ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

# Display of Ordinary Differential Equation

**Team #5**

KEJIA HU
(kejiahu2@illinois.edu)
ZHUOHAO LI
(zhuohao5@illinois.edu)
QIANHE YE
(qianhey2@illinois.edu)
QIRONG XIA
(qirongx2@illinois.edu)

Advisor: Prof. Pavel Loskot
TA: Yue Yu

2024-03-27

## Contents

# 1 Introduction

## 1.1 Problem and Solution Overview

Understanding the behavior of complex systems in scientific and engineering fields often requires analyzing interactions between multiple variables over time and space. While three-dimensional (3D) visualization offers superior expressiveness and effectiveness in representing data information [1]. Modern software capabilities include sophisticated 3D visualization tools that not only enhance data comprehension but also facilitate interactive exploration and analysis [2]. However, the current screen-based 3D visualization tools are confined to the dimensions of the display device, hindering the viewer's ability to perceive objects accurately from various angles and distances. There's a pressing need for innovative visualization tools capable of representing complex systems in 3D in real-time, providing a more intuitive understanding of their behavior. Such tools could find applications in various disciplines, from mathematics to physics, biology, engineering, and environmental science.

To address the challenges posed by traditional visualization methods, our solution proposes the development of a portable and user-friendly 3D real-time visualization system for accurately representing time-varying 2D differential equations. The system will dynamically visualize the changing behavior of the function over time, projecting color onto the surface from the top to enhance interpretation. It comprises four subsystems: User Interface subsystem, Control subsystem, Mechanical subsystem, and Coloring subsystem. The User Interface subsystem collects the user's differential equation input and transmits it to the Control system. The Control subsystem converts this information for mechanical devices to understand and adjust the height of sticks, creating a smooth visualization surface. Finally, the Coloring Subsystem projects images onto the canvas. This solution not only improves data comprehension but also facilitates interactive exploration and analysis, offering researchers a comprehensive tool for understanding complex systems' behavior.

## 1.2 Visual Aid

Based on the solution we propose, a pictorial representation of our project is shown in Fig.1.

## 1.3 High-Level Requirement List

- The device should provide a user-friendly interface that enables users to input differential equations easily and view the status of computations and solutions on the screen clearly.
- The 3D visualization system must update dynamically and respond to changes in the differential equation solution, ensuring that users can observe the behavior of the system.

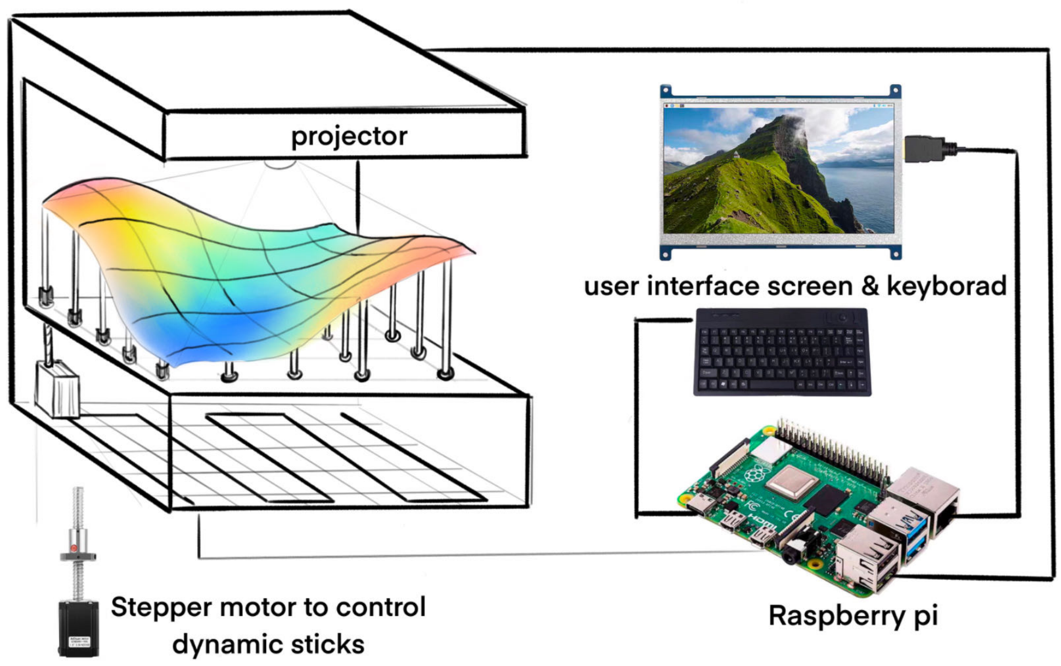

projector
user interface screen & keyborad
Stepper motor to control dynamic sticks
Raspberry pi

Figure 1: A pictorial representation of the device

- The entire system should be robust and reliable, capable of maintaining stable operation over extended periods. It should withstand environmental factors and variations in input conditions without compromising the accuracy or functionality of the visualization.

## 2 Design

### 2.1 Block Diagram

The block diagram of the device is shown in Fig. 2. There are four modules, representing four different subsystems respectively. The User Interface module enables the user to input differential equations and view the computation results. The Control module and Mechanical module ensure the sticks move to different heights based on the solution of the differential equation. The Coloring module will project color to the canvas.

Besides the block diagram illustrating the physical layout of our design, we also illustrate the overall logic workflow in Fig. 4.

### 2.2 Physical Design

As shown in Figure 4, the mechanical part of our design consists of 16 motors and the corresponding rods aligned on a 500*500mm board. More details are explained in the Mechanical subsystem part.

