**CM12002 – Coursework 1: Whac-a-Mole**

**Project Report**

**Morgan Connolly & Eddie Baker**

**1.0: Introduction to the project**

Whac-a-mole is an arcade style game in which objects, often represented by moles, sequentially arise from a grid of holes in which the player must ‘whack’ said mole whilst it is risen to score a point. Whac-a-mole has seen many adaptations from many different developers, and what started as a physical, real-life game was translated into a digital version as digital game variations become more popular. A variety of changes can be made to the original game structure to improve the fluency of gameplay for a digital version, more of which will be highlighted later as we proceed to analyse and explain our implementation of the game.

**2.0: Understanding our ideal implementation of the project**

We will be implementing our version of Whac-a-mole using an Arduino Uno. We aim to achieve all required core functionality, as well as the specified further functionality.

There are some fundamental rules that are crucial to the core game; rules that we will ensure we don’t break with our variation of the game.

Our first variation to the typical game comes through how players can obtain score. Our approach is that all players have two blue LEDs and one green LED, the objective of the game is for the player to press the button only if the green LED is on, not if either of the blue LEDs are. We are using this approach as after experimenting with the original game style, we felt it was too easy and not enjoyable enough, regardless of difficultly level (which will be mentioned later).

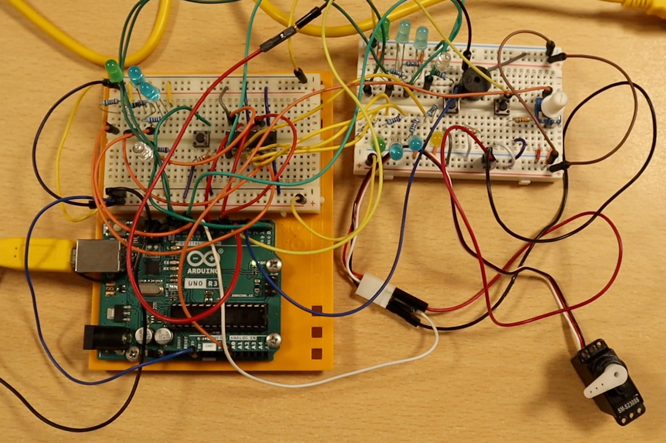
|  |  |  |  |
| --- | --- | --- | --- |
| **Objective** | **Achieved?** | **Objective** | **Achieved?** |
| Checking whether the button is pressed whilst an LED is on. | Yes | Include sound output for correct/incorrect button presses. | Yes |
| Incrementing a score when the button is pressed at the correct time. | Yes | Allow customisation of difficulty through changing the duration LEDs are active. | Yes |
| When the score reaches 10, flash all the LEDs and end the game. | Yes | Add a third player to the game. | Yes |
| Extend to 2 player mode. | Yes | Make use of a servo motor. | Yes |

**4.0: How does our approach work?**

As highlighted above, each player has two blue LEDs and one green LED in which they will only obtain a score point if it is the green LED which is currently lit up. We made use of a shift register as we had a limited number of digital pins to work with, allowing us to implement a third player. Our use of a potentiometer was to set the difficulty between rounds – a player could turn the potentiometer 90 degrees to increase the difficulty, in A circuit board with wires and wires

Description automatically generatedwhich the servo motor would then rotate 90 degrees to represent the difficulty change at the end of a round. Every player has their own score counter, set of LEDs and button, as using one button between multiple players would prove conflicting. Every player additionally has a white LED, which lights up upon a correct button press, scoring them a point. Once a player reaches 10 points, the winners LEDs will flash and instead of ending the game, a new round will start, giving players time to adjust the difficulty between rounds if they wanted to.

**5.0: Why did we choose this approach?**

****Whilst we had a strong initial plan for our approach and how we would utilise the provided components to achieve the required outcome, there were still adaptations we had to make throughout development to ensure we could meet our objectives. We initially didn’t intend to use a shift register, however after finding ourselves not being implement three player mode without one, we decided to make use of one and are now satisfied with its purpose.

Our main adaptation to the typical game is LED specific scoring, essentially the idea that a player can only score if it is specifically the green LED that flashed. We implemented this as during investigating how we can implement difficulty levels, we decided that our system would work better overall if we took out some of the ‘button mashing’ aspects from the game, achieved by requiring the player to have to be more precise as to when they can press the button.

**6.0: Future improvements / problems during development**

One of the biggest problems we encountered during development was one button press granting a player numerous score points when they should only get one. We were able to mitigate this bug by making use of global variables in our code in which each player had a unique boolean variable that would be set to True if the button is pressed and would then be set to false again when the LEDs change. This ideally would prevent our bouncing issue as the scoring mechanisms would only trigger for any given player if the player-specific variable was set to false, therefore after a button press if the button press algorithm attempted to trigger a second time, the player wouldn’t achieve score. Whilst this method fixed the majority of instances of this bug occurring, we acknowledged it is a memory-inefficient approach to fixing this bug and additionally in some instances, specifically when the green LED was to flash twice in a row and a player was to hold the button for a time period overlapping both LED flashes, then occasionally multiple score points would still be allocated, this is an area of our project we’d attempt to program differently in the future.

Additionally, we highlighted earlier our usage of a shift register to allow us to implement three player mode. Whilst this proved extremely useful, we still had to figure out work-arounds to maximise efficient digital pin usage on our Arduino, we achieved this through usage of code libraries that allowed us to alter which pins could handle interrupts.