NORTHWESTERN UNIVERSITY

Design and Deployment of Multistable Robotic Metamaterial

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Field of Mechanical Engineering

By

Maxwell Bryant Patwardhan

EVANSTON, ILLINOIS

December 2024

ABSTRACT

Design and Deployment of Multistable Robotic Metamaterial

Maxwell Bryant Patwardhan

To deploy robots in the real world, they must be able to adapt to changing environments that require different capabilities. Traditional rigid robots are typically designed for a specific task, making them very effective for a small range of applications. Robotic systems with coupled, active units, or robotic metamaterials have the potential to circumvent these issues by trading precision for adaptability. In this work, we present a novel robotic metamaterial that leverages the multistable states of compression springs in tandem with bistable mechanisms in order to exhibit exploitable behavior. We demonstrate this system's ability as a flexible robotic manipulator by grasping a variety of objects using a facile open-loop controller.

Acknowledgements

During my brief tenure at Northwestern, I have had the privilege of working alongside the brilliant individuals of the Murphey Lab and the Center for Robotics and Biosystems. Never before have I been surrounded by such a wealth of expertise and a constant willingness to offer support. I sincerely appreciate everyone who contributes to fostering such a rich culture committed to learning and engagement.

Todd, I am deeply grateful for welcoming me into your lab. Your patience, guidance, and generosity have enabled me to explore a new dimension of robotics that was previously foreign to me. You are an inspiring educator and researcher, and I feel incredibly fortunate to have conducted my thesis under your supervision. I greatly admire your resolute dedication to this craft, and I'm thankful for the opportunity to be a part of it.

Annalisa, working with you throughout this project has been a pleasure. Week after week, you demonstrated an immense capacity to help me overcome any hurdle. Your support and direction proved invaluable throughout the entire project—for this, I can't thank you enough. (Also, thank you for tolerating the slow takeover of your desk with nonfunctional prototypes!)

Ilya, you are truly the gold standard of educators. The wealth of information you can dispense at a moment's notice in a clear manner is absolutely wonderful. You constantly demonstrate passion not only for the content you teach but also for the act of teaching itself. Thank you for taking so much time this past year to debug my endlessly broken electronics.

To my family and friends, both new and old, this undertaking would not have been possible without the great support and care you've all shown me. I hope to one day return the favor.

'*Today I walked down the street I used to wander*
Yeah, shook my head and made myself a bet
There were all these things that I don't think I remember
Hey, how lucky can one man get?'
– John Prine, *How Lucky* (1979)

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1. Motivation

'*One needs only to study a certain positioning of the hand in relation to the keys to obtain with ease the most beautiful sounds, to know how to play long notes and short notes and to achieve certain unlimited dexterity.*'
-Frédéric Chopin

Well before the era of benchmarking and comparing robotic systems to their biological counterparts, Chopin understood the innate capability of the human hand. Such abilities are challenging to downplay when compared to robotic simulacrums. The human hand can grasp and move immense loads, as well as deftly manipulate small and fragile objects. It is difficult to find a comparably robust system capable of such strength, precision, and dexterity—if even possible at all.

Nevertheless, the field of classical robotics has historically worked to create unique robots that serve unique tasks. Rather than compete in generality, humans have designed specific machines that are highly capable of singular tasks. The particular nature of these machines' inherent purpose leaves them with a weakness: they quickly become unreliable and potentially dangerous in highly dynamic environments [**1**]. A common notion of a robot is one capable of a task chosen a-priori, such as navigating a crowd [**2**], manipulating components in an assembly line [**3**], or autonomously driving

on city streets [**4**]. However, slight environmental changes jeopardize the robot's performance. In recent years, robots that act in task-agnostic and environment-agnostic manners have been proposed, producing machines that could one day perform multiple unrelated primitives with relatively low cost, training, and investment in design engineering. Robots that act as metamaterials, a concept which originates from the field of material science [**5**] [**6**], are well suited for said environment-agnostic operation. Metamaterials are artificial structures that leverage an anisotropic material property to exhibit unique behavior in various applications, from thermodynamics to acoustics [**7**], [**8**].

In this thesis, I outline a robotic metamaterial system built of modular nodes, allowing for high scalability and configurability with little alteration in manufacturing technique. As will be discussed in Chapter 2, many robotic metamaterials require fine-tuned manufacturing and assembly, limiting scalability and experimentation with multiple configurations. This system has the potential to function for a variety of tasks, including manipulation, locomotion, and mechanical sensing, among other areas. Classically, manipulators are designed to handle families of objects that share a common characteristic. Some manipulators can easily pinch and lift large, rigid objects, others work with fragile and soft items, and some handle high-loading conditions [**9**]. We construe our robotic system as a manipulator capable of handling a breadth of objects. However, it simultaneously serves as a platform for future research in heuristic control, mechanical intelligence, and locomotion via collective behavior.

1.2. General Approach

Our metamaterial leverages the multistable states of compression springs with an additional bistable compliant mechanism to grasp objects in its environment. The robotic system is a sheet of individual housings (nodes) connected by compression springs, as shown in Figure 1.1. Selected nodes contain batteries, a servomotor, a

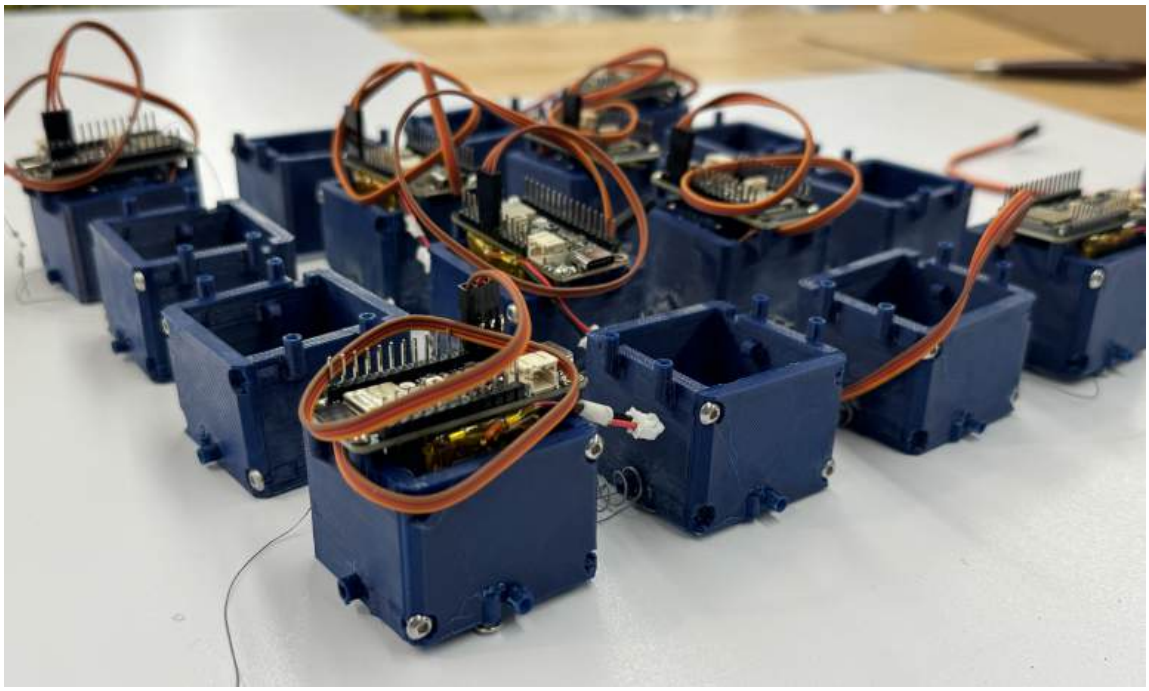


Figure 1.1. Robotic metamaterial sheet

bistable compliant spring, and a microcontroller. These nodes are referred to as "active nodes," and nodes without said electronics will be called "passive nodes." Select compression springs between nodes have a small cable running through their center, which can be tensioned by the servomotor in active nodes. Tensioning the compression springs causes them to enter a bifurcated state, pulling the nodes together. When performed in unison, this causes the metamaterial to form various shapes, making it useful for various tasks such as grasping, locomotion, and mechanical computing.

1.3. Summary

The contributions of this thesis are the design, manufacturing process, and software interface for a novel robotic metamaterial. Chapter 2 provides a background of current related work on robotic metamaterial systems. Chapter 3 discusses the platform's design, assembly, and operation. Chapter 4 demonstrates the functionality of a manipulation primitive. Lastly, Chapter 5 discusses future uses and considerations for the platform and what conclusions can currently be drawn from this research project. Further details regarding the design and in-depth instructions for assembling and operating the system can be found in the GitHub[1] repository.

