**Microcontrollers Assignment Badge**

Christopher Luddy

R00214236

**Protype Documentation**

**Design Overview :**

**Name: Double Jump Bro’s**

The prototype is a simple 2D arcade-style game built for an Arduino-based system with a Liquid Crystal display (LCD) and an ATiny, a buzzer for sound effects, and input controls such as a button and slide switch. The player controls two heroes who run and jump over terrain obstacles that randomly appear on the screen. The objective is to avoid colliding with the obstacles and accumulate a high score. The ATiny is used to show the level change and current level by turning on LEDs for each level. The sound can be adjusted using the slideswitch (Mute/Unmute).

The game also includes features like level progression based on the player's score, and it tracks the highest score using an EEPROM. The LCD displays the hero's movements, the terrain, the score, and other important messages (e.g., "Start Game" and "Game Over").

Once a user has lost the game the Game Over ! text is shown and a menu for the next actions is displayed. The user can then pick between these 3 actions either continue playing the game, display a text or debug mode to show debugging information.

I found this very enjoyable to make as I did encounter a lot of errors I also learned a lot and feel like my skills were improved as I did encounter a lot of errors both on the hardware and software sides.

**Hardware**

* **Arduino Board & ATtiny:** The microcontrollers used for the project.
* **LCD Display:** A 16x2 LCD used to show the game, the Hero, the Terrain, and the score / high score.
* **Button:** A push button to control the Hero’s jump.
* **LED:** A simple set of LEDs each connected to one of the digital pins to indicate when the score reaches each level.
* **Potentiometer:** Added to adjust the LCD’s display contrast for visuals.

**Software**

* **Programming Language:** The game is coded using C++ with the Arduino IDE and the ATtiny.
* **Libraries Used:** Liquid Crystal library for controlling the LCD.

**Important attributes:**

* **Terrain:** The terrain is generated by randomly placed blocks at the bottom of the screen, and when the hero advances, the landscape moves to the left.
* **Hero Animation:** Several sprite animations depict the hero's running and leaping positions as they run and jump.
* **Level Progression:** As the player advances, the game gets harder and harder, with faster speed and more dense terrain.
* **High Score:** At the conclusion of the game, the highest score is printed and saved in the EEPROM.
* **The buzzer Sound Effects:** Throughout the game, background music and sound effects are played these can be muted and unmuted by the user using a switch.
* **Input Controls:** The hero's jump action is activated by pressing a button. The player can toggle sound effects with a sliding switch. They can also turn on and turn off the badge using the battery’s on and off switch
* **Game Difficulty:** Over the course of the game the speed and chance of terrain spawn in are increase making the game harder as the levels increase.
* **Debug Mode button :** This button once activated displays Game State, Current level, Hero position, and other information about the current state of the system and game at that point in time.
* **Menu:** The player can select between continue playing game, text or debug mode. This menu gets called when the game is over. The text printed is game over and try again both moving for extra animation. I also added the debug mode, this can be triggered mid game by the left button press or at the end by selecting 3 and entering that into the serial interface.

**Issues Encountered During Development**

1. **LED and LCD Flickering Issue:**

During gameplay, the LED flickered, especially while the Hero was hopping, the connections seemed fine on the ATtiny and bread board so it must have been software issue.  
Cause: The usage of lcd.clear(), The code was calling a clear led state and recover led state over and over in a loop leasing to timing and LED control issues. The loop was meant to refresh or clear both LCD screen and all LED lights once the game started.  
Solution: To fix this, I replaced updates for lcd.clear() to the correct location and changed them in the loop that ran the LED code. As a result, there was less latency, and the LCD and LED displays could function smoothly without any flickering from both LCD and LED.

1. **Debouncing Button**Issue: Even though the button was only touched once, it occasionally read as numerous presses.

Cause: The button's mechanical design caused it to send out multiple notifications quickly when it was pressed or released (bouncing).  
Solution: To disregard several signals during the bounce phase, I implemented a debounce technique by adding a delay after the button is pressed. To make sure the button hit remained active after the wait, I also included a backup check. I tested this multiple times to find the right length of the delay.

