	130	71
3	120	66	15	200
45	95	140	80	190
67	180	5	170	110

Figure 4-18. Category 5 – Editing and Transformation of Existing Surfaces (5-4). The LLM transforms a random surface (5-4-1) into a left-right mirrored configuration (5-4-2), as reflected in the reordered values of the corresponding heightmaps.



5-5-1
make a random shape

		45	120	200
		25	175	90
		15	185	40
		200	140	92
		165	45	85



5-5-2
snap every pin to the nearest 10 mm

70	190	50	120	200
90	60	30	180	90
140	110	20	190	40
80	60	200	140	90
10	30	170	50	90

Figure 4-19. Category 5 – Editing and Transformation of Existing Surfaces (5-5). From a random surface (5-5-1), the LLM generates a second command (5-5-2) that snaps every pin to the nearest 10 mm increment, converting arbitrary heights into a quantized grid.

4.7.3 LLM Reasoning Channel Results

This subsection reports the observed behavior of the LLM reasoning channel, which produces a natural-language explanation of the already executed surface configuration. In contrast to the action channel, the reasoning channel is non-operative: it does not generate motor commands and its output is displayed only within the GUI. The analysis therefore focuses on whether the reasoning text (1) correctly references salient geometric properties of the executed 5×5 height field and (2) communicates the mapping between prompt intent and the resulting discretized configuration in a readable manner. Because no user study, recognition scoring, or inter-rater agreement analysis was conducted, the observations below should be interpreted as researcher-based plausibility checks of explanatory adequacy rather than validated measures of semantic correspondence.

Reasoning outputs were recorded for seven representative generation trials (R1–R7). For presentation clarity, the original reasoning texts were edited for length and formatting while preserving their substantive claims about the generated shapes. The full, unedited LLM reasoning outputs for R1–R7 are provided in Appendix A.

R1 Centered hill. For the prompt requesting a hill with a central maximum and decreasing height toward the edges, the reasoning text correctly identified a single highest value located at the center of the 5×5 grid and described a symmetric falloff toward the perimeter. The explanation emphasized stepped height levels as a discrete approximation of a radial mound, which is consistent with the representational limits of the low-resolution surface.

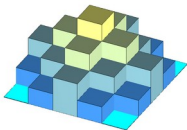
Physical Output	Action Prompt / Executed SET command
	Create a hill with the highest point at the center and decreasing toward the edges SET 0 50 100 50 0 50 100 150 100 50 100 150 200 150 100 50 100 150 100 50 0 50 100 50 0

Table 4-1. Prompt and SET command for reasoning example R1 (centered hill)

R2 Crater. For the crater prompt, the reasoning output described a low center point surrounded by a higher ring and a lower outer region. The explanation mapped this structure to crater-like topology by distinguishing between the “pit” at the center, the “rim” in the immediate neighborhood, and the background terrain toward the edges. This description aligned with the dominant global features of the executed configuration.

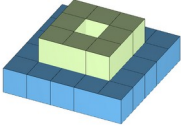
Physical Output	Action Prompt / Executed SET command
	Create a crater: low center, higher ring around it, and lower edges SET 75 75 75 75 75 75 150 150 150 75 75 150 0 150 75 75 150 150 150 75 75 75 75 75

Table 4-2. Prompt and SET command for reasoning example R2 (crater)

R3 Left-to-right ramp. For the ramp increasing from left to right, the reasoning channel characterized the configuration as a monotonic increase along the horizontal axis and noted that the same column-wise gradient is repeated across rows, yielding a planar incline rather than a localized ridge. The explanation therefore correctly articulated the intended directional structure at the grid level.