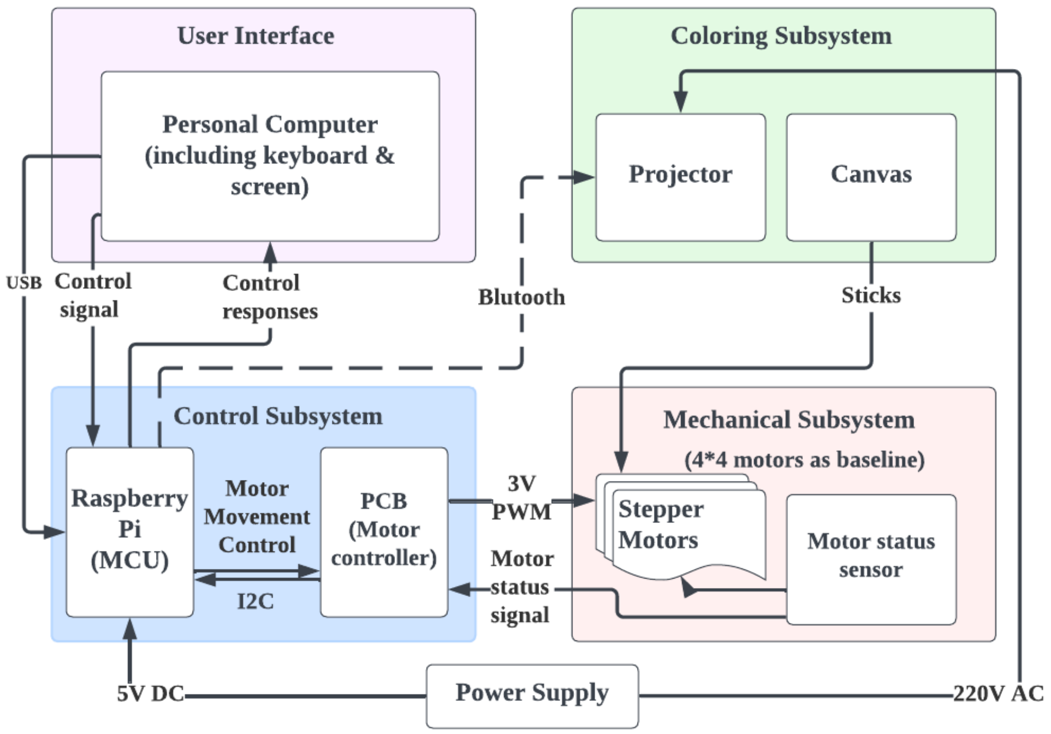


Figure 2: A pictorial representation of the device

## 2.3 User Interface Subsystem

The User Interface (UI) Subsystem is an essential component that bridges the gap between users and the visualization system, enhancing the interaction through a well-designed, user-friendly graphical interface. This subsystem is meticulously engineered to handle the input of Differential Equations, a critical task for users who need to convey complex mathematical models to the system for computation and subsequent visualization. At the core of this process lies the Graphical User Interface (GUI), which is thoughtfully equipped with various interactive elements. These elements include but are not limited to, text input fields designed for the precise entry of equations, buttons that facilitate the submission or cancellation of inputs, sliders that offer control over variable parameters, and a virtual keyboard to ensure users without access to a physical keyboard can still interact effectively. Upon the successful input of Differential Equations by a user, the system leverages the computing power of a Raspberry Pi, a compact yet powerful computing solution, to process and solve these equations.the Raspberry Pi transmits the data, including the time-varying solutions of the Differential Equations variables, to the Control Subsystem for visualiza- tion. Lastly, the Raspberry Pi will provide feedback on the screen regarding the status of the computation, such as progress indicators or error messages in case of invalid input or computation failures. See Table 1 for the requirement and verification.

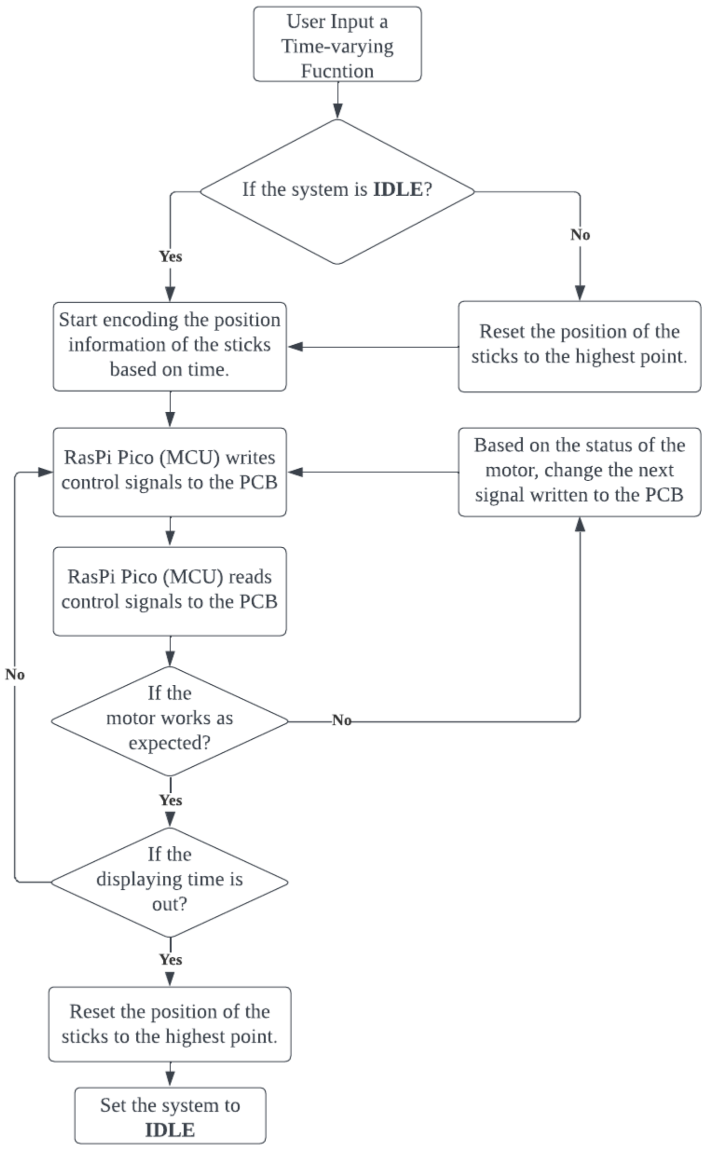

User Input a Time-varying Fucntion
If the system is IDLE?
Yes
No
Start encoding the position information of the sticks based on time.
Reset the position of the sticks to the highest point.
RasPi Pico (MCU) writes control signals to the PCB
Based on the status of the motor, change the next signal written to the PCB
RasPi Pico (MCU) reads control signals to the PCB
No
If the motor works as expected?
No
Yes
If the displaying time is out?
Yes
Reset the position of the sticks to the highest point.
Set the system to IDLE

Figure 3: The logic workflow

## 2.4 Control Subsystem

The Control Subsystem is in charge of the stick height adjustments. It translates solutions of Differential Equations received from the User Interface Subsystem into actionable control commands for the Mechanical Subsystem. This involves mapping the mathematical solutions to physical actions, such as the movement of the sticks.

