Robot Garden

Albert Author and Bernard D. Researcher

***Abstract*— Abstract will go here**

I. INTRODUCTION

Fig. 1. Robot Garden

A. *Related Work*

II. SYSTEM ARCHITECTURE

A. *Hardware*

*1) **Overview**:*

*2) **Pouch Actuated Flowers**:*

- Actuation Type A
- Actuation Type B
- Actuation Type C

*3) **Connecting to the System**:*

B. *Software*

*1) **Overview**:* Each of the 16 tiles in the garden can 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 2.

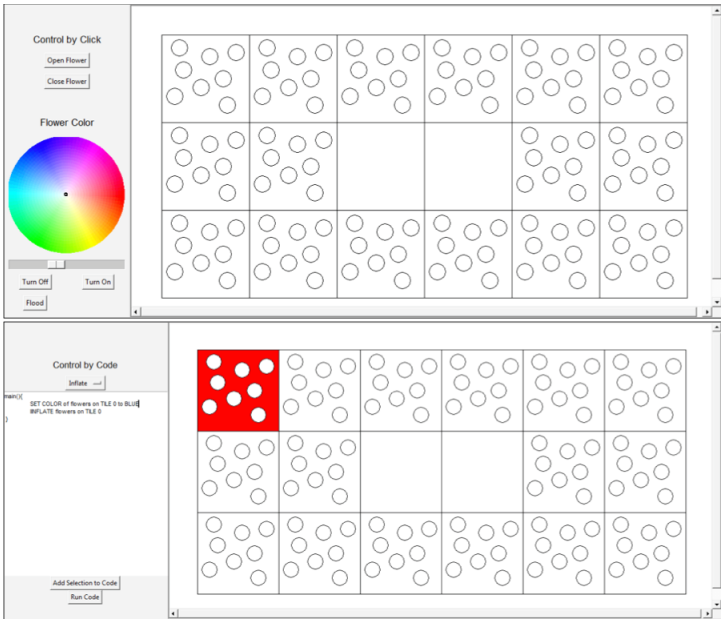

Fig. 2. Graphical User Interface

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 3. 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.

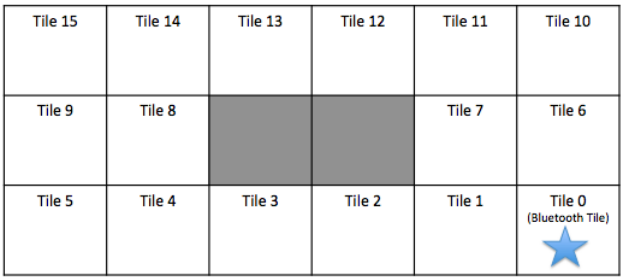

Fig. 3. Garden Grid Structure

To allow tile-to-tile communication and relay of commands 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.

*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. Algorithm 2 shows the command routing algorithm that was used.

Algorithm 1 Self Addressing

```
procedure ASSIGN EACH TILE A UNIQUE ADDRESS
    currentTile ← address of tile running the algorithm
    northNeighbor ← tile above currentTile in grid
    southNeighbor ← tile below currentTile in grid
    eastNeighbor ← tile right of currentTile in grid
    westNeighbor ← tile left of currentTile in grid

    loop:
    if currentTile does not have an address then
        if no eastNeighbor and no southNeighbor then
            currentTile ← Address 0
        else
            Ping all neighbors for address of currentTile
    else
        Send address currentTile+1 to westNeighbor
        Send address currentTile-1 to eastNeighbor
        Send address currentTile+6 to northNeighbor
        Send address currentTile-6 to southNeighbor
```

Algorithm 2 Command Routing

```
procedure ROUTE COMMAND TO CORRECT TILE AND CARRY OUT COMMAND
    serialData ← command and goal tile data
    currentTile ← address of tile running the algorithm
    commandTile ← goal tile for command
    myRow ← the row of currentTile in grid
    otherRow ← row of goal tile in grid
    northNeighbor ← tile above currentTile in grid
    southNeighbor ← tile below currentTile in grid
    eastNeighbor ← tile right of currentTile in grid
    westNeighbor ← tile left of currentTile in grid

    loop:
    listen to serial ports for incoming messages
    if serialData received then
        if currentTile = commandTile then
            carry out command
        else
            if currentTile knows its address then
                if myRow = otherRow then
                    if currentTile < commandTile then
                        send serialData to westNeighbor
                    else
                        send serialData to eastNeighbor
                else if myRow < otherRow then
                    send serialData to northNeighbor
                else
                    send serialData to southNeighbor
            else
                send serialData to all neighbors
```

III. SYSTEM PERFORMANCE

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 the changing the colors of LED-lit flowers. As an example, a flooding algorithm was implemented in the garden. A button in the GUI allows users to see the behavior of this algorithm 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 loops 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.

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 their 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 andcurriculum.

VII. CONCLUSIONS

APPENDIX

ACKNOWLEDGMENT