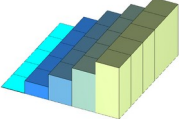
Physical Output	Action Prompt / Executed SET command
	Make a smooth ramp that increases from left to right SET 0 40 80 120 160 0 40 80 120 160 0 40 80 120 160 0 40 80 120 160 0 40 80 120 160

Table 4-3 . Prompt and SET command for reasoning example R3 (left-to-right ramp)

R4 Front-to-back ramp. For the ramp increasing from front row to back row, the reasoning output described the surface as row-wise bands of increasing height, with the front row at the lowest level and the back row at the highest. It further noted that uniform increments across rows produce a consistent incline. This reasoning was consistent with how the height values were organized into the 5×5 array

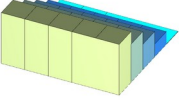
Physical Output	Action Prompt / Executed SET command
	Make a ramp that increases from the front row to the back row. SET 0 0 0 0 0 40 40 40 40 40 80 80 80 80 80 120 120 120 120 120 160 160 160 160 160

Table 4-4. Prompt and SET command for reasoning example R4 (front-to-back ramp)

R5 Stepped pyramid. For the stepped pyramid prompt, the reasoning channel reported three discrete levels: a low outer border, a higher middle region surrounding the center, and a single high peak at the center cell. The explanation explicitly framed the result as a stepped approximation rather than a continuous pyramid slope, which appropriately reflects the discrete actuation model.

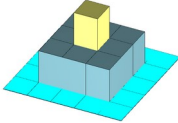
Physical Output	Action Prompt / Executed SET command
	Create a stepped pyramid with three levels: low outer border, medium middle ring, high center. SET 0 0 0 0 0 0 100 100 100 0 0 100 200 100 0 0 100 100 100 0 0 0

Table 4-5. Prompt and SET command for reasoning example R5 (three-level stepped pyramid)

R6 Checkerboard pattern. For the two-level checkerboard prompt, the reasoning output identified the use of two height levels and an alternating assignment rule across adjacent cells. The explanation correctly treated alternation as the governing structural constraint and presented the pattern as a high-contrast configuration suited to coarse spatial sampling.

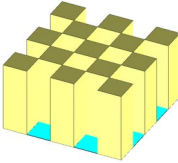
Physical Output	Action Prompt / Executed SET command
	Create a checkerboard pattern using two height levels (high and low) SET 200 0 200 0 200 0 200 0 200 0 200 0 200 0 200 0 200 0 200 0 200 0

Table 4-6. Prompt and SET command for reasoning example R6 (two-level checkerboard pattern)

R7 Symbolic letterform “H”. For the letter “H” prompt, the reasoning text mapped the 25 values to a 5×5 grid and described two vertical strokes on the left and right columns and a horizontal crossbar across the middle row, with remaining cells kept low for contrast. The explanation therefore captured the principal structural features required for symbolic legibility under a 5×5 constraint.

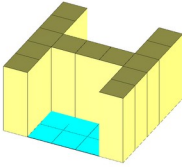
Physical Output	Action Prompt / Executed SET command
	Form the letter ‘H’ using high pins on a low background. SET 200 0 0 0 200 200 0 0 0 200 200 200 200 200 200 200 0 0 0 200 200 0 0 0 200

Table 4-7. Prompt and SET command for reasoning example R7 (symbolic letterform “H”)

4.7.4 Summary of Results

Across R1–R7, the reasoning channel consistently grounded its explanations in simple geometric primitives (peak, ring, monotonic slope, stepped levels, alternation, and stroke-based glyph formation). The explanations were generally traceable to the executed numeric patterns and aligned with the representational logic of a 5×5 discretized height field. Because no user-based study was conducted, these results should be interpreted as evidence of explanation coherence and traceability rather than as validated measures of how humans perceive semantic match.

5. Discussion

5.1 Overview

Chapter 4 demonstrated that the Motor-Based Kinetic Morphing System (MKMS) can execute stable, bounded static surface configurations from heterogeneous inputs, including image-derived heightmaps and language-mediated commands. It also showed that the dual-channel AI interface can separate actuation commands from explanatory text, supporting predictable hardware behavior while providing an optional interpretive layer. This chapter discusses the implications of these results by focusing on what the prototype suggests about (1) distributed actuation as a software-defined surface, (2) language-based interaction with physical morphing systems, and (3) the conceptual role of explanation when AI outputs are materialized.

