**Milestone 3**

Pong Game

Carlo Russo, [Nigel Gary](mailto:nigelgary@oakland.edu), [Michael Szpotek](mailto:mszpotek@oakland.edu)

Computer Science and Engineering Department

School of Engineering and Computer Science

Oakland University, Rochester, MI

crusso@oakland.edu, nigelgary@oakland.edu, mszpotek@oakland.edu

**Abstract-This project milestone report outlines the development of a Pong game on the Raspberry Pi utilizing the Sense HAT. The game features adjustable difficulty levels, real-time ball and bat movement, and joystick-controlled bat navigation. The objective is to enhance understanding of Python programming, hardware interaction, and basic game development principles.**

Ⅰ. INTRODUCTION

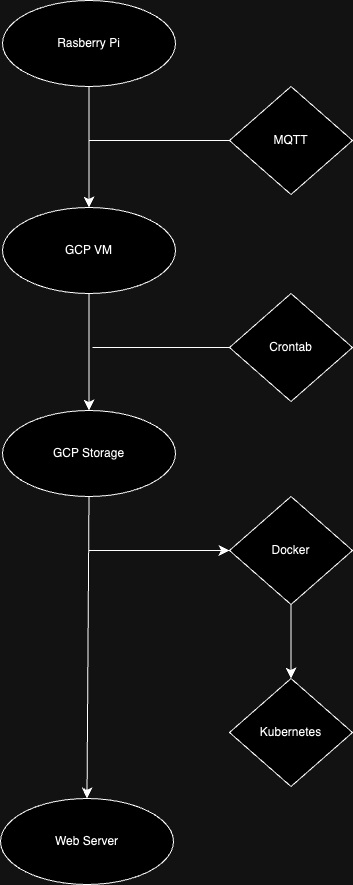
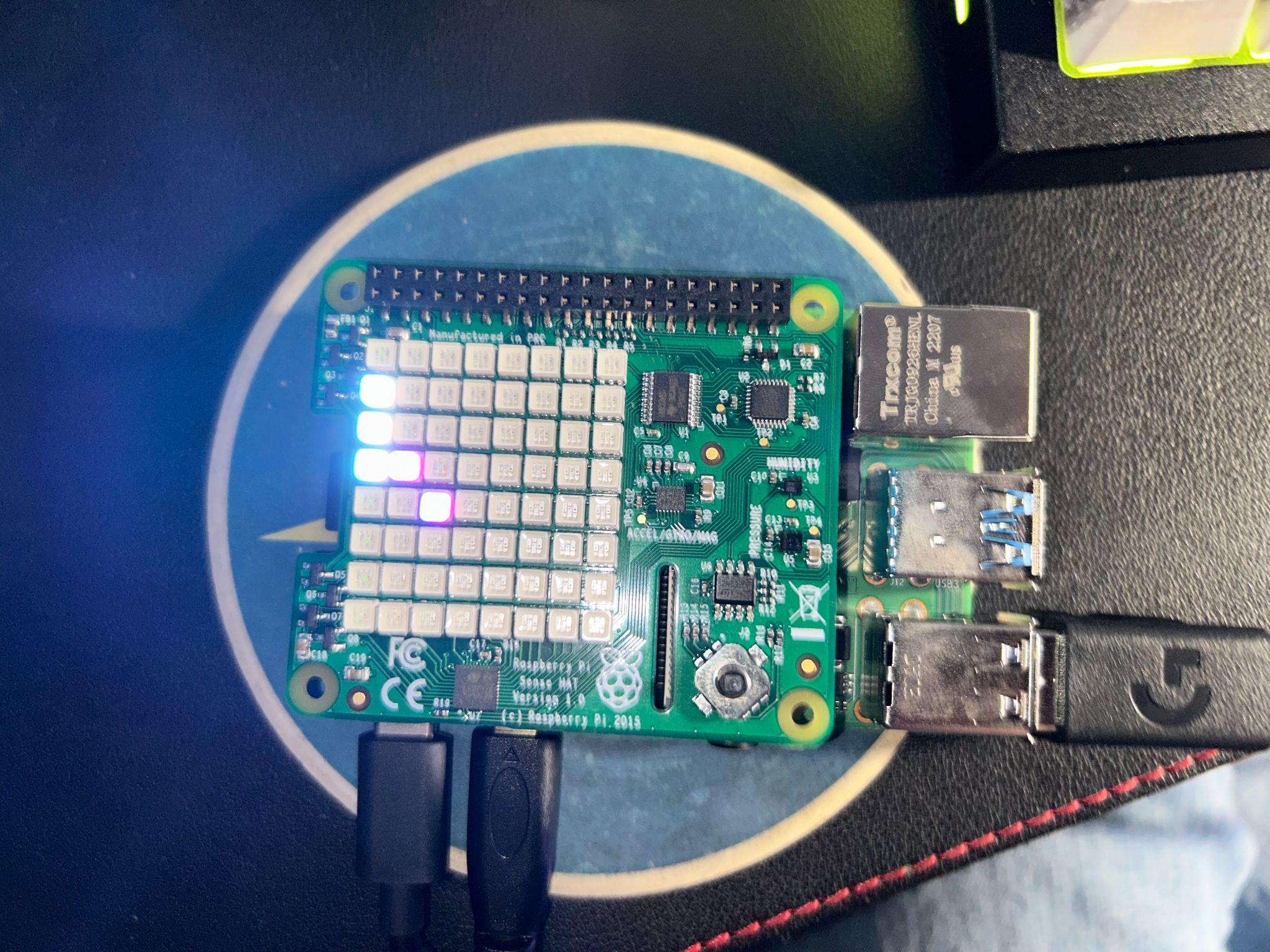
The integration of physical computing with game development presents an engaging approach to learning programming and electronics. The Raspberry Pi, combined with the Sense HAT attachment, offers a unique platform for such projects. Our Pong game project is relevant and interesting because it demonstrates how to create interactive applications that require real-time input and display feedback. This project is not only about coding but also about interacting with hardware, offering a comprehensive learning opportunity. The relevance of this topic is supported by the growing interest in STEM education, where hands-on projects play a pivotal role in enhancing learning outcomes.