[1]github.com/MaxPatwardhan/multistableRoboticMetamaterial

CHAPTER 2

Related Work

Modern advancements in material science, mechanical engineering, and robotics have given way to a burgeoning field: robotic metamaterials. These systems exhibit unique behaviors by exploiting properties not found in naturally occurring materials, allowing for novel capabilities in a diverse set of environments. The creation of new materials with properties such as negative stiffness [**10**], auxetic response [**11**], and programmable deformation [**12**], makes them highly suitable for a wide range of applications in robotics. Robotic metamaterials have seen significant advancement in recent years due to improvements in access to aforementioned novel materials, advanced manufacturing techniques, and current simulation/control strategies. The development of robotic metamaterials has brought about a new class of versatile, robust robotic systems that are well suited to handle dynamic environments compared to their prototypical counterparts.

A fascinating aspect of robotic metamaterials is their potential for mechanical intelligence [**13**]. Mechanical intelligence involves the integration of sensing, computing, and actuating functions within the material structure itself, enabling adaptive and responsive behaviors without the need for external control systems. By treating information processing as a material property, one can design systems that are inherently capable of complex, autonomous operations. This concept gives way to

robots interacting with and adapting to their environments in real time, enhancing their robustness and efficiency.

Here, we further explore examples of existing systems' design, strengths, and weaknesses.

2.1. Metamaterial Advantages in Robot Tasks

Robotic metamaterials exhibit behaviors not found in classical robots, enabling them to perform tasks that traditional robots would be incapable of. Traditional robots often have rigid components that limit their ability to safely and effectively interact with various environments. Robotic metamaterials offer a promising solution to this shortcoming due to their ability to undergo significant transformations in shape while maintaining structural integrity.

Some robotic metamaterials leverage the advantages of multistability to create easily controllable and energy-efficient systems [**14**], [**15**]. H. Morgan et al. (2021) present a pneumatic soft gripper that utilizes shape reconfiguration and actuation without the need for complex closed-loop control systems. The gripper uses a hierarchy of multistable structures to encode and transition between multiple stable states using an open-loop controller. The researchers effectively demonstrate the control of a complex, nonlinear system by collapsing a continuum of states into a handful of stable states. This paper focuses on conceptual design and simulation but displays limited physical demonstration in diverse environments.

In [**16**], Hu et al. (2020) propose a method of creating programmable, origami-inspired robotic structures. The programming of these structures changes the mechanical properties of the origami structure, which changes the robot's behavior when it is actuated. Hu et al. provide a solid basis for transforming origami models into programmable robotic systems and validated the proposal with practical applications such as a peristaltic crawling robot. Similar [**14**], this paper is mainly theoretical and lacks solutions to questions that arise regarding manufacturing and scalability.

Grossi et al. (2021) propose a bio-inspired robot, the Metarpillar, that uses metamaterial units to promote mechanical instabilities that drive motion in the form of buckling [**15**]. The Metarpillar is constructed of elastomer chambers whose stiffness is anisotropic. When deflated, the chambers buckle under negative pressure, enabling controlled deformations that mimic the movement of a caterpillar. This actuation method is efficient, safe, and environment-agnostic, demonstrating the innate advantages of metamaterial use in robot tasks.

2.2. Mechanical Intelligence

Mechanical intelligence is exemplified in systems where the material structure's sensing and computational capabilities are innate. Mechanical computers do not need external (often digital) control systems to exhibit valuable behaviors. For this reason, metamaterials are a well-posed medium for mechanical computers. Yasuda et al. (2021) explore the potential of mechanical computing systems that use flexible mechanical metamaterials. The authors demonstrate how mechanical bits, represented by bistable beams and origami structures, can be networked to perform complex computations. These systems can implement universal logic gates like AND, OR, and

NOT, which are essential for computational tasks. Integrating mechanical computing with robotic metamaterials enables the creation of robots that can process information and make decisions autonomously in the absence of a digital processor, ultimately enhancing their adaptability and functionality in dynamic environments [**13**].

Another notable application of mechanical intelligence is machine learning. As mechanical systems become more capable of computing complex information, this opens up the possibility of creating robots that can learn about their environment in real time. Bhagat et al. (2019) review various deep reinforcement learning algorithms applied to soft robotics, highlighting the potential for developing self-sufficient agents capable of learning from their environments. They emphasize the ability of soft robots to adapt and respond autonomously, using embodied intelligence to improve their interaction with unpredictable environments. This integration of machine learning with metamaterials comes with challenges, such as reliance on characterizing and classifying metamaterial robots, which, by design, operate in countless degrees of freedom. Bhagat et al. review promising results of using imitation learning algorithms to produce deep reinforcement learning models on soft robotic systems [**17**].

2.3. Key Takeaways

The aforementioned work captures only a small snapshot of this ever-broadening field. Yet even with the limited examples provided above, the advantages of robotic metamaterials are clear. Robotic metamaterials are inherently versatile, adaptable, and low-cost due to the design ideals they are predicated upon. However, these ideals also bring shortcomings that must be solved to increase their capabilities. These challenges include material limitations, poor control precision due to many systems'

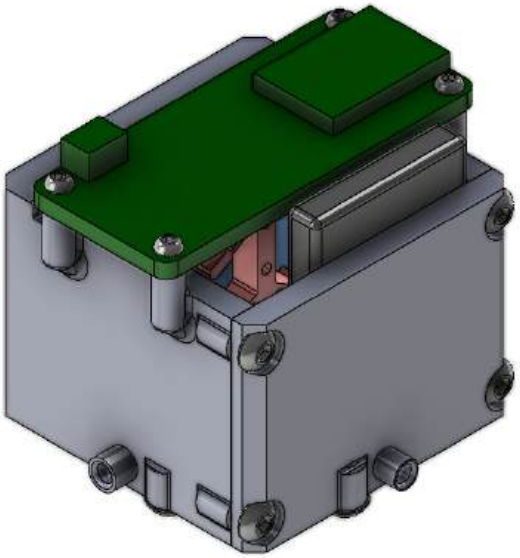
innate nonlinear nature, and scalability issues due to manufacturing complexity. Despite this, these robots provide a gateway to creating autonomous systems capable of real-time learning and adaptation in a robust manner. As manufacturing methods, material science, machine learning, and control theory all advance, these robots will continue to gain capabilities in a plethora of applications.

CHAPTER 3

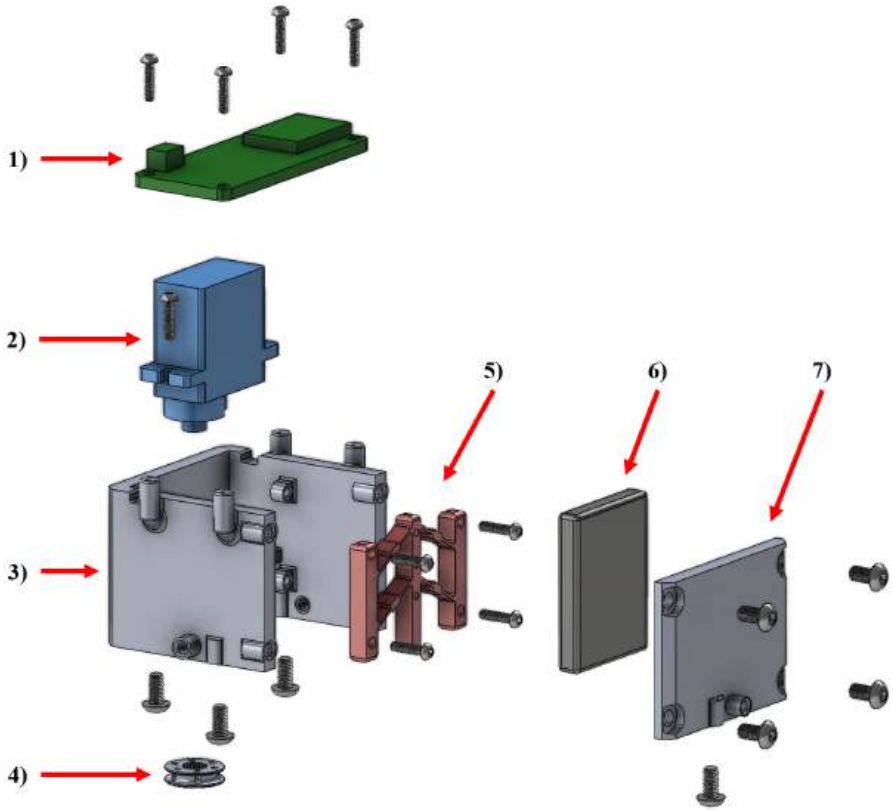
Metamaterial Design

The overall metamaterial system is highly configurable. Our initial design is a four-by-four rectangular grid, though this is far from the sole possible configuration. The specific arrangement of driven nodes can be easily modified, thus altering the metamaterial's behavior. The nodes are modular, and the overall system can be easily biased towards certain behaviors depending on the placement of driven nodes and the routing of the cables; an example of this can be found in 3.4.

Figure 3.1a shows an assembled node, and adjacent is Figure 3.1b, which shows an exploded view with labeled components.



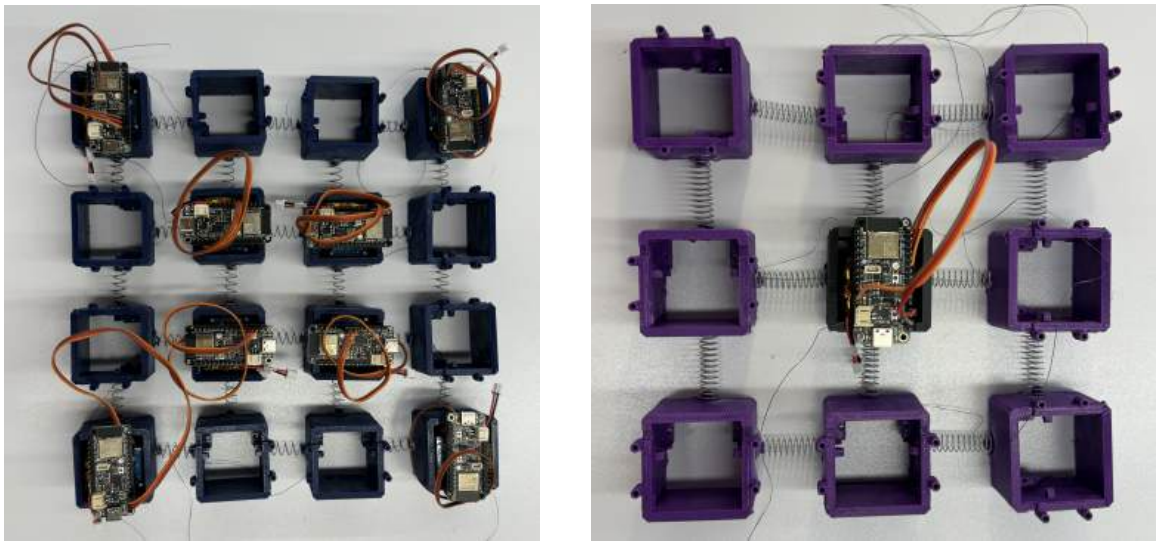
(a)



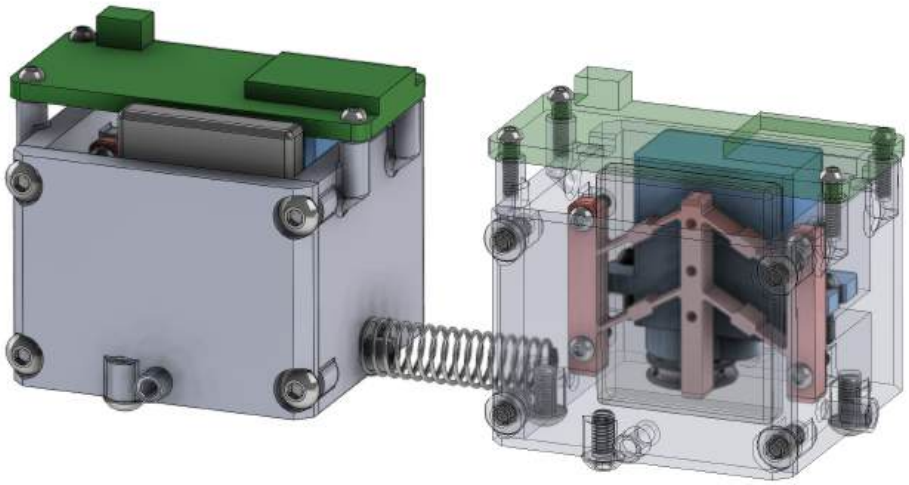
(b)

Figure 3.1. (a) CAD of an assembled active node. (b) Exploded view of an active node with all components. **1)** Adafruit Feather microcontroller. **2)** FeeTech Servo Motor. **3)** The main body of the node housing serves as the mounting point for all components. **4)** Cable pulley. **5)** Compliant bistable mechanism. **6)** Lithium polymer power supply battery. **7)** Removable lid for assembly and repair.

Figure 3.2 shows two possible configurations of robot layout, as detailed CAD perspectives of connected nodes.



(a) (b)



(c)

Figure 3.2. (a) Four-by-four configuration with eight active nodes and eight passive nodes. (b) Three-by-three configuration with one active node and eight passive nodes. (c) CAD representation of connected nodes with one transparent housing.

3.1. Mechanical Design

The metamaterial system's mechanical design is designed to be modular, compact, and simple to assemble or disassemble while maintaining the desired behaviors in prototyping. The nodes can be easily 3D printed and assembled with glue and hex keys. Necessary assembly hardware is listed in the bill of materials 3.4. The subsections below more thoroughly detail the key mechanical components that comprise the robot.

3.1.1. Modular Housing

The modular housing has a handful of functions in the system:

(1) The rigid component of metamaterial system.

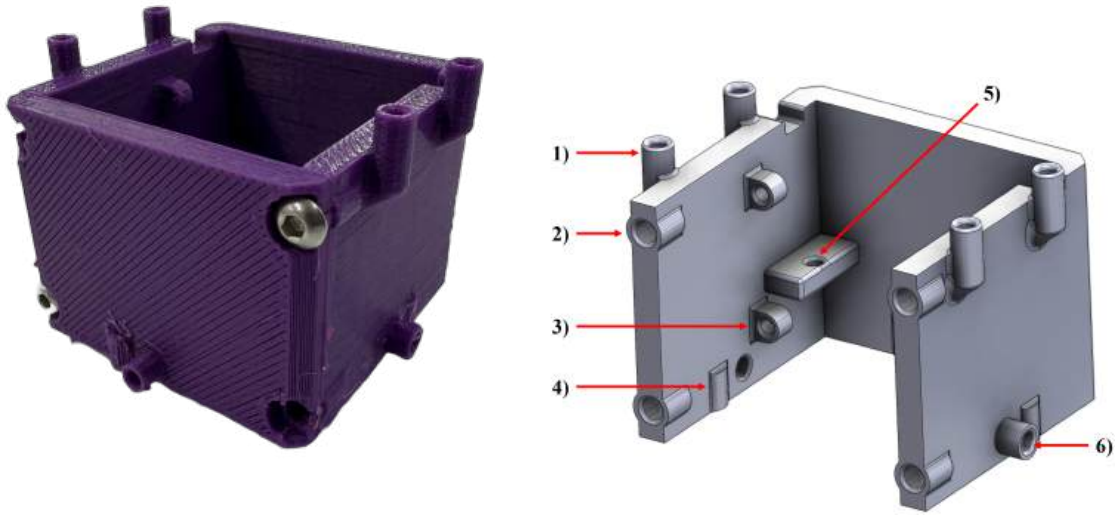
(2) Acts as a compact mount for electronic and mechanical components.

(3) Creates a bias when inducing multistability in bifurcated springs.

(4) Acts as an anchor for incoming cables from other active nodes.

A fully assembled node fits within a 5cm cube. Each active node holds a battery, a compliant bistable spring, a servo motor, an Adafruit Feather ESP32 V2 3.9, and the pulley system 3.4 used to tension the cables between compression springs. The housing can easily be 3D printed, but supports are required to allow the overhangs to print correctly. All holes are sized to self-tap with either M2 or M3 metric machine screws. The modular nature of the housing design allows for the metamaterial sheet to be assembled and driven in various configurations, as shown in allowing one to bias the metamaterial towards certain behaviors. The node housing has a removable wall,

allowing assembly and repair. Figure 3.3 shows a physical housing beside its labeled CAD counterpart. Note that the housing is symmetric, and the obscured interior section of 3.3b is identical to the visible portion.



(a) (b)

Figure 3.3. (a) 3D Printed passive node. (b) CAD Housing with labeled mounting points. **1)** Mounts for Adafruit Feather microcontroller. **2)** Mounting hold for removable wall. **3)** Mounting hole for compliant bistable mechanism. **4)** Anchor point for incoming cables. **5)** Mounting point for Servo. **6)** Cable port for incoming and outgoing wire. The cable port also fixes the location of the compression spring.

3.1.2. Tensioning System

The compression of the bistable compliant mechanism, as well as the compression of the springs, is performed by a small hobbyist servo in tandem with a custom-designed pulley. Fishing wire is used due to its low bend radius, high yield strength, and ease of assembly. The fishing wire is glued to the pulley for the compression springs and threaded between the housings. The fishing line is anchored at the terminal node

with an M3 screw that is threaded directly into the housing. Figure 3.4 depicts how cables are routed and anchored between active and passive nodes. Depending on both

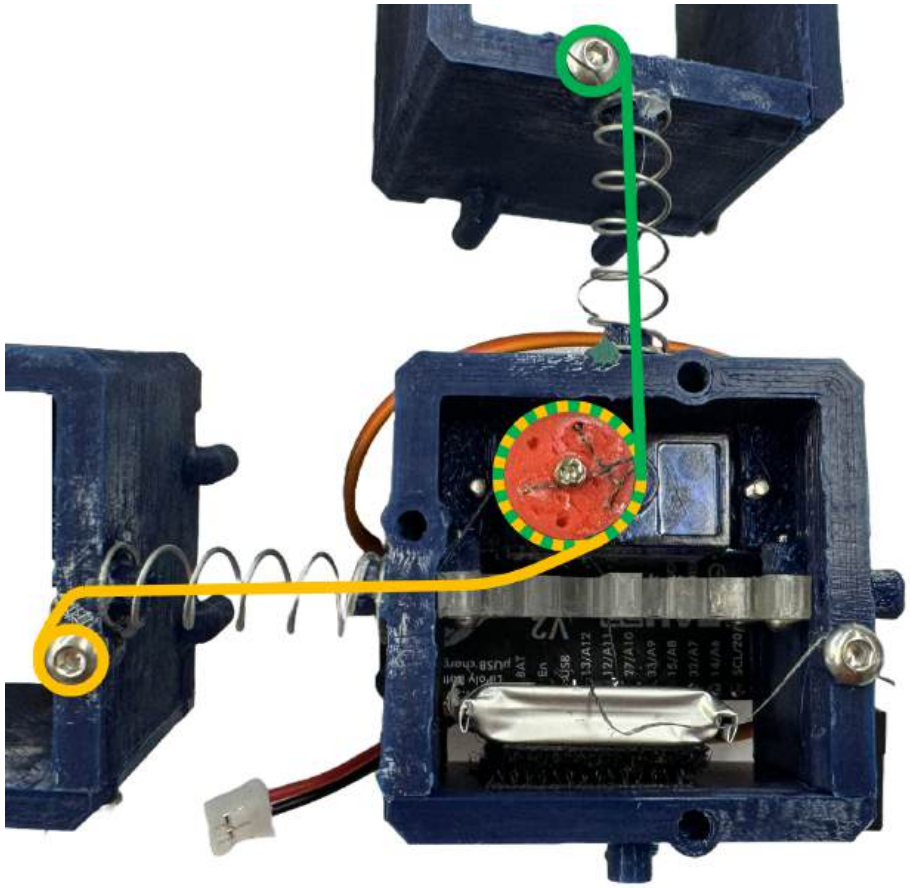


Figure 3.4. Example of tensioning between an active node and two passive nodes. Each color represents a different cable, and the dashed portion is where the two cables overlap. Each cable terminates at an M3 screw in the adjacent passive node.

the overall construction of the sheet and the configuration of the active nodes, there are a variety of ways in which the cables between active and passive nodes could be tensioned. Altering the direction of the cables can drastically change the robot's behavior and can be used to implement other primitives. One could reasonably think to "bias" the robot by introducing an asymmetry into which nodes are active and which nodes are passive or by only tensioning the compression springs in a certain

direction, ultimately introducing new behaviors and primitives. Figure 3.5 shows a small group of possible tensioning strategies.

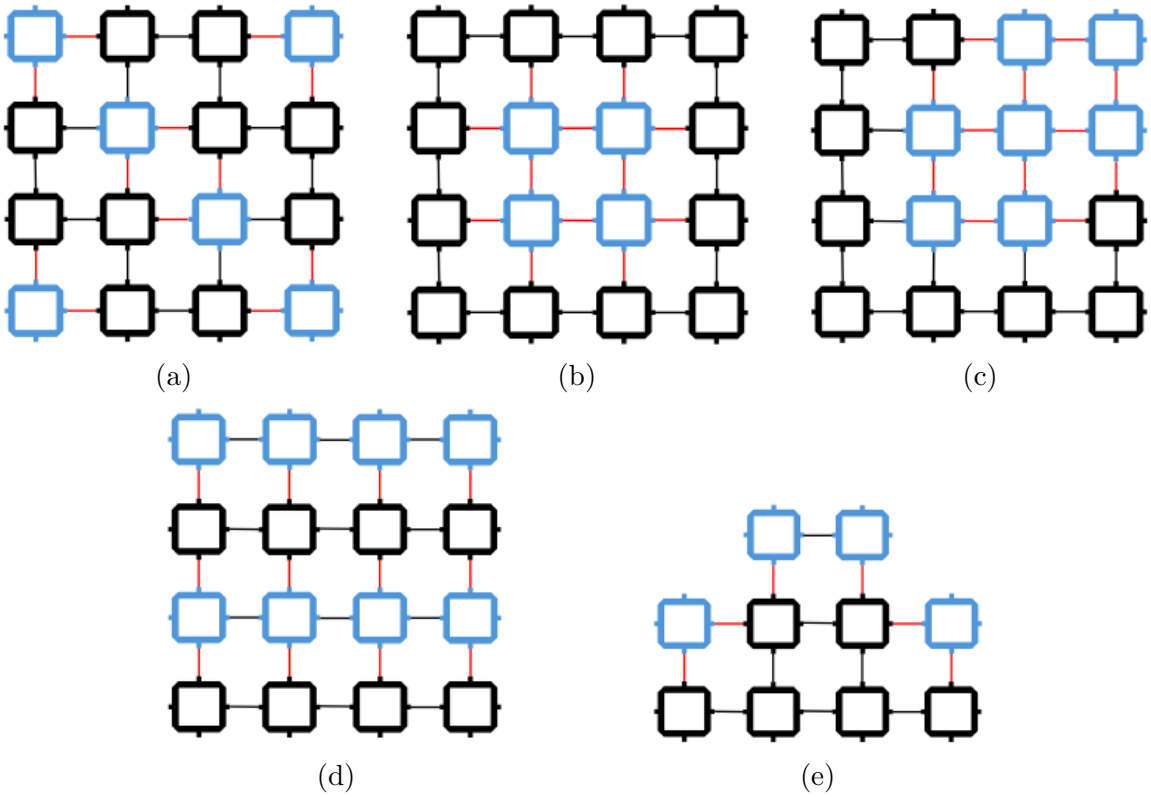
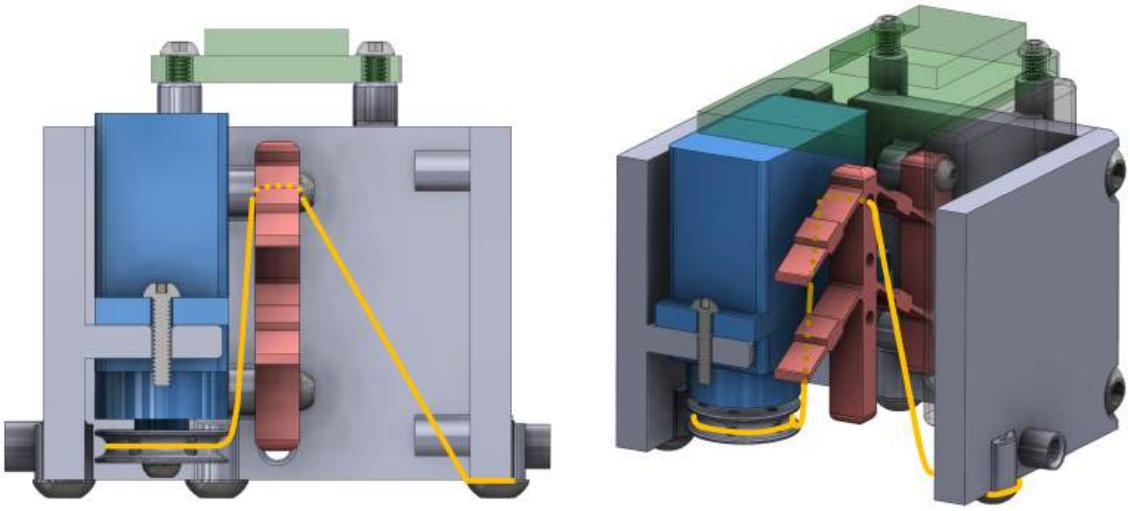


Figure 3.5. Blue nodes represent active nodes, black nodes are passive. Similarly, red connections are driven, and black connections are not. (a) Four-by-four configuration with eight active nodes and eight passive nodes. This configuration is used on the experimental platform. (b) An alternate symmetric configuration (c) A biased configuration. (d) A second biased configuration with potential for locomotion. (e) An example of a possible non-rectangular configuration.

3.1.3. Bistable Compliant Spring

Each active node has a compliant bistable spring mechanism driven by the servo. Like the system's compression springs, the compliant mechanism is routed by a cable

connected to the servo's pulley. The cable is anchored to the removable wall, as shown in Figure 3.6.



(a) (b)

Figure 3.6. (a) A profile section view of how the bistable mechanism is actuated. (b) An alternate view of how cables are routed through the bistable mechanism.

The design for this mechanism was referenced from Brian Jensen et al. [**18**]. Figure 3.7 shows the mechanism in its two states. This mechanism must be printed from a compliant material that does not fatigue easily. Polypropylene (PP) and Thermal Polyurethane (TPU) were both tested during prototyping, and TPU was ultimately selected, and TPU was chosen for its superior fatigue resistance and manufacturability, ensuring that the mechanism can endure repeated cycling without significant wear.

A bistable mechanism of this form is desirable as it only consumes energy when transitioning between states. In the context of manipulation, these mechanisms provide compliant yet forceful points of contact that provide friction and help keep the object stationary. This component of the metamaterial system is also viable as a

(a) (b)

Figure 3.7. (a) The bistable mechanism in its lowest energy state. (b) The bistable mechanism in higher energy state. It is driven to this state by the servo when an active node tensioning its cables.

dynamic actuator for tasks requiring quick and repeatable transitions between stable states. Additionally, the bistable mechanism's ability to maintain its state without continuous power makes it highly efficient for applications where energy conservation is critical.

The integration of bistable compliant springs into the metamaterial system enhances its versatility, enabling it to adapt to various operational requirements while maintaining high efficiency and reliability. This approach leverages the inherent properties of bistable mechanisms to create a robust and adaptive robotic system capable of performing complex tasks with minimal energy expenditure. Section

3.2. Electrical Design

The electrical systems design of the metamaterial is comprised of entirely off-the-shelf components, making it easy to assemble and repair. In an individual active node, the electronic system contains an Adafruit Feather ESP32 V2 (microcontroller), a 3.7 volt nominal Lithium Polymer (LiPo) battery, and a FeeTech FS90R continuous rotation servo motor. This system can receive remote commands from a base computer and send commands or information about its state if requested. Figure 3.8 shows a functional block diagram displaying a high-level view of the electronic system.

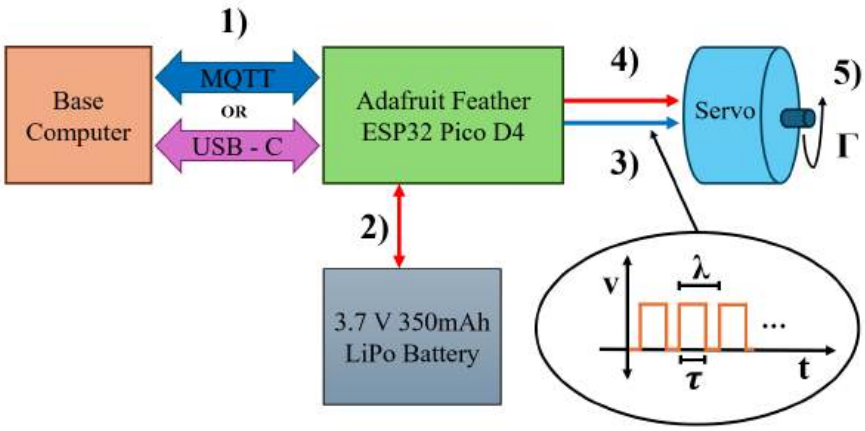


Figure 3.8. Electrical functional block diagram displaying power (red), data (blue), and dual (violet) connections. **1)** The base computer programs and powers the microcontroller over USB-C. This is also used to charge the LiPo battery. MQTT is used to communicate with the base computer over WiFi during typical robot operation. **2)** The LiPo battery serves as a power supply for the actuator and microcontroller during wireless operation. The microcontroller recharges the LiPo battery when connected to USB-C. **3)** The microcontroller sends a pulse width modulation (PWM) control signal to the servo. The pulse width value, τ, which is some fraction of the total period, λ, is used to set the speed and direction of rotation. **4)** The servo is powered by a 3.3-volt DC linear regulator onboard the microcontroller. **5)** Torque, Γ, commanded by the PWM control signal.

3.2.1. Feather ESP32 Microcontroller

The Adafruit Feather ESP32 V2 is an ideal microcontroller board for this robot due to its size, low cost, high configurability, and inclusion of an antenna. The Feather serves these main functions:

(1) Controls of the servo actuator.

(2) Communicates with the central controller via MQTT.

(3) Distributes and regulates power to onboard peripheral devices.

(4) Displays state using the NeoPixel LED.

The microcontroller is programmed using C through the Arduino IDE. This is possible as the board has a built-in programmer, eliminating the need for an external USB-to-UART module. The main processor on the board is an ESP32-WROOM Pico Mini D2, a small and powerful microcontroller with an onboard antenna, making it ideal for use in our WiFi-based communication protocol. The Feather board has a 3.3-volt DC linear power regulator, which is used to power the node's actuators. This regulator also has the capacity to power other low-current peripheral devices (such as a sensor) that the user would like. Notably, the Feather contains all the necessary circuitry to charge the LiPo battery and "hot-swap" the battery as the main power source when the USB-C cable is connected and supplying power. Figure 3.9 shows each of the aforementioned functions.

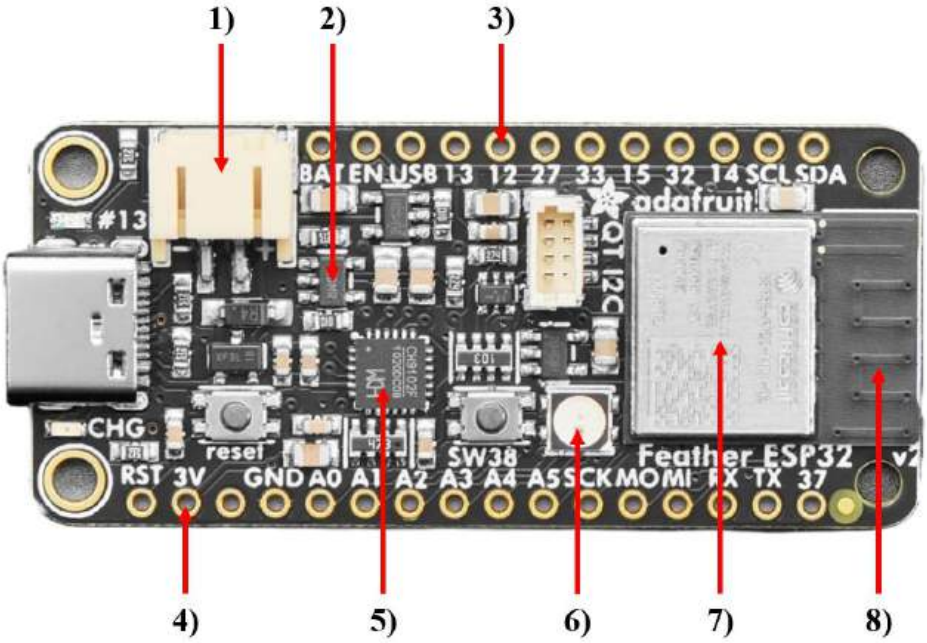


Figure 3.9. Adafruit Feather ESP32-WROOM V2 **1)** Two-pin LiPo battery connector. **2)** LiPo battery management circuitry. **3)** Example pin for generating PWM control signal. **4)** 3.3-volt DC power supply for the servo. **5)** USB to UART programming microcontroller. **6)** NeoPixel RBG Indicator used to display state. **7)** Espressif ESP32 Pico Mini D2. **8)** WiFi antenna.

The Feather board sits on top of the metamaterial, allowing the NeoPixel RGB LED to declare the state of a given driven node. Thus, the operator can see whether a given node is tensioning in a given direction, unwinding, or at rest.