5.2 Technical Implications of a Distributed Morphing Surface

The prototype validates a practical architecture for a small-scale reconfigurable height-field surface using distributed embedded control. By allocating five actuators per Arduino Mega and coordinating them through a host application, the system achieves reliable execution for static target configurations without requiring inter-board communication. Within the evaluation scope, this supports the feasibility of modular pin-surface construction using relatively accessible components and a simple command contract.

At the same time, the results clarify the current technical boundary of the platform. The system operates as an open-loop device with no sensing, no encoder feedback, and no verification of achieved height beyond visual inspection. As a consequence, reliability is best understood as repeatable execution of commanded motions under controlled conditions, rather than guaranteed geometric accuracy in the engineering metrology sense. In this configuration, the system functions most credibly as a proof-of-

concept for integrated pipeline behavior, not as a precision motion platform or a forming tool.

5.3 Interaction Implications of Language-Mediated Control

The evaluation indicates that language prompts can serve as an effective high-level input modality when the system constrains outputs to a strict command format and bounded actuation range. In this thesis, the action channel is intentionally framed as a constrained command generator that produces one static SET instruction per request, rather than as an optimizer, planner, or adaptive controller. This design choice aligns with the prototype’s static actuation model and reduces ambiguity at the hardware boundary.

The results also suggest that the LLM interface is most dependable when the requested intent can be expressed as a global structure on a discretized 5×5 grid, such as monotonic gradients, centered extrema, bilateral symmetry, alternation patterns, or ring-like distributions. Requests that imply thin strokes, diagonal curvature, or fine local detail are more likely to yield ambiguous or degraded outputs, which is consistent with the representational limits of a low-resolution height field. These observations should be read as interface-level performance characteristics within the prototype’s constraints, not as general claims about language models.

5.4 Role of Explanation and Interface Transparency

A central contribution of the MKMS interface is the explicit separation between action and explanation. The reasoning channel does not affect hardware behavior and is therefore positioned as an interpretive layer that can help users understand how a prompt was mapped into a discrete surface configuration. In the present study, reasoning outputs are treated as interface transparency artifacts rather than as evidence of semantic correctness.

Because this thesis does not include a formal perceptual evaluation of whether observers interpret the resulting shapes as matching prompt meaning, the explanatory text is discussed in terms of its communicative function. In other words, the reasoning channel can clarify what the system attempted to do and why, but it does not validate that the produced geometry is semantically “correct” in a statistically supported sense. This framing is important for keeping the interpretation of results aligned with the evaluation scope.

5.5 Limitations and Future Directions

Several limitations follow directly from the hardware and compute configuration used in this prototype. First, the system is limited to static configurations: it does not perform continuous motion, real-time responsiveness, or sensor-driven adaptation, and therefore cannot be evaluated as a dynamic kinetic display or an interactive closed-loop cyber-physical platform. Second, the AI integration remains exploratory. The local model (gpt-oss 20B) runs offline via a local API on an RTX 4060 Ti (16 GB) and is neither trained nor fine-tuned specifically for pin-surface control, so the latency observed during operation constrains the interaction tempo and makes the AI mode more appropriate for demonstration and reflective exploration than for time-critical use. Third, scalability remains an open engineering question: while the distributed approach is effective at 25 actuators, larger arrays would require rethinking power distribution, wiring complexity,

and communication architecture, likely moving toward networked embedded nodes and more structured synchronization strategies.

Future work should therefore focus on three directions: (1) scalable control architectures for higher-resolution arrays, (2) sensing and closed-loop verification methods that support repeatable positioning, and (3) model-adaptation strategies that encode system constraints more directly, whether through training data, constraint-aware decoding, or hybrid rule-based post-processing.