Ⅱ. BACKGROUND

To fully appreciate this project, one must understand the Raspberry Pi and Sense HAT. The Raspberry Pi is a small, affordable computer popular in education and hobbyist projects for its versatility. The Sense HAT is an add-on device for the Raspberry Pi, featuring an 8x8 RGB LED matrix, joystick, and various sensors. This project leverages the LED matrix for the game display and the joystick for bat control.

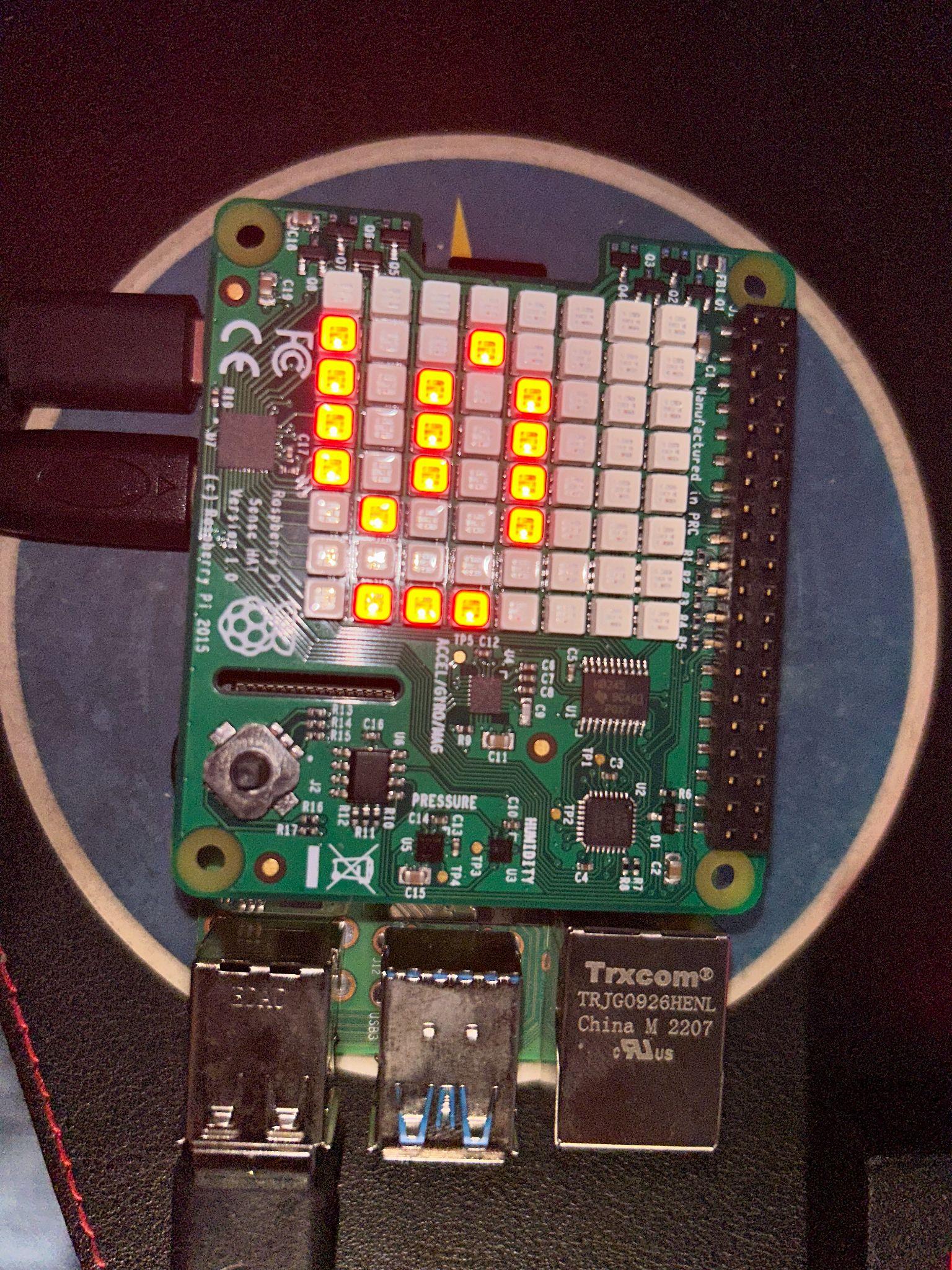
Ⅲ. OVERVIEW

The Pong game project was designed with simplicity and interactivity in mind. Utilizing Python, we developed a game loop that continuously updates the game state, including the positions of the bat and ball and handling user input through the Sense HAT's joystick. The game's difficulty can be adjusted by changing the ball's speed, allowing users to select from three levels of difficulty to suit their skill level.