To ensure the successful completion of the control subsystem, we think of two methods to control 16 motors with limited IO pins. The baseline method is to use two Raspberry Pi (one is Raspberry Pi Pico W and the other is Raspberry Pi Pico) to communicate with each other. The two development boards are connected to each other using UART channel 0, with the TX pin connecting to the RX pin on the other board. To ensure that the control data is in order, we build an extra abstract layer upon the physical layer of data

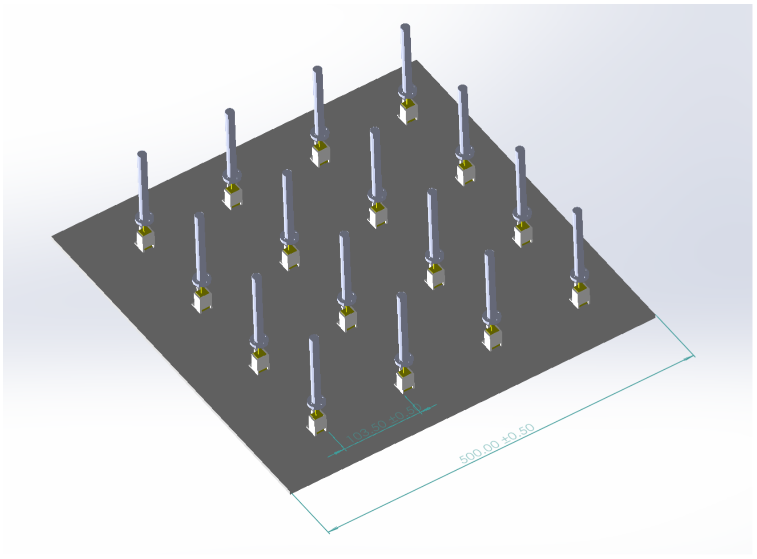


Figure 4: The mechanical part of design

transmission, by introducing a workqueue for each motor. We currently set the Baud rate to 9600Hz.

A more advanced way to achieve the same effects is to use a Raspberry Pi Pico as the central microcontroller unit (MCU), and a PCB serves as the motor controller. The MCU and the PCB are communicating using $I^2C$ protocol, by setting the $I^2C$ expander chip MCP23017 on the PCB as the slaves, and the MCU as the host system. The MCU has two $I^2C$ channels, each can take multiple slave chips, sending commands parallelly from the MCU. However, only two motors can take the control signals simultaneously due to the sequential essence of the $I^2C$ protocol. According to the datasheet of MCP23017, we designed the byte flow when transmitting data from the MCU to the PCB in the physical layer, which is shown in Fig.6. The *DataIn* and *DataOut* in the Fig.6 signals are sent as a byte, controlling 4 motors' movements for every data transmission to the MCP23017 chip.

The hardware architecture of the MCP23017 chip is shown in 5. There are 16 GPIO pins on this chip in total, divided into two groups. The GPIO-A is set as output latches, which will directly control four motors. The GPIO-B is set as input latches, which will be written by the sensors, and read by the host MCU. In that way, the MCU can check the status of motors, ensuring that every motor is working as expected. An incomplete circuit schematic is given in 7 for better illustration. For software development, we will use MicroPython for prototyping, and use open-source C/C++ SDK [3] to control the actuators based on received data and feedback from sensors. An example code of driving $2\times2$ sticks to represent the heat equation is given in Appendix C.

See Table 2 for the requirement and verification.

| **Requirement** | **Verification** |
|---|---|
| The GUI should be well-designed and include components like text input fields, buttons, and sliders for user interaction. This interface should be intuitive and user-friendly to allow users to easily input Differential Equations. | Conduct a thorough inspection of the GUI to ensure that all the specified components (text input fields, buttons, sliders) are present and organize a series of usability testing sessions with participants who represent the end-users of the system. |
| The subsystem should be designed for reliability, including robust error handling and validation of user inputs to prevent crashes and ensure accurate computation results. | Implement tests that intentionally input erroneous data or create scenarios where errors are likely to occur to see how the system handles these situations |

Table 1: R&V table for User Interface Subsystem

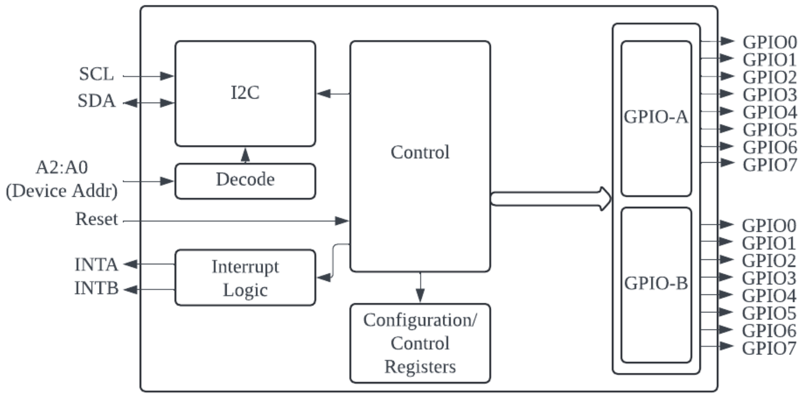


Figure 5: The hardware architecture of MCP23017 chip

## 2.5 Mechanical Subsystem

The Mechanical Subsystem consists of rods that can move up and down dynamically in accordance with the displayed solution. Each rod is connected to a motor, which is fixed in a base and aligned on a board, forming a 5*5 grid. The stepper motor and a screw were used to transfer rotation motion into translation, and the base was created by CAD modeling and 3D printing. The detailed design is shown below. A layer of rubber foil is secured to the top of each rod, ensuring the visualization remains seamless. Inputs from the control subsystem dictate the movements of this subsystem, which then executes these commands. Additionally, it provides feedback to the control subsystem for adjustments and fine-tuning. The system is designed to adhere precisely to the commands from the control system, aiming for high accuracy and precision in movements. Moreover, it's built to be sturdy, effectively reducing minor vibrations and environmental noise. Fig 8 shows the mechanical design of one translating structure. See Table 3 for the requirement

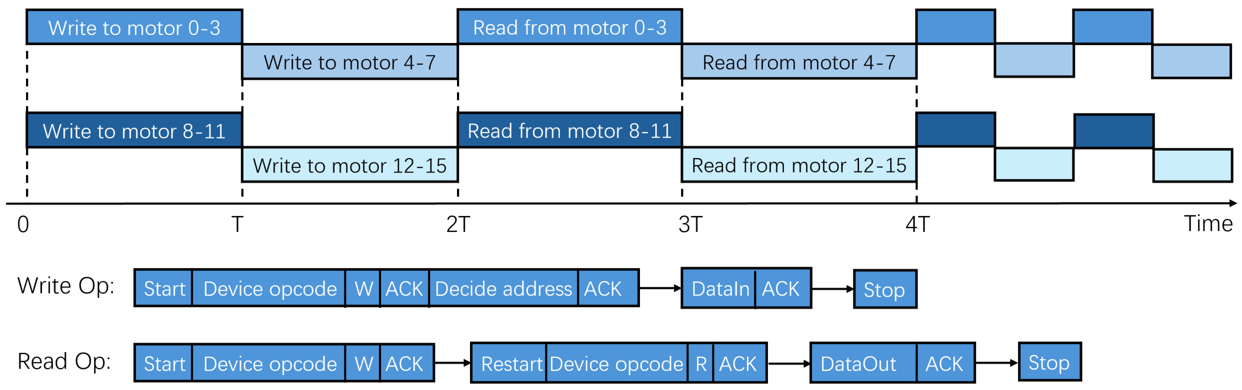


Figure 6: The byte flow in the physical layer

and verification.

