Whack-A-Mole Coursework

What is Whack-A-Mole?

Whack-A-Mole is a Japanese arcade game created in 1975 by Kazuo Yamada of TOGO. The game involves having to have quick reflexes in order to hit animals on the head using a mallet when they pop up out of a hole.

My Game

Our coursework was to implement the actual game into the Arduino board, using lights as the 'moles' and the button as the 'mallet'. My game includes all the core features and for the two additional functions I used a piezo to show incorrect and correct button presses as well as servo to indicate who is in the lead while the game is going on. To include the servo I had to add the servo library in my code. The game was meant to use white LEDs to indicate when a correct button press happened but as I had no white LEDs I used blue LEDs for both of the players to indicate when they have scored a point. My game can be seen in the Appendix A,B,C and the schematic for my game is Appendix D and the breadboard layout can be seen at Appendix E. To design the schematic I used fritzing. The schematic and breadboard layout as well as the pictures should help to give a visual idea of how the game was built and set up as well as how it functions.

My Code

For my code I used three main functions in the void loop function. playerOneGame, playerTwoGame and ScoreCompare. These three functions would be on loop unless the score for either player was equal to 10. The score variables and the boolean variables for each player are global variables as they are used and updated in different functions so instead of having to keep passing the variables through the function creating global variables was easier.

For player one the function playerOneGame runs the game for them. The function randomly lights up an LED for a random time and when the LED is lit it sets the boolean variable for P1 to 1 so that a correct button press can occur, this will be talked about more in the button functionality. For player two, the function playerTwoGame runs the game for them the same way as it does for P1 but lights up P2's LEDs and changes P2's boolean values.

Within the loop function there is a scoreComapre function. This function uses the servo along with the values of each players score. After the loop function runs both players game it compares the score. The servo is set to 90 degrees at default to point to the middle, if player one is winning the servo rotates to 0 degrees to point to the greens LEDs (Player One) if player two is winning the servo rotates to 180 degrees to point to player two LEDs. If the scores are tired the servo goes back to 90 degrees. For the servo I had to include the servo library which can be seen on line 7, the library allowed me to use in built functions to rotate the servo.

As I mentioned before the loop functions run the three functions in a while loop until either players score is equal or greater to 10. If the score is greater than 10 the function gameOver is run which passes both players score as parameters. The function uses a for loop to flash all the LEDs of both players to indicate the game is over, it also checks who has reached 10 first to flash their score LED (Blue LED) which can be seen in the video. For my code player one and player two functions are very similar, the code is almost identical just that the variables are different for each player, such as the boolean variable and the score variables. The codes are almost identical as this reduces bias and in the game and tries to keep it as random as possible.

Button Functionality

To implement my button presses I used interrupts. I think interrupts was the best way to go as it allowed for button presses outside of an LED flashing to be registered. I also used boolean logic along side interrupts to check whether the button press was correct or not. When any of the LEDs were lit for any player their own boolean was set to 1 otherwise their boolean was set to 0. So the interrupt for the button when pressed ran the function checkPlayerOneInput for P1 and checkPlayerTwoInput for P2. The two functions checked whether the boolean value was 0 or 1 and acted accordingly, when the boolean value was 0 the function caused the piezo to output the incorrect sound. When the boolean value was 1 another function is run, the functions lights up the blue LED to indicate a score and outputted a sound on the piezo to indicate that the button press was valid.

The downside of using interrupts was that sometimes when the button was pressed when the boolean value was 1 it registered it as an incorrect button press due to the delay and feedback when pressing the button. To overcome this problem I added a timestamp and a while loop to make the boolean value 1 the entire duration the LED was lit, this can be seen on line 130 and 131 of my code. I also added some small delay to account for the feedback on the button but overall using interrupts was the best option as it allowed me to use the piezo to output incorrect and correct sound.

Additional Functions

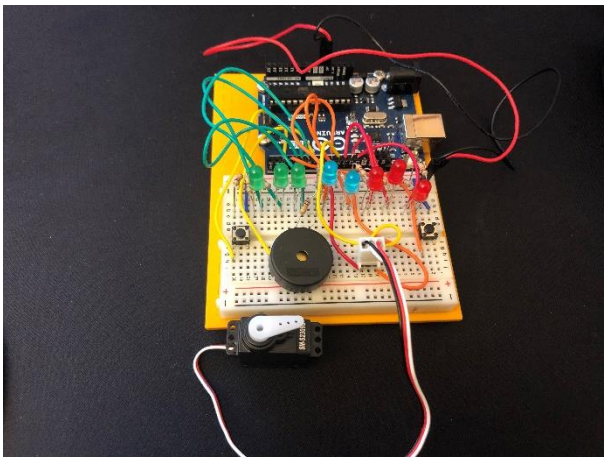
For the additional functions as mentioned before I used a servo to indicate whos in the lead and a piezo as an output for incorrect and correct button presses.

