A Distributed Robot Garden System

Albert Author and Bernard D. Researcher

***Abstract*— Abstract will go here**

I. INTRODUCTION

Fig. 1. Robot Garden

A. *Related Work*

II. SYSTEM ARCHITECTURE

A. *Garden Hardware Design*

*1) **Overview**:* In this section, we introduce the overall hardware system of the origami-inspired robotic flower garden. The garden bed is 86cm tall and 170cm wide and has six rows and three columns for a total of 18 tiles. Tiles are 28x28cm. Two tiles in the center of the garden are empty and are reserved for a small pond and the other 16 tiles are full of robotic flowers (Fig. 2(a)). Each of the 16 operational tiles has eight holes for connectors (Fig. 2(b)).

One set of robotic flowers has eight primary mechanical and electrical components (Fig 3), which are as follows:

- Origami flower: Flowers are made of thin color papers or thin acrylic sheets, 0.2 mm in thickness. The designed flowers are manually cut and folded.
- Pouch motors: Pouch motors are soft pneumatic actuators used to actuate the blooming motion of petals. Their shape, dimension and patterns are programmed on a desktop computer and manufactured using a heat sealing method [1].
- Tubes and wires: Tubes and wires provide electrical signals and air pressure to the flowers. They are coated with green-colored liquid rubber, which is cured at room temperature for a day after application. The rubber coating increases the stiffness of the stem, and the flower and stem can maintain its pose.
- Connector: The connector has three main functions. First, it fixes the rubber-coated stem to the plate. Second, it connects the air pressure and electrical signals from the system below the tile to the air pouches and LEDs on the flower. Finally, it allows flowers to be interchangeable throughout the garden. This accommodates multiple configurations of flowers in the garden and allows a broken flower to easily be exchanged for a new flower (see the right flower in Fig. 3). Connectors are made by using a 3-D printer or rubber molds.
- Acrylic plate: The plate has eight holes to support connectors. The position of holes can be programmed using a CAD tool and rapidly manufactured by using a laser cutter.
- Pneumatic and Electrical System: Each tile has one pneumatic and electrical setup to actuate the pneumatic pouches and LEDs inside the flower. Details about this setup are introduced in section II.A.3.

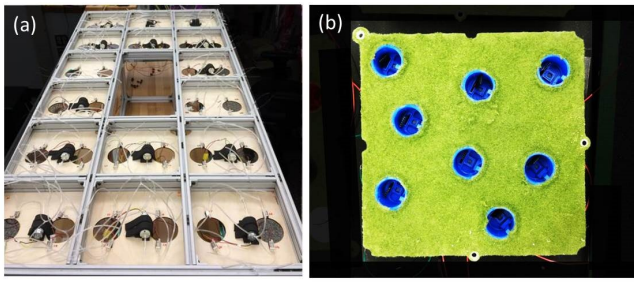

Fig. 2. (a)The robotic garden bed consists of 16 identical modular tiles. (b) Top view of a modular tile

The above components are mass-producible using a laser cutter, a 3-D printer, a computer-controlled heal sealing machine, and molds. Thus, we made 100 origami flowers, pouches, and connectors and 16 tiles and pneumatic control systems.

*2) **Pouch Actuated Flowers**:* The robotic flowers exhibited in the garden are all actuated by pouch motors. We made eight types of flowers, seven made by folding origami paper and one made with an acrylic sheet and decorated with LEDs. The flower types are as follows: robot tulip, robot lotus, robot lily, robot spiral flower, robot bird of paradise, robot fireworks flower, robot clematis and robot LED flower. In order to facilitate different foldings and to achieve different blossom effects, pouch motors were tailored to the different designs and actuation mechanisms. The flower and pouch motor types are shown in Figure 4. The following list describes each type of flower:

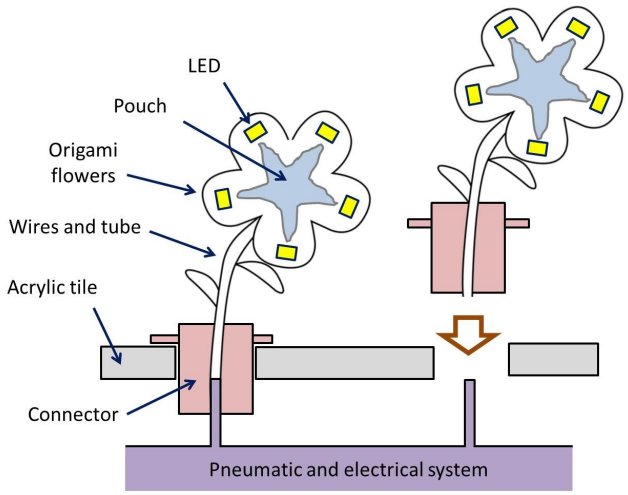

Fig. 3. A schematic figure (side-view) of a pouch-actuated robotic flower. Each robotic flower has one connector, wires, a tube, a printable and inflatable pouch, and some flowers have LEDs. The pneumatic and electrical system below the acrylic tile provide air pressure and electric signals to the pouch and LEDs in the flower. Flowers are interchangeable throughout the garden.

- Tulip: The tulip has a pouch hidden inside the side petals. When the pouch is inflated, the originally folded pouch straightens and opens up the flower.
- Lotus: Inside the lotus, a smaller square pouch is stacked on top of a larger square pouch. When they are inflated, they push up the stamen and pistils.
- Flower Lily: The flower lily has multiple pouches connected in a cross beneath the petals. Although the design looks like a linear mode pouch system as introduced by Niiyama et al. [1], it actually acts like a discrete rotational mode single pouch, because the whole system is adhered to the petal not just the end of each of the pouch stems. When pouches are inflated, the individual pouches on the pouch stem "bend" the adhered paper petals to open.
- Spiral Flower: Unlike other robot followers on which pouch motors were adhered onto folded origami structures, we took a very different approach to make Spiral Flowers. For each Spiral Flower, we folded 12 simple petals and adhered them onto a spiral shape pouch. When the pouch inflates, the length of the spiral changes and presents a twisting effect on petals.
- Bird of Paradise: The actuation mechanism on the Bird of Paradise is similar to the method used for the lotuses in that the pouches are stacked. In this case, the bottom of the individual pouches are bonded together. When they are inflated, the pouches separate the petals to open.
- Fireworks Flower: Like the flower lily, the pouch motors for the fireworks flower are hidden beneath the petals of the flower. The change in shape of the pouch during inflation pushes the flower open.
- Clematis: The clematis uses the same pouch motor design as the lotus. The stamen and pistils are pushed outward with pouch inflation.
- LED Flower: Unlike the other flowers, the LED flower is made with acrylic sheets. There is one LED on each petal. The blooming motion of the petals is actuated by rotational mode single unit pouch motors, discussed in detail by Niiyama et al. [1]. The wires connecting the LEDs are wrapped around the stem and attached to the plug connector.

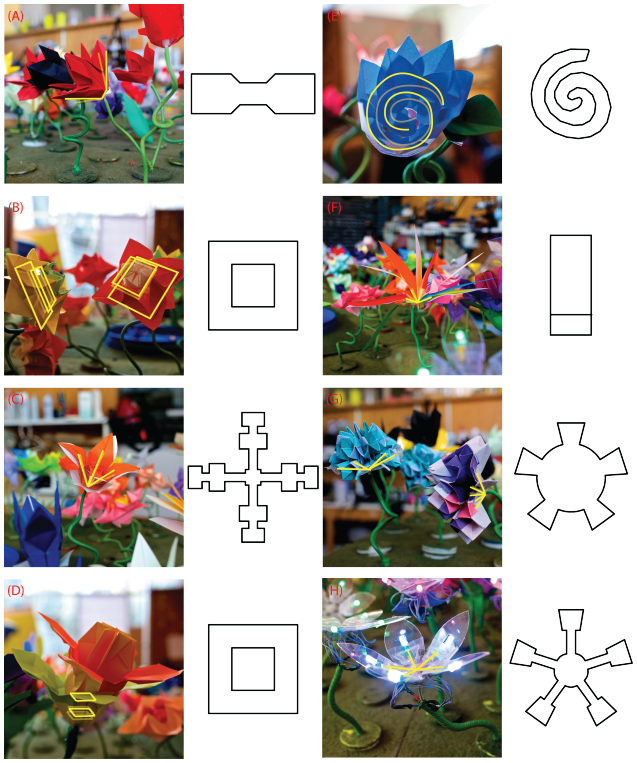

Fig. 4. Pictures of inflated flowers and corresponding pouch motor designs. The locations of the pouches are marked in yellow. (A)Tulip (B) Lotus (C) Flower Lily (D) Clematis (E) Spiral Flower (F) Bird of Paradise (G) Fireworks Flower (H) LED Flower

*3) **Computer-Aided Pouch Motor Design and Fabrication**:* We developed a pouch-motor manufacturing method to allow rapid design, fabrication, and iterative design modification [2]. The detailed process is as follows: The 2D pouch motor CAD design is converted into Numerical Control (NC) codes. Then the NC codes are sent to a custom-made CNC machine to make pouch systems. This machine "draws" pouch patterns onto two layers of 4mm thick polyethylene film simultaneously using a heated soldering iron [2].