5.6 Summary

This chapter interpreted the Chapter 4 results as evidence that the MKMS functions as an integrated proof-of-concept for translating heterogeneous inputs into deterministic static surface states. The technical contribution lies in demonstrating a workable small-scale distributed actuation stack coordinated by a host application. The interaction contribution lies in showing that language-mediated control can be operationally reliable when it is constrained by a strict command contract and bounded actuation rules. Finally, the conceptual contribution lies in treating explanation as part of the interface, where reasoning is separated from action and framed as transparency rather than as validation of semantic correctness.

6. Conclusion

6.1 Research Summary and Outcomes

This thesis examined how mechanical actuation, computational control, and large language models can be integrated into a single embodied system for height-based shape formation. The outcome of this research is the Motor-Based Kinetic Morphing System (MKMS), a 5×5 reconfigurable pin surface driven by 25 stepper-motor actuators and controlled through a two-layer software stack: embedded Arduino firmware and a Python host application.

The MKMS was developed as a proof-of-concept platform rather than an industrial forming tool. Accordingly, the evaluation emphasized operational validity and end-to-end integration. Across the reported trials, the system demonstrated that (1) image-derived heightmaps can be mapped to a 5×5 grid and executed as stable static configurations, and (2) a locally hosted LLM can be constrained to generate hardware-compatible commands and provide optional, non-actuating explanations through a separated reasoning channel. The findings support the feasibility of a language-mediated, explainable interaction paradigm for an embodied morphing surface under open-loop constraints and low spatial resolution.

6.2 Contributions

The contributions of this thesis are primarily architectural and methodological. They are summarized as follows.

(1) **Integrated morphing-surface prototype with distributed actuation.** The research presents a functioning 5×5 morphing surface and a distributed control approach in which five Arduino Mega boards each execute homing and motion for a fixed subset of actuators. This configuration demonstrates a practical method for scaling a pin array beyond a single controller while keeping firmware responsibilities localized and deterministic.

(2) **Host-side software pipeline for heterogeneous inputs.** The Python application unifies multiple input modalities (manual height specification and image-to-heightmap conversion) with visualization and serial communication. This establishes a reproducible pipeline from user input to actuator commands, suitable for iterative testing and demonstration of static topographic output.

(3) **Dual-channel LLM interface for constrained actuation and optional explanation.** A central contribution is the separation of the AI subsystem into two roles: an action channel that produces strictly formatted motor commands, and a reasoning channel that produces explanatory text while being prohibited from generating actuation commands. This separation provides a concrete interaction pattern for embedding an LLM into a physical system where safety and determinism are required at the command layer, while interpretability can remain optional and interface-scoped.

6.3 Limitations

The MKMS is intentionally bounded, and the results should be interpreted within the constraints of the prototype and evaluation design.

(1) **Low spatial resolution and discretization.** With only 25 actuators arranged as a 5×5 grid, representable geometries are inherently coarse. Thin strokes, diagonal features, and fine local detail are difficult to express, and many shapes remain ambiguous when reduced to a small height field.

(2) **Open-loop operation with minimal physical state verification.** Aside from bottom limit-switch homing, the system does not measure actuator position, load, or contact. As a result, the system cannot detect missed steps, obstruction, or mechanical drift during motion, nor can it correct deviations through feedback control.

(3) **Static actuation model.** The system executes single target configurations rather than continuous motion or real-time responsiveness. This limits the scope of interaction to discrete “set-and-settle” behaviors and excludes dynamic waveforms, reactive patterns, or time-based choreography.

(4) **AI behavior is constrained but not semantically validated.** The LLM is a general-purpose model running locally and is not trained for shape generation. While command validity and bounded outputs can be enforced through prompt constraints and parsing, this thesis does not include a formal perceptual evaluation (for example, user studies or inter-rater agreement) to verify whether observers consistently interpret produced shapes as matching prompt intent. Therefore, any discussion of “match” should be read as a researcher-observed plausibility assessment rather than a statistically validated measure of perception.

(5) **Latency and local deployment constraints.** Local inference enables offline operation but introduces practical constraints in response time and runtime stability. In the reported trials, response latency could be substantial for interactive use, and performance depended on hardware resources, model configuration, and prompt length.

