Final Report Document

Carson Wagner, Roberto Solis

Friday, May 3, 2024

1 Introduction

For our design, we have chosen to develop a game inspired by the classic "dinosaur game" found on Google when the internet is down. We named our game, "Circuit Dash" which will be implemented using the LPC1769 micro-controller, along with additional components to create a fully functional gaming experience.

Components

- **Microcontroller:** We will be using the LPC1769 micro-controller, which we have been using throughout the semester, which will serve as the central processing unit for the game.
- **HD44780 Display:** We will utilize an HD44780 graphical LCD to display the video game.
- **Joystick:** A joystick interface will allow players to control the movement of the character, which will also to jump through the obstacles.
- **Piezo Buzzer:** To have a better gaming experience, we will incorporate an 8-note song played through a piezo buzzer as an our intro song.

Gameplay Features

- **Animated Real-time Game:** Players will will be able to experience an animated real-time game, which will earn us (1pt).
- **Non-Graphical LCD Output:** The HD44780 will help us display the game and it will also earn us (1pt) and to move the character, we plan on implementing a joystick (0.5pt).
- **End Game Song:** Before the game-play starts, an 8-note song will be played through the piezo buzzer. We got 0.5 points instead of one because we had trouble implementing into the game code, but on separate file it did run correctly. (0.5pt total).

2 Gameplay Instructions

To play Circuit Dash, follow these instructions:

- **Game:** Will start when joy-stick is pressed. Obstacles will come towards the player, but it is the duty of the player to make sure the obstacle is jumped over with joy-stick.

3 Controls

- **Joystick:** When joystick is moved to the up position will cause the ball to jump.

4 Complete Schematic

Here is our complete schematic preliminary design for our hardware.

