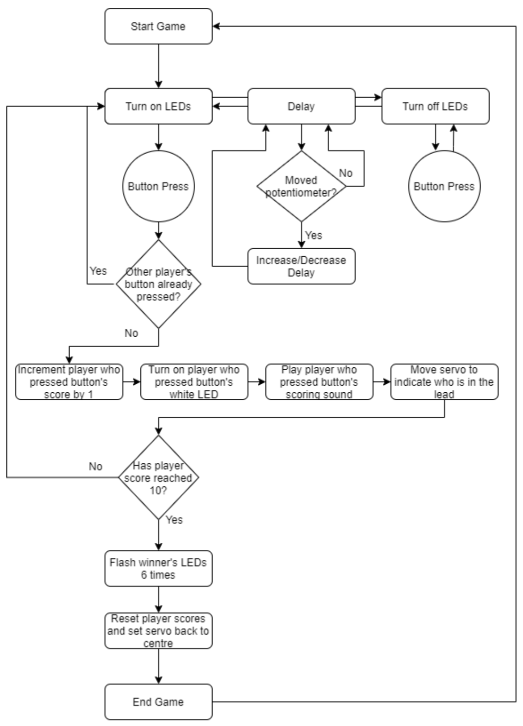
WHAC-A-MOLE

Project Introduction:

We were tasked to create a game of whac-a-mole using an Arduino Uno. The features that we have included in our game are: a 2 player game, 3 coloured LEDs per player representing the ‘moles’, a button per player, a white LED per player, a servo which indicates which player is in the lead, a potentiometer which changes the difficulty level, and a piezo which outputs a tone indicating which player has scored a point.

Description of Gameplay:

The gameplay begins with both players’ sets of coloured LEDs lighting in random sequence and both players are then free to press their buttons. The player can turn the potentiometer to change the speed at which the LEDs flash, and therefore the difficulty level. We decided to make the game a race to see which player can press the button for each ‘mole’ first in order to increase the competitiveness of the game and player enjoyment. If a player presses their button faster than the other player when one of their LEDs is lit, then their white LED will light, their buzzer note will sound and their score will be incremented by 1. If a button is pressed slower than the other player or an LED is not lit, then it will be ignored. Throughout the game the servo will move on the dial to indicate which player is in the lead or if it is a tie. Once one player reaches a score of 10, the game ends and the LEDs of the winning player will flash to indicate they have won, then the game will reset and restart. This is all represented in the flowchart to the right.


Start Game
Turn on LEDs
Delay
Turn off LEDs
Button Press
Moved potentiometer?
No
Yes
Button Press
Increase/Decrease Delay
Yes
Other player's button already pressed?
No
Increment player who pressed button's score by 1
Turn on player who pressed button's white LED
Play player who pressed button's scoring sound
Move servo to indicate who is in the lead
No
Has player score reached 10?
Yes
Flash winner's LEDs 6 times
Reset player scores and set servo back to centre
End Game

Implementation:

Buttons: We decided to use interrupts to process the button presses. These work by executing a specified function as soon as the interrupt is triggered by a signal from one of the specified pins. We decided to use this method as it is the most efficient way to quickly execute code on a button press as it means the processor does not need to finish executing what it is currently doing. This means the timing of the button presses is more accurate than what it would be with other methods. However, the drawbacks of this method are that interrupts can only be used for pins 2 and 3 meaning that adding a third player and a third button would not be possible without additional code.

LEDs: For the LEDs we used one array of pin numbers rather than having separate arrays for players one and two’s LEDs or having each LED have its own separate variable. This was in order to reduce repeated code and make the code more readable as the LEDs could be set up and turned on and off inside of for loops. This method enabled us to use a shared function for both players 1 and 2 called playerInput() as we could use the player value to choose which LEDs in the array should be checked or changed.