In the case of Lotus and Bird of Paradise, the pouches are stacked and interconnected. Thanks to the versatile fabrication process, these multilayer pouch motors can be made simply layer by layer. We added four alignment features on the CNC machine mounting board. We made the multilayer pouch by firstly making the "air doors" in between the neighbor pouches, and then thermally bonded the top pouch with fiberglass separating it from the bottom one.

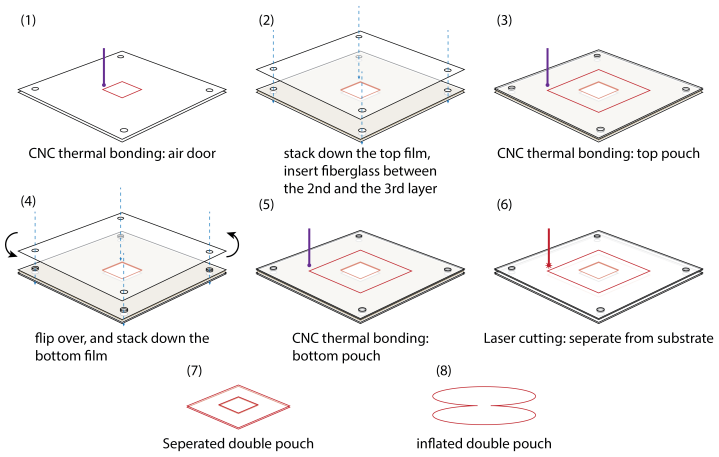

Fig. 5. An illustration of the fabrication method of a two-layer pouch motor

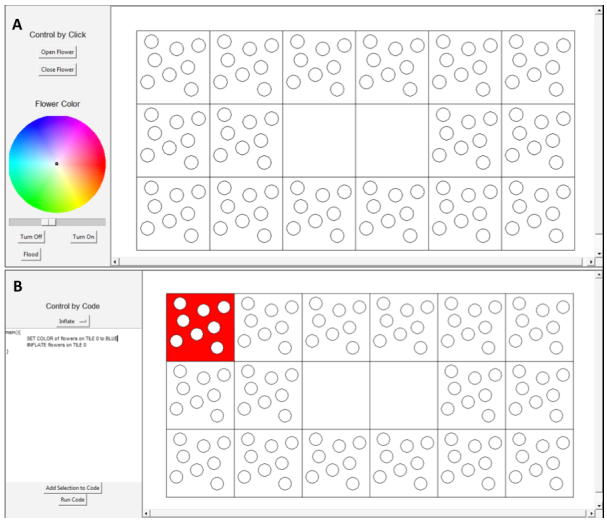

Fig. 6. Graphical User Interface (A) "Control by Click" component (B) "Control by Code" component

B. *Software*

*1) **Overview**:* The computational hardware used in the garden allows each of the 16 tiles to support up to eight flowers with LEDs and pouch motors. There are four pouch motors per tile that each inflate two flowers at a time. Each tile in the garden is controlled using an Arduino Mega2560 microcontroller equipped with additional custom boards designed to service all flower pumps and LED connections. Each Arduino board is connected to a Bluetooth chip to allow Bluetooth communication between the tile and a computer running a Graphical User Interface developed in Python. The PyBluez library (cite this) for Python was used for communication between the computer and the Bluetooth devices on each tile. Additionally, the Arduino boards are connected in a wired mesh network to allow direct communication between tiles. Algorithms have been developed for tile addressing and communication and to demonstrate distributed behaviors in the garden.

*2) **Computational Hardware**:* The hardware components required for computation and control of the garden include the following for each tile: an Arduino Mega2560 board, two custom printed circuit boards, a Bluetooth chip, and wired connection to pumps, LEDs and to other Arduino boards. In addition, a power supply is required to provide power to all Arduinos and pumps and a computer that can run Python is required for external control of the garden using the GUI.