1. **Movement of Terrain :**Issue: It was challenging to keep track of the score because the Terrain quickly moved back to the right after reaching the leftmost column. The issues I was facing were some level difficulty logics and laggy transitions.  
   Cause: Every 200 milliseconds, the Terrain moved, and each time it reset, the score increased. This resulted in an abrupt score boost that didn't seem to match the movement of the cactus. The transition or movement from each terrain block was laggy and slow. The code to increase difficulty was not implemented correctly.  
   Solution: To make sure the Terrain travelled more slowly, and the score increased more steadily, I changed the collision checks and speed. I changed the delay time to smoothen out the movement of Terrain
2. **Game Difficulty :**

Issue: I had to increase difficulty as the game progressed, doing this was challenging at first as increasing speed or terrain spawn rate altered the quality of the game.

Cause: Each time the game was sped up the collision timing was off, also at higher levels speed and spawn rate were difficult to balance as I wanted the game to be difficult not impossible but also keeping the game difficult enough to give the player a challenge.

Solution: For the collisions, timing adjustments were made to the login to handle rapid terrain updates effectively without allowing the hero and terrain to overlap. To solve the speed issue and difficulty balance, I added a speed limit and tested the percentage of spawn rates over and over to ensure game was difficult enough to challenge the users but also possible.

1. **Menu Integration:** When I began putting the game mode selection menu into practice, I encountered some trouble integrating it straight into the main game loop. First, I tried to add the menu outside the loop and call it after the game had started, then I tried the same before the game had started. Both tried failed. I then tried to add menu login into the main loop while putting the game function code into a separate function called Game(). This also proved to be unsuccessful. Finally, I added the user input function that handles the users’ inputs into a function outside the main loop while keeping the running code for the game in the main loop. This runs the game from the start and the once the game is over the if (GameOver) function handles the menu display and uses the user input to determine what option s selected and what task to do next. From here the users can continue playing the game by selecting 1 or 2 for Text or 3 to show the debug mode that prints all information to the serial interface.

**Link to Tinker cad Project:**

**Project link:**

[**https://www.tinkercad.com/things/4xQWtVClrG6-copy-of-updated-06122024-microcontroller-badge/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard**](https://www.tinkercad.com/things/4xQWtVClrG6-copy-of-updated-06122024-microcontroller-badge/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard)

**First Design Draft:**

A circuit board with wires and a rectangular screen

Description automatically generated

Here we have my first draft of the badge I started off by connecting an attiny to an lcd screen, I used one resistor to control the wattage of the led in so the Lcd screen didn’t over load and blow. I powered the attiny, breadboard and the LCD using the 6v battery pack. A button was connected to turn on an led. The button was connected to pb2 and the led was connected to pb 3 of the attiny.

**Second Design:**

**A circuit board with wires and a screen

Description automatically generated**

Realising my design was flawed and needed a bigger microcontroller with more features I decided to restart and use an Arduino to power my lcd. This time the button was connected to pin 2 of the Arduino and the lcd was directly connected to the 5V with a resistor of 220 ohms. A potentiometer was added for contrast visibility.

I decided to stick with my second design as I had more options to expand my Badge. I then added an Attiny with LEDs for level tracking in the game. I currently had a demo working looking something like this on the Lcd screen

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

“D” was the character, “#” was the object and Score: printed the score you were on.

When the character and the object collide the game state is changed to “Game Over!” and the score is printed

**Final Design:**

All components working correctly, menu installed for extra LCD activities for user including reading of a text, debug mode and a game and the option to play again.

A diagram of a circuit board

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A circuit board with wires

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

3. A screenshot of a computer program

Description automatically generated

Note you must be in the Arduino Uno interface to see serial monitor display

2. A screen shot of a computer

Description automatically generated

1. Plays game again.

**In conclusion:**The game prototype successfully incorporates a series of features designed to make for an engaging arcade-style experience. Sound effects, score tracking, hero movement and animation, and terrain scrolling are all implemented successfully. However, several difficulties were encountered throughout the development process, such as button debouncing issues, inconsistent difficulty progression, and collision detection concerns. Specific code changes and optimisations were made to address each of these errors. The product is an engaging yet challenging game that offers valuable insight on game development procedures for embedded devices such as Arduino and the ATtiny. I really enjoyed the process of making this badge and will continue to make more in the future with the knowledge I have learned from making this one. !!