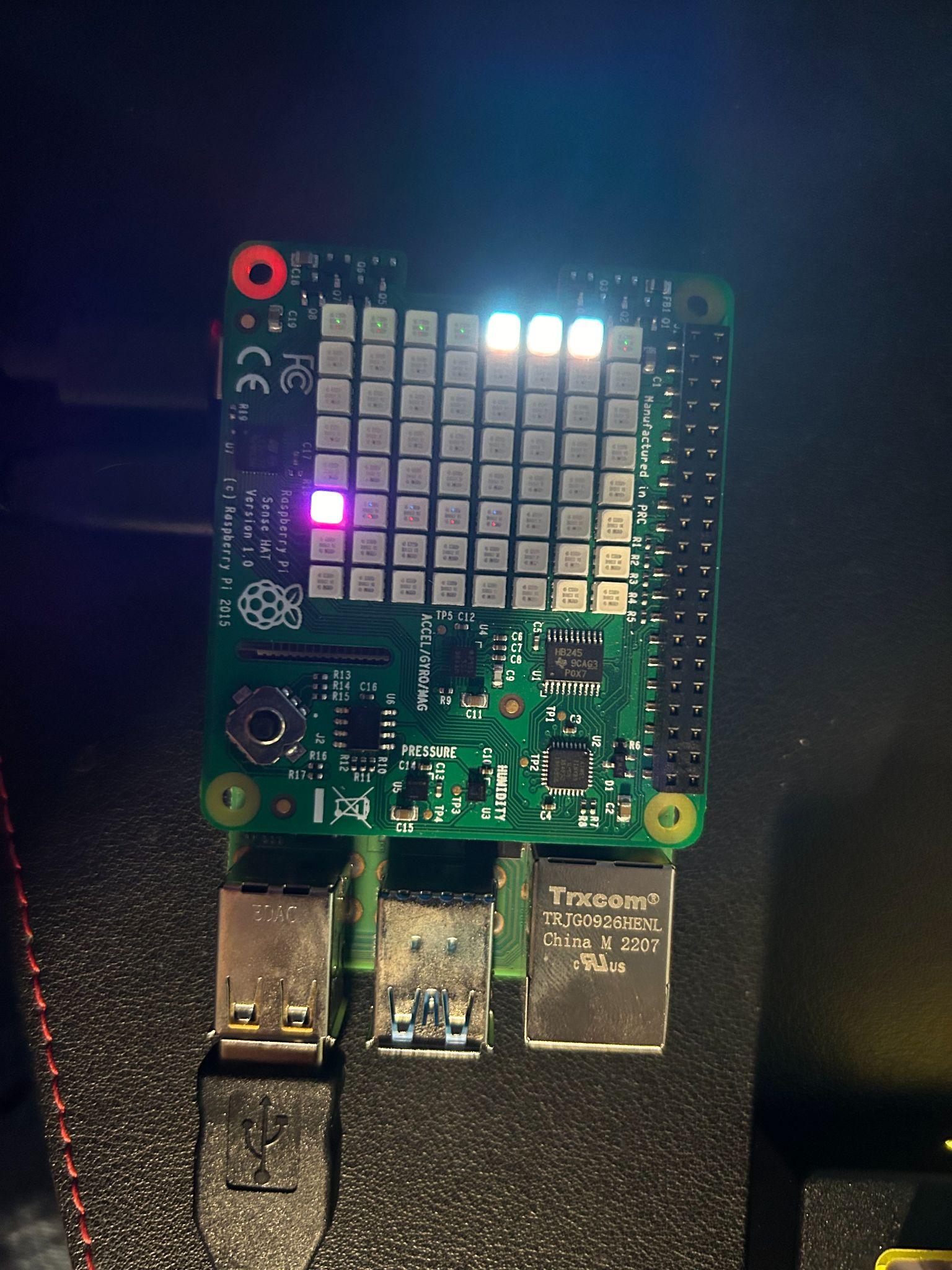
#### *A. Game Mechanics*

The core game mechanics involve moving a bat vertically to intercept a bouncing ball. The ball's motion is determined by its velocity, which reverses upon collision with the screen's edges or the bat. If the ball passes the bat and reaches the screen's edge you’ll lose and the game will restart. This simple setup provides a foundation for understanding game loops, collision detection, and user input handling.



#### *B. User Interaction*

User interaction is facilitated through the Sense HAT's joystick. Players move the bat by pushing the joystick up or down, demonstrating how hardware inputs can be integrated into software applications. This interaction model is critical for developing interactive applications beyond traditional keyboard and mouse inputs.



Ⅳ. SETUP

To replicate this project, one needs a Raspberry Pi with the Sense HAT attached. The Python script provided is executed on the Raspberry Pi, assuming Raspbian or a similar OS is installed. Key steps include installing the sense\_hat Python package and ensuring the Sense HAT is correctly mounted on the Raspberry Pi. Once the script runs, the game initiates with a prompt for selecting the difficulty level, showcasing real-time bat and ball movement on the Sense HAT's LED matrix.

Ⅴ. RESULTS

The implemented game successfully demonstrates the capabilities of the Raspberry Pi and Sense HAT for developing interactive applications. The adjustable difficulty levels function as intended, affecting the game's pace and challenging the player's reaction time. The joystick controls provide a responsive and intuitive interface for bat movement. While the project met its primary objectives, further enhancements could include score tracking and a win condition to increase the game's complexity and replay value.

VI. Milestone 2