6.4 Future Research Directions

The MKMS provides a foundation for future work along four main directions.

(1) **Higher-resolution arrays and scalable communication architectures.** Future systems should explore higher actuator counts and improved coordination mechanisms, such as bus-based communication or distributed nodes designed for synchronization, power distribution, and serviceability at scale.

(2) **Sensor integration and closed-loop control.** Adding sensing (position feedback, force/contact sensing, or vision-based reconstruction) would enable error detection, state verification, and adaptive behavior. This would also allow evaluation based on measured shape accuracy rather than observation alone.

(3) **Expanded actuation and surface strategies.** Alternative actuation methods and hybrid surfaces (for example, membranes or compliant layers that smooth discrete pins) could improve geometric continuity and reduce discretization artifacts. This would broaden the representational range beyond pixel-like relief patterns.

(4) **Domain-specific AI and formal evaluation.** A domain-specific model or prompting strategy informed by system constraints could reduce invalid interpretations and improve consistency. Equally important, future research should include formal evaluation of semantic correspondence through user studies, recognition tasks, or structured rating protocols, separating operational success from perceptual validity.

6.5 Closing Statement

This thesis demonstrated that an embodied morphing surface can be meaningfully integrated with a constrained LLM interface when the system is designed around clear separation of responsibilities: deterministic execution at the firmware layer, validated command handling at the host layer, and optional explanation confined to the interface. The MKMS does not claim industrial capability, dynamic motion, or perceptual correctness of generated shapes. Instead, it contributes an implemented proof-of-concept and a reproducible interaction model showing how language, computation, and physical form can be coupled within a single workflow.

Bibliography

- DeVOR, Richard, Robert Graves, and John J. Mills. “Agile Manufacturing Research: Accomplishments and Opportunities.” *IIE Transactions* 29, no. 10 (1997): 813–23. <https://doi.org/10.1080/07408179708966404>.
- Everitt, Aluna. *Digital Fabrication Approaches for the Design and Development of Shape-Changing Displays*. n.d.
- Follmer, Sean, Daniel Leithinger, Alex Olwal, Akimitsu Hogge, and Hiroshi Ishii. “inFORM: Dynamic Physical Affordances and Constraints through Shape and Object Actuation.” *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, ACM, October 8, 2013, 417–26. <https://doi.org/10.1145/2501988.2502032>.
- Goldstein, Seth C, Jason Campbell, and Todd C Mowry. *Programmable Matter*. n.d.
- Han, Changyo, Ryo Takahashi, Yuchi Yahagi, and Takeshi Naemura. “PneuModule: Using Inflatable Pin Arrays for Reconfigurable Physical Controls on Pressure-Sensitive Touch Surfaces.” *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ACM, April 21, 2020, 1–14. <https://doi.org/10.1145/3313831.3376838>.
- Kinetic Sculpture — The Shapes of Things to Come*. n.d. Accessed December 20, 2025. <https://artcom.de/en/?project=kinetic-sculpture>.
- Koc, Bahattin, and Sridhar Thangaswamy. “Design and Analysis of a Reconfigurable Discrete Pin Tooling System for Molding of Three-Dimensional Free-Form Objects.” *Robotics and Computer-Integrated Manufacturing* 27, no. 2 (2011): 335–48. <https://doi.org/10.1016/j.rcim.2010.07.017>.
- Leithinger, Daniel, Sean Follmer, Alex Olwal, and Hiroshi Ishii. “Shape Displays: Spatial Interaction with Dynamic Physical Form.” *IEEE Computer Graphics and Applications* 35, no. 5 (2015): 5–11. <https://doi.org/10.1109/MCG.2015.111>.