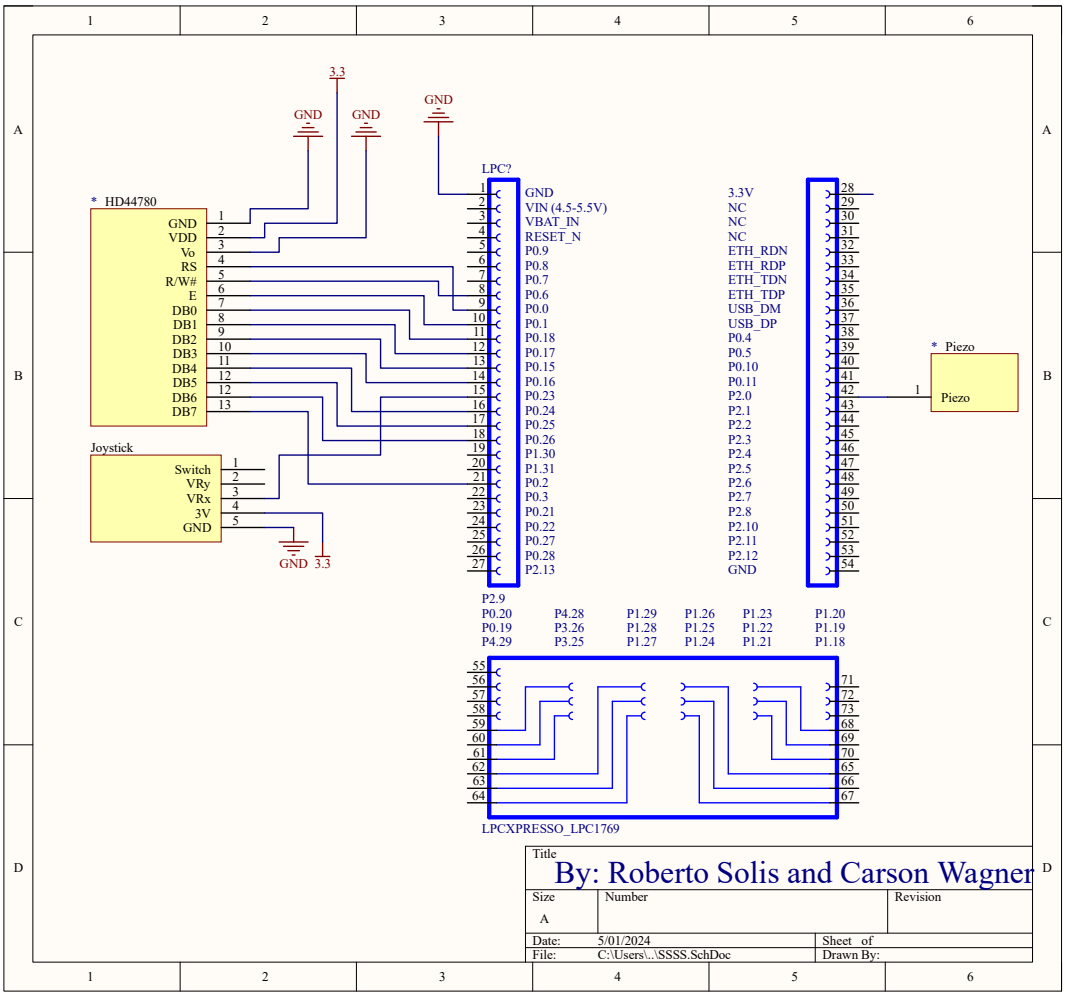

Figure 1: Final Hardware Design

5 Picture of Hardware

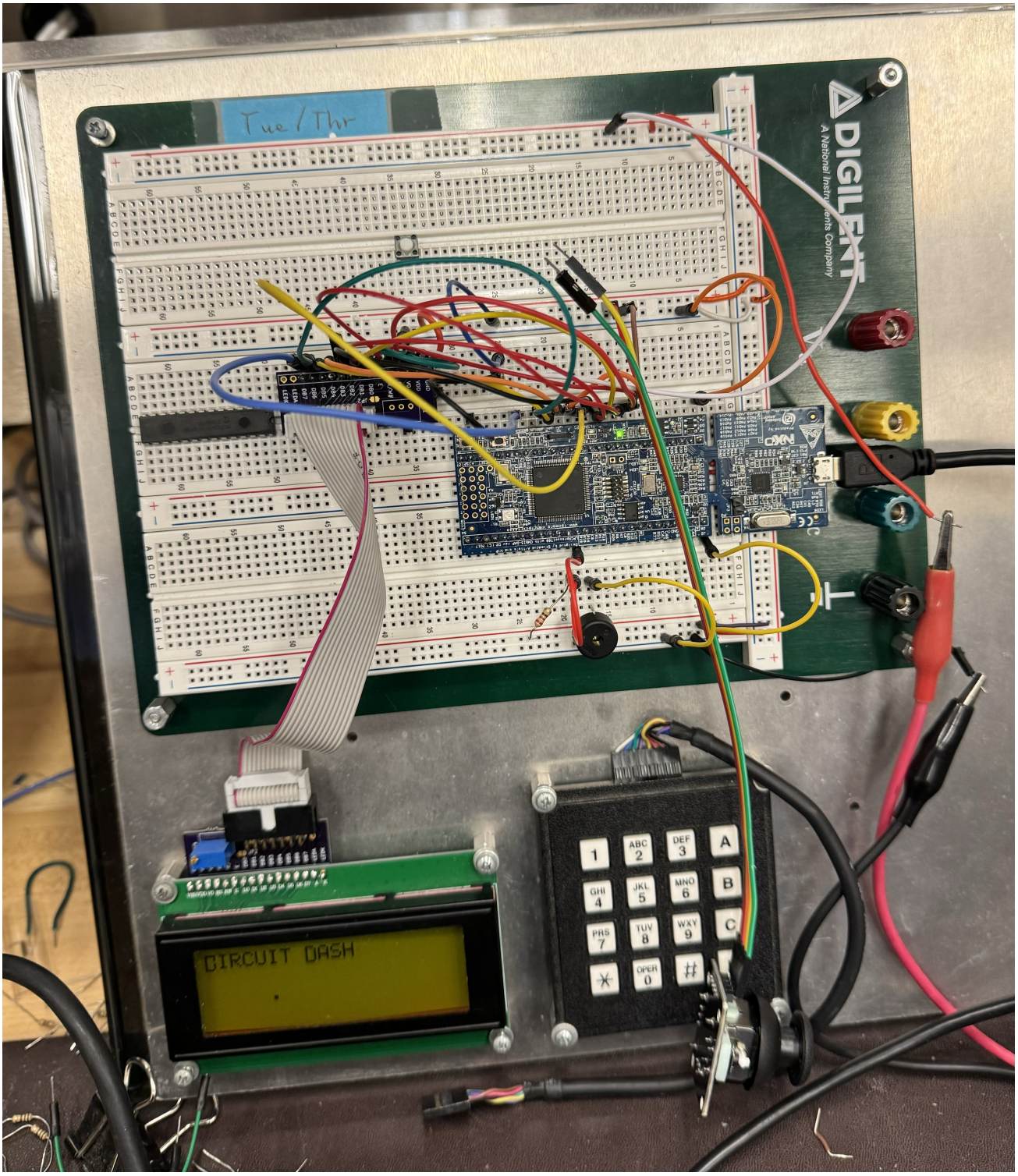

Figure 2: Hardware Picutre

6 Software Section

Software Code For Game:

We got 2.5 points here

```
1
2 #ifdef __USE_CMSIS
3 #include "LPC17xx.h"
4 #endif
5
6 #include <cr_section_macros.h>
7
8 // FOR LCD
9 #define FIO0DIR (*(volatile unsigned int *)0x2009c000) // Used for E and RS pin of LCD
      Display
10 #define FIO0PIN (*(volatile unsigned int *)0x2009c014)
11
12
13 // FOR ANALOG-TO-DIGITAL CONVERSION
14 #define PCONP (*(volatile unsigned int *) 0x400FC0C4)
15 #define PCLKSEL0 (*(volatile unsigned int *)0x400FC1A8)
16
17 #define AD0CR (*(volatile unsigned int *)0x40034000)
18 #define AD0GDR (*(volatile unsigned int *) 0x40034004)
19
20 #define PINSEL1 (*(volatile unsigned int *)  0x4002C004)
21 #define PINSEL4 (*(volatile unsigned int *)0x4002C010)
22
23 #define PWM1TCR (*(volatile unsigned int *) 0x40018004)
24 #define PWM1MR0 (*(volatile unsigned int *) 0x40018018)
25 #define PWM1MR1 (*(volatile unsigned int *) 0x4001801C)
26 #define PWM1MCR (*(volatile unsigned int *) 0x40018014)
27 #define PWM1PCR (*(volatile unsigned int *) 0x4001804C)
28 #define PWM1LER (*(volatile unsigned int *) 0x40018050)
29
30
31 float dataADC;
32 int count = 0;
33
34 // Port 0.18, 17,15, 16, 25, 26, 21
35 int DB[] = {18, 17, 15, 16, 24, 25, 26, 2};
36
37 // Used to start game
38 int start = 1;
39
40
41 void initializeLCD() {
42
43   // Initializing P0.0 (RS), P0.1 (E), and P0.6 (R/W) to be outputs
44   FIO0DIR |= (1 << 0); // RS
45   FIO0DIR |= (1 << 1); // E
46   FIO0DIR |= (1 << 6); // R/W
47
48   // Drive RS, E, and R/W to Low
49   FIO0PIN &= ~(1 << 0); // RS
50   FIO0PIN &= ~(1 << 1); // E
51   FIO0PIN &= ~(1 << 6); // R/W
52
53
54   // Setting DB Array Elements to be outputs corresponding with DB0 - DB7 pins of LCD
       display
55   for(int i = 0; i < 8; i++) {
56
57     FIO0DIR |= (1 << DB[i]);
58   }
59
60
61   wait_ms(4);
62
```

```
63
64    // Write LCD Commands (0x38, 0x06, 0x0c, 0x01)
65    LCDwriteCommand(0x38); // Use full 8-bit bus
66    LCDwriteCommand(0x06); // Automatically Advances Cursor
67    LCDwriteCommand(0x0c); // Enable Display Only (Cursor Off)
68    LCDwriteCommand(0x01); // Clear Display and Move Cursor to Left
69
70
71    // Wait 4 milliseconds
72    wait_ms(4);
73
74 }
75
76
77 void LCDwriteCommand(int command) {
78
79
80    // Update DB0-DB7
81    for(int i = 0; i < 8; i++) {
82
83      if((command >> i) & 1) {
84        FIO0PIN |= (1 << DB[i]);
85      }
86      else {
87        FIO0PIN &= ~(1 << DB[i]);
88      }
89    }
90
91    // Drive R/W to Low
92    FIO0PIN &= ~(1 << 6); // R/W
93
94    // Drive RS Low
95    FIO0PIN &= ~(1 << 0); // RS
96
97    // Drive E to High then Low
98    FIO0PIN |= (1 << 1);
99    wait_ms(0.1);
100   FIO0PIN &= ~(1 << 1);
101
102   // Wait 100 microseconds (0.1 ms)
103   wait_ms(0.1);
104
105 }
106
107
108 void LCDwriteData(int data) {
109
110   // Update DB0-DB7 to match ACII Code
111   for(int i = 0; i < 8; i++){
112
113     if((data >> i) & 1){
114       FIO0PIN |= (1 << DB[i]);
115     }
116
117     else{
118       FIO0PIN &= ~(1 << DB[i]);
119     }
120   }
121
122   // Drive R/W to Low