Created a web server on a Raspberry Pi, hosted locally, and implemented the pong game onto the web server using HTML, CSS, and JavaScript. Created a bash script to monitor CPU, RAM, and Storage using Crontab to automate it into a log file. Implemented a leaderboard that tracks the players top scores. Removed the level of difficulty and implemented a constant speed for the ball.

VII. Milestone 3

In this phase, we focused on enhancing our Pong game project through automation and cloud integration, significantly improving data handling and application scalability. We implemented the MQTT protocol to enable real-time data transfer from the Raspberry Pi to the Google Cloud Platform VM. The game’s data, which includes logs of the system data and player scores, are automatically backed up using Crontab. We containerized our web server and the application into a Docker image and deployed it on a Kubernetes cluster in order to enhance reliability.

ⅥII. CONCLUSION

This project milestone has shown the feasibility of creating interactive games with the Raspberry Pi and Sense HAT, emphasizing the importance of integrating software with physical computing. Future work could explore more advanced game features, such as AI opponents, multiplayer functionality, and improved graphics using external displays. Learning outcomes include a deeper understanding of Python programming, game design principles, and the practical application of computer science and electronics knowledge. The remaining challenges include refining the game's user interface and expanding its features to increase educational value and engagement.

VIV. REFERENCES

[1] Raspberry Pi Foundation. “Sense HAT Pong.” *Projects.Raspberrypi.Org*, projects.raspberrypi.org/en/projects/sense-hat-pong. Accessed 5 Feb. 2024.

[2] Jais, Nilisha. “Nilisha-Jais/Pong: Sense Hat Pong.” *GitHub*, github.com/nilisha-jais/Pong. Accessed 5 Feb. 2024.

Contribution for Milestone 3:

Carlo Russo: Created Docker image and pushing to Kubernetes Clusters to off load data

Nigel Gary: Google Doc, diagram and developed crontab for the automation of Google operations

Michael Szpotek: Created MQTT connection between Raspberry Pi and GCP VM for live-data updates