Arduino UNO Whack-a-mole Game

Overview

This project is a basic whack-a-mole game involving 2 players with light, sound, and variable difficulty. It also makes use of a servo to show which player is winning. There are 3 LEDs for each player; red for player 1, and green for player 2. In turns, the players have one of their LEDs, or “moles”, lit and have to press their corresponding button before the LED turns off. If they are successful, a white LED in front of their moles will light, a success tone will play from a piezo speaker, and the servo will rotate in their favour. Their score (starting from 0) will also increase by 1. If they press the button when no mole is lit, a failure tone will play. The first player to reach 10 points (or any other chosen max score) will win the game, which will cause a winning tune to play, all of their LED moles will light, and the servo and score will reset.

LEDs

The LEDs in this game were coded in such a way that only one would turn on at a time for each player, and the players would take it in turns to play. The duration for which they turn on for is decided by the player by use of inputting through the potentiometer. The time *between* each player’s turn however was consistently random in the range 0.3 to 1.3 seconds. We did this as we felt that the time between the moles lighting was not important for difficulty, however the duration for which they were lit would have a drastic effect, with this part being solely dependent on reaction time of the player.

Buttons

To implement button presses, we used the built-in interrupt pins 2 and 3 on the Arduino UNO to attach interrupts to each button. When the button is in a “falling” state, i.e. goes from un-pressed to pressed, an input function for that player is called which checks if each of their moles is on, and if any of them are, turns that player’s corresponding white LED on, which then causes their score to increment.

The use of interrupts in our game was useful for allowing the game to immediately check whether the player inputs were right or wrong. The downside to this approach however was that it is then difficult to add any more players as both built-in interrupt pins have been used and an alternative solution would have to be found.

Potentiometer

To make the difficulty variable, we used a potentiometer that outputs values between 0 and 1023. When the potentiometer is turned all the way down (anticlockwise), the value is 1023, and if it is turned all the way up (clockwise), the value is 0. We then wrote a function that takes the value outputted and multiplies it by 0.9, which we then used as the delay time that the LEDs (moles) were lit for. However, if the potentiometer gave less than 50, we would simply set the delay value to 50, so as to not make the game impossible. Our delay times therefore ranged between 0.05 seconds and 0.92 seconds, the latter being useful for example for children or someone just starting out at the game.

This approach was highly useful as it allowed a full and continuous range of difficulty levels, however we could have further added an RGB LED that indicates the current difficulty level to give a visual guide accompanying this.

Servo

For the servo, we wrote a function that is called when either player's score is increased. This function turns the servo on for a tiny fraction of a second (in microseconds) and then turns it off again. We calculated the values of the microseconds so that it would turn anticlockwise for player 1 and clockwise for player 2, in equal proportions. We wrote the function so that if either player was leading by more than 3 points, the servo would not turn, to keep it inside it's range of motion and prevent damage. We used each player's LED colour as a placeholder on each side of the servo arm. When the game is won by either player, we reset the servo by turning between 1 and 3 pulses in the losing player's favour, depending on where the final position of the servo arm was. This then puts the servo arm back to its original horizontal position.