Piezo: To produce a sound output we decided to use a piezo buzzer. Due to the fact that we decided to make it a race to who presses the button first, we decided each player should have their own pitch and then the pitch of whichever player gets the point would be played. We decided to do this because having the buzzer play a sound for each correct or incorrect button press for both players would be too overwhelming and not very informative as it wouldn’t indicate which player made the correct or incorrect button press.

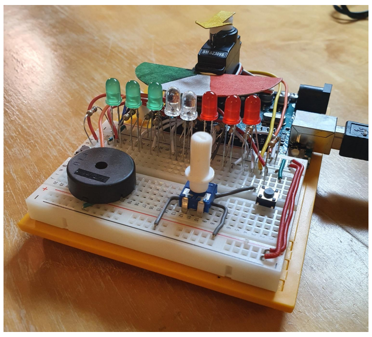
Servo: We chose to have the servo and a dial to indicate which player is in the lead. At the beginning of the game and when the scores are equal the arrow will point towards the centre of the dial. This gives a clear indication of who is in the lead, however it is not accurate in the sense that it does not tell the player how much they are in the lead by. If we wanted to display the exact scores of the players we could have used an LCD, however we evaluated and decided that attempting this would be too time consuming for the constraints of the project.

Potentiometer: We decided to use a potentiometer to select our difficulty levels. It is a variable resistor which changes the time between each LED blink. We decided to use the potentiometer instead of a button, for example, to toggle the difficulty level as the potentiometer offers a continuous selection of difficulty levels rather than a discrete number that a button would offer. This means there is more selection and customizability for the player, making the game more enjoyable.

Challenges:

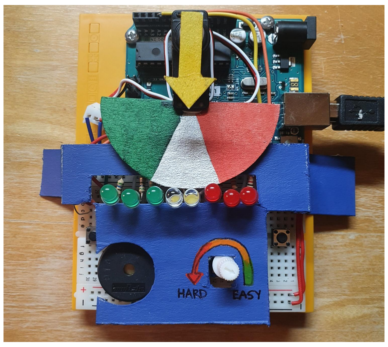
As previously mentioned due to the nature of using interrupts for the button presses, it made trying to implement a third player very challenging. This, combined with the challenge of the limited number of pins on the Arduino, meant that we were not able to implement the three player mode and instead focused on optimising our code for the two player mode.

Images:

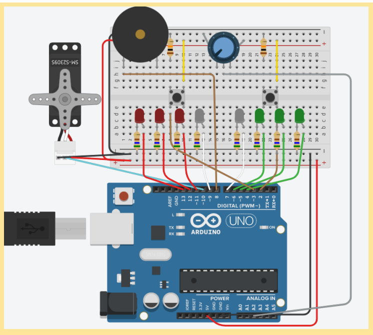


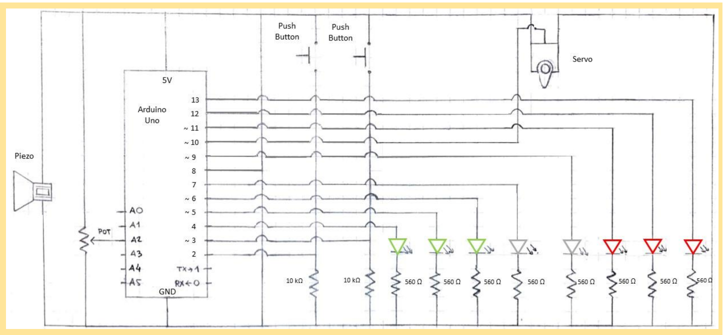
<- Image showing our arduino and breadboard demonstrating the layout of components.

Image showing the view the players would **->** get when playing the game, with breadboard cover on to optimise game play.



Diagrams:





Arduino diagram

Schematic

Above are an arduino diagram and a schematic. They both show the components involved in the physical circuit and how they are connected to the Arduino. The schematic shows representations of components whereas the Arduino diagram shows what the components actually look like in the circuit. The Arduino diagram is useful to show how the circuit is built in real life and the schematic is useful to represent how the circuit works logically.