I decided to go with the servo to show whos in the lead. The servo allows the players to visually see whos in the lead which creates a competitive atmosphere. However the servo can sometimes be delayed when quick scores are scored in succession. This caused the servo to rotate multiple times in a matter of a few second which can cause confusion, but the functionality of the servo worked perfectly fine as shown in the video. To code the servo I used the inbuilt functions within the servo library, these functions allowed me to change the direction the arrow was pointing to. An example can be seen on line 104

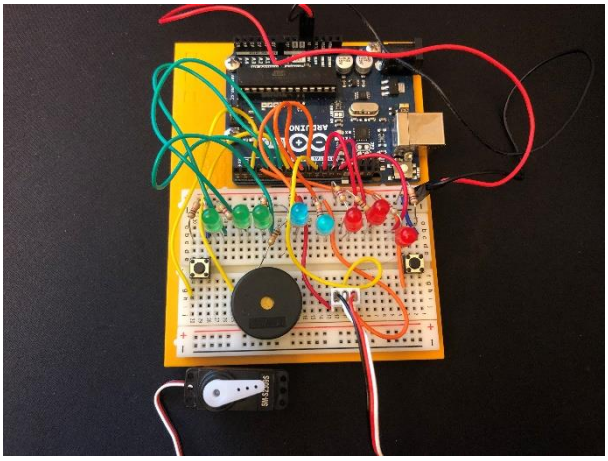
I chose the piezo as a sound ouput for correct and incorrect button presses as it helps the users know when they have scored a point or have the missed the timing on the button. The piezo outputs the same sound for incorrect presses for both users and a different sound for correct button presses and each players correct button press has a different sound to each other. I think having a piezo is useful as from the video we can see that the blue LEDs flash quickly so sometimes the user might not see that they have scored the point and the sound output would help them to notice that they have scored the point. Also having a sound for incorrect button presses helps the players to understand the timing of the button and when they have pressed the button at the wrong time, which in turn will help them to score more points. Coding the piezo was pretty simple as I used the tone feature which enables you to set the hertz of the output and for how long the sound is outputted for. An example of this can be seen on line 163

My code and game has been completed to fulfil all of the core functionalities and has achieved two of the four additional functionalities. To improve the game I would've liked to have used the LCD screen to indicate the score for each player as well as show the record for how fast the game has been completed in, like a leader board system. I also would have liked to add a third LED that flashes randomly and the first player to hit the button when the LED flashes gets 2 points, however the Arduino board did not have enough pins to include this feature but this would've been a nice feature to include as it makes the game more competitive.

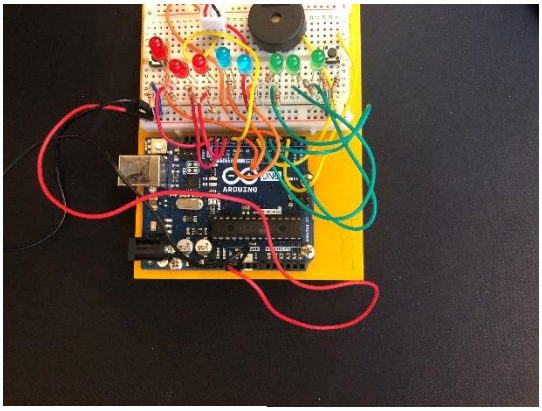
Appendix



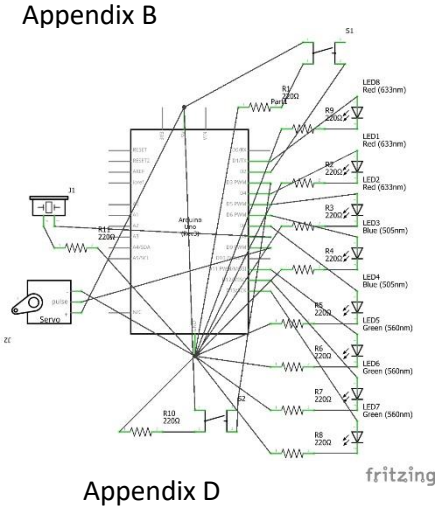
Appendix A



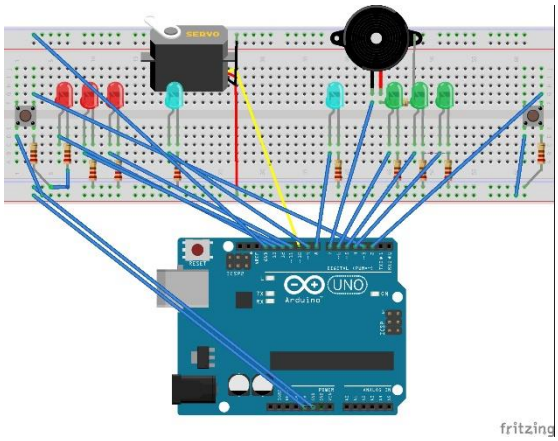
Appendix B



Appendix C



Appendix D



Appendix E