The computational hardware used for the robot garden is extensible. The custom printed circuit boards allow the Arduino to service additional flowers and different types of flowers, so hardware can be reused if there are design changes in future versions of the garden.

*3) **Graphical User Interface**:* A Graphical User Interface was developed in Python so that inexperienced users can easily control the garden. The Graphical User Interface shows the layout of the garden and has two components: the "Control by Click" component and the "Control by Code" component. The two components of the GUI are shown in Figure 6.

The "Control by Click" component has buttons that allow the user to open and close the flowers using the pouch motors, a color wheel that allows the user to select a color for LEDs on the flowers, and buttons to turn the LEDs off and on. Additionally there are buttons that allow users to demonstrate distributed behaviors throughout the garden. Users can select one or multiple flowers or tiles to control and then select a command on the interface, and the selected tiles or flowers will execute the provided command.

The "Control by Code" component of the GUI allows users to select a flower or tile and a command from a drop-down menu. Users then add their selected tile and command to the text using the "Add Selection to Code" button, and can run the commands they have chosen in sequence using the "Run Code" button.

*4) **Mesh Network and Addressing**:* Control of the garden through the GUI requires communication between a computer acting as a master device and all tiles in the garden acting as slave devices. Although each tile supports Bluetooth communication, a maximum of seven slave devices are available per Bluetooth master device (cite Bluetooth Core Specification here), so use of Bluetooth for all 16 tile connections was not possible. As a result, we decided to establish one Bluetooth tile connection to the computer and to create a mesh network including the remaining tiles in the garden using wired serial connections for information distribution. Each tile in the garden is a node in the mesh network and is connected to the neighboring tiles directly above, below, to the right and to the left of itself in the garden grid structure, shown in Figure 7. The two center tiles were not included in the network to allow a pond or other robots to be displayed in the center of the garden. A modified version of the SoftwareSerial library (citation) for Arduino was used to facilitate communication between Arduino boards in the network.

To allow tile-to-tile communication and relay of com-

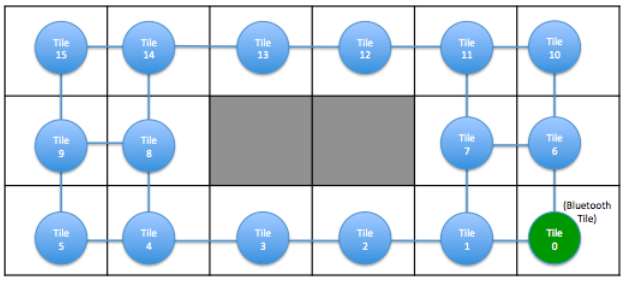

Fig. 7. Garden Grid Structure

mands throughout the garden, an addressing scheme was developed so each node in the network was assigned a unique address. Each individual tile has information about the topology of the garden mesh network, and tiles self-address accordingly. Algorithm 1 shows the self-addressing algorithm that was used.

Algorithm 1 Self Addressing

```
1:  procedure ASSIGN EACH TILE A UNIQUE ADDRESS
2:      currentTile ← address of tile running the algorithm
3:
4:      loop:
5:      Ping neighboring tiles for heartbeat
6:      if currentTile does not have an address then
7:          if no neighbors right or below currentTile then
8:              currentTile ← Address 0
9:          else
10:             Ping all neighbors for address of currentTile
11:     else
12:         Send appropriate addresses to neighboring tiles
```

*5) **Arduino-to-Arduino Communication**:* When a specific flower or tile and a command are selected using the GUI, the command is routed through the Bluetooth-connected tile to the appropriate tile with minimum hops in the mesh network. Each tile has information about the layout of the garden and can determine the next tile in the shortest path to the goal tile. Algorithm 2 shows the command routing algorithm that was used.

III. SYSTEM PERFORMANCE

We evaluated the performance aspects of the garden including power requirements for the whole garden, each tile, and each flower, total time required for self-addrssing, total messages sent and received per tile for self-addressing, and total time required to send a command to each tile. The following tables show our results.

Put tables here.

What conclusions can we draw about these things...

