Snake 3D LED Tower

Diogo Chaves
up202004540
MEEC-FEUP
Contribuição-33%

Joana Pereira
up202004651
MEEC-FEUP
Contribuição-33%

Lucas Freitas
up202006938
MEEC-FEUP
Contribuição-33%

***Abstract*—This report presents the design, implementation, and evaluation of a 6x6x6 LED tower system controlled by a Raspberry Pi 4. The project uses three I2C expanders and two shifters for floor selection, the project achieves interactive visual effects, including a Snake game, rain, and fireworks. The report outlines the system architecture, task scheduling, and results, emphasizing the software design quality, execution, and alignment with course objectives. The implemented system demonstrates efficient task scheduling with 93% CPU utilization in the worst-case scenario, confirming the feasibility and responsiveness of the design.**

I. Introduction

This project explores real-time scheduling and operating system concepts, such as process and thread management, to implement a game and visual effects for a 6x6x6 LED tower. The system is powered by a Raspberry Pi 4 running Raspberry Pi OS and uses three expanders and two shifters to extend the microcontroller's input/output (I/O) pin capacity.

A snake game and various visual effects are executed through six distinct tasks, each defined by parameters like priority, period, and deadline. These values are used to assess CPU utilization and ensure the system remains schedulable. The expanders and shifters are crucial for enabling the LED tower's full functionality.

II. System Description

Context and Objectives

The LED tower provides an interactive medium for visual effects. The primary objectives are:

- **Modular Design:** Ensuring scalability and maintainability.
- **Efficient Scheduling:** Implementing periodic and one-shot tasks for responsive operation.
- **Interactive Features:** Enabling user-controlled gameplay and dynamic effects.

Hardware Specifications

- **Controller:** Raspberry Pi 4.
- **Expanders:** Three I2C expanders (MCP23017)
- **Shifters:** Two 74HC164 shifters for floor selection, configured in a cascaded (daisy-chained) arrangement. This configuration allows the outputs of one shifter to feed into the next, effectively expanding the number of output pins controlled by the microcontroller.
- **Buttons:** 12 buttons total, with 6 on the left side for gameplay and 4 on the right side for swapping between modes of operation.
- **Transistors:** Transistors are used to provide the correct power to the LEDs. Some of these transistors are connected to the MCP23017 expanders to light up the rows, while 6 transistors are connected to the 74HC164 shifters to light up the floors of the cube.
- **LEDs:** 216 LEDs are used. As only one layer is active at each time, the phenomenon of persistence of vision is utilized by rapidly flickering the LEDs. This causes the human brain to perceive the light as steady and continuous.

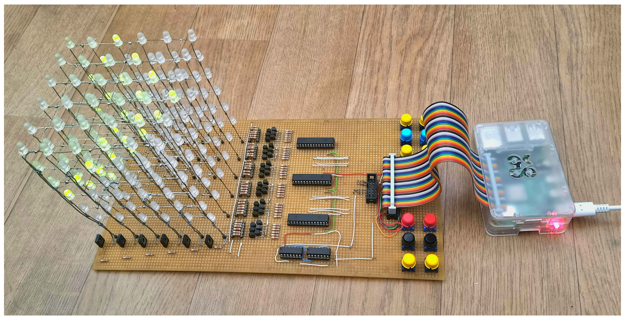
Fig. 1: LED Cube hardware overview

A pre-assembled system with expanders and shifters was provided by our professor. To study the management of processes and threads following POSIX standards, we decided to work with the Raspberry Pi 4.

III. Software Architecture

Tasks

The software architecture consists of six tasks with distinct functions: five periodic tasks and one one-shot task, which is executed only once at the start of the operation.

1) τ_1 **- Create Cube System:** One-shot task that initializes the system's components, including shared memory.
2) τ_2 **- Display Cube:** Reads shared memory and updates the outputs of expanders and shifters, turning the LED's on or off (periodic).
3) τ_3 **- Read Buttons:** Captures user input while debouncing button presses (periodic).
4) τ_4 **- Update Snake Direction:** Updates the Snake direction based on input (periodic).
5) τ_5 **- System State Transition:** Handles transitions between system's states (periodic).

6) τ_6 **- System State Action:** Executes state-specific actions (periodic).

A single process creates all six tasks. The task τ_1 has the highest priority in the system and operates under the FIFO scheduling policy, ensuring it is never blocked by lower-priority tasks. The remaining tasks, τ_2, τ_3, τ_4, τ_5, and τ_6, are executed using the Round Robin scheduling policy, all with equal priority.