| **Requirement** | **Verification** |
|---|---|
| The stick and the connection of which with the motor must be robust to support the canvas without noticeable wobbling during and after the motion. | Do optimization of the 3D model and test the motor motion when connecting all parts to the motor. |
| The motors should act accurately and nearly identically according to the signal input from the control subsystem. | Connect the motors to the power supply and measure if they move identically. If not, attach sensors to give feed back about position directly. |
| The canvas needs to be elastic and resistant to tearing, while reflecting the position of different sticks accurately. | Test different materials and choose the most suitable one. |

Table 3: R&V table for Mechanical Subsystem

## 2.6 Coloring Subsystem

The Coloring Subsystem is on the top of the device and it gets the solution of the Differential Equations from the Raspberry Pi. Given the height data, the subsystem employs logic to assign colors based on the height of the sticks. For example, the highest sticks could be assigned to the color red, while the lowest points are assigned to the color blue. As the height of the sticks changes, the subsystem must dynamically adjust the projections in real time to reflect these changes. This is achieved by having a tiny projector connected to the Raspberry Pi, taking the time-varying solutions of the Differential Equations, and coloring the surface with the time and height of the sticks changing. We plan to implement this feature at the end since it needs to synchronize accurately with the mechanical part. The implementation of this subsystem will greatly enhance the 3D visualization and

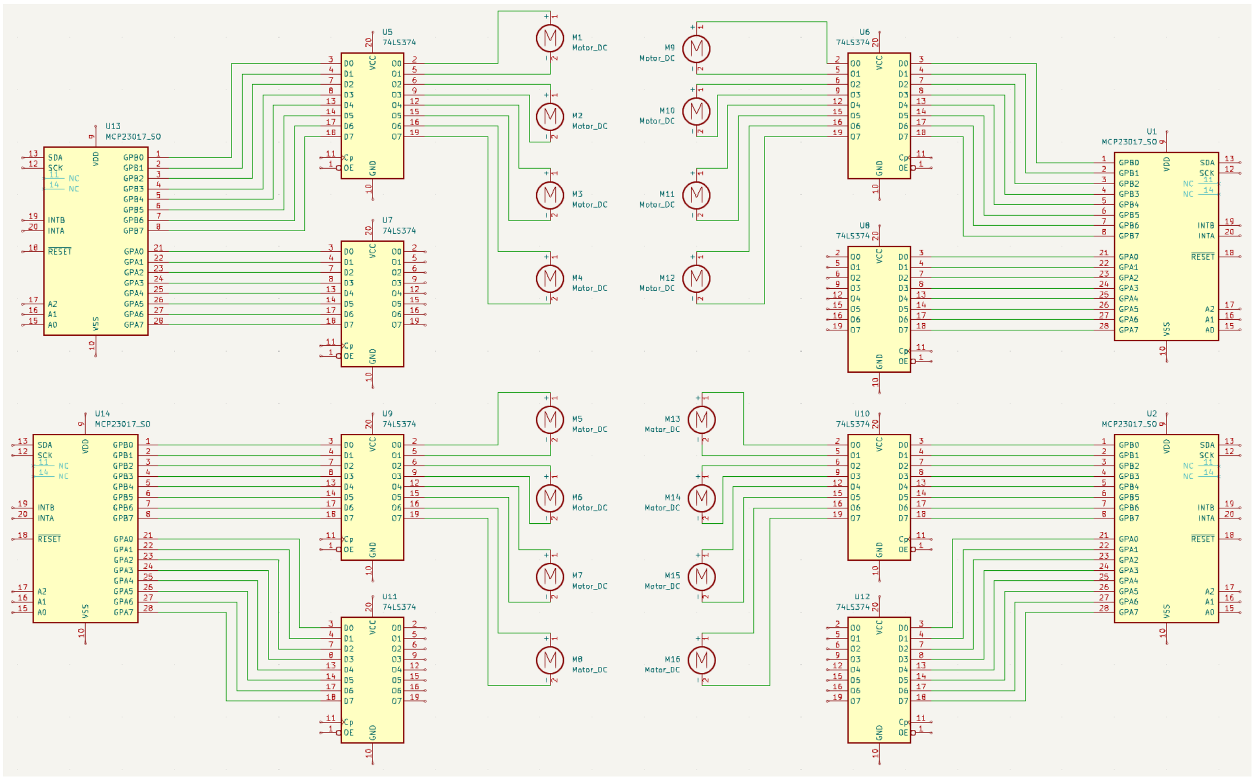


Figure 7: The circuit schematic

enable observers to easily discern patterns within the solution, leading to more informed insights. An example of what the Coloring Subsystem will be projecting for the case of the heat equation is shown in Fig 9. See Table 4 for the requirement and verification.

## 2.7 Point Summary

The point allocation is shown in Table 5.

## 2.8 Tolerance Analysis

### 2.8.1 Timely responses of the motor

The responsiveness of the motors to the microcontroller influences the visual impact significantly. To achieve precise vertical movement, we adopted stepper motor linear actuators for each dynamic stick. In Appendix B, we delve into a detailed simulation of a heat wave function solver. With the sticks' vertical range set at 10 cm, we can tailor the time-varying function to an optimal visual spectrum by pinpointing its highest and lowest values and calculating the median for each display horizon.

Our control over stick movement is constrained to directing them via the motor's poles, and we can use the duty cycle of the Pulse Width Modulation (PWM) signals to adjust their speeds. We modeled stick motion as uniform linear motion, but as depicted in Fig-

| **Requirement** | **Verification** |
|---|---|
| The Raspberry Pi and PCB will be used to control 16 stepper motors, each of them given a stable 3.3V voltage, allowing it to achieve a 500 RPM (Revolution Per Minute) rotation rate. | Use an oscilloscope to measure the voltage output from the PCB and stepper motor drivers |
| The Control Subsystem is responsible for synchronizing the surface coloring and the movement of the sticks. That said, every time the microcontroller sends the signals to the sticks, it should also send the generated coloring image to the projector simultaneously. | Send signals to move the sticks and observe if the coloring image is synchronized with the stick movements. Then capture images of the projected surface using the camera simultaneously with stick movements. |
| The Control Subsystem must ensure that the height adjustment of each stick remains within 0.3cm of the true value. | Use the sensor to get the position of the sticks, then compare the measured height with the reference value. |

Table 2: R&V table for Control Subsystem

ure 10, the distance traveled varies between frames with the same time interval. Consequently, we propose two major constraints for determining the device's frame rate:

1. The maximal stick's achievable moving velocity.
2. The maximal time interval that the microcontroller dispatches the motion commands to the motor.