123   FIO0PIN &= ~(1 << 6);
124
125   // Drive RS High
126   FIO0PIN |= (1 << 0);
127
128   // Drive E High then Low
129   FIO0PIN |= (1 << 1);
130   wait_ms(0.1);
131   FIO0PIN &= ~(1 << 1);
132
133   // Wait 100 microseconds (0.1 ms)
```

```
  wait_ms(0.1);

}


initializeCustom() {

  // Custom Character C for "Circuit"
  LCDwriteCommand(0x40);
  LCDwriteData(0b11111); //Byte 1
  LCDwriteData(0b10000); //Byte 2
  LCDwriteData(0b10000);
  LCDwriteData(0b10000);
  LCDwriteData(0b10000);
  LCDwriteData(0b10000);
  LCDwriteData(0b10000);
  LCDwriteData(0b11111); //Byte 8

  LCDwriteCommand(0x80); // Position 0x80
  //Show character 0
  LCDwriteData(0x00);



  // Display "ircuit" with normal characters
  LCDwriteCommand(0x81);
  LCDwriteData(0x49); // I

  LCDwriteCommand(0x82);
  LCDwriteData(0x52); // R

  LCDwriteCommand(0x83);
  LCDwriteData(0x43); // C

  LCDwriteCommand(0x84);
  LCDwriteData(0x55); // U

  LCDwriteCommand(0x85);
  LCDwriteData(0x49); // I

  LCDwriteCommand(0x86);
  LCDwriteData(0x54); // T

  LCDwriteCommand(0x87);
  LCDwriteData(0x10); // Space between words


  // Custom Character D for "Dash"
  LCDwriteCommand(0x48);
  LCDwriteData(0b11110); //Byte 1
  LCDwriteData(0b10001); //Byte 2
  LCDwriteData(0b10001);
  LCDwriteData(0b10001);
  LCDwriteData(0b10001);
  LCDwriteData(0b10001);
  LCDwriteData(0b10001);
  LCDwriteData(0b11110); //Byte 8
  LCDwriteCommand(0x80);

  LCDwriteCommand(0x88); // Put in position 0x88
  //Show character 1
  LCDwriteData(0x01);



  //LCDwriteCommand(0x88);
  //LCDwriteData(0x44); // D

  LCDwriteCommand(0x89);
  LCDwriteData(0x41); // A