- MacLennan, B.J. “Universally Programmable Intelligent Matter Summary.” *Proceedings of the 2nd IEEE Conference on Nanotechnology*, IEEE, 2002, 405–8. <https://doi.org/10.1109/NANO.2002.1032276>.
- Munro, Chris, and Daniel Walczyk. “Reconfigurable Pin-Type Tooling: A Survey of Prior Art and Reduction to Practice.” *Journal of Manufacturing Science and Engineering* 129, no. 3 (2007): 551–65. <https://doi.org/10.1115/1.2714577>.
- Nakagaki, Ken, Yingda (Roger) Liu, Chloe Nelson-Arzuaga, and Hiroshi Ishii. “TRANS-DOCK: Expanding the Interactivity of Pin-Based Shape Displays by Docking Mechanical Transducers.” *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, ACM, February 9, 2020, 131–42. <https://doi.org/10.1145/3374920.3374933>.
- Olson, Eric G., and Deepak Sharma. “Beating the Commoditization Trend: A Framework from the Electronics Industry.” *Journal of Business Strategy* 29, no. 4 (2008): 22–28. <https://doi.org/10.1108/02756660810886971>.
- Owodunni, Oladele O., Javier Diaz-Rozo, and Srichand Hinduja. “Development and Evaluation of a Low-Cost Computer Controlled Reconfigurable Rapid Tool.” *Computer-Aided Design and Applications* 1, nos. 1–4 (2004): 101–8. <https://doi.org/10.1080/16864360.2004.10738248>.
- Pelin, George, Maria Sonmez, and Cristina-Elisabeta Pelin. “The Use of Additive Manufacturing Techniques in the Development of Polymeric Molds: A Review.” *Polymers* 16, no. 8 (2024): 1055. <https://doi.org/10.3390/polym16081055>.
- Sá, Asla Medeiros E, Karina Rodriguez Echavarria, Nico Pietroni, and Paolo Cignoni. “State of The Art on Functional Fabrication.” In *Eurographics Workshop on Graphics for Digital Fabrication*. Preprint, The Eurographics Association, 2016. 9 pages. <https://doi.org/10.2312/GDF.20161073>.

“TRANSFORM.” Accessed December 29, 2025.

<https://tangible.media.mit.edu/project/transform>.

Walczyk, Daniel F., and Yong-Tai Im. “A Hydraulically-Actuated Reconfigurable Tool for Flexible Fabrication: Implementation and Control.” *Journal of Manufacturing Science and Engineering* 122, no. 3 (2000): 562–68. <https://doi.org/10.1115/1.1286258>.

Walczyk, Daniel F., and Randy S. Longtin. “Fixturing of Compliant Parts Using a Matrix of Reconfigurable Pins.” *Journal of Manufacturing Science and Engineering* 122, no. 4 (2000): 766–72. <https://doi.org/10.1115/1.1314599>.

Wang, Zhijian. *Rapid Manufacturing of Vacuum Forming Components Utilising Reconfigurable Screw-Pin Tooling*. n.d.

Xu *, X. W., H. Wang, J. Mao, et al. “STEP-Compliant NC Research: The Search for Intelligent CAD/CAPP/CAM/CNC Integration.” *International Journal of Production Research* 43, no. 17 (2005): 3703–43. <https://doi.org/10.1080/00207540500137530>.

Yin, Yue H., Jin Y. Xie, Li Da Xu, and Hao Chen. “Imaginal Thinking-Based Human-Machine Design Methodology for the Configuration of Reconfigurable Machine Tools.” *IEEE Transactions on Industrial Informatics* 8, no. 3 (2012): 659–68. <https://doi.org/10.1109/TII.2012.2188900>.

□ □ □ □

based kinetic morphing system) (motor-
(AI)
Mega , , - . 25 Arduino
. Python , , AI
(LLM) . 25
SET ,
LLM ,
, AI
(programmable matter), -AI

Appendix (Listing to R-1 to R-7)

R-1. Centered Hill

R-2. Crater

R-3. Left-to-right ramp

R-4. Front-to-back ramp

R-5 Stepped pyramid

R-6 Checkerboard pattern

R-7 Symbolic letterform “H”