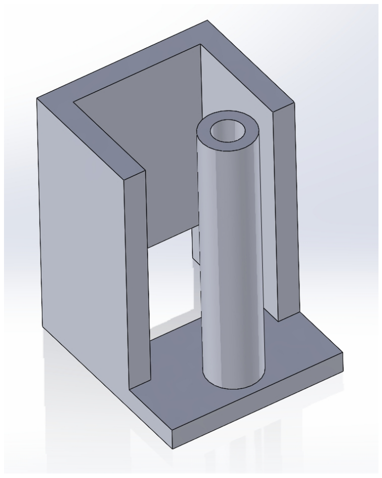
A reasonable breakdown of the working period of the motor consists of (1) the screw performing rotation, and (2) waiting for the next moving signal sent from the microcontroller. However, one working period can consist of multiple periods of data transmission between the MCU and the PCB, and the granularity of the motion control is determined by the clock frequency of data transmission.

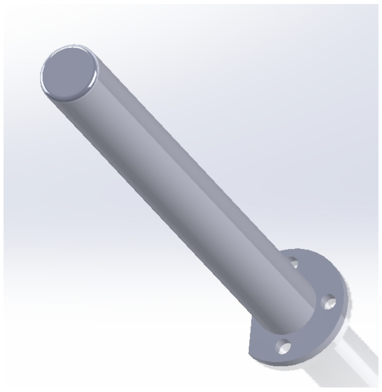
The RPM of the stepper motor shaft is directly related to the linear speed of the external nut (and consequently the stick attached to it) in a proportional manner. The screw lead is the distance the nut moves parallel to the screw axis when the screw makes one complete revolution, with unit mm/rev. The formula to calculate the linear speed $V_{linear}$ (in unit mm/min) based on the motor's RPM and the screw's lead $L$ is:

$$V_{linear} = RPM \cdot L \tag{1}$$

According to the actual measurement, the screw lead is 0.6mm/rev, and the linear speed of the stick is:

$$V_{linear} = 500RPM \cdot 0.6mm/rev = 300mm/min \approx 0.5cm/s$$


(a) Motor base


(b) Rod

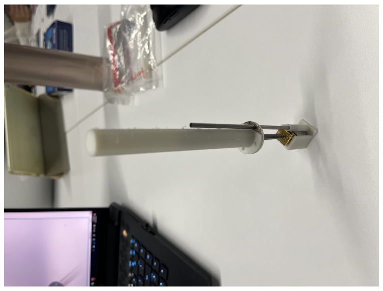

(c) Assembled translating structure

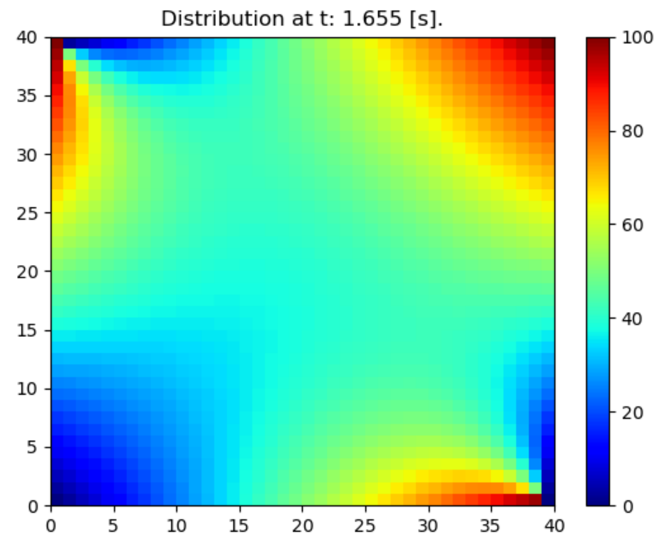
Figure 8: Mechanical design of one translating structure

| **Requirement** | **Verification** |
|---|---|
| The Coloring Subsystem must accurately project color onto the surface corresponding to the mathematical solutions of the differential equations. | Compare the projected color with the Matlab Simulation results. |
| The Coloring Subsystem must maintain synchronization with the movement of the sticks controlled by the Control Subsystem. | Test the system while simultaneously adjusting stick heights and projecting colors. Verify that the projected colors remain synchronized with the movement of the sticks without noticeable lag or inconsistency. |

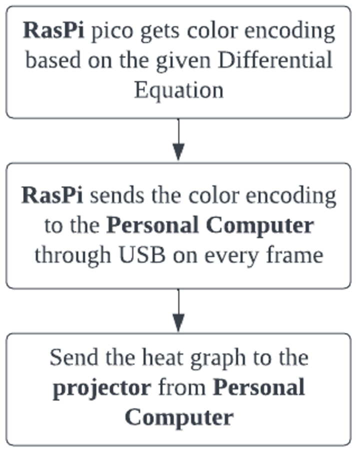
Table 4: R&V table for Coloring Subsystem

Therefore, given that the maximum travel distance per work cycle is set to 0.5 cm, the minimum duration of a work cycle must be $t_{motor} = \frac{0.5cm}{0.5cm/s} = 1s$. Since the time taken by the microcontroller to transmit signals to the motor is negligible relative to the movement latency of the actuators given a reasonable clock frequency, a time interval of 1 second is sufficient to guarantee the device's full operational capability.

We conducted some experiments to investigate the influence of varying duty cycles on the time it takes for a motorized stick to reach its highest position. By altering the PWM signals applied to the motor, we aimed to discern how changes in signal intensity affect the speed of the motor's movement. Fig 11 shows the relationship between the duty cycle and the time the motor takes to make a stick to reach the highest position. As we can see, the larger the duty cycle, the less time it takes. However, as the duty cycle drops to $85\%$, the motor fails to drive the sticks.



(a) Example of the projected color for the heat function


**RasPi** pico gets color encoding based on the given Differential Equation
**RasPi** sends the color encoding to the **Personal Computer** through USB on every frame
Send the heat graph to the **projector** from **Personal Computer**

(b) The data flow of coloring subsystem

Figure 9: The illustrations of coloring subsystem

### 2.8.2 Synchronization between the sticks and the projection

Achieving precise synchronization between the movement of the sticks within the Mechanical Subsystem and the color projection in the Coloring Subsystem is critical for accurate 3D visualization. Any inconsistency or delay in this synchronization could result in misleading visualizations of the differential equation solution. Therefore, we need to ensure that the movement of the sticks and the projection of colors occur within an acceptable time. Assume that the mechanical movement of the sticks takes $t_m$ seconds to complete, and the coloring subsystem requires $t_c$ seconds to project the colors onto the surface. Through testing and adjusting the RPM of the sticks, we try to minimize the time difference $|t_m - t_c|$ to less than $0.5$ seconds.

