For this project, my group has developed an alternative version of the arcade game “Whac-A-Mole” as shown in **Figure 1**. We’ve done this by building an Arduino circuit and extending an Arduino program. For each player, we implemented different coloured LED lights as shown below in **Figure 2.** The yellow and green LED lights turn on at random times for a short period of time. Within this short time, each player has to press their respective buttons’ to score a point, which is exhibited by the red or blue LED lights flashing. When 10 points have been scored, all LED lights of that player will flash, showing the player has won.

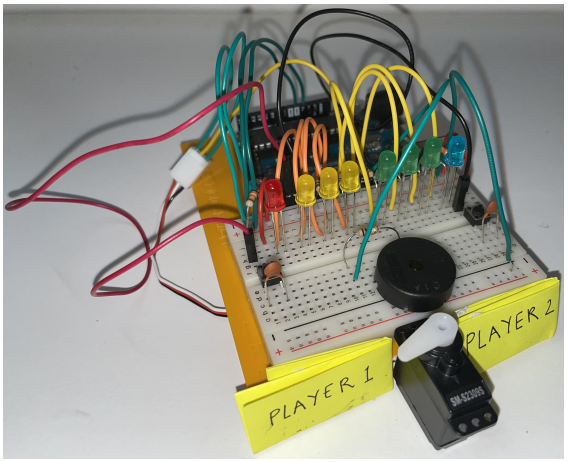

Figure 1: Arduino circuit of the solution[1]

Player 1	Player 2
Yellow LEDs - Acts as the moles - Connected to pins 5, 6, and 7	Green LEDs - Acts as the moles - Connected to pins 8, 9, and 10
Red LED - Acts as the score indicator - Connected to pin 4	Blue LED - Acts as the score indicator - Connected to pin 11
Button - Acts as the mallet - Connected to pin 2	Button - Acts as the mallet - Connected to pin 3
Servo - Points towards the winner of the game - Connected to pin 12	
Buzzer - Makes a sound when the game is over - Connected to pin 13	

Figure 2: Assigning pins for player 1 and 2

In our code, we declared different variables, some include `delayTime` and `randNumber`. An object for the servo was created to allow us to control it. When the Arduino turns on, the `setup()` function runs once. We’ve formatted the digital pins to be either inputs or outputs through a function called `pinmode()`. The pins connected to LEDs are outputs while the button pins are inputs. We used the `Serial.begin()` function to create a link between the Arduino and the computer. `9600` is used within the parameters to show the speed at which the Arduino will communicate in bits per second. The use of `Serial.println()` in setup is to send information from the Arduino to the connected computer, presenting it in the Serial monitor, which is where our score count and the winner of the game is shown for our project.

After the `setup()` function has been completed, the `loop()` function runs continuously, where it checks for voltage on the input and turns the output on and off. Within this function, we use commands such as `digitalRead()` to check the voltage input, and `digitalWrite()` which enables us to send either 5V or 0V to an output pin. `Delay()` functions were also used to stop all functionalities from continuing. We used `attachInterrupt()` functions as it allows the code to be carried out automatically in an efficient manner. We created two interrupt functions, one for each player, stating the requirements for a point to be counted.

[1] Own image

We connected a cable from the 5V pin on the Arduino to one of the positive bus lines on the breadboard. We then connected another cable from the ground pin to the neighbouring bus line. Resistors are used to absorb some electrical energy, allowing less energy to pass through the connected component. Each LED is connected to the breadboard with a 560Ω resistor attached in series, effectively dimming it and preventing it from burning out. A cable is used to connect the LEDs' anode to a pin on the Arduino making it accessible to use. We then connected a 220Ω resistor to each button in series, grounding the pin when the switch is open.

Figure 3: Debouncing error[2]

We attached one button on two opposite ends of the board. However, there was a debouncing issue which was a drawback. When the button is pressed, the spring inside of the button may oscillate, creating an unwanted bounce, causing it to not cleanly change states. This is shown in **Figure 3.**

The additional bounce can cause a microcontroller to add extra presses to the count score that are not intentional. And so, the score monitor may display the players' score as a higher number than it truthfully is. My partner and I chose to reduce the error by adding a capacitor to each button. The capacitor maintains the voltage from the power source, causing the unwanted bounce drop to be more acute as well as more attenuated.

Furthermore, we have added a servo that points towards the winner of the game. The servo has 3 wires, the live wire (red wire) which is connected to the positive vertical strip used for power, the ground wire (black wire) which is connected to the negative vertical strip used for ground connections, and the remaining line on the servo connecter (white wire) is connected to one of the digital pins on the Arduino board (pin 12).

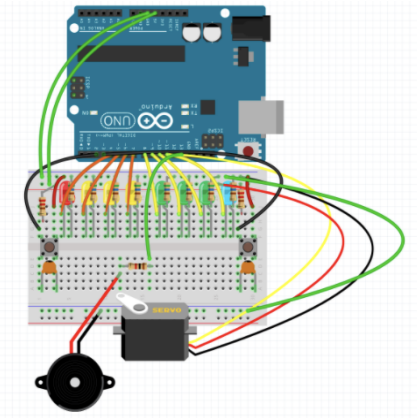

Figure 4: digital version of the Arduino breadboard[3]

Lastly, my partner and I added a piezo speaker (buzzer) to our circuit. We programmed the buzzer to emit a low frequency sound when the player presses the button when all the LED lights are off. When the game is over, the buzzer will emit a higher frequency sound three times in a row.

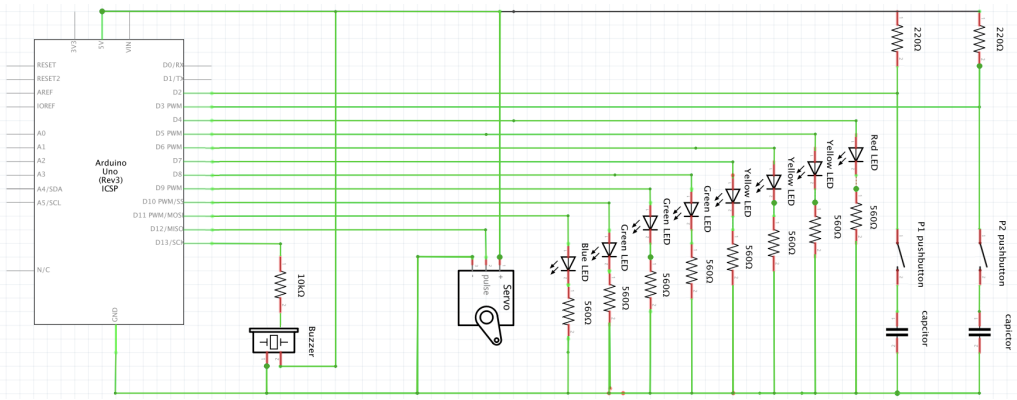

Figure 5: Schematic diagram of the circuit[4]

[2] Parks, M., 2015. *Find Electronic Components – Mouser Europe*. <https://www.mouser.com/blog/battling-the-bouncing-button> [8 November 2021].
[3] Own image made using fritzing
[4] Own image made using fritzing