Written Report on Arduino Coursework

This report will include our implementation to reach the end goal of constructing a fully functional game of “whack a mole” using the Arduino Uno and the components provided.

My partner and I were presented with the skeletal code which included the functionality of generating a random LED flash, as well as the functionality for making the white turn on when pressed. We were also helped to construct a circuit which included a button and all of the lights including the white LED. For requirement one, we needed to slightly modify the skeletal code by adding an if condition stating that when the button was pressed and given that a random LED was on the white light would turn on.

Requirement one didn’t pose as a problem to us at the time so we then proceeded to requirement two which included a score counter that would be incremented each time the player pressed the button at the correct time. For the purpose of completing the core functionalities, we started by printing the updated score on the screen by using the serial port. However, we soon realised that when the button was pressed the counter would over count and, when held down, the white light would come on when a random LED was lit. After speaking to one of the teaching assistants, we later discovered that we needed to consider the case of debouncing. We found that when releasing the button, there was a slight delay before the button effect and occasionally, the button triggered multiple times. After research, we found this was due to a delay in the metal disconnecting from the breadboard, hence a voltage is recorded. We had to implement a debouncing feature to our code to counter this effect. We created the state for player one’s button and created a function that returned a boolean value based on a slight delay of two milliseconds. We then added that condition into our requirement one condition stating that if the state of the button changed and the random LED was on to increment the score one by one. [1]

For requirement three, we proceeded by creating a conditional in our interrupt printing the score as long as it is less than or equal to ten. When the score reached ten, the console would print “You win”. In our main function, we created a global boolean storing the state of the game and setting the state of the game to false(gameOver) when the score is equal to ten in the interrupt function. This initialised the score to zero and flashed all the lights for three times to display this to the user. The potential area of improvement in this section would be understanding the relationship between the interrupt and the loop as we were beginning to find some bugs in our code at this stage.

Before we added multiplayer functionality to our system, we realised that reusing the same randomising function multiple times for every players LED pin, time delay for the LED to stay on and time delay for the LED to stay off, would result in an unfair game as it would come down to luck as to how many times your LED came on in a set time period. On top of this we had to think about how we could avoid players from clicking quickly, relatedly while their LED was on to get multiple points while that LED had been on once. To resolve this, we decided that the lit LEDs position, and the delay should be generated once and be used for all players. This would ensure all players got the same number of chances to click with a light. We also only restricted one point per LED light to be collected across all players, to make the game more competitive. This meant only one player could collect a point per set of LED lights that lit and then unlit, which also avoided the outcome of a draw between players.

To make the game two player we connected a wire from player one’s LEDs to player two’s LEDs to ensure that both players were playing under the same circumstances and we added another button connected to pin three. In our code, we had to add another interrupt function for player two and then we had to check which player was the fastest in responding to the

LED. We had to include the previous code from requirement two to ensure that overcounting wasn't an issue. One problem at this stage was that the code became challenging to manage as the debounce checking conditions had to be replicated and our the size of our code increased. Another noticeable consideration to improve the neatness of the circuit would be to use LEDs with in-built resistors in order to make space for other components such as the servo. The servo implementation was straightforward. For two players, we set it to point in the centre if it was currently a draw, or towards the player who currently had a higher score. For three players, it points towards the current player with the highest score, otherwise it points between player two and three if two or more players have the same score.

We implemented a mis-click counter for each player that, similar to a score counter, counted the number of times a player clicked when the light was not currently on. This allowed us to increase the difficulty and reduce the players score when the mis-clicked reached a set point. This stopped the player from pressing the button repeatedly to get the point first.

For the extra requirements my partner and I decided that it would be a good idea for the user to input their difficulty, this difficulty setting managed the delay time between the flashing LEDs and the maximum times a player could mis click before they got penalised. We continued by adding a piezo electric sensor that would output sounds, sounds for collecting correct clicks and sounds collecting incorrect clicks. Regarding 3 player mode we included a library which has the ability to change any pin into an interrupt pin. [2] We then proceeded by programming the player three button to respond to the lights much like our implementation of player one and player two's button. One setback in our opinion was the fact that we had to use another breadboard as most of our pins were taken up by the components used in our circuit. In future, we could explore ways of minimising the pins needed on the breadboard while retaining the same functionality.

In conclusion, this experience has opened our eyes to the possibilities of Arduino configurations and has indicated improvements to be made in understanding how code flows for future projects.

Reference List:

1)edu.8s.tv,2015**.** Arduino Button Debounce Tutorial[Online]. Available from: https://www.youtube.com/watch?v=jYOYgU2vlSE [Accessed 1st November 2018].

2)Anon, How to have unlimited interrupt Pins on your Arduino. *Brainy-Bits*. Available at: https://www.brainy-bits.com/make-any-arduino-pin-an-interrupt-pin/ [Accessed November 3, 2018].

Pictures: