APPLIED PHYSICS

Pneumatic coding blocks enable programmability of electronics-free fluidic soft robots

Sergio Picella[1,2], Catharina M. van Riet[1,3], Johannes T. B. Overvelde[1,2]*

Decision-making based on environmental cues is a crucial feature of autonomous systems. Embodying this feature in soft robots poses nontrivial challenges on both hardware and software that can undermine the simplicity and autonomy of such devices. Existing pneumatic electronics-free soft robots have so far mostly been approached by using system fluidic circuit architectures analogous to digital electronics. Instead, here we design dedicated pneumatic coding blocks equivalent to *If*, *If...break*, and *For* software control statements, which are based on the analog nature of nonlinear mechanical components. We demonstrate that we can combine these coding blocks into programs to implement sequences and to control an electronics-free autonomous soft gripper that switches between behaviors based on interactions with the environment. As such, our strategy provides an alternative approach to designing complex behavior in soft robotics that is more reminiscent of how functionalities are also encoded in the body of living systems.

INTRODUCTION

Soft robotics branches off more traditional robotics and offers a hardware platform for diverse contexts where adaptivity and sensitivity are important, ranging from agriculture to medicine and human interaction (*1*–*4*). In particular, compliant hardware enhances robustness to unstructured and unknown tasks by embodying mechanical behavior that responds directly to external stimuli (*5*, *6*). Nevertheless, many soft robotic systems still rely on electronic sensors and controllers that, although potentially accurate, cheap, and precise, impose additional fabrication challenges for the integration into the soft hardware. Furthermore, integration of electronics also limits the potential broad application perspective of soft robotics in domains where electronics would fail, including high radiation environments (*7*, *8*), implantable medical devices (*9*), demining applications (*10*), or explosive environments in general (*11*).

As such, a recent trend in soft robotics points toward reproducing the response of electronic components using a pneumatic electronics-free approach (*12*, *13*). Several studies have focused on the development of components with logic-port functionalities (*14*, *15*), which often depend on mechanical nonlinearities and instabilities to generate complex behavior (*16*–*20*). On the basis of these components, a few robots have been built with programmed functionalities that arise when combining multiple components (*21*, *22*). However, although these components and circuits share similar features with their electronic counterparts, a more general approach for achieving high-level programmability to build pneumatic circuits is still missing. A strategy to effectively convert preferred behavior to a pneumatic circuit design would make electronics-free soft robots even more appealing as it would simplify and allow the embodiment of more complex programs in the soft robots that can interact with the environment directly.

To enable the design of electronics-free soft robots that can autonomously make decisions (i.e., switch between different behaviors or programs) based on interactions with the environment, we draw inspiration from mechanical programming. Machines and robots designed prior to the advent of software had to rely exclusively on hardware programming, thus basing their working principles on interactions of mechanical and pneumatic components with themselves and operators. The schematics of these robots and automated machines were often based on the intuition of so-called wired intelligence. According to this design principle, different behaviors from the machines can be obtained by directly connecting actuators and sensors, with no need for dedicated controllers (*23*). In our case study, we design our systems to follow similar design rules and constraints to avoid the use of external electronic controllers in pneumatic circuits.

To enable mechanical programming, we take inspiration from the modularity of coding statements used in software programs. We aim to design a general library of pneumatic circuit blocks that enable the programmability of pneumatic soft robots and allow them to run versatile pseudocodes that can be designed following our circuit design approach. We set the basis for a structured coding language by identifying the minimal requirements for our pneumatic equivalent coding language to comply with the hypotheses of the Böhm-Jacopini theorem (*24*). According to this programming language theorem, every computable program can be computed if the control system has the following capabilities: (i) executing commands in sequence, (ii) conditionally going to subprograms through selection statements, and (iii) repeating subprograms as long as a Boolean condition is met. Therefore, we focus on prototyping statement-equivalent circuits that can manifest these functionalities to write and run pneumatic electronics-free programs for soft robots.

To meet this purpose, we introduce pneumatic circuits for *If*, *If...break*, and *For* and demonstrate the sequential execution of instructions. We then introduce the designs for both continuous and Boolean variables that can be used in our fully pneumatic programming environment. Last, we demonstrate that our modular approach enables more complex programming, where we focus on a soft gripper that changes its behavior when detecting an object, as we schematically represent in Fig. 1. Through our demonstrator, we show that our modular coding approach allows the integration of actuation, sensing, and feedback within the same pneumatic platform.

[1]Autonomous Matter Department, AMOLF, Amsterdam 1098 XG, Netherlands. [2]Institute for Complex Molecular Systems and Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven 5600 MB, Netherlands. [3]Department of Industrial Design, Eindhoven University of Technology, Eindhoven 5600 MB, Netherlands.
*Corresponding author. Email: overvelde@amolf.nl

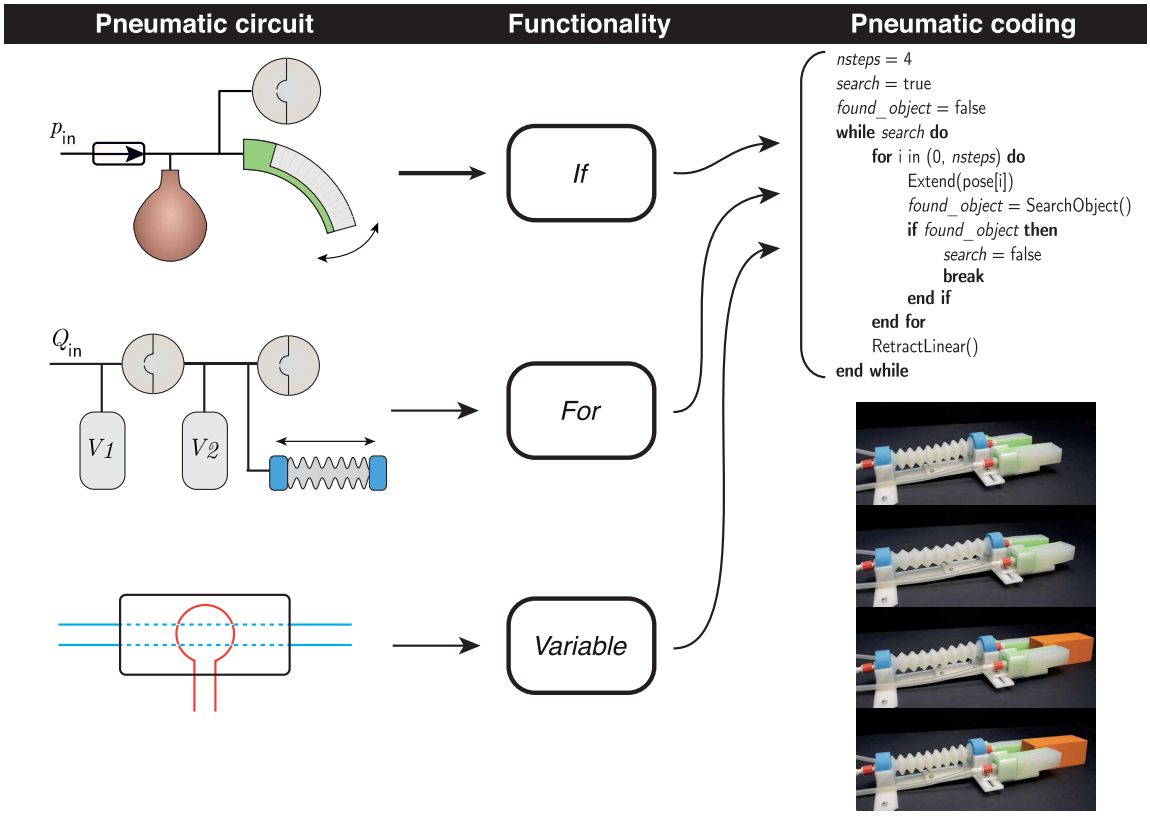

Fig. 1. Overview of our design approach to enable pneumatic coding. Our approach is based on a library of dedicated pneumatic circuits that represent specific coding statements. These statements can be combined to develop pneumatic programs that can respond to the environment.

RESULTS

Programming environment

To introduce our soft pneumatic programming environment, in the following section, we present our statement equivalent coding blocks for the *If*, *If...break*, and *For* control flow statements and show their operation in movie S1. Furthermore, we demonstrate that our soft pneumatic coding approach allows for introducing variables that can be written, read, and used through mechanical interaction with the pneumatic system. Analogous to coding, we give the equivalent pseudocodes of the soft pneumatic programs we introduce in this section. We include additional examples of control statements and circuit designs implemented through our electronics-free strategy in the Supplementary Materials.

Fluidic *If* implementation

The first fluidic circuit we design reproduces the functionality of an *If* statement. To design our fluidic circuit, we make use of a soft hysteretic valve that was previously developed by our group (*25*). The hysteretic valve is based on an elastomeric dome with three cuts at its apex. When the Δp pressure difference between the pressure upstream and downstream of the valve is below a critical value Δp_{open}, the valve remains in a closed state. When the pressure on the initially closed valve reaches a critical value $\Delta p > \Delta p_{open}$, the dome snaps and the valve transitions to its open state in which air can flow through the valve. Upon reducing the pressure, the dome snaps back to its closed state when the upstream pressure drops below a Δp_{close} value. We report examples of the open and closed state of the soft hysteretic valve in Fig. 2A.

To enable the *If* functionality using this hysteretic valve, we make use of the critical opening pressure of the valve, combined with pressure fluctuations that can trigger the opening. We demonstrate the functionality in a fluidic circuit in Fig. 2B, where we show both the experimental realization and a schematic. Additional setup parameters are reported in table S17. We connect a compliant volume that we can interact with before the soft hysteretic valve and pressurize it to p_{in} so that the pressure difference across the valve is Δp_{if}, being lower than the critical value $\Delta p_{if} < \Delta p_{open}$. In addition, we connect a one-way valve to prevent backflow upon interaction with the compliant volume. Now, when we squeeze the compliant volume, the increase in pressure causes the soft hysteretic valve to snap open, triggering the *If* statement (Fig. 2B). Note that, in the circuit in Fig. 2 (B to D), we use a soft actuator to indicate the pressure when the *If* condition is met and after it when pressure equilibrates back to p_{in}.

As the whole circuit is pressure controlled, the pressure automatically resets to Δp_{if} after an opening event as long as the pressure difference dips below the closing pressure Δp_{close} of the valve (Fig. 2C). The timescale of the reset can be controlled by including additional fluidic resistances in series with a one-way valve. Moreover, the sensitivity of the *If* condition can be increased or decreased by varying Δp_{if} to be closer or further from Δp_{open}, respectively (fig. S11). Using this principle, soft hysteretic valves can be used to detect interaction with the environment as we show in Fig. 2.

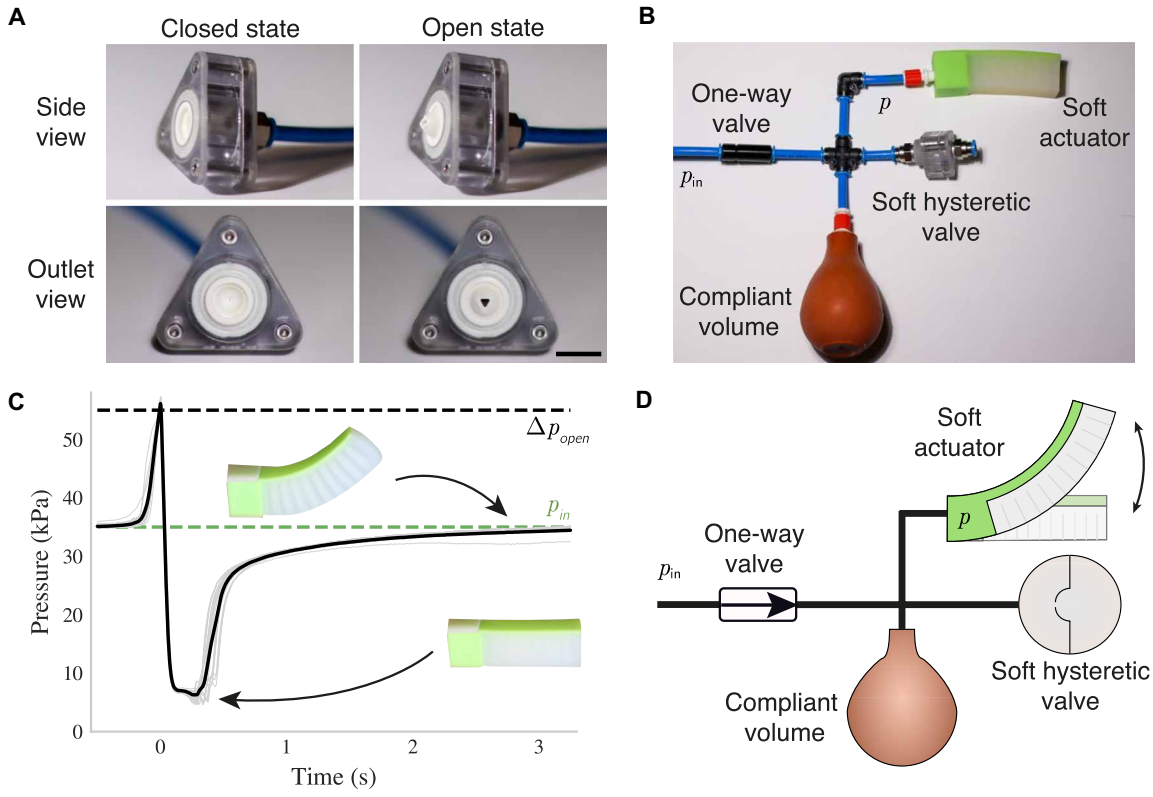

Fig. 2. Pneumatic equivalent of an *If* statement. (**A**) Experimental realization of a soft hysteretic valve in its holder in its closed and open states. Scale bar, 12 mm. (**B**) Experimental realization of the *If* pneumatic circuit. (**C**) Measurement of the pressure p upon interacting with (squeezing) the compliant volume. The interaction occurs at $t = 0$ s. The green and black dashed lines indicate $p_{in} = 35$ kPa and $\Delta p_{open} = 55$ kPa of the hysteretic valve, respectively. The gray solid lines are individual pressure measurements at 10 different interaction events, and the black solid line indicates their average. The inset photos represent the state of the soft actuator at the specific moment in time. (**D**) Schematic of pneumatic circuit used for the *If* functionality in (B).

This circuit can be regarded as an *If* statement that is triggered by a mechanical interaction and that automatically resets afterward. The equivalent pseudocode for this specific configuration is shown in Algorithm 1. Through our fluidic implementation of the *If* pneumatic coding block, we materialize the selection control statement required to meet the selection hypothesis of the Böhm-Jacopini theorem.

Algorithm 1 Pseudocode for the *If* circuit in Fig. 2

```
Pressurize(actuator)
if interaction then
    Deflate(actuator)
end if
```

Fluidic *If...break* implementation

The *If* condition alone serves the purpose of a trigger event for which input signals must overcome a certain programmable threshold to activate the conditional statement. To harness the analogy with coding statements further, we next design an *If...break* statement. To achieve this, we introduce a pneumatic normally open (N.O.) valve that draws inspiration from microfluidic valves (*26*). Our valve is shown in Fig. 3A and is fabricated by heat sealing two pairs of thermoplastic polyurethane (TPU) sheets together (fig. S13). The channels and pouches that reproduce the N.O. valve functionality are reported in Fig. 3B, as also indicated by the source-drain and gate channels. We stack the TPU sheets and confine them between two acrylic sheets (Fig. 3B) to guarantee the mutual influence of the channels on one another when they are pressurized. If the gate channel is not pressurized, then air can freely flow between the input (source) and output (drain) ports. When the gate channel is pressurized at a sufficiently high pressure, it will close the channel connecting the source to the drain.

To achieve the *If...break* functionality, we place the source and drain of the N.O. valve in series with the pressure input of the *If* circuit. We make use of the triggered output pressures from the hysteretic valve to control the gate pressure of the N.O. valve. Note that we include an R_{out} fluidic resistance in parallel with the actuator that is venting to the atmosphere. This ensures that, once the gate is pressurized, the soft actuator can deflate. With this setup, it is possible to control pressures and flows in other parts of the circuit as we demonstrate through the circuit in Fig. 3C. Additional setup details in table S17. Similar to the *If* circuit, we first pressurize $\Delta p_{if} < \Delta p_{open}$, which results in the bending of the soft actuator. When we now interact with the compliant volume, the valve opens, deflating the actuator and pressurizing the gate of the N.O. valve. As a result, the actuator does not repressurize because the gate pressure prevents airflow through the source-drain channel (Fig. 3D).

The functionality of this pneumatic circuit is described by the pseudocode in Algorithm 2, which is characteristic of an *If...break* statement. Note that larger (or smaller) R_{out} values result in slower (or faster) deflation times as well as in higher (or lower) actuator

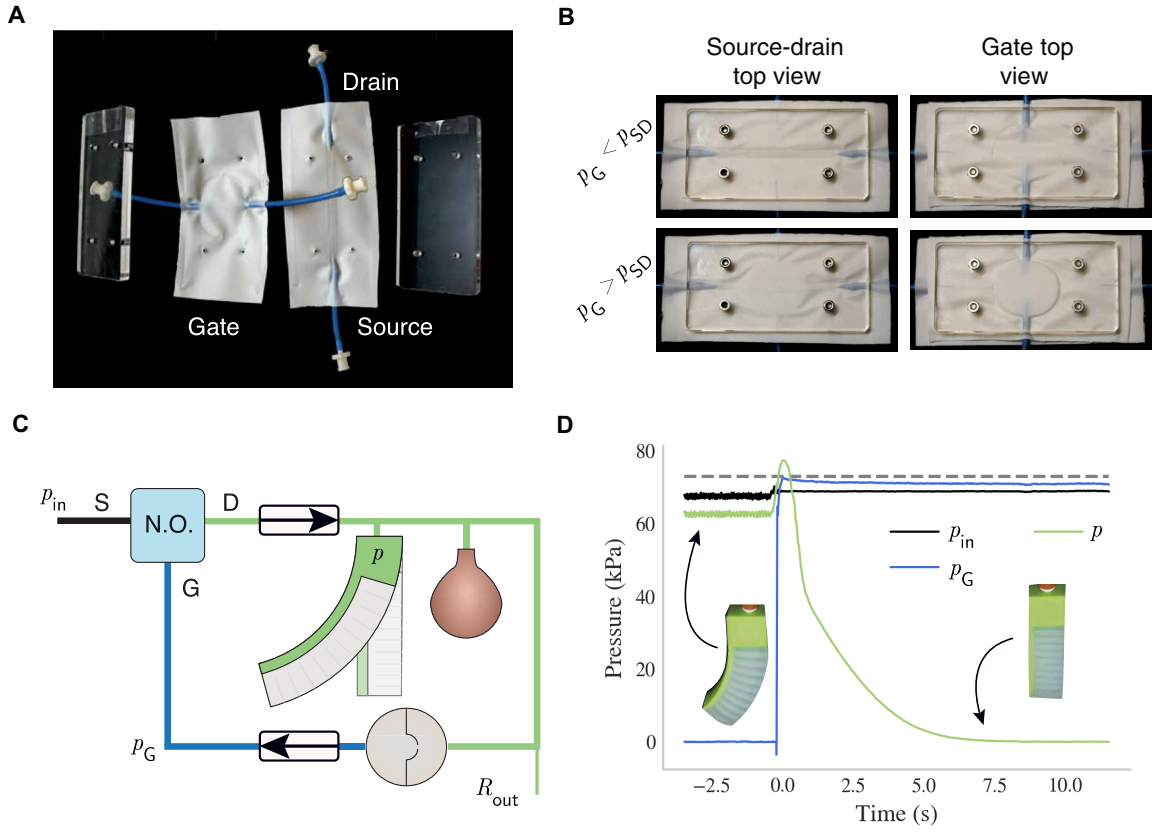

Fig. 3. Pneumatic equivalent of an *If...break* statement. (**A**) Schematic of the components used to assemble a pneumatic N.O. valve. (**B**) Assembled N.O. valve, for different gate pressures. (**C**) Schematic of the *If...break*. (**D**) Experimental measurement for a pneumatic circuit based on the schematic in (C), for an interaction event with the compliant volume that occurs at $t = 0$ s. The black, blue, and green solid lines represent the approximately constant input pressure $p_{in} \approx 68$ kPa, the pressure at the gate p_G of the N.O. valve, and the actuator pressure p, respectively. The dashed line represents the opening pressure $\Delta p_{open} = 70$ kPa of the hysteretic valve. Snapshots of the actuator's state are taken from the experimental realization of the setup at the corresponding moments in time.

pressures when the N.O. valve is in the open state, similarly to what previously has been demonstrated (*27*). Moreover, note that by adding different fluidic resistors venting to the atmosphere in the gate branch (fig. S17), we find different timescales for automatic reset of p_G. Thus, both long-term and short-term memory of the triggered state can be achieved. On top of that, if no fluidic resistor is used in the gate branch, p_G is held indefinitely, as we show in fig. S18.

Algorithm 2 Pseudocode for the *If... break* circuit in Fig. 3C

```
while true do
    Pressurize(actuator)
    if interaction then
        Deflate(actuator)
        break
    end if
end while
```

Fluidic *For* implementation

The ability to iteratively perform a task is a key feature for designing compact and versatile software. To obtain a fluidic implementation of the iterations needed to meet the requirements of the Böhm-Jacopini theorem, we implement a pneumatic circuit representative of the *For* control statement in our electronics-free environment. Similar to the *If* statement, we base the *For* functionality on circuits containing hysteretic valves. A previous work from our group demonstrated that, when a hysteretic valve is supplied with a continuous flow Q_{in}, it starts to act as a relaxation oscillator by opening and closing cyclically at a fixed frequency (*25*). The frequency can be tuned by varying a volume V_1 placed before the valve or by controlling the inflow Q_{in} of air. Therefore, this device converts a continuous input flow into a pulsatile output flow so that, at each snapping event, a finite amount of air traverses the orifice of the valve.

The pneumatic *For* circuit we design is based on two hysteretic valves connected in series, using only a single constant flow Q_{in} as input (Fig. 4A and table S17). We found that, when connecting the output of a first hysteretic valve with the input of a second hysteretic valve, the air released from the first valve pressurizes a volume V_2 placed between both valves in discrete steps (Fig. 4B). This stepwise increase in pressure occurs until the buckling pressure $\Delta p_{open,2}$ of the second valve is reached. At that moment, the trigger event of the first valve also triggers the opening of the second hysteretic valve, hence automatically resetting the whole system. We demonstrate the potential for applications with the example in Fig. 4 (A to C), in which we inflate a soft actuator in discrete steps, after which the position of the linear actuator resets before the cycle starts (Fig. 4B).

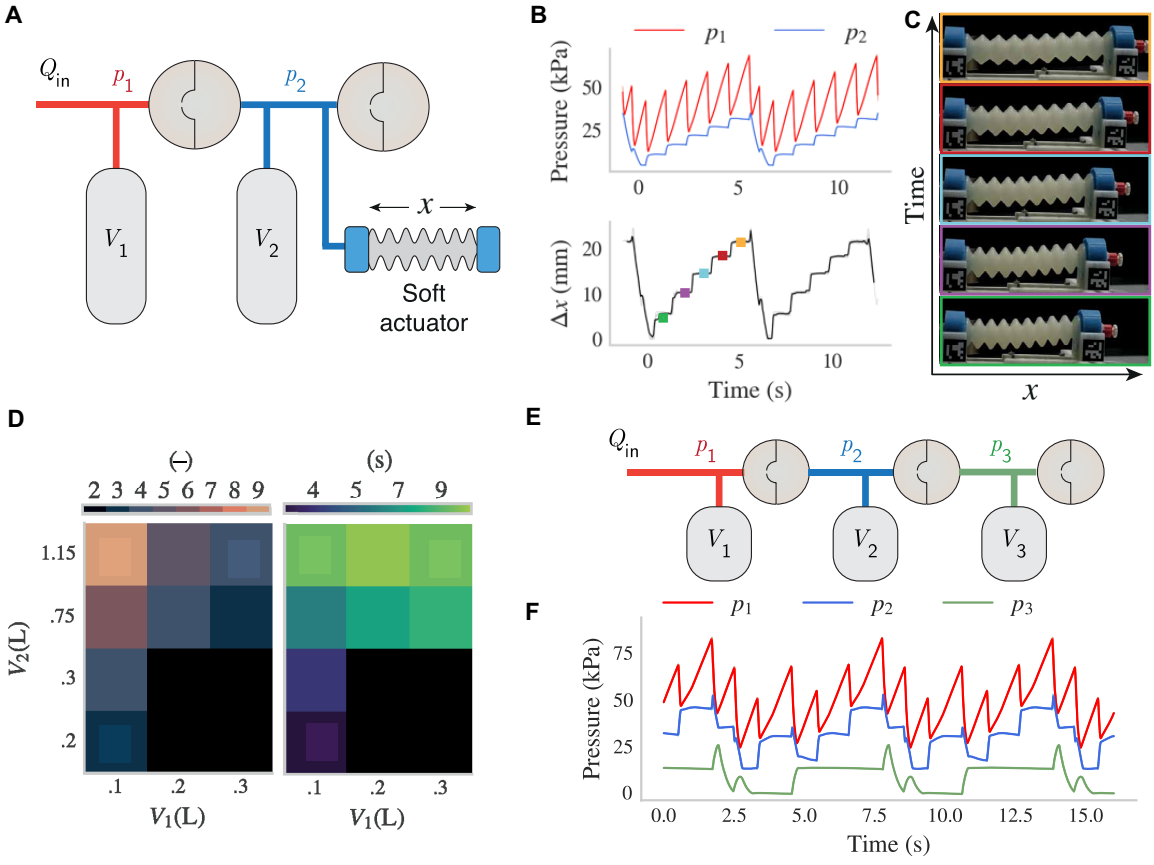

Fig. 4. Pneumatic equivalent of a *For* statement. (**A**) Schematic of the pneumatic *For* circuit. *x* indicates the linear extension of the soft actuator. (**B**) Experimental measurement of the averaged pressure in volumes V_1 and V_2 and tracking of the extension Δx of the linear actuator with respect to its resting position over 20 different repetitions of the *For* loop. (**C**) Snapshots of the linear actuator used for the corresponding highlighted positions as indicated by the colored dots in (B). (**D**) Different number of iterations (left) and periods (right) of the *For* loop for different volume combinations of V_1 and V_2. The black entries on the heatmaps represent V_1 and V_2 combinations for which the *For* behavior in (B) was not observed. Note that, for this experiment, the soft actuator was removed from the setup. (**E**) Schematic and (**F**) experimental data of the nested *For* loop.

From a functionality point of view, we obtain a stepwise increase in pressure that automatically resets after a number of discrete steps, hence mimicking the *For* control statement functionality. The functionality of this fluidic circuit is indicated in the pseudocode in Algorithm 3. Moreover, besides a *For* loop, the introduced pneumatic coding block can also be used as a time trigger or internal clock for pneumatic soft robots. In our fluidic circuit, note that the V_1 and V_2 volumes dictate both the number of iterations before the reset mechanism is triggered and the duration in time of the *For* loop. In Fig. 4D, we report the different number of steps in the *For* loop and the duration of the loop for different combinations of V_1 and V_2. More details about design rules for the number of iterations timing are given in fig. S20. We separately highlight three cases in our experiment in fig. S21B.

Algorithm 3 Pseudocode for the *For* circuit in Fig. 4A

```
nsteps = 5
while true do
    for i in (0,nsteps) do
        Extend(pose[i])
    end for
end while
```

The connection in series of multiple soft hysteretic valves can be further exploited to obtain nested *For* loops. We report an example case in Fig. 4 (E and F), where we connect three valves in series to obtain two nested, stepwise increasing pressures with two distinct states for p_2 and three for p_3, being able to describe overall six distinct states. Note that pressure not only is the actuating medium but also encodes the state of the system. Thus, pressure can be directly used by the circuit itself with no need for an external processor or transducer. Yet, this also puts constraints on the number of nested loops that can be achieved.

Sequential execution of instructions using the *For* statement

Before continuing with our fluidic implementation and representation of variables, we demonstrate the potential for application of the *For* pneumatic circuit by using it as a clocking device to enable simultaneous readout and execution of sequential instructions. With this demonstration, we show that we meet the first requirement for the Böhm-Jacopini theorem. We materialize this functionality by introducing the design of a demultiplexer in Fig. 5. The demultiplexer device we design is based on the working principles of

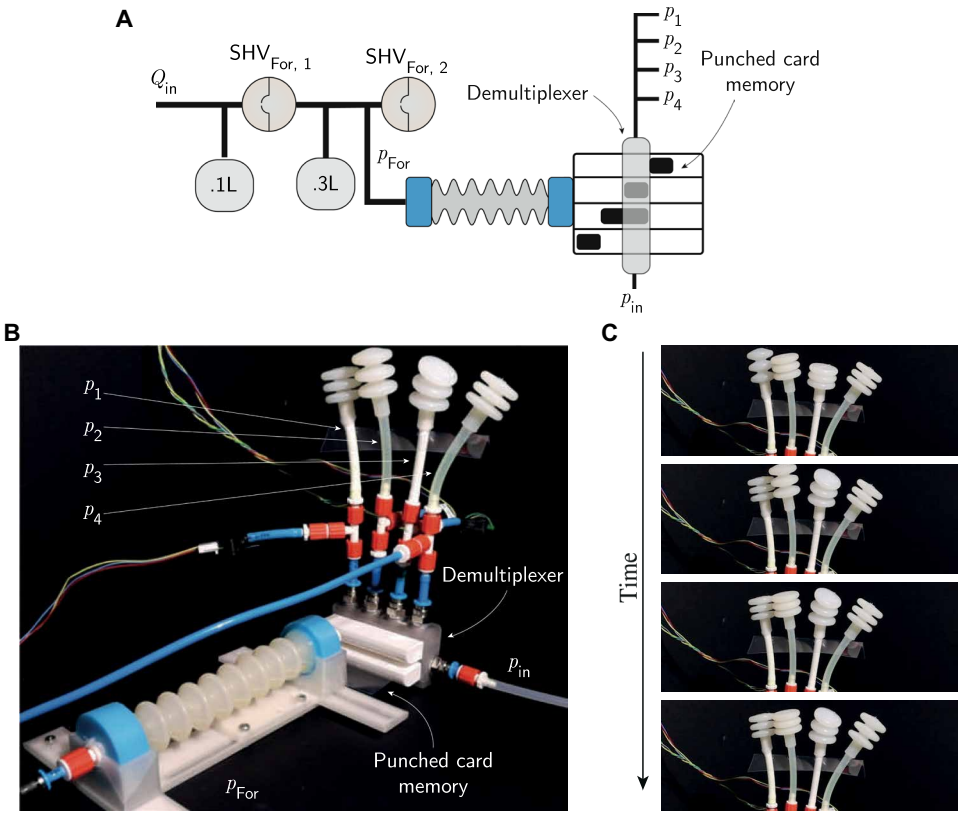

Fig. 5. Schematics of a pneumatic circuit and component to obtain sequential execution of instructions. The *For* loop enables clocking capabilities and feeds a 0.1 mm PVC punched card that encodes the instructions to be executed. By connecting an input pressure p_{in} = 367 kPa, the four different output channels can be pressurized following the sequence of instructions in the physical support memory. (**A**) Circuit design and physical support memory. In black, the punched holes provide the change of fluidic resistance required to pressurize a specific channel. The duration of operation can be encoded in the different lengths of each aperture. (**B**) Experimental realization of the setup. (**C**) Highlight of bellow deformation as a consequence of the sequential readout of instructions in time. We report the data from this experiment in fig. S4 and its realization in movie S1.

cardboard punched cards, particularly popular in traditional book organs, automatic looms, and early computers.

The demultiplexer strategy works by sliding a polyvinyl chloride (PVC) punched card through a slot, which, based on the positions of holes, connecting p_{in} to four different outputs. To address the pressure outputs in sequence and execute instructions sequentially, we drive a punch card with several holes using the pneumatic *For* coding block that generates the stepwise inflation of a linear extension of the bellow actuator as we demonstrated in Fig. 4B. Note that, to reach larger extensions of ~10 cm, we used a hysteretic valve with an opening pressure of 55 kPa, which is higher than the one we used in Fig. 4B. As a result, using the pneumatic circuit in Fig. 5A, the punched card is fed into the readout device shown in Fig. 5B at well-defined positions in time.

Using this demultiplexer, we can implement sequences and sequential execution of coding statements such as selection (*If*, *If…break*, and *If else*) and iteration (*For*) statements. Specifically, we aim to use this component to implement a device that can execute the pseudocode in Algorithm 4. As we highlight in this algorithm, the strategy implemented for executing the sequences of instructions can be rewritten in terms of *For* and *If* statements.

Algorithm 4 Pseudocode for the execution of sequential instructions as represented in Fig. 5B and movie S1. *StepTime* variable can be modified by operating on the frequency of the *For* loop feeding the instructions into the decoder.

```
while true do
    for i in (0,8) do
        if i in [0,1] then
            Pressurize(p1)
        end if
        if i∈[2,3] then
            Pressurize(p2,p3)
        end if
        if i∈[4,5] then
            Pressurize(p3)
        end if
        if i∈[6,7] then
            Pressurize(p4)
        end if
        Wait(StepTime)
        Depressurize(p1,p2,p3,p4)
    end for
end while
```

To design a pneumatic circuit to run this code and to demonstrate the workings of the demultiplexer, we connect its output channels to four bellow actuators as shown in Fig. 5B and use a specific design of a punch card that addresses all of the four output channels that are connected to the bellows. The resulting response when providing an input of $p_{in} = 367$ kPa is visually shown in Fig. 5C and movie S1, whereas we report the data for this experiment in fig. S4. As expected, instructions that originate from the design of the punched card are executed sequentially in time at the frequency dictated by the fluidic clock mechanism enabled by the *For* loop (fig. S4, A to C, and movie S1).

Note that, in our design, the number of instructions that can be read in time mostly depends on the number of extension steps executed by the *For* loop. Instead, the number of channels (different sets of instructions to be executed) is determined by the design of the demultiplexer and depends on the number of output channels of the readout device. Moreover, note that, as the position of the punched card is determined by the extension of the linear actuator, once the *For* loop restarts, the punched-hole card also resets to initial conditions. During this retraction phase, all the instructions are sequentially re-executed in reverse order when the linear actuator retracts. However, the retraction occurs relatively quickly, limiting the time that each channel is addressed as can also be seen in fig. S4. In addition, by using internal delays and resistances, the input pressure can potentially effectively be disconnected during the retraction phase. Design parameters for the experimental realization of this experiment are reported in figs. S5 and S6 and table S17.

Fluidic implementation of variables

Variables are fundamental building blocks of any program as they allow encoding of both the results of internal memory and external inputs. We can already identify the use of variables in previously introduced coding statements. For example, for the *If…break* statement, the N.O. valve describes a discrete variable as it can be in an open or closed state, analogous to a true or false state. By setting the gate pressure p_G of the N.O. valve to intermediate levels between the fully open and fully closed states, we can regulate the flow from source to drain, therefore encoding nonbinary information through the pressure at the gate branch. To demonstrate this, we leverage this approach to write continuous pneumatic variables to control both the amplitude and frequency of oscillation in a pneumatic soft actuator.

In our demonstrator in Fig. 6A and movie S4, we tune the gate pressures for the frequency control $p_{G,f}$ and for the amplitude control $p_{G,Ampl.}$ of two different N.O. valves. Here, $p_{G,f}$ dictates the fluidic resistance of the first N.O. valve that converts an input pressure p_{in} coming from a pressure regulator into a tunable, continuous input flow for the downstream branch. The resulting flow feeds a soft hysteretic valve that converts the continuous input flow into a pulsatile one. Moreover, we use $p_{G,Ampl.}$ to isolate the actuator branch from the remainder of the circuit, therefore controlling the amplitude of the oscillations of the linear actuator. Note that we connect the output of the soft hysteretic valve in parallel to a fixed fluidic resistor $R_{out,S}$. Moreover, the drain of the second N.O. valve is connected in parallel with a second fixed resistor $R_{out,D}$ and the soft actuator. The roles of $R_{out,S}$ and $R_{out,D}$ are to vent to the atmosphere after each oscillation, allowing pressure to automatically reset after each snapping event of the soft hysteretic valve. With this demonstrator, we have therefore introduced the notion of continuous variables in our pseudocode, as described in Algorithm 5.

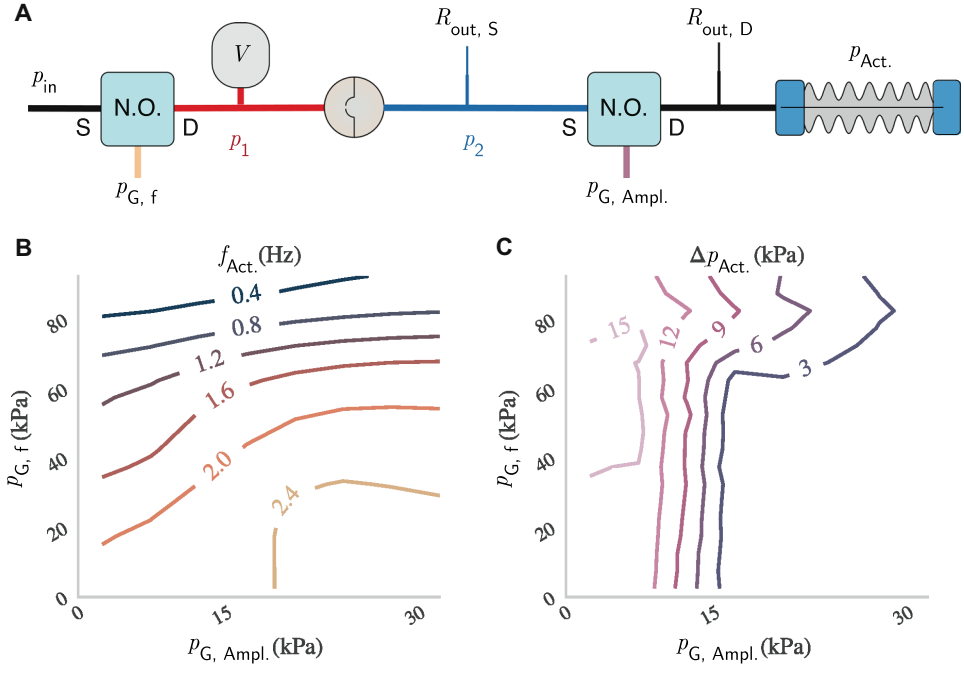

Fig. 6. Encoding pneumatic continuous variables using N.O. valves. (**A**) Schematic of the pneumatic circuit. (**B**) Isofrequency f_{Act} and (**C**) isoamplitude of oscillations Δp_{Act} curves as a function of the control parameters $p_{G,Ampl.}$ and $p_{G,f}$.

Algorithm 5 Pseudocode for the circuit using continuous variables in Fig. 6A

while true **do**
  Extend(amplitude($p_{G,Ampl.}$),frequency($p_{G,f}$))
end while

We demonstrate in Fig. 6 (B and C) that it is possible to control the frequency, amplitude, and absolute pressures of oscillations. Our circuit characterization shows that it is possible in the parameter design space to tune frequencies between 0.1 and 5 Hz while controlling the amplitude of the oscillation at the same time. We report extended data on this experiment in fig. S19. Moreover, we push further the parallelism between traditional programming and our hardware fluidic coding by introducing Boolean variables in fig. S14.

Programming behavior and feedback in a soft gripper

To demonstrate the applicability of our approach toward designing fluidic programs using the previously introduced fluidic coding blocks, we next combine our statement-equivalent circuits to assemble a soft fluidic demonstrator that runs a basic program. This program focuses on enabling an independent interaction with the environment. Our aim is to integrate embedded sensing in the gripper so that it is able to detect an object and change behavior accordingly. Specifically, we aim to develop a program for a single pneumatic soft robotic gripper with two pneumatic degrees of freedom. One degree of freedom activates two bending actuators that can grip an object, whereas the second degree of freedom controls a soft extension actuator that is used to extend and retract the gripper. These soft actuators are assembled on a polylactic acid (PLA) 3D-printed frame (Fig. 7A).

To design our demonstrator, we start by writing the pseudocode as given in Algorithm 6. This pseudocode describes the behavior of a soft gripper that is searching for an object by extending along a line in multiple steps. If an object is detected, then the object is grasped, after which the gripper retrieves the object by moving back to its initial position.

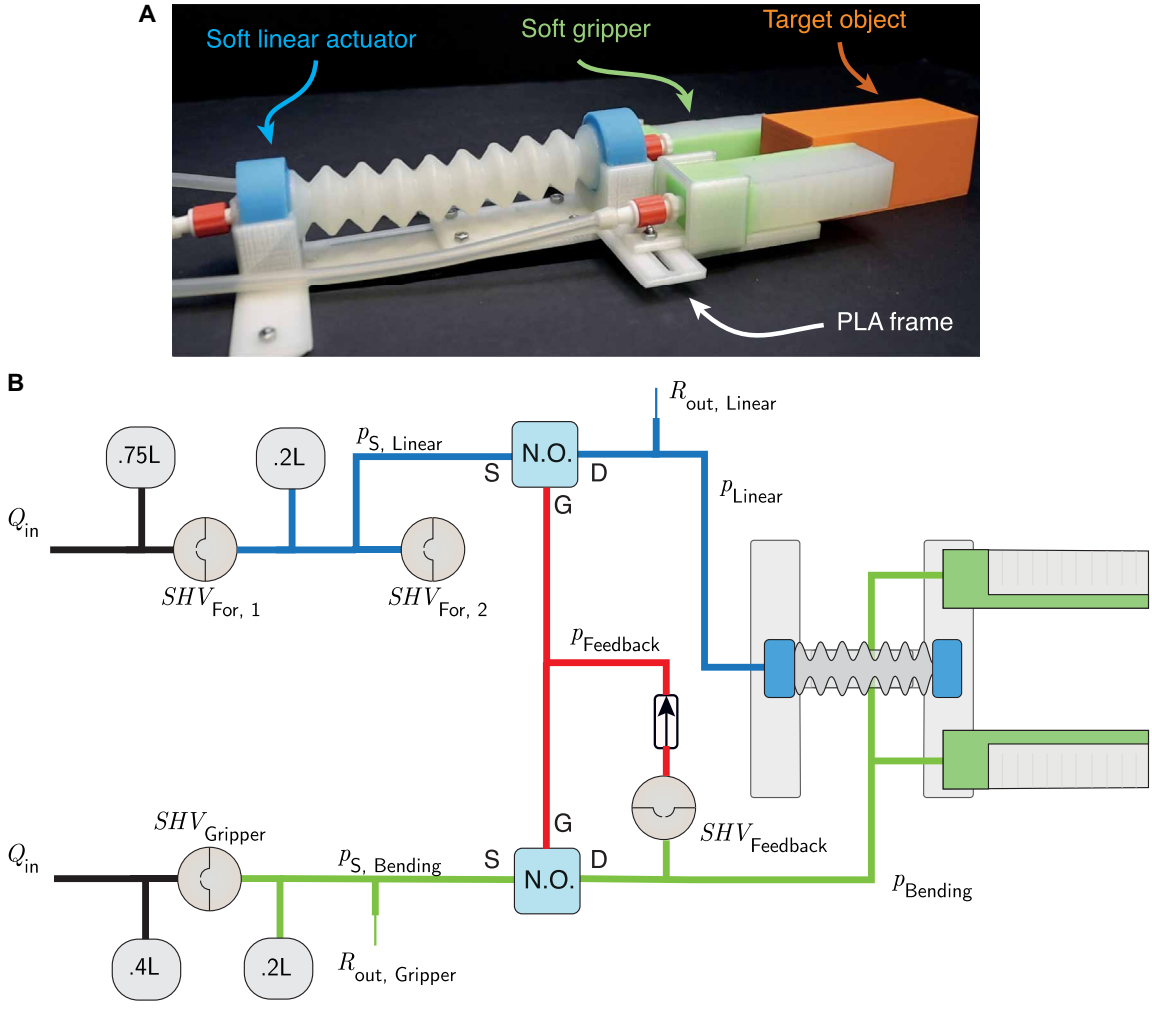

Fig. 7. Assembly of pneumatic coding blocks to design a circuit for a soft robotic retriever demonstrator. (**A**) Highlight of the hardware components. We assemble a soft linear actuator with two bending actuators through a PLA frame. This allows us to obtain a pneumatic soft robot that can elongate and grab target objects. (**B**) We combine the *For* (blue) and *If...break* (red) pneumatic coding blocks to functionalize the actuators so that the pneumatic soft retriever executes the program in Algorithm 6.

Algorithm 6 Pseudocode for the pneumatic soft retriever in Fig. 7

```
nsteps = 4
search = true
found_object = false
while search do
    for i in (0,nsteps) do
        Extend(pose[i])
        found_object = SearchObject()
        if found_object then
            search = false
            break
        end if
    end for
    RetractLinear()
end while
```

Next, we transform the pseudocode into a fluidic circuit as shown in Fig. 7B, for which we combine the previously developed fluidic coding statements. We combine the pneumatic coding blocks representative of the *For* and *If…break* statements. Note that we only need a constant power input to the system, and no control is further needed. This power is provided by two flow sources as shown in Fig. 7B, which feed into the two separate branches of our fluidic program that run in parallel. One of the branches incrementally extends the position of the gripper by pressurizing the linear actuator similar to the example in Fig. 4B. The second branch periodically inflates and deflates the soft gripper, similar to the overall behavior explained in the *For* coding block, but with only one iteration before reset. The range of motion of the gripper can be tuned by changing the $R_{out,Gripper}$, following the same approach we presented while studying pneumatic variables in Fig. 6.

Moreover, we use the *If…break* circuit to embed sensing in the soft gripper. This feature can be achieved by harnessing the fact most soft actuators can also directly be used as a sensor (*28*). As an example, when the deformation of the soft gripper is constrained, pressure will (slightly) increase as internal volumetric changes are now limited by the interaction with the environment. Comparing this observation with the examples we provided in the *If…break* circuit, this means that we do not need the additional bulb. Instead, we can use the increase in pressure in the soft actuator due to interactions with the environment as the trigger event for the hysteretic valve used in the *If…break* statement.

At the same time, the trigger signal from the *If…break* can be used to isolate the linear actuator following the example circuit we demonstrated in Fig. 6, preventing it from oscillating once the object is detected. As a result, when the *If…break* statement is triggered, the feedback loop should change the preprogrammed response of the soft robotic unit. From a functionality point of view, the states of the Boolean variables we report in the pseudocode Algorithm 6 are encoded in the pressure at the gates of the N.O. valves. By using a fluidic resistance $R_{out,Linear}$ in parallel with the linear actuator, once it is isolated from the *For* branch, it deflates through $R_{out,Linear}$ and thus retracts. Note that we do not use this strategy in the gripper branch so that the pressure inside the gripper is retained once the *If…break* condition is triggered. By doing so, the retriever keeps on gripping the target object while retracting instead of scanning the environment.

The fully measured behavior of the soft gripper with embedded sensing is shown in Fig. 8. When the program is initialized, the inflow at the gripper is regulated to match the actuation frequency of the linear actuator. This subprogram loops until the *If…break* condition is triggered by the presence of the target object, which we place after the 18 loops of the full program, and occurs after ~370 s in Fig. 8 (A to C). At this point, we see a clear change in behavior as the N.O. gates in the fluidic circuit are triggered. This caused the soft extension actuator to retract and the soft gripper to remain closed. It is important to note that we still observe some pressure oscillations in the linear actuator and soft gripper branch of the program. In movie S2, we report the execution of this experiment together with the autonomous switch from the scan and search (Fig. 8, D to F) to the grip and retrieve behaviors (Fig. 8, E to G). We do note that, during the operation of the gripper, the operator is manually touching the soft gripper, which could aid the detection of the object by increasing the deformation of the gripper and thus the internal pressure. To demonstrate that the gripper can autonomously detect an object, we performed an additional run of this experiment with a slightly modified circuit design in the *For* pneumatic coding block (fig. S23 and movie S2).

Although not directly implemented in the demonstrator of the retriever in Fig. 7, we can devise additional strategies to increase the complexity of the task performed, for example, to release the object after a certain time. These strategies can remain relatively straightforward: already including a fluidic resistance in parallel with the gripper and venting to the atmosphere could be an effective way to incorporate such timing. Such timing is also shown in figs. S17, S18, and S25 and movie S3, where we demonstrate that, by including increasingly larger fluidic resistances in parallel with the gate branch of the N.O. valve and venting to the atmosphere, we can retain pressure for increasingly longer times. Through this approach, the pressure in the gripper branch could be retained long enough to allow the full retrieving of the target object before releasing the grip by lowering the pressure in the gripper branch.

DISCUSSION

Inspired by traditional coding statements, we introduce a strategy to materialize algorithms using electronics-free pneumatics circuits. Through our design rules, we program responses and switch between responses in soft robotic systems. Furthermore, from a functionality point of view, the pneumatic circuits we propose fulfill the requirements for the Böhm-Jacopini theorem (*24*). For each of our designs, we illustrate the potential for applications in soft robotics, highlighting features in the parameter space useful to encode complex responses in pneumatic soft robots. The pneumatic coding blocks we introduce can perform operations in series (as in the case of the nested *For* loop in Fig. 4F) and in parallel (Algorithm 6 and Fig. 7 for the retriever demonstrator). This last feature lends itself well to further investigation of computational interest, as well as the possibility of introducing sensory feedback to different branches of the pneumatic branches, similar to biological processes that can be executed and triggered in parallel.

Through our design choice of working in an electronics-free environment, we ensure that pressurized air volumes work both as an actuating medium and state control variables of the system. This represents a valuable asset, e.g., in scenarios where electrostatic discharges can compromise the operation of robotic systems. Drawing inspiration from the self-sensing strategy in pneumatic grippers (*28*), we introduce sensing through the actuators themselves, lifting the requirement for dedicated external (electronic) sensors to

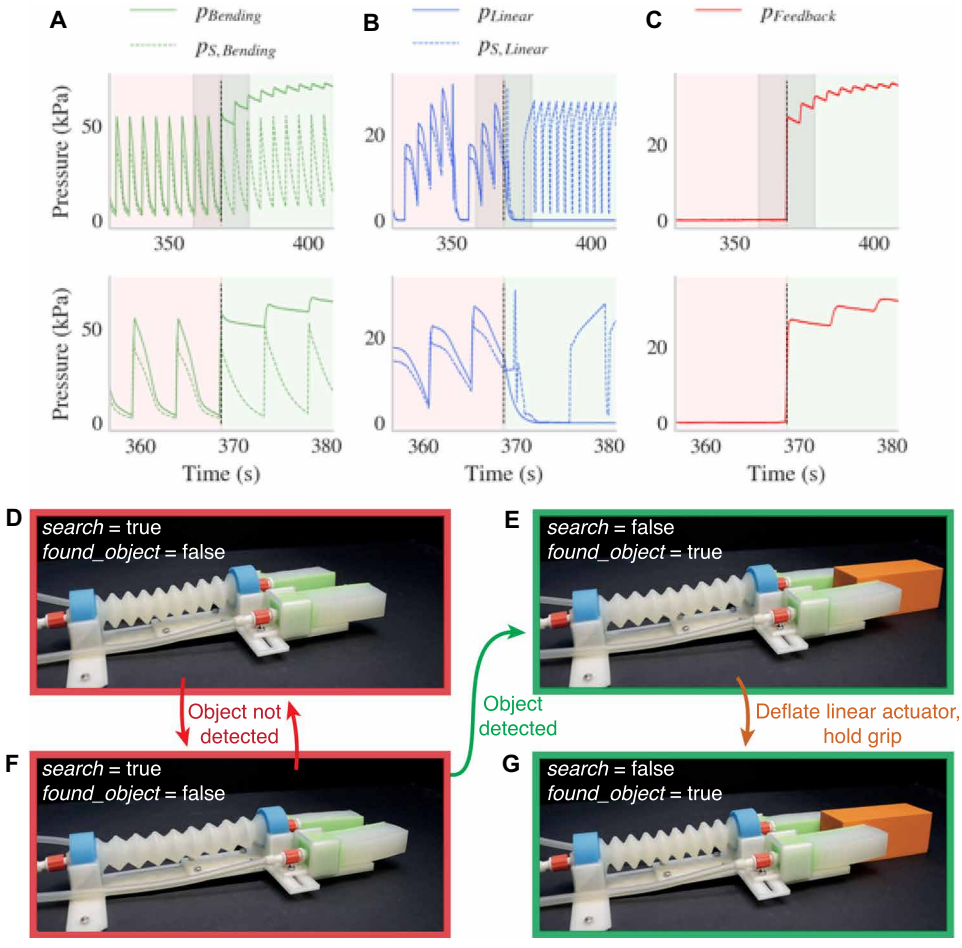

Fig. 8. Experimental data and schematic representation of the obtained behavior of the soft robotic retriever demonstrator. (**A** to **C**) Pneumatic responses of the circuit before and after the *If* condition is triggered upon gripping the orange object for the bending actuator branch (green), the linear actuator branch (blue), and the feedback branch (red), respectively. In the top row, we report an overview of the two different pneumatic responses before and after the *If...break* condition is triggered, with the successful gripping event occurring at $t^* = 368$ s. The shaded time window of the interaction event is shown in more detail in the bottom row. (**D** to **G**) Schematic representation of the behavior of the pneumatic soft retriever unit running Algorithm 6. Highlights of behavior before and after the *If...break* condition is triggered are highlighted respectively in red and green.

achieve feedback. Therefore, we embed actuation, sensing, and control within the same pneumatic, electronics-free platform, with no need for signal transducing.

Nevertheless, the design choice to not use signal transduction to control the pneumatic circuits imposes some limitations in the context of the scalability of our designs. As each component in a given branch of the circuit causes a pressure drop, it reduces the expendable energy to actuate and control the branches both downstream and in parallel to it. We report that this is not an issue at the scale of component types and numbers we investigated. A potential solution to overcome this potential limitation is to provide separate pressure inputs for different branches or decouple parallel branches using the temporary switches such as the N.O. valve we presented.

Although the individual pneumatic coding blocks we design can reproduce high-level functionalities of control statements in a coding language, combining such circuits does not directly translate into traditional programming. On the one hand, the analog and continuous nature of the pneumatic medium can impose some limitations while combining pneumatic circuits to achieve the desired functionalities. As an example, the physical properties of the hysteretic valves and actuators in the example in Fig. 7 must be simultaneously taken into account while designing the pneumatic circuit able to execute Algorithm 6 for the soft retriever. Such fundamental differences with traditional programming require an alternative approach to directly build a compiler able to return the optimal circuit design starting from a preferred high-level functionality. On the other hand, while combining subprograms in the very same example, the possibility to run programs in parallel and let them affect one another emerges from the design of the pneumatic circuits itself. Through this modular approach, we were able to build a soft gripper demonstrator that was able to run a program that we would not have been able to develop without insights gained from each coding block.

Using the design of the fluidic demultiplexer we introduced in Fig. 5A, we materialize flash memory-like properties in our demultiplexer that uses punched cards. Such physical memory can

relatively easily be implemented with off-the-shelf components and can address (re)programmability of sequential instructions through a different design of the punched holes. Furthermore, it is worth highlighting that our strategy can be used to extend the execution of instructions past the pure digital output signal, e.g., by designing slots with variable width or shape in the punched cards. However, it is technically not feasible with such punched-hole memory to reach the same order of magnitude of instruction density of Static Random-Access Memory (SRAM), Dynamic Random-Access Memory (DRAM), and flash memory that exists in more general-purpose computers today. Still, we expect that our electronics-free approach can still be useful to integrate redundancy and simple autonomous responses through hardware solutions.

In our study, we did not investigate whether we used the minimal required number and type of components for our designs. We opted instead to focus on the general hardware programming strategies rather than the specific circuit optimization. Efforts have been made in the community to tackle circuit optimization and create compilers to map specific actuator outputs to given pneumatic inputs. However, the programmability of pneumatic responses is generally addressed within the context of binary logic (*29*). Such an approach can be useful in many other studies both at the macrofluidic scale (*27*, *30*) and at the microfluidic scale (*15*, *31*). To our knowledge, only a few studies (*32*, *33*) have focused on nonbinary logic to program the behavior of soft robots so far. Although, in the current work, we only focused on showing the use of the fluidic coding blocks in a gripper demonstrator, other applications should be in reach. For example, the ability to exhibit locomotion patterns, as well as the capability to switch between them under given environmental cues, is a crucial feature for control and actuation autonomy in soft robots. We expect that our control strategy lends itself well to both programming such patterns and changing behavior as a response to environmental changes. This capability could be beneficial while designing locomotion patterns in (soft) robotic walkers. The pneumatic circuits we design could be used as a passive, redundant system to, e.g., detect impacts or sense the strength of interaction between the (soft) robotic system and its environment. Moreover, the control and design strategy we propose could be used, e.g., to obtain (re)programmability of the position of an actuator for rehabilitation devices (*34*) and wearable smart textiles (*35*, *36*).

We envision it would be possible to reproduce at least some of the pneumatic coding blocks that we presented in our work at smaller and larger scales, provided that different components with a similar nonlinear response in the pressure-flow diagram can be developed. Nevertheless, fabrication limitations and experimental setups constraints often set the pressure and dimensions to be in the same range we investigated. This is particularly true when zooming in on the pneumatic soft robotics community we address in our work, with only relatively few examples exploring other high-pressure (*37*), larger-size (*38*, *39*), or smaller-size (*26*, *40*) scales. To scale down our control strategies into the domain of microfluidic valves and actuators compatible to similar works (*41*, *42*), we believe that other nonlinearities and hysteresis could be used to obtain similar functionalities. As an example of such functionalities, microfluidic bistable and differential amplifiers and delay and signal generation capabilities (*43*, *44*) could be used as foundations for microfluidic coding blocks.

Despite our approach not being straightforward to implement for complex algorithms, it is often true that it is a small set of basic instructions that drives the actuation and life cycle of living beings in elementary yet complex scenarios. This is particularly true when considering systems that can navigate natural environments by means of simple interactions with their ecosystems. As a well-known example from the literature, we mention the case of the Venus flytrap plant. This carnivorous plant stays in an idle state until a prey triggers its sensory hairs more than once within a predefined time window. When these conditions are met, its leaves snap shut, thus trapping the prey (*45*, *46*). From a coding perspective, the plant is idle until the input signal overcomes a certain threshold twice, a behavior we can physically represent using the pneumatic *For* and *If* coding blocks. We replicate functionality similar to the Venus flytrap case study in fig. S24. Such a fundamental capability gives the Venus flytrap the capability to survive and grow in natural environments by manifesting autonomous behavior in actuation, sensing, and feedback from its environment. Similarly, we also believe that the responsiveness we implemented in pneumatic, electronics-free soft robots can help move toward autonomous, soft robots capable of decision-making while maintaining a limited number of components (*47*).

The aforementioned studies focus on how to use the response of individual nonlinear components to obtain programmable behaviors. Conversely, our efforts are instead aimed at defining a library of circuits to be used and combined in both binary and nonbinary logic. We believe that such an approach can encourage the opportunity to intuitively design responses in pneumatic soft robots starting from a desired pseudocode, in a modular electronics-free environment, even starting from different hardware solutions. Therefore, our approach paves the way for the realization of pneumatic coding blocks to be used for programming responses in soft robots that are meant to interact with their environment and manifest decision-making capabilities.

MATERIALS AND METHODS

Fabrication of the N.O. valve

To fabricate the N.O. valve, we use two plexiglass plates that enclose two pairs of TPU sheets (ECL000009 Ecoseal Film T150 85A), heat sealed together at 240°C. We heat seal the TPU to obtain the source-drain channel (fig. S1A) and gate channel (fig. S1B). The heat sealing procedure is executed using a modified 3D printer (Felix Tec 4 3D printer), of which we use the heat generated from a custom-modified printhead to heat seal two sheets of TPU following the designed patterns (*48*). We highlight fabrication details in fig. S2.

We fabricate the plexiglass plates by laser cutting (Speedy 300, Trotec) an 8-mm-thick plexiglass following the design in fig. S1C. Once the two identical plexiglass sheets have been laser cut and the TPU sheets heat sealed, the individual pieces are assembled together. We follow the step-by-step protocol in fig. S3. To be able to fluidically control the channels, we use a FESTO tube with an outer diameter of 4 mm with a 4-cm length, and we slide it inside the TPU channels as in fig. S3D. The TPU channel size is chosen so that the TPU channels always compress the FESTO tube, forming an airtight connection without additional elements required. We then connect a 2.25-mm Luer Lock at the free end of the FESTO tubes to connect each channel to our pneumatic circuits.

Fabrication of the interacting volumes

The heat sealing procedure we use to fabricate enclosed volumes is closely related to procedure we reported in fig. S2, the difference being the material used. The pattern we heat seal is shown in fig. S7A. To fabricate these TPU-coated nylon fabric pouches, we use a nylon-coated TPU (Nylon, 70den, TPU-coated one side, sealing temperature of 270°C). We connect the heat-sealed pouch to a FESTO 6-mm tube to connect it to our pneumatic circuits as we show in fig. S7 (B to D). As both our pouch design and the FESTO tube are made of TPU, we can heat seal the two elements together using a custom-designed heat sealer made up of a circular heater, controlled at 190°C. The design of the custom heat sealer is reported in fig. S7 (G to I). After placing a heat-resistant layer (commercial baking paper) between the circular heating element and the connection point of the TPU pouch with the FESTO tube (fig. S7E), we wait about 15 s to obtain an airtight connection. The resulting component can withstand pressures higher than 2 bar without leakages.

Fabrication of the soft actuators

Linear actuator

The design of the linear actuator is based on the schematics of bellow actuators (*49*) and reported in fig. S8. To fabricate this component, we use outer molds in VeroClear on a PolyJet 3D printer (Eden260VS, Stratasys) and a BVOH (Butenediol Vinyl Alcohol Copolymer) water-soluble inner mold. We assemble the inner and outer mold together and align the two using a metal rod that fits the inner diameter of the inner mold. After assembling the molds as in fig. S8A, we inject Smooth-On Dragon Skin (DS) 20. We inject this two-component silicon using a cartridge (AF 400-01-10-01, Sulzer) after the two components have been degassed. Before casting silicones, we spray the surface of the molds with Ease release 200, Smooth-On. We use a static mixing nozzle (MFQ 05-24L) using a pneumatic extrusion gun to inject the DS20 silicon through the injection hole highlighted in fig. S8A. After this step, we use a cartridge filled with Elite Double (ED) 32 Fast silicone (Zhermack) to fill the outer molds of the end-caps as in fig. S8B. When the ED32 silicon is fully cured, we remove the inside rod used for the alignment and use a water pump in parallel with a tunable flow resistor for venting to flush the water-soluble support material from inside the actuator. We report the dimensions of the linear actuator in fig. S8 (C and D).

Bending actuator

We fabricate the bending actuators following a procedure similar to the one already developed within our group (*28*). We report fabrication details and dimensions in figs. S9 and S10, respectively. For the fabrication of the air chambers we highlight in fig. S10B, we use a sacrificial inner mold in BVOH, highlighted in fig. S9. We fabricate this actuator using of two different silicones (ED32 and DS20). Therefore, a two-step procedure is required for allowing two different silicones to bond together. First, we assemble the molds as in fig. S9A using M6 screws to guarantee alignment. We position the inner mold in the slot of the bottom mold. We inject the DS20 silicone after spaying mold release and the same equipment as described in the fabrication of the linear actuator. We wait 3.5 hours of the 5 hours required for the DS to cure, and we disassemble the top and bottom molds of the first step (fig. S9A) to substitute them, respectively, with the top and bottom molds of the second step (fig. S9B) onto which we apply mold release. Casting the ED32 on the DS20 after this time guarantees a solid interface between the two materials. We insert the plug to obtain an entry point to pneumatically actuate the resulting actuator. We then inject the ED32 and wait for the DS20 to be fully cured. We then use a water pump in parallel with a tunable flow resistor for venting to flush the water-soluble support material from inside the actuator. We report the geometrical parameters of our design in fig. S10.

Supplementary Materials

The PDF file includes:

Supplementary Material
Figs. S1 to S26
Tables S1 to S17
Legends for movies S1 to S3
References

Other Supplementary Material for this manuscript includes the following:

Movies S1 to S3

REFERENCES AND NOTES

1. E. Navas, R. Fernández, D. Sepúlveda, M. Armada, P. Gonzalez-de Santos, Soft gripper for robotic harvesting in precision agriculture applications, in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)* (IEEE, 2021), pp. 167–172.
2. M. Arfaee, A. Vis, J. Kluin, Future technologies in total artificial heart development: Can a robot become as good as a donor heart? *Eur. Heart J.* **43**, 4970–4972 (2022).
3. S. Jadhav, V. Kannanda, B. Kang, M. T. Tolley, J. P. Schulze, Soft robotic glove for kinesthetic haptic feedback in virtual reality environments. *Electron. Imaging* **29**, 19–24 (2017).
4. Y. Yim, Y. Noguchi, F. Tanaka, A wearable soft robot that can alleviate the pain and fear of the wearer. *Sci. Rep.* **12**, 17003 (2022).
5. D. S. Shah, J. P. Powers, L. G. Tilton, S. Kriegman, J. Bongard, R. Kramer-Bottiglio, A soft robot that adapts to environments through shape change. *Nat. Mach. Intell.* **3**, 51–59 (2021).
6. E. W. Hawkes, L. H. Blumenschein, J. D. Greer, A. M. Okamura, A soft robot that navigates its environment through growth. *Sci. Robot.* **2**, eaan3028 (2017).
7. Y. Zhang, P. Li, J. Quan, L. Li, G. Zhang, D. Zhou, Progress, challenges, and prospects of soft robotics for space applications. *Adv. Intell. Syst.* **5**, 2200071 (2023).
8. C. Lee, M. Kim, Y. J. Kim, N. Hong, S. Ryu, H. J. Kim, S. Kim, Soft robot review. *Int. J. Control. Autom. Syst.* **15**, 3–15 (2017).
9. M. Cianchetti, C. Laschi, A. Menciassi, P. Dario, Biomedical applications of soft robotics. *Nat. Rev. Mater.* **3**, 143–153 (2018).
10. M. Hassani, Mine Kafon Ball (MINE KAFON LAB, 2011); https://minekafon.org/index.php/mine-kafon-ball/.
11. S. T. Mahon, A. Buchoux, M. E. Sayed, L. Teng, A. A. Stokes, Soft robots for extreme environments: Removing electronic control, in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)* (IEEE, 2019), pp. 782–787.
12. S. Song, S. Joshi, J. Paik, CMOS-inspired complementary fluidic circuits for soft robots. *Adv. Sci.* **8**, 2100924 (2021).
13. Y. Masuda, M. Ishikawa, Review of electronics-free robotics: Toward a highly decentralized control architecture. *J. Robot. Mechatron.* **34**, 202–211 (2022).
14. D. J. Preston, P. Rothemund, H. J. Jiang, M. P. Nemitz, J. Rawson, Z. Suo, G. M. Whitesides, Digital logic for soft devices. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 7750–7759 (2019).
15. S. Ahrar, M. Raje, I. C. Lee, E. E. Hui, Pneumatic computers for embedded control of microfluidics. *Sci. Adv.* **9**, eadg0201 (2023).
16. J. T. B. Overvelde, T. Kloek, J. J. A. D'haen, K. Bertoldi, Amplifying the response of soft actuators by harnessing snap-through instabilities. *Proc. Natl. Acad. Sci. U.S.A.* **112**, 10863–10868 (2015).
17. B. Gorissen, E. Milana, A. Baeyens, E. Broeders, J. Christiaens, K. Collin, D. Reynaerts, M. De Volder, Hardware sequencing of inflatable nonlinear actuators for autonomous soft robots. *Adv. Mater.* **31**, 1804598 (2019).
18. N. Vasios, A. J. Gross, S. Soifer, J. T. B. Overvelde, K. Bertoldi, Harnessing viscous flow to simplify the actuation of fluidic soft robots. *Soft Robot.* **7**, 1804598 (2020).
19. B. Van Raemdonck, E. Milana, M. De Volder, D. Reynaerts, B. Gorissen, Nonlinear inflatable actuators for distributed control in soft robots. *Adv. Mater.* **35**, 2301487 (2023).
20. Y. Chi, Y. Li, Y. Zhao, Y. Hong, Y. Tang, J. Yin, Bistable and multistable actuators for soft robots: Structures, materials, and functionalities. *Adv. Mater.* **34**, 2110384 (2022).
21. Z. Jiao, Z. Hu, Y. Shi, K. Xu, F. Lin, P. Zhu, W. Tang, Y. Zhong, H. Yang, J. Zou, Reprogrammable, intelligent soft origami lego coupling actuation, computation, and sensing. *Innovation* **5**, 100549 (2024).