```

```
  LCDwriteCommand(0x8A);
  LCDwriteData(0x53); // S

  LCDwriteCommand(0x8B);
  LCDwriteData(0x48); // H


}

// Function to create a delay in milliseconds
void wait_ms(float ms) {

  volatile float i;

  float m = 0.002719;
  float b = 0.1;

    ms = (ms-b)/m;

    for (i = 0; i < ms; i++) {
    //do nothing
    }
}


void initializeConvert() {

  // Setting the A/D Converter power/clock control bit (Turning A-to-D on)
  PCONP |= (1 << 12); // Setting the A/D Converter power/clock control bit (Turning A-
    to-D on)

  // Enable the ADC by setting PDN Bit in AD0CR Register
  AD0CR |= (1 << 21);

  // Selecting AD0.0
  PINSEL1 |= (1 << 14);

  // Set AD0.0 to Input
  AD0CR |= (1 << 0);

  // Disable Burst Bit (Set to 0)
  AD0CR &= ~(1 << 16);


}


void initializePWM() {

  // PWM Initializing as down in Lecture Slides
  PWM1TCR = 0;
  PCLKSEL0 |= (1<<12); // PWM PCLK = CCLK/1
  PWM1MR0 = 80000; // 256 PCLK cycle period (8-bit equivalent)
  PINSEL4 |= (1<<0); // Configure P2.0 as PWM1.1
  PWM1MCR = (1<<1); // Reset counter on MR0 match
  PWM1PCR = (1<<9); // Single edge mode and enable PWM1.1 only
  PWM1TCR = (1<<0) | (1<<3); // Enable counter and PWM mode


}


unsigned int readConversion() {

  // Start A-to-D Conversion
  AD0CR |= (1 << 24);


  // Wait for Done bit to go to 1
  while(!((AD0GDR >> 31) & 1)) {
    // Do nothing
```

```
275   }
276
277   // Clear A-to-D Start
278   AD0CR &= ~(1 << 24);
279
280   // Read Converted Result
281   int data = ((AD0GDR >> 4) & 0xFFF);
282
283   return data;
284
285 }
286
287
288 int main(void) {
289
290
291   FIO0DIR &= (1 << 28); // Joystick Switch set to Input
292
293
294   // Initialize Display
295   initializeLCD();
296
297
298   // Display "Circuit Dash" on LCD
299   initializeCustom();
300
301
302   // Initialize ADC and PWM
303   initializeConvert();
304   initializePWM();
305
306   // Array to go through positions of 3rd row to make objects moves across LCD
307   int objectPos[20] = {0xA7, 0xA6, 0xA5, 0xA4, 0xA3, 0xA2, 0xA1, 0xA0, 0x9F, 0x9E, 0
       x9D, 0x9C, 0x9B, 0x9A, 0x99,
308       0x98, 0x97, 0x96, 0x95, 0x94};
309
310   int ballJump[2] = {0x99, 0xC5};
311
312   int ball = 0x2E; // Ball Hex Code
313   int obstacle = 0x5B; // Obstacle Character
314   int empty = 0x10; //Empty Character
315
316   int ifJump = 0; // Check if ball is in middle of jump
317
318
319     // Enter an infinite loop, just incrementing a counter
320     while(1) {
321
322       // Conversion and Calculation for PWM
323       dataADC = readConversion();
324       int checkJump = ((204.75 + 0.05 * dataADC) * 80000/4095);
325       PWM1LER = (1 << 1);
326       wait_ms(1);
327
328     // Fill 99 with Ball Character
329     LCDwriteCommand(ballJump[0]);
330     LCDwriteData(ball); // Ball Character
331     int onGround = 1;
332
333
334     LCDwriteCommand(objectPos[count]);
335     LCDwriteData(obstacle);
336     wait_ms(5);
337
338
339     // Using Joystick to make Ball Jump
340     if(checkJump == 4000) {
341
342
343       // Empty 99 (Ground) with Empty Character
344       LCDwriteCommand(ballJump[0]);
```

```
345         LCDwriteData(empty); // Empty Character
346         wait_ms(10);
347         onGround = 0;
348 
349         // Fill C5 (Above) with Ball Character
350         LCDwriteCommand(ballJump[1]);
351         LCDwriteData(ball); // Ball Character
352         wait_ms(10);
353 
354         // Indicate Ball is in air
355         onGround = 0;
356 
357         // Update to 1 to indicate ball is jumping
358         ifJump = 1;
359 
360       }
361 
362       // Continue Writing Objects across Display
363       LCDwriteCommand(objectPos[count]);
364       LCDwriteData(empty);
365       wait_ms(1);
366 
367 
368       // Keep Ball in Air if Player Jumped
369       if(ifJump == 1) {
370 
371         // Fill C5 (Above) with Ball Character
372         LCDwriteCommand(ballJump[1]);
373         LCDwriteData(ball); // Ball Character
374         wait_ms(10);
375       }
376 
377 
378       LCDwriteCommand(objectPos[count + 1]);
379       LCDwriteData(obstacle);
380       wait_ms(1);
381 
382 
383 
384       // Keep Ball in Air if Player Jumped
385       if(ifJump == 1) {
386 
387         // Fill C5 (Above) with Ball Character
388         LCDwriteCommand(ballJump[1]);
389         LCDwriteData(ball); // Ball Character
390         wait_ms(10);
391 
392       }
393 
394 
395       LCDwriteCommand(objectPos[count + 1]);
396       LCDwriteData(empty);
397       wait_ms(1);
398 
399       // Keep Ball in Air if Player Jumped
400       if(ifJump == 1) {
401 
402         // Fill C5 (Above) with Ball Character
403         LCDwriteCommand(ballJump[1]);
404         LCDwriteData(ball); // Ball Character
405         wait_ms(10);
406 
407 
408       }
409 
410 
411       LCDwriteCommand(objectPos[count + 2]);
412       LCDwriteData(obstacle);
413       wait_ms(1);
414 
415       LCDwriteCommand(objectPos[count + 2]);
```

```
    LCDwriteData(empty);
    wait_ms(1);

    LCDwriteCommand(objectPos[count + 3]);
    LCDwriteData(obstacle);
    wait_ms(1);

    LCDwriteCommand(objectPos[count + 3]);
    LCDwriteData(empty);
    wait_ms(1);

    if(ifJump == 1) {

      // Empty C5 with Empty Character
      LCDwriteCommand(ballJump[1]);
      LCDwriteData(empty); // Empty Character
      wait_ms(10);

      // Update to 0 to indicate ball is no longer Jumping
      ifJump = 0;
    }

    count++;

    if(count > 18) {
      count = 0;
    }

    }
    return 0;

}
```