### 2.8.3 Accuracy of the numerical solutions

Many of the Differential Equations do not have an explicit solution. Hence, we need to use a numerical method to get an approximate solution. To ensure the solution solved by a numerical method is similar to the true solution, we need to perform a tolerance analysis on the difference between the two.

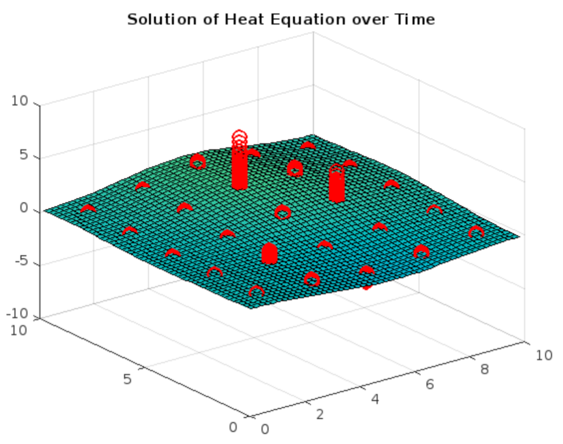
Let's denote:

- $u(x_i, y_j, t_k)$ as the true solution of the Differential Equation at each grid point.
- $u_{\text{num}}(x_i, y_j, t_k)$ as the solution obtained numerically in Python at each grid point.

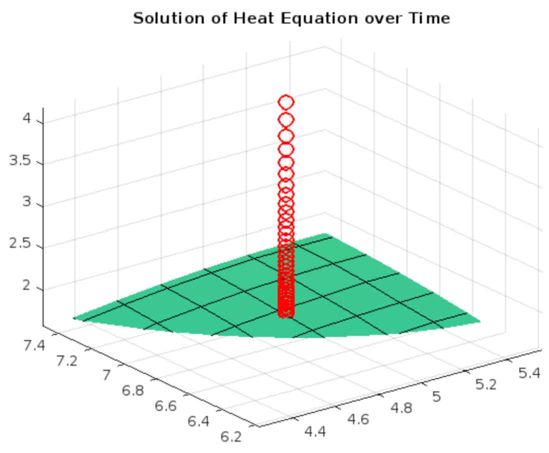
We can measure the difference between the true solution and the numerical solution us-

| **Module Name** | **High Level Requirement** | **Point** |
|---|---|---|
| User Interface Module | User can input a differential equation and our device can solve it | 10 |
| Control Module | The sticks can move to different heights, representing the differential equation solution | 20 |
| Mechanical Module | The canvas should be elastic enough to represent a smooth surface | 10 |
| Coloring Module | Correct colors are projected to the canvas | 10 |
| | **Total** | 50 |

Table 5: Point Summary Table



(a) The trajectories of 25 sticks



(b) The trajectory of a single stick

Figure 10: The trajectories of sticks

ing a suitable metric such as the L2-norm. The error at each grid point $(x_i, y_j, t_k)$ can be calculated using the absolute difference between the true solution and the numerical solution:

$$e_{i,j,k} = |u(x_i, y_j, t_k) - u_{\text{num}}(x_i, y_j, t_k)| \tag{2}$$

Then, the $L^2$ norm of the error over the entire grid can be approximated by summing the squared errors at each grid point and taking the square root:

$$\|e\|_2 \approx \sqrt{\sum_{i,j,k} e_{i,j,k}^2} \tag{3}$$

where the sum is taken over all grid points $(x_i, y_j, t_k)$.

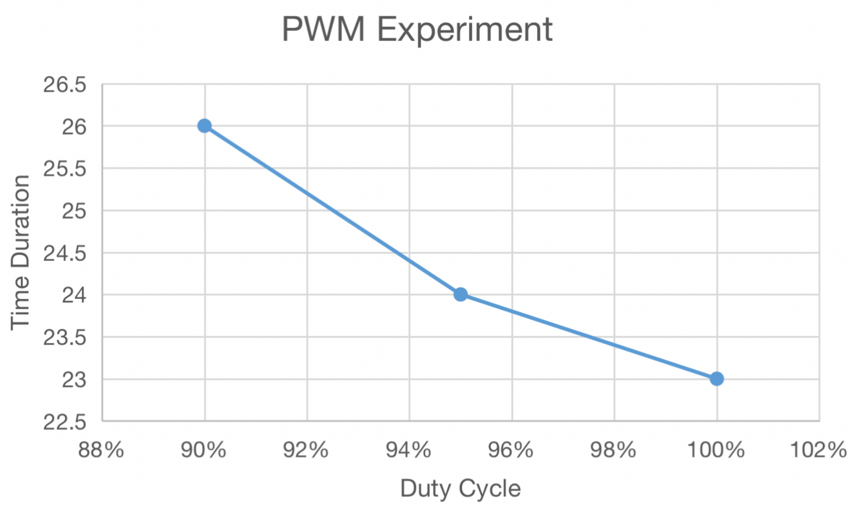


Figure 11: Relationship between the duty cycle and time duration

When we implement the Differential Equation solving algorithm, we need to calculate the error difference based on some Differential Equations that we know the true solutions. We need to ensure the $\|e\|_2 \leq T_{\text{tol}}$, where $T_{\text{tol}} = 10^{-2}$.

## 3 Cost and Schedule

### 3.1 Cost Analysis

| Components | Vendor | Quantity | Cost (RMB)/unit | Total Cost (RMB) |
|---|---|---|---|---|
| DC Motor | YongChuangXin Actuator | 17 | 22 | 374 |
| Raspberry Pi Pico | Raspberry Pi | 1 | 29 | 29 |
| Raspberry Pi Pico W | Raspberry Pi | 1 | 59 | 59 |
| I2C Expander Chip | Microchip | 4 | 9.5 | 38 |
| PCB | Custom | 1 | 30 | 30 |
| Tiny projector | Hantangke | 1 | 200 | 200 |
| Canvas | Shengshi Textile and Leather | 1 | 6.5 | 6.5 |
|  |  |  | **Total** | 736.5 |

Table 6: Cost for components

#### 3.1.1 Labor

According to public statistics of the average salary for a graduate from Illinois ECE [4], the average salary per hour is roughly $45/hour, which is 315 RMB/hour. Assuming that we are given 8 weeks to finish the project, every member works for at least 20 hours per week, the total number of hours for each member is given by

$$20 hours/week \times 8 weeks = 160 hours$$

Multiplying the total number of hours worked by the hourly rate gives the total labor cost for a person, which is

$$315 RMB/hour \times 160 hours = 50400 RMB$$

Therefore, the total labor cost for this project will be

$$50400 RMB/person \times 4 persons = 201600 RMB$$

#### 3.1.2 Parts

The cost for each component for our project is given in 6.

#### 3.1.3 Sum of costs into a grand total

The total cost given by summing the parts cost and labor costs

$$201600 RMB + 736.5 RMB = 202,336.5 RMB$$

### 3.2 Schedule

The weekly schedule is shown in Table.7.

## 4 Discussion of Ethics and Safety

Prioritizing safety in the creation and application of any product is paramount. To ensure "the safety, health, and welfare of the public" as outlined in IEEE's ethical guidelines[5], we should strictly adhere to relevant regulations throughout the research and development phases, as well as inform users about the proper usage and to communicate the potential risks with misuse.

Furthermore, to be forthright and grounded in reality when making claims or estimates based on the data at hand[5], we should make it clear that the visualized differential equation solution is an approximation. Despite aiming to mirror real-life situations as closely as possible, these solutions cannot replace actual real-world solutions.

The essence of design lies in simplifying life and enhancing work efficiency. It's crucial for society to aim for respect, inclusivity, fairness, and balance, guaranteeing that everyone can access the necessary tools and resources for a rewarding life, free from discrimination related to race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or expression[5]. Fundamentally, we are dedicated to providing individuals with an enhanced ability to interact with and learn from differential equations.

| Week | Task | Member |
|---|---|---|
| 3/25/2024 | Design a PCB that can (1) take signals from RasPi Pico, (2) send out signals to the motors. | Qirong Xia |
| | Work with Qirong to design the PCB and verify the correctness of the PCB | Kejia Hu |
| | Translate the user inputted time-varying function to control signals sending to the motors | Zhuohao Li |
| | 3D print and optimize the design of motor base | Qianhe Ye |
| 4/1/2024 | Build a prototype of the PCB and test it with breadboard | Qirong Xia |
| | Refine the control signal translation process based on initial testing results | Kejia Hu |
| | Adjust the range of solvable equations based on the limitations of the physical structure | Zhuohao Li |
| | Search for usable type of sensors | Qianhe Ye |
| 4/8/2024 | Get the PCB and make it soldered, incorporating all the components to allow end-to-end control | Qirong Xia |
| | Begin debugging any hardware issues or communication issues between the Raspberry Pi and motors | Kejia Hu |
| | Design the sampling intervals and sampling locations for the solutions | Zhuohao Li |
| | Implement sensor to the control system | Qianhe Ye |
| 4/15/2024 | Debug the hardware control subsystem if previous week's work is not done perfectly. | Qirong Xia |
| | Optimize the performance of the control subsystem to ensure smooth and accurate movement of the sticks | Kejia Hu |
| | Optimize the signals sent to the motor to make its operation more stable | Zhuohao Li |
| | Optimize 3D model to place sensor | Qianhe Ye |
| 4/22/2024 | Help find the right projector & connect the projector to the MCU | Qirong Xia |
| | Write code to make the projector project wanted colors | Kejia Hu |
| | Ensure that structures covered by canvas can still work. | Zhuohao Li |
| | Search canvas material | Qianhe Ye |
| 4/29/2024 | Debug the overall system to make sure it can work from end-to-end | Qirong Xia |
| | Address any synchronization issues between the control subsystem and other subsystems | Kejia Hu |
| | Help debug the system | Zhuohao Li |
| | Fix the mechanical system to the board | Qianhe Ye |
| 5/6/2024 | Prepare for mock demonstration. | Qirong Xia |
| | Conduct final debugging and testing of the entire system to ensure it is ready for the mock demonstration | Kejia Hu |
| | Prepare for mock demonstration. | Zhuohao Li |
| | Prepare for mock demonstration | Qianhe Ye |
| 5/13/2024 | Rehearse the final presentation and demonstration with the team | Qirong Xia |
| | Rehearse the final presentation and demonstration with the team | Kejia Hu |
| | Rehearse the final presentation and demonstration with the team | Zhuohao Li |
| | Rehearse the final presentation and demonstration with the team | Qianhe Ye |

Table 7: Weekly Schedule for team members

## References

[1] J. Wood, S. Kirschenbauer, J. Döllner, A. Lopes, and L. Bodum, "Using 3d in visualization," in *Exploring geovisualization*, Elsevier, 2005, pp. 293–312.

[2] A. R. Teyseyre and M. R. Campo, "An overview of 3d software visualization," *IEEE transactions on visualization and computer graphics*, vol. 15, no. 1, pp. 87–105, 2008.

[3] raspberrypi. "Pico-sdk." (2023), [Online]. Available: https://github.com/raspberrypi/pico-sdk (visited on 03/26/2024).

[4] U. E. Department. "Salary averages." (2024), [Online]. Available: Salary%20Averages (visited on 03/27/2024).

[5] IEEE. "Ieee code of ethics." (2020), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/05/2024).

[6] Matlab. "Simple heat equation solver." 2024-03-03. (2023), [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/59916-simple-heat-equation-solver.

[7] Younes-Toumi. "Heat equation." (2024), [Online]. Available: https://github.com/Younes-Toumi/YoutubeChannel/tree/main/Simulation%20with%20Python/Heat%20Equation.

## Appendix A Define and Solve 2D Time-Dependent PDE

Below is a pseudocode of how to obtain a partial differential equation (PDE) from the user and solve it using numerical methods.