Shared resources

The single process contains resources that can be read and write by threads. Each task receives a void pointer to a struct, shown in Listing 1, created after running τ_1 or Create Cube System thread. It is worth mentioning that **Shifter2** is initialized together with **Shifter1** as a cascaded (daisy-chained) arrangement mentioned in Section II.

```
typedef struct {
  LedValues LedArray;
  Expander Expander1;
  Expander Expander2;
  Expander Expander3;
  Shifter Shifter1; // and Shifter2
  Button Button11;
  Button Button12;
  Button Button13;
  Button Button14;
  Button Button15;
  Button Button16;
  Button Button21;
  Button Button22;
  Button Button23;
  Button Button24;
  Button Button25;
  Button Button26;
  SystemStates SystemState;
  pthread_mutex_t StateMutex;
  pthread_mutex_t directionMutex;
  int SnakeDirection;
} CubeSystem;
```

Listing 1: Shared memory between threads.

Communication Protocols

Communication between the Raspberry Pi and the hardware components is established through the following protocols:

I2C Protocol: The Inter-Integrated Circuit (I2C) protocol is a synchronous, multi-master, multi-slave communication protocol commonly used in embedded systems. It uses two lines:

- **SDA (Serial Data Line):** Transmits data between the master and slave devices.
- **SCL (Serial Clock Line):** Synchronizes data transmission with the clock pulses.

The Raspberry Pi acts as the master, while the MCP23017 expanders function as slaves. Each expander is assigned a unique address, enabling the master to communicate with multiple devices on the same bus efficiently.

Serial Clock with Shifters: The 74HC164 shifters use a serial clock (SCLK) to receive and shift data into their internal registers. The process involves:

- Feeding the data bit-by-bit into the serial input pin.
- Using the clock pulse (SCLK) to shift the bits through the register.
- Latching the data to update the outputs and control the corresponding LEDs.

Therefore, cycling between each layer can be achieved using the shifters in cascade as mentioned in II, minimizing the number of GPIO pins required from the Raspberry Pi.

Hardware Communication Summary:

- The Raspberry Pi communicates with the MCP23017 expanders via the I2C protocol, leveraging its ability to handle multiple devices on the same bus with unique addresses.
- The 74HC164 shifters are controlled using a serial clock (SCLK) signal to shift data into the registers, enabling precise control over the cube's layers.

IV. Results

Task Scheduling

The tasks are managed with deadlines (D) equal to their periods (T), ensuring deterministic behavior. The characteristics of each task are shown in Table I.

Task	Worst-Case Execution Time (C)	Period (T)	Deadline (D)
τ_1	5.6 ms	-	-
τ_2	21.8 ms	30 ms	30 ms
τ_3	6.2 ms	30 ms	30 ms
τ_4	0.4 μs	30 ms	30 ms
τ_5	0.9 μs	30 ms	30 ms
τ_6	8.2 μs	15 ms	15 ms

TABLE I: Task Characteristics

CPU Utilization

The system's utilization is calculated as:

$$U = \sum_i \frac{C_i}{T_i} = 0.93 \qquad (1)$$

This 93% utilization ensures sufficient CPU headroom for future scalability and real-time performance in the worst-case scenario. Therefore, the system is schedulable.

Functional Evaluation

The system successfully achieves its objectives:

- **Snake Game:** Smooth gameplay with user inputs via buttons.
- **Rain Effect:** LEDs light up in a cascading pattern.
- **Fireworks Effect:** Simulated bursts of light resembling fireworks.

Analytical vs Observed Results

The observed behavior matches analytical estimations, with tasks adhering to their deadlines and transitions between states occurring seamlessly.

The CPU usage was checked using the command `top -i`. It revealed that the process uses about 60% when initializing and drops to about 5% after a few seconds in every mode of the system, never hitting the 93% obtained in Section IV.

System Responsiveness

The periodic tasks (30 ms periods) provide a responsive system for user interaction and visual updates, with no observable lag during operation or changing states.

V. Conclusion

The project's complexity stems from:

- **Task Scheduling:** Managing multiple periodic tasks within deadlines.
- **I2C Integration:** Ensuring seamless communication between the Raspberry Pi and expanders.
- **Game Logic:** Implementing dynamic gameplay and visual effects.
- **Hardware Understanding:** Overcoming initial difficulties in understanding how the shifters work and manipulating the expanders.

The thread *DisplayCube* was identified as having a high execution time. A possible solution to improve performance would be to divide this thread into simpler threads: one thread could calculate the output values based on the memory, while another thread writes these values to the components.

The project utilized a significant portion of the course content, including real-time systems and task scheduling, reflecting a high degree of learning application.

One interesting aspect to explore would be to force the execution of the process using only one core instead of four. By limiting the computational resources, we could gain valuable insights into its performance and behavior, opening up opportunities for optimization and analysis.

This project successfully implements a Raspberry Pi-controlled LED tower, achieving interactive effects with efficient task management. The modular architecture and low CPU utilization ensure scalability, while the observed results validate the system's design and operation. Future work could explore additional effects and integration with external sensors for enhanced interactivity.

VI. Relevant Links

A short demonstration video was published at: youtube.com/SEMB_FEUP-Led_Tower. This video shows the two effects(rain and fireworks) and the snake game.

The project code is available on GitHub at this link: github.com/SEMB-LED_Tower

Appendix A: Circuit Diagram

The circuit diagram is available for download at this link: circuit_diagram.pdf