Software Code for Music:

We got 0.5 points here

```

#include <stdint.h>

#define FIO2DIR (*(volatile unsigned int *)0x2009c040)
#define FIO2PIN (*(volatile unsigned int *)0x2009c054)
#define PCONP (*(volatile unsigned int *)0x400fc0c4)
#define T0TCR (*(volatile unsigned int *)0x40004004)
#define T0TC (*(volatile unsigned int *)0x40004008)

// Define note
typedef struct {
    int frequency;
    int duration;
} Note;

// Define an array of notes
Note notes[] = {
     {659 ,  200}, // E5
     {659 ,  200}, // E5
     {880 ,  200}, // A5
     {784 ,  200}, // G5
     {698 ,  200}, // F#5
     {698 ,  200}, // F#5
     {784 ,  200}, // G5
     {698 ,  200}, // F#5
     {659 ,  200}, // E5
     {587 ,  200}, // D5
     {587 ,  200}, // D5
     {659 ,  200}, // E5
     {698 ,  200}, // F#5
     {784 ,  200}, // G5
```

```
    {880 ,  200}, // A5
    {987 ,  400}, // B5
};

// Function to play a tone on the buzzer
void playTone(int frequency, int duration) {
    int halfPeriodMicros = 1000000 / (frequency * 2);

    for (int i = 0; i < duration * 2; i++) {
      // piezo on
        FIO2PIN |= (1 << 0);
        delayMicroseconds(halfPeriodMicros);

        // piezo off
        FIO2PIN &= ~(1 << 0);
        delayMicroseconds(halfPeriodMicros);
    }
}

// Functions
void playTone(int frequency, int duration);
void delayMilliseconds(int ms);
void delayMicroseconds(int us);

// will generate milliseconds
void delayMilliseconds(int ms) {
    delayMicroseconds(ms * 1000);
}

// will generate microseconds
void delayMicroseconds(int us) {
    int start = T0TC;
    T0TCR |= (1 << 0);
    while ((T0TC - start) < us) {}
}

int main(void) {
    PCONP |= (1 << 22);
    FIO2DIR |= (1 << 0); // will set as input

    while (1) {
        // play the note in array
        for (int i = 0; i < sizeof(notes) / sizeof(Note); i++) {
            playTone(notes[i].frequency, notes[i].duration);
            delayMilliseconds(50);
        }
        delayMilliseconds(1000);
    }

    return 0;
}
```

7 Contributions

Carson Wagner (Computer Engineering)

Worked on code for the Circuit Dash game, implementing LCD display and joystick control logic. Also, worked on final document and poster.

Roberto Solis (Electrical Engineering)

Worked on Music section of the code. Also worked on final document and poster.