IV. DISTRIBUTED ALGORITHMS

The robot garden can be used to demonstrate distributed algorithm behavior and depict graph traversal algorithms in a visually pleasing way. Distributed behaviors are visualized by inflation and deflation of flowers on different tiles and by

Algorithm 2 Command Routing

```
1:  procedure ROUTE COMMAND TO CORRECT TILE AND
    CARRY OUT COMMAND
2:      serialData ← command and goal tile data
3:      currentTile ← address of tile running the algorithm
4:      commandTile ← goal tile for command
5:
6:      loop:
7:      Listen to serial ports for incoming messages
8:      if serialData received then
9:          if currentTile = commandTile then
10:             Carry out command
11:         else
12:             if currentTile knows its address then
13:                 Send serialData to next tile in shortest path
14:             else
15:                 Send serialData to all neighbors
```

the changing colors of LED-lit flowers. We implemented a flooding algorithm and a graph coloring algorithm in the garden as examples. Two buttons in the GUI, a "Flood" button for the flooding algorithm and a "Graph Coloring" button for the graph coloring algorithm, allow users to observe the behaviors of these algorithms throughout the garden.

When the "Flood" button is pressed in the GUI and the flooding algorithm is called, a command is first sent to the Bluetooth-connected tile to trigger the start of the flooding behavior. On the first tile, all flowers are inflated, and the tile sends the "flood" command to all of its neighbors. When the neighbors receive the command, they inflate all of their flowers and pass the command on again. A boolean value is used to keep track of whether or not a tile has already received a flooding command to prevent infinite looping in cycles in the garden. After 30 seconds, enough time for the algorithm to traverse the garden, the boolean value is changed back to its original value, allowing the tile to receive the "flood" command if it is sent again.

When the "Graph Coloring" button is selected, the graph coloring algorithm is executed. To demonstrate the graph coloring behavior, we placed at least one LED flower on each tile, so LED color on each tile and distribution throughout the garden could easily be distinguished. The graph coloring algorithm does this...

V. EDUCATIONAL APPLICATIONS

The robot garden can act as a platform for art and education, particularly focused on teaching computational thinking. The garden allows users to see their commands or code running in the physical world, linking programming to the real world. This aspect of the garden and its colorful presentation might be especially beneficial in getting young children excited about learning to program and think computationally. Using the garden could help young children to start thinking about the prevalence and importance of code in their daily lives. At the same time, the garden allows children

to be creative in their designs and provides them a medium for creating a new type of art using programming concepts.

Through the ”Control by Click” portion of the user interface, users can either control the garden directly by clicking on flowers or tiles and choosing commands or they can click on one of the algorithm visualization buttons to see algorithms demonstrated in a visual way using actuation and changing of colors of flowers in the garden. This part of the GUI acts as a teaching tool for basic algorithm concepts. So far, one algorithm has been demonstrated in the garden, and by leveraging the capabilities of the flower design and infrastructure of the system, many other distributed and graph traversal algorithms can be depicted in interesting ways.

Through the ”Control by Code” section users are able to create art using basic programming concepts. Currently users select a tile or tiles and a command and press the ”Add to Code” button, and they can see their commands pop up in the text box in sequential order. Selecting ”Run Code” runs the commands they have chosen in sequence in the garden, demonstrating basic sequential programming concepts. In the future, we envision adding functionality to the ”Control by Code” section that allows users to use logical statements and looping in their garden code with the aim of teaching these additional programming concepts through garden use.

VI. FUTURE WORK

In addition to adding functionality to the GUI, we plan to develop curriculum materials that leverage the capabilities of the garden to teach computational thinking. We have already developed a versatile curriculum for middle school students that covers basic programming concepts and Finite State Machines. This curriculum has been adapted to both Lego Mindstorms robots (cite) and the MIT SEG robot (cite) We plan to adapt the curriculum to the robot garden by adding basic sensing to the tiles and using the GUI we have already created. In addition to using materials we already have, we will add a unit covering basic algorithms to the curriculum, making use of the visual algorithm demonstration capabilities of the garden. Finally, we plan to invite children from local schools to use the garden and provide feedback on the garden user interface and curriculum.

VII. CONCLUSIONS

APPENDIX

ACKNOWLEDGMENT

REFERENCES

[1] R. Niiyama, D. Rus, and S. Kim, ”Pouch motors: Printable/inflatable soft actuators for robotics,” in *Proc. of IEEE International Conference on Robotics and Automation*, 2014.

[2] R. Niiyama, X. Sun, C. Sung, B. An, D. Rus, and S. Kim, ”Pouch motors: Printable soft actuators integrated with computational design,” *International Journal of Robotics Research*, submitted.