3.2.2. Servo Actuator

The servo used in this system is the FeeTech FS90R continuous rotation servo, a common hobbyist servo that is lightweight, small, and powerful. As previously mentioned, the servo's speed and direction are determined by a PWM signal generated

by the ESP32. Depending on the PWM signal's duty cycle, D, the servo will rotate either counterclockwise or clockwise over a continuous distribution of speeds.

It is worth noting that the values we use for D do not match the recommended values on the FS90R datasheet. Nevertheless, a uniform distribution of input D was tested and measured on an oscilloscope to find values that give the fastest clockwise and counter-clockwise rotation speeds. Table 3.1 shows the spectrum of measured pulse width values, τ, the corresponding duty cycle values, D, and the measured rotational velocity, ω, with the selected values in bold.

Input (integer)	τ (μs)	D (%)	**Output, ω (RPM)**
0	0	0	0.0
10	40	4	-35.5
25	100	10	-84.5
35	**140**	**14**	**-95.0**
50	200	20	-84.4
75	296	29.6	-41.4
100	392	39.2	-9.8
125	492	49.2	-1.5
150	588	58.8	-1.5
175	688	68.8	-0.5
200	784	78.4	-0.5
225	880	88	0.0
250	**980**	**98**	**95.0**
255	1000	100	90.0
275	72	7.2	-62.5

Table 3.1. Input 8-bit integer versus measured output velocity, ω. Note that a negative value of ω implies a counterclockwise rotation. For typical operation, values of $\mathbf{D = 14\%}$ and $\mathbf{D{=}98\%}$ are used for clockwise and counter-clockwise rotation, respectively, as they produce the highest torque.

3.3. Software and Communication

The system's software is separated into the central controller, the MQTT broker, and the embedded programming on a given active node. The central controller and

microcontrollers use the Message Queuing Telemetry Transport (MQTT) communication protocol to send and receive information. The central controller and the MQTT broker run on the user's laptop. The controller runs in a Python environment, and the broker simultaneously runs in the Windows terminal. Overall, this architecture is highly configurable and scalable, enabling the robot to operate in a broader spectrum of environments without needing a tethered connection to the controller and power source.

3.3.1. Wireless Communication via MQTT

MQTT is a lightweight, publish-subscribe, broker-based network protocol ideal for communication between the base controller and separate microcontrollers. MQTT is highly configurable and reliant on a local WiFi network to transmit data. Our system uses Mosquitto, an open-source broker that runs on a local environment Windows. The broker's job is to gather and relay messages to any clients on the network. In contrast with other cloud-based alternatives, the Mosquitto broker is lightweight and can also be run on the ESP32 microcontroller boards themselves, allowing one to implement a control scheme and communication protocol that lives only on the embedded systems themselves. Our system utilizes a central Python controller to command each node and monitor the state of the robotic system. Figure 3.10 shows the physical architecture adopted for this project. A local router hosts a WiFi network to which the laptop and nodes connect. The central controller communicates with the Mosquitto broker internally and can then publish a message on an arbitrary topic. Each Adafruit Feather microcontroller connects to the local WiFi network and

establishes a connection to the MQTT broker using the broker's Network IP address and port 1883 (unless otherwise specified by the user).

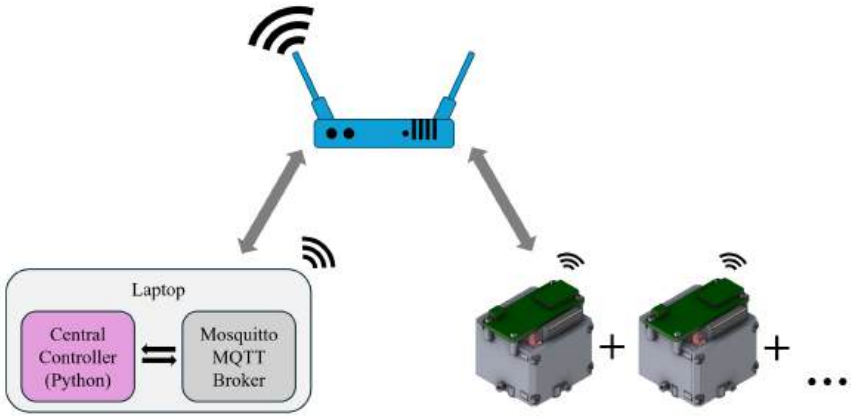


Figure 3.10. Physical layout of MQTT network. The router creates a local WiFi network, and the broker distributes messages between the clients in the system. An MQTT network can handle hundreds of devices at a time, which is more than sufficient for our purposes.

In an MQTT network, clients publish messages, also called payloads, on designated topics. These messages are sent to the broker, which allocates the message to all other clients subscribed to that topic. For instance, a node might publish data to a topic named **nodes/'address'/state**. In this scenario, it is essential to note that **'address'** would be specific to a particular node or set of nodes. However, a device could also publish to a different topic that every node subscribes to in the system, e.g., **nodes/all/state**. The Python client on the laptop and other nodes can subscribe to this topic to receive updates whenever new data is published. This functionality is particularly useful for real-time monitoring and control with high granularity. The topics address clients and their objects on the network, and the messages change the state of information relevant to the addressed object.

```
1 import paho.mqtt.client as mqtt #mqtt library
```

```
2
3 client = mqtt.Client(protocol=mqtt.MQTTv5)  #create client obj.
4 client.connect("192.168.1.7", 1883) #connect to broker IP
5
6 topic = "nodes/3/servo/set" #address node w/ topic
7 message = '{"drive": "clockwise"}'  #specify message
8
9 client.publish(topic, message)  #publish message to broker
10 client.disconnect() 3Disconnect from the broker
```

Listing 3.1. Example Python script that commands an individual node to drive the servo.

Currently, the robot is controlled by inputs from a human user. The central controller receives an address and a payload as input and publishes the message to the corresponding address. The following steps further detail the process by which a message is published by a Python controller and received by a WiFi client via the Mosquitto broker:

Step 1: **Python Controller Publishes a Message.** The Python controller, acting as an MQTT client, establishes a connection with the Mosquitto broker using the broker's IP address. Once connected, the controller publishes a message to a specific topic.

Step 2: **Broker Receives the PUBLISH Message.** The Mosquitto broker receives the published message. It parses the message to extract the topic and payload and identifies which clients are subscribed to the topic.

Step 3: **Broker Forwards the Message to Subscribed Clients.** The broker forwards the message to all clients subscribed to the topic. This involves sending a PUBLISH message from the broker to each subscribed client.

Step 4: **Node Client Receives the Message.** Each subscribed Node client receives the forwarded message. The client parses the message to extract the topic and payload and then processes the message accordingly.

The controller for the manipulation task is quite facile. The controller will intake one of three values from the user:

User Input:	Node Action
0	Halt all movement
1	Drive in clockwise direction
2	Drive in counterclockwise direction

Table 3.2. Example inputs for a facile open-loop control scheme, which switches between a relaxed state and a contracted state.

3.3.2. Embedded Programming

The embedded programming of this system is currently quite simple and highly configurable. As stated in Subsection 3.2.1, the ESP32 is programmed in C using the Arduino IDE. The Feather board contains a microcontroller that automatically handles resetting and programming the ESP32 over UART.

The Feather declares variables, pins, and necessary objects when initially powered on. The ESP then waits for a WiFi connection and a subsequent connection to the

broker. When the system is running, the Feather runs a main loop. Algorithm 1 details all steps taken during startup and the main loop that follows.

