

Micro:bit Gaming Project Report

Project Title: Interactive Games on Micro:bit using Python

Table of Contents

1. Project Overview
 2. Game Implementations (Python)
 - 2.1 Dino Game (Single-player)
 - 2.2 Battleship (Two-player)
 - 2.3 Hunter-Enemy Tag Game (Two-player)
 3. Challenges and Solutions
 4. Future Improvements
 5. Team Members
-

1. Project Overview

This project showcases the integration of hardware-level control and creative game design using the BBC Micro:bit. We developed three interactive games using MicroPython and an LED animation controller purely in ARM Assembly. The objective was to explore both high-level prototyping and low-level system control, leveraging the Micro:bit's capabilities such as:

- 5x5 LED Matrix Display
 - Buttons A & B
 - Microphone
 - Accelerometer (tilt sensing)
 - Radio Communication
 - GPIO and timer control
-

2. Game Implementations (MicroPython)

2.1 Dino Game (Single-player)

Description:

Inspired by the Chrome Dino Game, this version has the player avoid obstacles like ground blocks and flying birds. It includes motion sensing, sound input, and visual feedback on the LED grid.

Features:

1. **Controls:**
 - Button A → Jump
 - Button B → Crouch
 - Microphone Input (Clap) → Jump
2. **Obstacles:** Randomly generated birds and blocks displayed at various vertical levels
3. **Scoring:** Increases with time/obstacles avoided
4. **Visuals:** Bird and block symbols, Dino movement on LED



2.2 Battleship (Two-player)

Description:

An engaging two-player version of Battleship where each player selects and defends a hidden ship position on the 5x5 LED grid. The game uses **Micro:bit's radio feature** to enable wireless communication between the two devices.

Features:

- **Ship Placement:**
 - Players can **choose and adjust** their ship's shape and position using button inputs.
 - **Button A:** Move ship left
 - **Button B:** Move ship right
 - **Both Buttons Together:** Select diagonal or custom ship patterns
 - If a player feels their ship is too easy to find, they can press **Button A again to change the pattern** before the game starts.
- **Starting the Game:**

- Once ship placement is done, players press **Button B to signal readiness**.
- A message saying “**Start Battle!**” appears to begin gameplay.
- **Turn-Based Attacks (via Radio):**
 - One player starts by selecting a coordinate.
 - **Button A / B / Both** used to move and choose target.
 - To lock in the target, the player **shakes their Micro:bit**.
 - The selected position is **sent over radio** to the opponent.
- **Feedback Mechanism:**
 - If the guess is correct, a  **tick symbol** is shown, and the **same player gets another turn**.
 - If incorrect, a  **cross symbol** appears, and **the turn switches** to the other player.
- **End Condition:**
 - The game ends when a player **successfully hits all parts of the opponent's ship**.
- **Implementation Notes:**
 - Implemented in MicroPython.
 - Radio communication is done using `radio.send()` and `radio.receive()` in Python.
 - LED indicators provide feedback for each action.

2.3 Hunter-Enemy Tag Game (Two-Player)

Description

A real-time chase game where one player (Hunter) and the other (Enemy) navigate a grid using BBC micro:bits. The game ends after 60 seconds, with no explicit tagging mechanic implemented.

Key Features

1. Role Assignment

- Game starts by scrolling **"HUNTER GAME"**
- Players choose roles:
 - **Press A → Hunter** (flashing high-intensity LED)
 - **Press B → Enemy** (steady high-intensity LED)
- **Conflict resolution:** If both pick the same role, displays **"SAME"** and resets.

2. Movement Controls

- **Button A:** Move left
- **Button B:** Move right
- **A+B:** Move down
- **Tilt forward (face up):** Move up

3. Time Limit

- Fixed **60-second rounds**
- Automatic game end when time expires

4. LED Indicators

To clearly indicate roles on each micro:bit:

- **Hunter's micro:bit:**
 - **Hunter (self):** Bright LED
 - **Enemy (opponent):** Dim LED
- **Enemy's micro:bit:**
 - **Enemy (self):** Bright LED
 - **Hunter (opponent):** Dim LED

5. Reset Condition

- Immediate reset if both players select identical roles initially
-

3.Challenges:

1. Radio Communication Reliability: Ensuring consistent data transmission between devices
2. Resource Management: Working within Micro's limited memory and processing power
3. Input Detection: Creating responsive controls with limited input options
4. Game Logic Complexity: Balancing feature richness with code simplicity
5. Power Efficiency: Maintaining battery life during extended gameplay

Solutions:

1. Implemented optimized radio protocols with error checking
2. Used efficient data structures and minimized variable usage
3. Combined multiple input methods (buttons, tilt, sound) with appropriate thresholds
4. Separated core game logic from display/input handling
5. Utilized sleep modes and reduced unnecessary processing loops

4.Future Improvements

- **Game Menu System:** Create a game selector UI with scrollable icons.
 - **Multiplayer Expansion:** Add 3+ player modes using mesh networking.
 - **Bluetooth Integration:** Use mobile app to control or view game stats.
 - **Persistent Scores:** Store high scores using external EEPROM or flash
 - **Sound Effects:** Trigger buzzers or external speakers using GPIO PWM.
 - **Motion-Based Power-ups:** Detect shakes or complex gestures for bonuses.
-

5. Team Members

1. P Jaya Raghunandhan Reddy - BT2024029
2. Pasham Godha - BT2024082
3. Varanasi Harshith Raj - BT2024177
4. Hasini Samudrala - BT2024113
5. Polareddy Harshavardhan Reddy - BT202406
6. Gangavarapu Jaswant - BT2024010