Liam Verrier & Nicholas Rask
EE 10200

EE10200 Final Project Report

1. Executive Summary:

For this project, we designed a “Whack-A-Mole”-style game using the Arduino Uno and various microelectronic components. The game requires the player to quickly press a button (or not press any buttons) depending on which of the three LEDs on the device are illuminated. The player gains a point each time they execute the correct action within the time that the corresponding LED is lit up, and they are allowed three “mistakes” before the game ends. The device runs the game continuously, keeping track of and displaying the current game score and the all-time high score.

2. Background:

- Motivation: The Whack-A-Mole game device is not intended for practical use but rather for entertainment purposes. It is entertaining as it provides a simple pastime that can be enjoyed by people young and old alike. It could be used to entertain children who are old enough to understand basic games, and it also would still be effective for adults or young adults who want something to take their mind off things or wish to take a break at work or school.
- Attribution: The idea for this device spawned from discussion between group members rather than drawing inspiration from another previously created project, but there does exist an Arduino Education project called “Whack A Mole” which shares some similarities with ours. The game works fundamentally similarly except that the previous one doesn’t have a screen to count score like ours does and instead just has red and green LEDs that indicate whether the player pressed the right button or not, and it also doesn’t have a light that indicates that no button should be pressed. Their version also requires many more resources to build, including a frame to build around the Arduino board as well as an “Education Shield” that must be placed on top of the board and connected to the components. The device our group created uses fewer resources and has the

added functionality of keeping track of score and high score (while still notifying the user when they make a mistake, as the other device does).

- Project Evolution: The differences between the device proposed in the initial 1-page project proposal and our final product are as follows: the addition of a high-score tracker, the reworking of the game to reset repeatedly after each loss (until the Arduino loses power or is reset), the addition of a 'neither' LED that means *neither* button should be pressed when it is lit, and the replacement of showing an 'X' on the LCD when a mistake is made with a mistake counter.
- Environmental Scan: Several Arduino-based "Whack-A-Mole" games exist in the marketplace, but most of them use the buttons themselves popping up as the indicators of when they should be pressed rather than using LEDs like our project, and the buttons are typically stylized/decorated to look like a mole or another animal. These products also generally have 6 or more of these buttons/moles.

3. <u>Functional Description and Product Operation:</u>

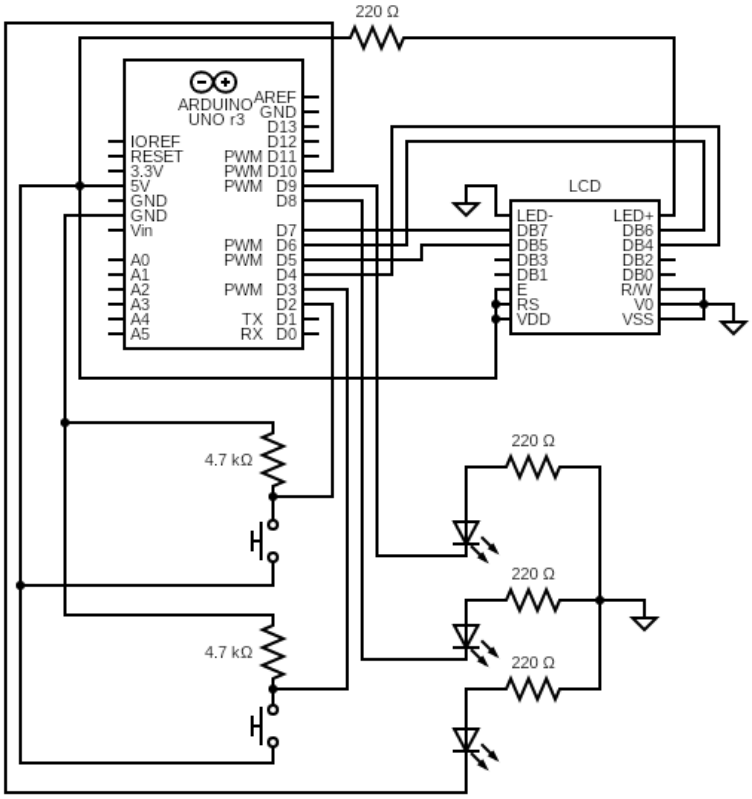
- Function: The service which this device provides is entertainment in the form of a game of "Whack-A-Mole". It repeatedly runs the game as long as it is connected to power.
- Operation: To operate the device, the user must plug the Arduino into a power source (and upload the program onto the Arduino if necessary.) Then the game will begin automatically after a three-second countdown. After the game has started, all the user needs to do is continually push the button corresponding to whichever LED is currently lit or push neither button when the third LED is lit in order to keep the game going and score as many points as possible.

4. <u>Hardware:</u>

- Sensors Used:
    - 2x push button
- Actuators Used:
    - 3x LEDs

- 1x LCD

- Circuit Diagram:



- Hardware Description: The device operates around an Arduino Uno R3, with the 3 LEDs and LCD powered through resistors and controlled by the digital pins on the Arduino. The push buttons have pull-down resistors to prevent floating when they aren't pressed and their output is read through the Arduino's digital pins as well. We used the LCD so that score could be tracked and displayed along with messages and the LEDs and buttons so that the user could visually see and physically interact with the game. The resistors with the LEDs and the LCD are for limiting the current going through them and preventing shorting.

5. <u>Software:</u>

- The program first initializes a variable for the LCD, integer variables for overall high score and LED pin numbers, and a boolean variable to keep track of whether a button has been pressed.

- **setup()** : This function takes no parameters and has a void output type so it returns no value. When it is run, it sets up the 3 LED pins to output mode, creates a random seed for the game's randomness factor using the floating value at digital pin 5, opens the LCD display for writing, and displays the title of the game.
- **loop()** : This function encompasses the playing of the game and repeats every after each game ends. It first initializes variables for the current game run including the integers for mistakes and score as well as a boolean for whether the game should end. It then displays the current high score, counts down to start the game, and then begins, repeatedly choosing a random LED to light up and displaying the current score and mistakes while incrementing them based on the player's response. It ends the game if the mistake count is at 3 and updates the high score if necessary. The function takes no parameters and returns no value.
- **buttonPressed()** : This function is called when an interrupt is detected. It returns no value and takes no parameters. The function debounces the button press and sets the global variable "button_pressed" to true so that the score will be incremented.
- **light_LED()** : This function lights an LED for a short period and then turns it off, attaching and detaching interrupts depending on which LED is being lit so that an interrupt only occurs when the score needs to change (for the normal LEDs) or the mistake count needs to change (for the trap LED). It returns no value but takes an integer parameter corresponding to the LED to be lit.
- Program flow: After the first initializations, the **setup()** function is automatically called. This doesn't call any of the other functions. **loop()** is then automatically called, in which **light_LED()** is repeatedly called multiple times every run of the game. Within **light_LED()** itself, **buttonPressed()** is called as a response to interrupts each time that a relevant button is pressed.

6. <u>Project Post-Mortem:</u>

- The final product of our project differs significantly from our original ideas for how it should operate. Most notably, the third "mole" LED was turned into a trap LED,

and the third button that was meant to correspond to it was removed. We decided to implement this twist on the original game for two reasons: for one, the Arduino only allows for interrupts on two pins, digital pins 2 and 3, so having a third button corresponding to a third LED was not possible under our setup. Also, we decided it would make the game more challenging and engaging if we added this special feature. Under similar circumstances, we decided to remove the “X” feature for mistakes and add a mistake counter as well as a high-score counter. These changes were implemented because the “X” feature did not work properly on the LCD. Despite many improvements on our code, each time the LCD began the game as empty and the “neither” LED lit up first, the display on the LCD would start to fill with corrupted information. We decided it would be best to have the LCD always display text, and so we reworked the program so that a mistake counter and score counter would be displayed on the screen throughout the entirety of the game. Out of this reworking came the idea to add a high-score tracker, to add a more competitive aspect to the game. As for making the game a continuous loop, we decided that it would be best for the pacing of the game to leave out a start button, and instead add a three-second countdown timer to the start of the game to let players prepare for the game.

- I believe that much of this project’s novelty comes from the competitive aspects of the game. The LED’s flash on and off fast, giving you a small window to react. The high-score tracker reminds players of the maximum extent to which they can succeed at the game, driving them to compete for the high-score spot.
- The most difficult part of this project was trying to implement the buttons as interrupts in the program, since we found that the interrupts being called at certain points in the code would break the entire program.
- If we were to do this project again, we would attempt to implement a ‘both’ LED into the game. While this would pose a serious challenge with the way we currently have the program set up, we would work to add this feature to further increase the complexity and competitiveness of the game.

7. Appendix:

```
#include <LiquidCrystal.h>

//create display object and initialize global variables
LiquidCrystal lcd(11, 12, 4, 5, 6, 7);

int high_score = 0;
bool button_pressed = false;
int LED1 = 8;
int LED2 = 9;
int LED3 = 10;

void setup() {
  //setup pin modes, random seed, and lcd screen with title
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);

  randomSeed(analogRead(5));

  lcd.begin(16, 2);
  lcd.print("WHACK-A-MOLE");
  delay(4000);
  lcd.clear();
}

void loop() {
  bool game_over = false;
  int mistakes = 0;
  int score = 0;
  int LED_num;

  //Display high score at the start
  lcd.clear();
  lcd.print("High Score: ");
  lcd.print(high_score);
  delay(1000);

  lcd.clear();
  for (int i = 3; i >= 1; i--) {
    lcd.print("New Game in: ");
    lcd.print(i);
    delay(1000);
    lcd.clear();
  }

  //Display the score continuously so the LCD is always updated
  lcd.clear();
  lcd.print("Score: ");
  lcd.print(score);
  lcd.setCursor(0, 1);  //Move to the second line for mistakes display
  lcd.print("Mistakes: ");
  lcd.print(mistakes);

  while (!game_over) {
    LED_num = random(1, 4);  //Randomly select between 1, 2, or 3
    light_LED(LED_num);  //Turn on the chosen LED
    delay(500);
```

```
58
59      //Check if the trap LED was lit
60      if (LED_num == 3) {
61        if (button_pressed) {
62          mistakes++;
63        } else {
64          score++;
65        }
66      } else {  //Normal LEDs
67        if (button_pressed) {
68          score++;
69        } else {
70          mistakes++;
71        }
72      }
73
74      //Reset button_pressed for next round
75      button_pressed = false;
76
77      //Update the LCD with new score and mistakes
78      lcd.setCursor(7, 0);
79      lcd.print(score);
80
81      lcd.setCursor(10, 1);
82      lcd.print(mistakes);
83
84      //End game if mistakes are too high
85      if (mistakes >= 3) {
86        game_over = true;
87        lcd.clear();
88        lcd.print("GAME OVER");
89        delay(1000);
90        lcd.clear();
91        lcd.print("Final Score: ");
92        lcd.print(score);
93
94        if (score > high_score) {
95          high_score = score;
96        }
97
98        delay(1500);
99      }
100   }
101 }
102
103 void buttonPressed() {
104   //Debounce the button press
105   static unsigned long lastInterrupttime = 0;
106   unsigned long currentInterrupttime = millis();
107   if (currentInterrupttime - lastInterrupttime > 250) {
108     button_pressed = true;
109     lastInterrupttime = currentInterrupttime;
110   }
111 }
112
```

```
113  void light_LED(int LED_num) {
114    //Light up the correct LED and interrupt if correct button is pressed while it's lit (or if any button is pressed for trap LED)
115    if (LED_num == 1) {
116      attachInterrupt(digitalPinToInterrupt(2), buttonPressed, RISING);
117      digitalWrite(LED1, HIGH);
118      delay(500);
119      digitalWrite(LED1, LOW);
120      detachInterrupt(digitalPinToInterrupt(2));
121    }
122    else if (LED_num == 2) {
123      attachInterrupt(digitalPinToInterrupt(3), buttonPressed, RISING);
124      digitalWrite(LED2, HIGH);
125      delay(500);
126      digitalWrite(LED2, LOW);
127      detachInterrupt(digitalPinToInterrupt(3));
128    }
129    else if (LED_num == 3) { //trap LED
130      attachInterrupt(digitalPinToInterrupt(2), buttonPressed, RISING);
131      attachInterrupt(digitalPinToInterrupt(3), buttonPressed, RISING);
132      digitalWrite(LED3, HIGH);
133      delay(500);
134      digitalWrite(LED3, LOW);
135      detachInterrupt(digitalPinToInterrupt(2));
136      detachInterrupt(digitalPinToInterrupt(3));
137    }
138  }
```