**Algorithm 1** Solving 2D Time-Dependent PDE $z(x, y, t)$

```
1: **Input:** User-provided PDE
2: **Prompt User Input:**
3: $pde\_input \leftarrow$ Get input from the user in the format $'a\frac{\partial^2 z}{\partial x^2} + b\frac{\partial^2 z}{\partial y^2} + c\frac{\partial z}{\partial t} = f(x, y, t)'$
4: **Parse Input:**
5: Parse $pde\_input$ to extract coefficients $a$, $b$, $c$ and the function $f(x, y, t)$
6: **Spatial and Temporal Domain:**
7: Define spatial and temporal domain by generating arrays $x\_values$, $y\_values$, $t\_values$
   using desired parameters
8: **Initialize Solution Array:**
9: Initialize array $z\_values$ of size $(\text{num\_points\_x}, \text{num\_points\_y}, \text{num\_points\_t})$ with ze-
   ros
10: **Boundary and Initial Conditions:**
11: Set initial conditions for $z(x, y, t)$ at $t = 0$ using input initial condition function
12: **Time-stepping Loop:**
13: **for** $n$ **in** $[0, \text{num\_points\_t} - 1]$ **do**
14:     **for** $i$ **in** $[1, \text{num\_points\_x} - 1]$ **do**
15:         **for** $j$ **in** $[1, \text{num\_points\_y} - 1]$ **do**
16:             **Implement Numerical Scheme:**
17:             $z\_values[i, j, n + 1] \leftarrow$ Update rule for $z(x, y, t)$ at $(x_i, y_j, t_{n+1})$
18:         **end for**
19:     **end for**
20: **end for**
21: **Output:** Solution of the PDE $z(x, y, t)$
```

# Appendix B Matlab Animation of Heat Equation

2D time-dependent PDE appears a lot and has wide applications. One common PDE that exhibits time variation in two spatial dimensions is the heat equation. The heat equation describes how a quantity (such as temperature) changes over time in a given region, based on its spatial distribution and the rate at which it diffuses.

The heat equation in three dimensions (including time) is given by:

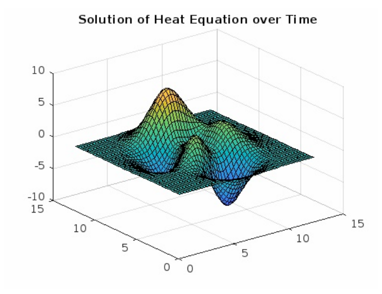
$$\frac{\partial u}{\partial t} = \alpha \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

Where:

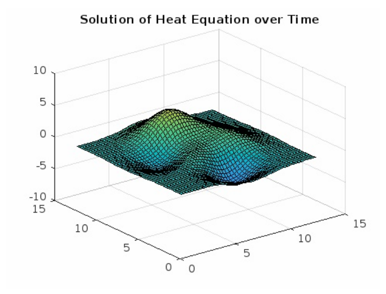
- $u(x, y, t)$ is the temperature (or another quantity) at position $(x, y)$ and time $t$.
- $\alpha$ is the thermal diffusivity coefficient, a constant that characterizes the material's ability to conduct heat.
- $\frac{\partial u}{\partial t}$ denotes the partial derivative with respect to time.
- $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial y^2}$ are the second partial derivatives with respect to $x$ and $y$respectively. These terms describe how the temperature changes along each spatial dimension.

This equation describes how the temperature $u(x, y, t)$ evolves over time due to the diffusion of heat in three dimensions. The spatial derivatives account for the change in temperature along the $x$ and $y$ directions, while the time derivative describes how the temperature changes over time.

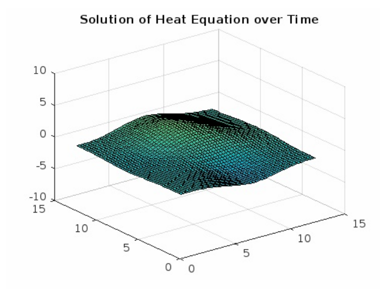
The animation depicting the solution of the heat equation in MATLAB [6], is illustrated in Fig 12.



(a) frame at t = 1



(b) frame at t = 12



(c) frame at t = 35

Figure 12: Animation of Heat Equation

# Appendix C Prototype code of driving 2 x 2 sticks

We leverage the idea from [7] to solve the differential equation numerically. And the following code shows how to drive 2 x 2 sticks to represent the heat equation.

```python
import time
from machine import Pin

# Define constants
a = 110
length = 20  # mm
total_time = 0.5  # seconds
nodes = 20

# Initialization
dx = length / nodes
dy = length / nodes
dt = min(dx**2 / (4 * a), dy**2 / (4 * a))

# Initialize the plate temperatures
u = [[20 for _ in range(nodes)] for _ in range(nodes)]

# Boundary Conditions
for i in range(nodes):
    u[0][i] = 10 * i / nodes
    u[-1][i] = 10 * i / nodes
    u[i][0] = 10 * i / nodes
    u[i][-1] = 10 * i / nodes

# Simulating
counter = 0

while counter < total_time:
    w = [row[:] for row in u]

    for i in range(1, nodes - 1):
        for j in range(1, nodes - 1):
            dd_ux = (w[i - 1][j] - 2 * w[i][j] + w[i + 1][j]) / dx**2
            dd_uy = (w[i][j - 1] - 2 * w[i][j] + w[i][j + 1]) / dy**2

            u[i][j] = dt * a * (dd_ux + dd_uy) + w[i][j]

    counter += dt

    avg_temp = sum(sum(row) for row in u) / (nodes * nodes)
    print("Temperature at [5,5], [5,15], [15,5], [15,15] is ", u[5][5],
        u[5][15], u[15][5], u[15][15])
    print("t: {:.3f} [s], Average temperature: {:.2f} Celsius".format(
        counter, avg_temp))

    # Updating the plot
    # time.sleep(0.5)  # Pause for better visualization

stick1 = int(u[5][5])
stick2 = int(u[5][15])
stick3 = int(u[15][5])
stick4 = int(u[15][15])
print(stick1)
print(stick2)
print(stick3)
print(stick4)

pin1_1 = Pin(0, Pin.OUT)
pin1_2 = Pin(1, Pin.OUT)

pin2_1 = Pin(2, Pin.OUT)
pin2_2 = Pin(3, Pin.OUT)

pin3_1 = Pin(4, Pin.OUT)
pin3_2 = Pin(5, Pin.OUT)

pin4_1 = Pin(6, Pin.OUT)
pin4_2 = Pin(7, Pin.OUT)

while stick1 > 0 or stick2 > 0 or stick3 > 0 or stick4 > 0:
    if stick1 > 0:
        pin1_1.value(0)
        pin1_2.value(1)
        stick1 -= 1
    else:
        pin1_1.value(0)
        pin1_2.value(0)

    if stick2 > 0:
        pin2_1.value(0)
        pin2_2.value(1)
        stick2 -= 1
    else:
        pin2_1.value(0)
        pin2_2.value(0)

    if stick3 > 0:
        pin3_1.value(0)
        pin3_2.value(1)
        stick3 -= 1
    else:
        pin3_1.value(0)
        pin3_2.value(0)

    if stick4 > 0:
        pin4_1.value(0)
        pin4_2.value(1)
        stick4 -= 1
    else:
        pin4_1.value(0)
        pin4_2.value(0)

    time.sleep(1)

# stop the motor
pin1_1.value(0)
pin1_2.value(0)
pin2_1.value(0)
pin2_2.value(0)
pin3_1.value(0)
pin3_2.value(0)
pin4_1.value(0)
pin4_2.value(0)
```