Algorithm 1 High-Level Servo and LED Control with WiFi and MQTT Communication

Initialization:
Set up constants for hardware pins and communication credentials
Initialize hardware and communication interfaces
function SETUP
    Connect to WiFi
    Connect to MQTT broker
end function
function ROTATESERVO(direction, duration)
    Rotate servo in specified direction for given duration
end function
function PUBLISHSTATE(state)
    if MQTT client connected **then**
        Publish state message
    else
        Reconnect and publish state message
    end if
end function
function UPDATENEOPIXEL(color)
    Update NeoPixel colors based on color input
end function
while true **do**
    Check Serial Input:
    if serial input available **then**
        Read and process serial input
        if input matches "0" **then**
            Stop servo, call PUBLISHSTATE(stopped), and UPDATENEOPIXEL(Red)
        else if input matches "1" **then**
            Rotate servo clockwise, call PUBLISHSTATE(clockwise), and UPDATENEOPIXEL(Green)
        else if input matches "2" **then**
            Rotate servo counterclockwise, call PUBLISHSTATE(counterclockwise), and UPDATENEOPIXEL(Blue)
        end if
    end if
    Maintain MQTT Connection:
    if MQTT client not connected **then**
        Reconnect to MQTT broker
    end if
    Call MQTT client loop to maintain connection
end while

3.4. Bill of Materials & Unit Cost

Part:	**Vendor:**	**Part No.:**	**Unit Cost ($):**	**Quantity:**	**Total Cost ($):**
Feather ESP32 V2	Adafruit	5400	19.95	1	19.95
350mAh 3.7V LiPo Battery	Adafruit	4237	5.95	1	5.95
FS90R Servo Motor	Adafruit	154	11.95	1	11.95
M3 × 6.00mm Screw	McMaster Carr	92095A179	0.0583	8	0.47
M2 × 6.00mm Screw	McMaster Carr	92095A454	0.2732	10	2.73
Compression Spring, 1” × 0.3 OD	McMaster Carr	9657K301	0.965	4	3.86
10 Lb. Fishing Line	Amazon	Find a cheap one.	0.02 ($/ft)	4 ft	0.08
Matte PLA+	Overture	Matte Black	0.019 ($/g)	16.83 g	0.32
Fast TPU	Overture	Clear	0.032 ($/g)	1.75 g	0.06
				Total:	**45.37**

Table 3.3. Bill of Materials (BOM) for Project

The overall cost of one active node is $45.37.

CHAPTER 4

Task Primitives

We demonstrate the capability of our robotic metamaterial system to perform various tasks such as manipulation, locomotion, and stable shape formation. The system's unique design, leveraging multistable states and compliant mechanisms, allows it to adapt its shape to grasp and manipulate objects of different geometries, as well as locomote in a given direction when mechanically biased.

The first set of experiments involved the robotic metamaterial gripping a cone and a rectangular prism. The active nodes were programmed to adjust the tension in the connecting cables, enabling the system to conform to the shapes of the objects. Figure 4.1 shows the robot successfully gripping a cone, while Figure 4.2 depicts the robot gripping a rectangular prism. The adaptability of the metamaterial structure is evident in these tasks, showcasing its potential for handling objects with varied geometries without the need for reconfiguration.

The experimental results indicate that the robotic metamaterial can effectively adapt to and manipulate objects of different shapes. This adaptability is crucial for applications where the robot must handle diverse items without needing specific end-effectors for each geometry.

Another key capability of the robotic metamaterial is its ability to morph into various shapes. This is particularly useful for tasks requiring surface conformity or

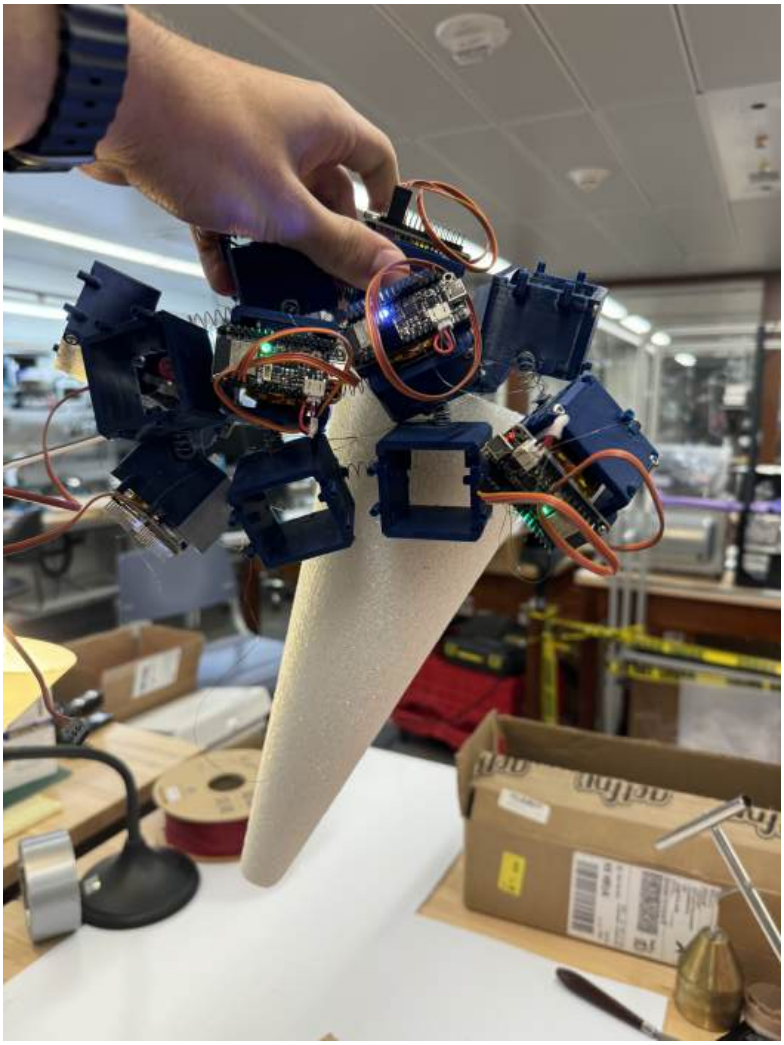


Figure 4.1. Robot system gripping a cone

specific geometric configurations. When actuated in two select trials, the system formed a dome shape and folded surface, demonstrating its potential application in programmable structures or adaptive surfaces [**19**]. Figure 4.3a illustrates the robot taking on a dome shape as well as a folded surface, highlighting the adaptability and reconfigurability of the system.

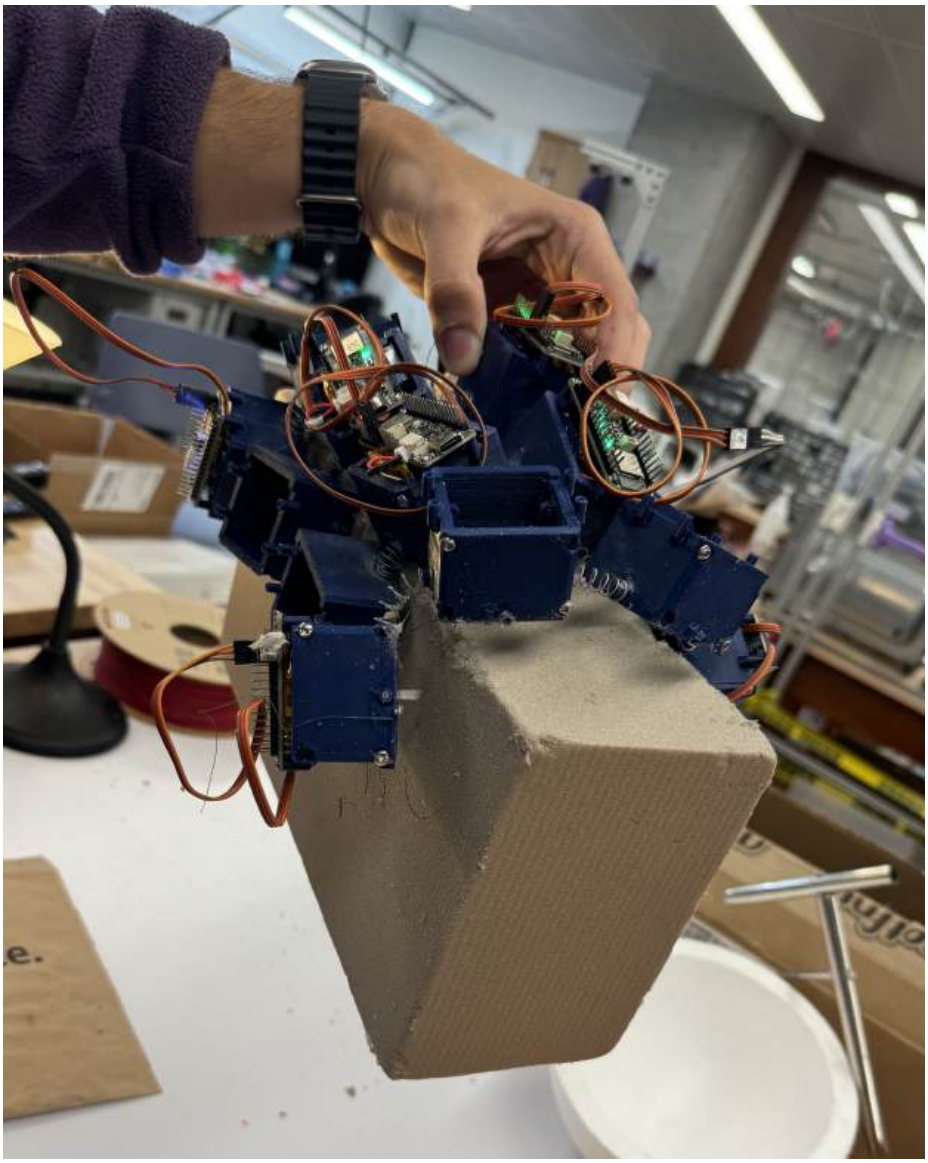
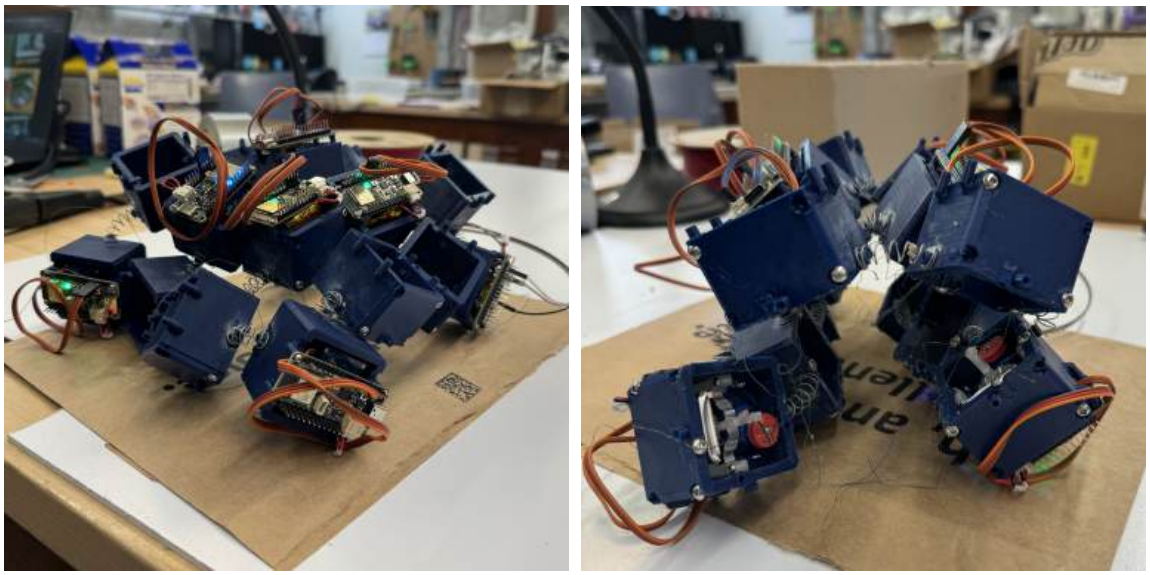


Figure 4.2. Robotic metamaterial gripping a rectangular prism. Note that the visible bistable mechanism helps grip the side of the prism.

Though limited in number of trials, these experiments underline the robotic metamaterial's adaptability and robustness in performing complex manipulation tasks and morphing into various shapes, making it suitable for a wide range of applications.



(a) (b)

Figure 4.3. (a) Robot metamaterial taking on a dome shape in the absence of an object to grasp. (b) Metamaterial mimics a folded surface.

CHAPTER 5

Conclusions and Other Potential Applications

This thesis has presented the design, development, and testing of a novel robotic metamaterial system capable of performing a manipulation task through the exploitation of compressed springs. The integration of multistable states and compliant mechanisms has enabled the system to handle objects of varying shapes and sizes and to morph into different geometric configurations as required by the task. The robot has also shown an ability to morph into two distinct shapes even in the absence of an object to grasp – further demonstrating the inherent advantages of robotic metamaterials.

A primary strength of this system is its high adaptability in contrast to its relatively facile control scheme. The modular design allows for easy scalability and reconfiguration, making it possible to tailor the system for specific applications without extensive redesign, manufacturing processes, or programming. The use of off-the-shelf components and 3D printing techniques further enhances its accessibility and ease of assembly. The system's ability to undergo significant shape transformations while maintaining structural integrity makes it particularly suitable for applications requiring versatile manipulation capabilities. This adaptability also extends to its potential use in dynamic environments, where traditional rigid robots may struggle to perform effectively. Despite its strengths, the system has certain limitations that need

to be addressed in future work. As mentioned in Chapter 2, this system lacks precision in favor of environmental adaptability. Additionally, while the current system demonstrates the feasibility of the approach, the current system struggles to grasp objects with higher mass. The system also suffers from inconsistency. Due to the relatively trivial control scheme, the simultaneous activation of every node does not always produce the same effect. The manipulator can fail to grasp objects or form a stable structure, and instead, active nodes will twist in-plane, not being significantly affected by the bistable mechanism. This is undoubtedly an interesting behavior but is not particularly useful in the context of manipulation. This system might be well suited for 2D and 3D locomotion. This would require more complex control schemes coupled with further research into biasing the placement of active nodes. Another potential avenue for this system is one of mechanical sensing. By receiving feedback from nodes, the system could potentially stand to learn the environment it is interacting with.

In conclusion, the robotic metamaterial system developed in this thesis provides evidence that robotic metamaterials can be useful in a variety of contexts. Its ability to perform manipulation tasks with high adaptability and trivial open-loop control makes it a promising candidate as a manipulator and leaves the possibility for other complex tasks.

References

[1] H. Kazerooni, B. J. Waibel, and S. Kim, "On the Stability of Robot Compliant Motion Control: Theory and Experiments," *Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 417–426, 09 1990.

[2] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 797–803, 2010.

[3] P. Chutima, "A comprehensive review of robotic assembly line balancing problem," *J Intell Manuf*, vol. 33, 2022.

[4] A. Faisal, M. Kamruzzaman, T. Yigitcanlar, and G. Currie, "Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy," *Journal of Transport and Land Use*, vol. 12, no. 1, pp. 45–72, 2019.

[5] S. E. Rodriguez, E. Calius, A. Khatibi, A. Orifici, and R. Das, "Mechanical metamaterial systems as transformation mechanisms," *Extreme Mechanics Letters*, vol. 61, p. 101985, 2023.

[6] J. Qi, Z. Chen, P. Jiang, W. Hu, Y. Wang, Z. Zhao, X. Cao, S. Zhang, R. Tao, Y. Li, *et al.*, "Recent progress in active mechanical metamaterials and construction principles," *Advanced Science*, vol. 9, no. 1, p. 2102662, 2022.

[7] J. E. H. J. au2, H. T. Schaef, B. P. McGrail, and Q. R. S. Miller, "Review of foundational concepts and emerging directions in metamaterial research: Design, phenomena, and applications," 2022.

[8] T. J. Cui, D. R. Smith, and R. Liu, *Metamaterials*. Springer, 2010.

[9] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.

[10] H. Li, Y. Li, and J. Li, "Negative stiffness devices for vibration isolation applications: A review," *Advances in Structural Engineering*, vol. 23, no. 8, pp. 1739–1755, 2020.

[11] M. Balan P, J. Mertens A, and M. V. A. R. Bahubalendruni, "Auxetic mechanical metamaterials and their futuristic developments: A state-of-art review," *Materials Today Communications*, vol. 34, p. 105285, 2023.

[12] B. Florijn, C. Coulais, and M. van Hecke, "Programmable mechanical metamaterials," *Physical Review Letters*, vol. 113, Oct. 2014.

[13] G. A. e. a. Yasuda H., Buskohl P.R., "Mechanical computing," *Nature*, vol. 598, no. 7879, pp. 39–50, 2021.

[14] J. C. H. Morgan, Osorio, H. Morgan, and A. F. Arrieta, "Programmable multistable soft grippers," *Journal of Soft Robotics*, vol. 8, no. 2, pp. 123–134, 2021.

[15] B. Grossi, H. Palza, J. Zagal, C. Falcón, and G. During, "Metarpillar: Soft robotic locomotion based on buckling-driven elastomeric metamaterials," *Materials Design*, vol. 212, p. 110285, 2021.

[16] F. Hu, W. Wang, J. Cheng, and Y. Bao, "Origami spring–inspired metamaterials and robots: An attempt at fully programmable robotics," *Science Progress*, vol. 103, no. 3, pp. 1–19, 2020.

[17] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, 2019.

[18] *Design Optimization of a Fully-Compliant Bistable Micro-Mechanism*, vol. Micro-Electro-Mechanical Systems (MEMS) of *ASME International Mechanical Engineering Congress and Exposition*, 11 2001.

[19] M. Pieber, R. Neurauter, and J. Gerstmayr, "An adaptive robot for building in-plane programmable structures," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, IEEE, 2018.