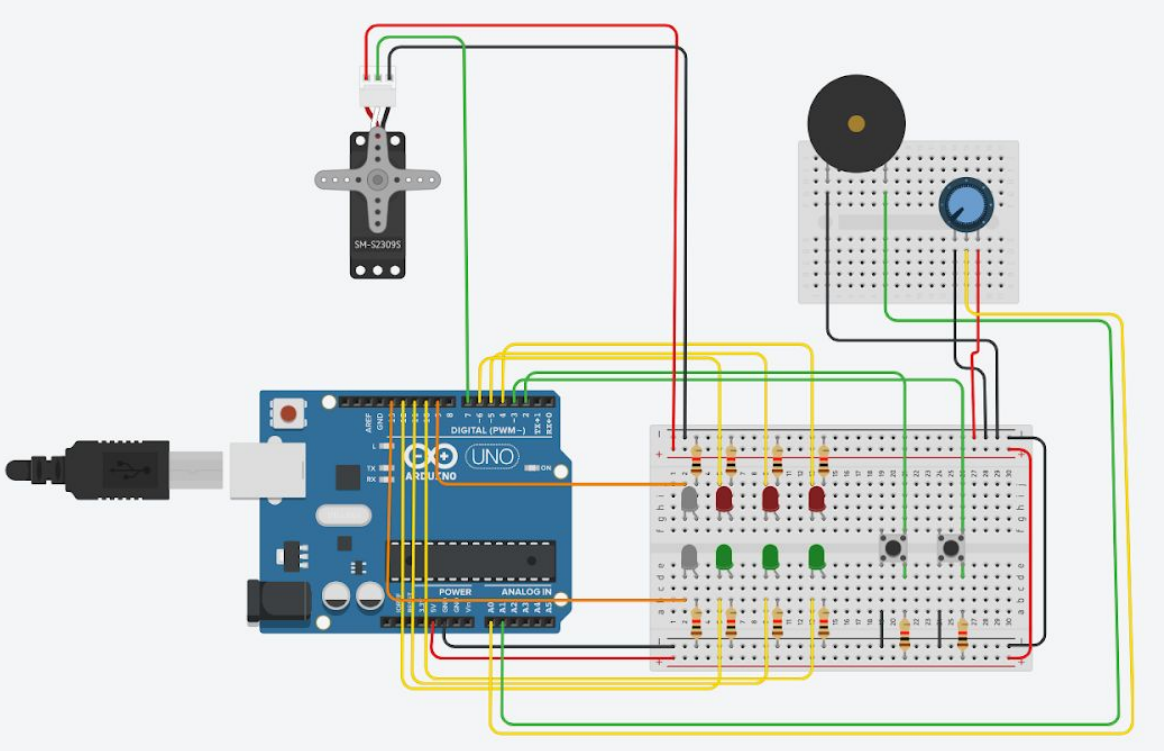
Sound

In terms of sound, we used the in-built "tone" function in Arduino. We wrote our code so that a high-pitched "happy" tone is played when the player presses their button successfully, and a lower-pitched "wrong" tone is played when the player presses their button at the wrong time. When either player's score reaches 10, we created a melody of 5 tones that will play to signify victory.

Breadboard

Player output LEDs (moles) are in digital pins 4-6 (inclusive) and 9-10 (inclusive). Buttons are in pins 2 and 3 making use of the interrupt functionality. We added an additional breadboard to the design which makes access and wiring components easier. It also acts as a separation between general game inputs/outputs and player specific inputs/outputs.

Circuit Diagram



(Created on: https://www.tinkercad.com)

Photo of the solution showing all LEDs, buttons, potentiometer, and piezo speaker

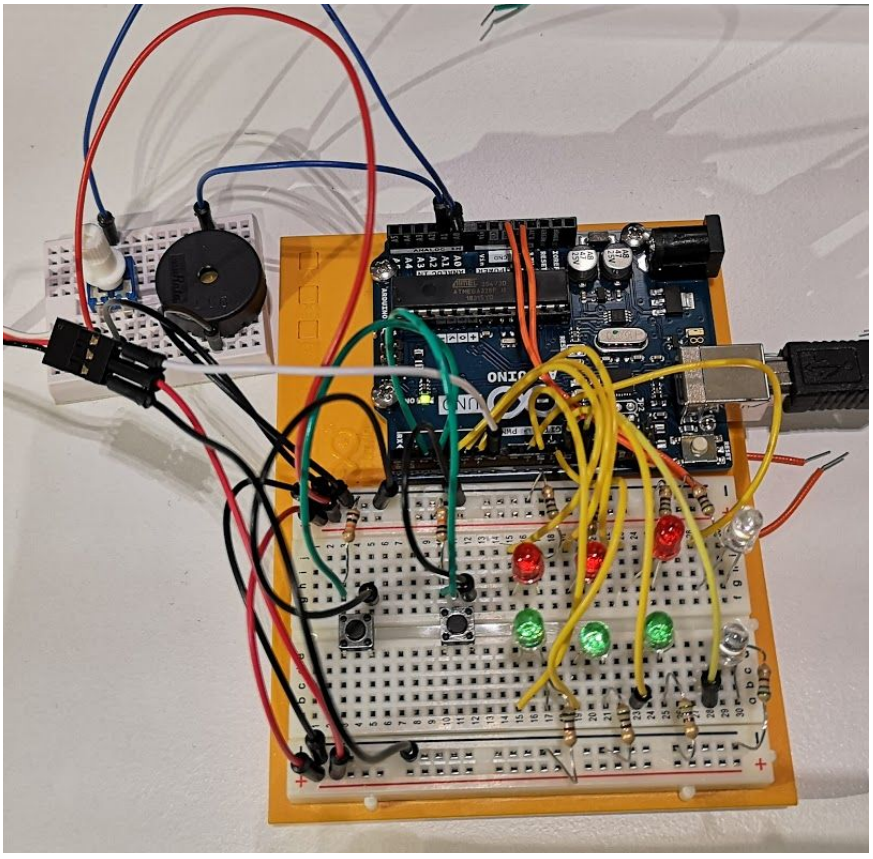


Photo showing close-up view of potentiometer and piezo speaker

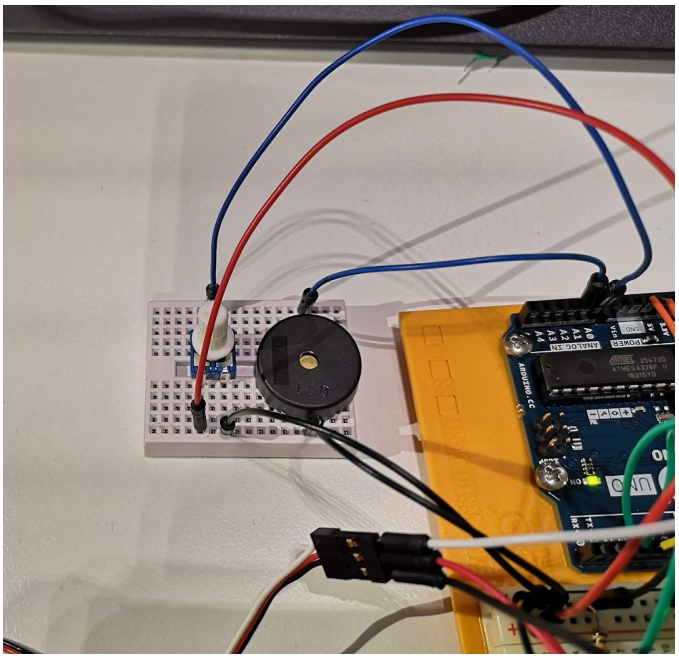


Photo showing servo with coloured LEDs indicating each player