22. W. Lee, D. J. Preston, M. P. Nemitz, A. Nagarkar, A. K. MacKeith, B. Gorissen, N. Vasios, V. Sanchez, K. Bertoldi, L. Mahadevan, and G. M. Whitesides, A buckling-sheet ring oscillator for electronics-free, multimodal locomotion. *Sci. Robot.* **7**, 100549 (2022).
23. D. C. Dennett, V. Braitenberg, Vehicles: Experiments in synthetic psychology. *Philos. Rev.* **95**, 137–139 (1986).
24. W. W. Peterson, T. Kasami, N. Tokura, On the capabilities of while, repeat, and exit statements. *Commun. ACM* **16**, 503–512 (1973).
25. L. C. van Laake, J. de Vries, S. M. Kani, J. T. B. Overvelde, A fluidic relaxation oscillator for reprogrammable sequential actuation in soft robots. *Matter* **5**, 2898–2917 (2022).
26. M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, S. R. Quake, Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science* **288**, 113–116 (2000).
27. B. Jumet, Z. A. Zook, A. Yousaf, A. Rajappan, D. Xu, T. F. Yap, N. Fino, Z. Liu, M. K. O'Malley, D. J. Preston, Fluidically programmed wearable haptic textiles. *Device* **1**, 100059 (2023).
28. S. Zou, S. Picella, J. de Vries, V. G. Kortman, A. Sakes, J. T. B. Overvelde, A retrofit sensing strategy for soft fluidic robots. *Nat. Commun.* **15**, 539 (2024).
29. S. V. Kendre, L. Whiteside, T. Y. Fan, J. A. Tracz, G. T. Teran, T. C. Underwood, M. E. Sayed, H. J. Jiang, A. A. Stokes, D. J. Preston, G. M. Whitesides, M. P. Nemitz, The soft compiler: A web-based tool for the design of modular pneumatic circuits for soft robots. *IEEE Robot. Autom. Lett.* **7**, 6060–6066 (2022).
30. A. Rajappan, B. Jumet, R. A. Shveda, C. J. Decker, Z. Liu, T. F. Yap, V. Sanchez, D. J. Preston, Logic-enabled textiles. *Proc. Natl. Acad. Sci. U.S.A.* **119**, e2202118119 (2022).
31. K. A. Gopinathan, A. Mishra, B. R. Mutlu, J. F. Edd, M. Toner, A microfluidic transistor for automatic control of liquids. *Nature* **622**, 735–741 (2023).
32. C. J. Decker, H. J. Jiang, M. P. Nemitz, S. E. Root, A. Rajappan, J. T. Alvarez, J. Tracz, L. Wille, D. J. Preston, G. M. Whitesides, Programmable soft valves for digital and analog control. *Proc. Natl. Acad. Sci. U.S.A.* **119**, e2205922119 (2022).
33. E. G. Hevia, C. M. McCann, M. Bell, N.-s. P. Hyun, C. Majidi, K. Bertoldi, R. J. Wood, High-gain microfluidic amplifiers: The bridge between microfluidic controllers and fluidic soft actuators. *Adv. Intell. Syst.* **4**, 2200122 (2022).
34. P. Polygerinos, K. C. Galloway, E. Savage, M. Herman, K. O'Donnell, C. J. Walsh, Soft robotic glove for hand rehabilitation and task specific training, in *IEEE Internatinal Conference on Robotics and Automation (ICRA)* (IEEE, 2015), pp. 2913–2919.
35. R. A. Shveda, A. Rajappan, T. F. Yap, Z. Liu, M. D. Bell, B. Jumet, V. Sanchez, D. J. Preston, A wearable textile-based pneumatic energy harvesting system for assistive robotics. *Sci. Adv.* **8**, eabo2418 (2022).
36. L. Paterno, L. Lorenzon, Soft robotics in wearable and implantable medical applications: Translational challenges and future outlooks. *Front. Robot. AI* **10**, 1075634 (2023).
37. K. Ikuta, Y. Matsuda, D. Yajima, Y. Ota, Pressure pulse drive: A control method for the precise bending of hydraulic active catheters. *IEEE ASME Trans. Mechatron.* **17**, 876–883 (2012).
38. N. S. Usevitch, Z. M. Hammond, M. Schwager, A. M. Okamura, E. W. Hawkes, S. Follmer, An untethered isoperimetric soft robot. *Sci. Robot.* **5**, eaaz0492 (2020).
39. N. Oh, H. Rodrigue, Toward the development of large-scale inflatable robotic arms using hot air welding. *Soft Robot.* **10**, 88–96 (2023).
40. Y.-F. Zhang, C. J.-X. Ng, Z. Chen, W. Zhang, S. Panjwani, K. Kowsari, H. Y. Yang, Q. Ge, Miniature pneumatic actuators for soft robots by high resolution multimaterial 3D printing. *Adv. Mater. Tech.* **4**, 1900427 (2019).
41. A. Pal, D. Goswami, R. V. Martinez, Elastic energy storage enables rapid and programmable actuation in soft machines. *Adv. Funct. Mater.* **30**, 1906603 (2020).
42. A. C. Glavan, R. V. Martinez, E. J. Maxwell, A. B. Subramaniam, R. M. D. Nunes, S. Soh, G. M. Whitesides, Rapid fabrication of pressure-driven open-channel microfluidic devices in omniphobic RF paper. *Lab Chip* **13**, 2922 (2013).
43. Q. Zhang, M. Zhang, L. Djeghlaf, J. Bataille, J. Gamby, A.-M. Haghiri-Gosnet, A. Pallandre, Logic digital fluidic in miniaturized functional devices: Perspective to the next generation of microfluidic lab on chips. *Electrophoresis* **38**, 953–976 (2017).
44. I. A. Loe, T. Zheng, K. Kotani, Y. Jimbo, Controlling fluidic oscillator flow dynamics by elastic structure vibration. *Sci. Rep.* **13**, 8852 (2023).
45. M. S. Imtiaz, "Calcium oscillations and pacemaking" in *Advances in Experimental Medicine and Biology* (Springer, 2012), pp. 511–520.
46. G. M. Durak, R. Thierer, R. Sachse, M. Bischoff, T. Speck, S. Poppinga, Smooth or with a snap! Biomechanics of trap reopening in the Venus flytrap. *Adv. Sci.* **9**, e2201362 (2022).
47. F. J. Tauber, V. Slesarenko, Early career scientists converse on the future of soft robotics. *Front. Robot. AI* **10**, 1129827 (2023).
48. M. Arfaee, J. Kluin, J. T. B. Overvelde, Modeling the behavior of elastic pouch motors, in *2023 IEEE International Conference on Soft Robotics (RoboSoft)* (IEEE, 2023).
49. J. de Vries, "Energy Efficiency of Soft Pneumatic Extension Actuators," thesis, Delft University of Technology, Delft, Netherlands (2021).
50. L. C. van Laake, A. Comoretto, J. T. B. Overvelde, On the coexistence of pressure regulation and oscillation modes in soft hysteretic valves. *J. Fluids Struct.* **126**, 104090 (2024).

Acknowledgments: We thank all members of our Soft Robotic Matter Group for the invaluable discussions. We are grateful to N. Commandeur for technical support. **Funding:** This project has received funding from the European Union's 2020 ERC-STG under grant agreement no. 948132. This work is part of the Dutch Research Council (NWO) and was performed at the research institute AMOLF. **Author contributions:** Conceptualization: S.P. and J.T.B.O. Methodology: all authors. Investigation: S.P. and C.M.v.R. Data analysis: S.P. Visualization: S.P. and C.M.v.R. Supervision: J.T.B.O. Writing: S.P. and J.T.B.O. Review and editing: all authors. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Data underlying this work can be found via: https://zenodo.org/records/14196385.

Submitted 21 June 2024
Accepted 15 November 2024
Published 20 December 2024
10.1126/sciadv.adr2433