Yu-De Chen
Henry Doyle

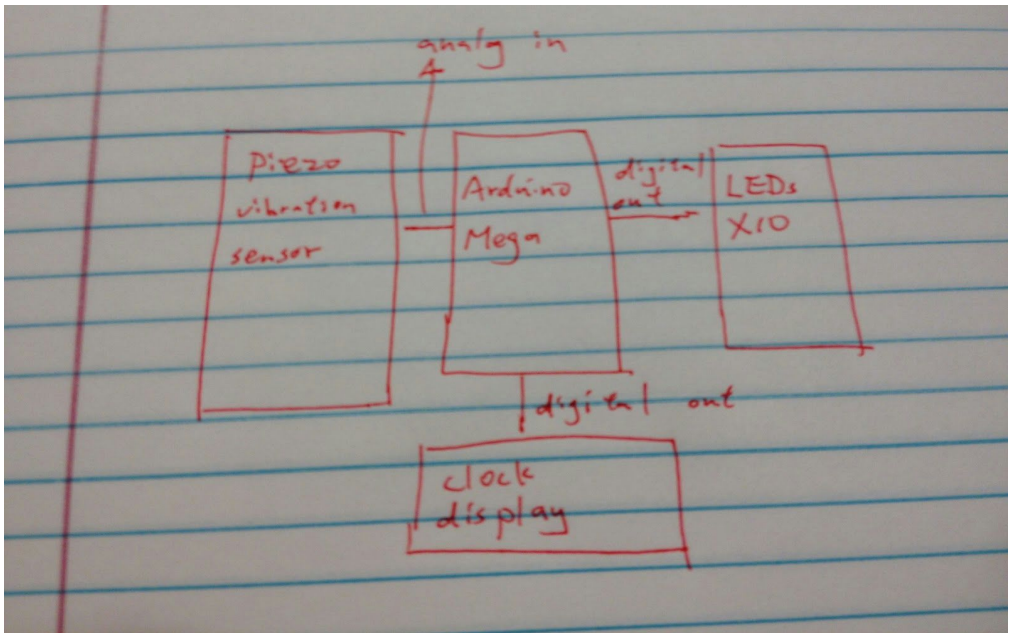
Ultimate Drummers

Introduction:

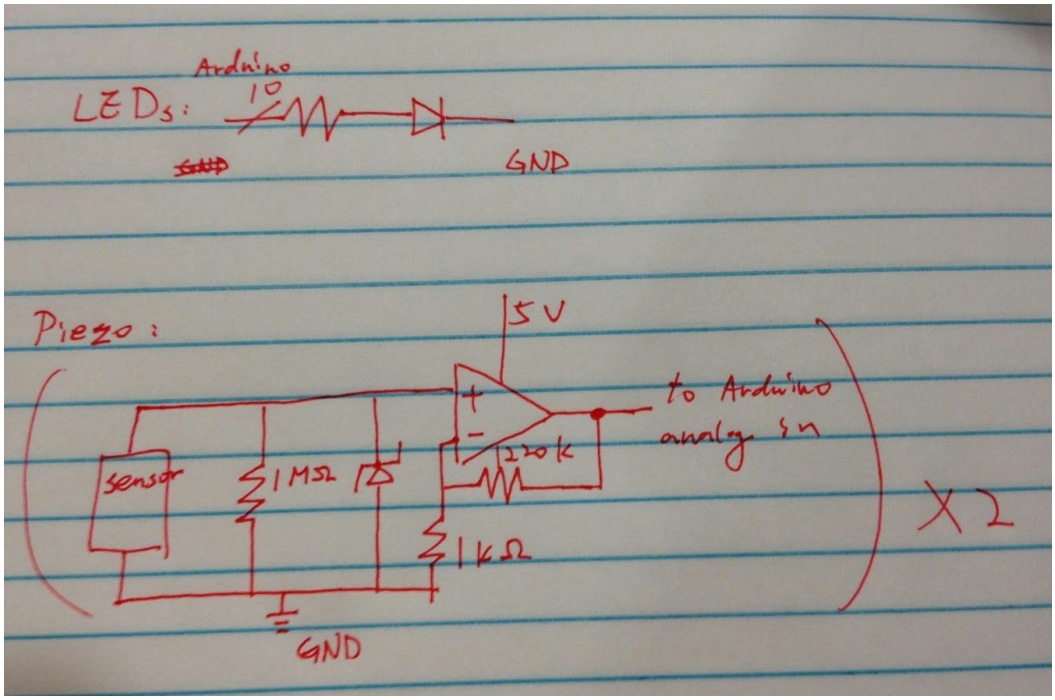
We began our project with the intention of creating an interactive rhythm game that reflects a player's reaction time. Our game consists of two drums, each of which is associated with a string of lights that light up either randomly or according to a pattern. The drums are constructed with Jell-O boxes, with piezo vibration sensors attached to sponges inside. The player should hit the drum when the final light in the string is illuminated, and his/her score will be calculated based on his/her accuracy, ranging from zero to five. The score is displayed on an LCD display, and it accumulates as the game continues. A maximum possible score is 200 points.

Design:

BLOCK DIAGRAMS AND THEIR DESCRIPTIONS

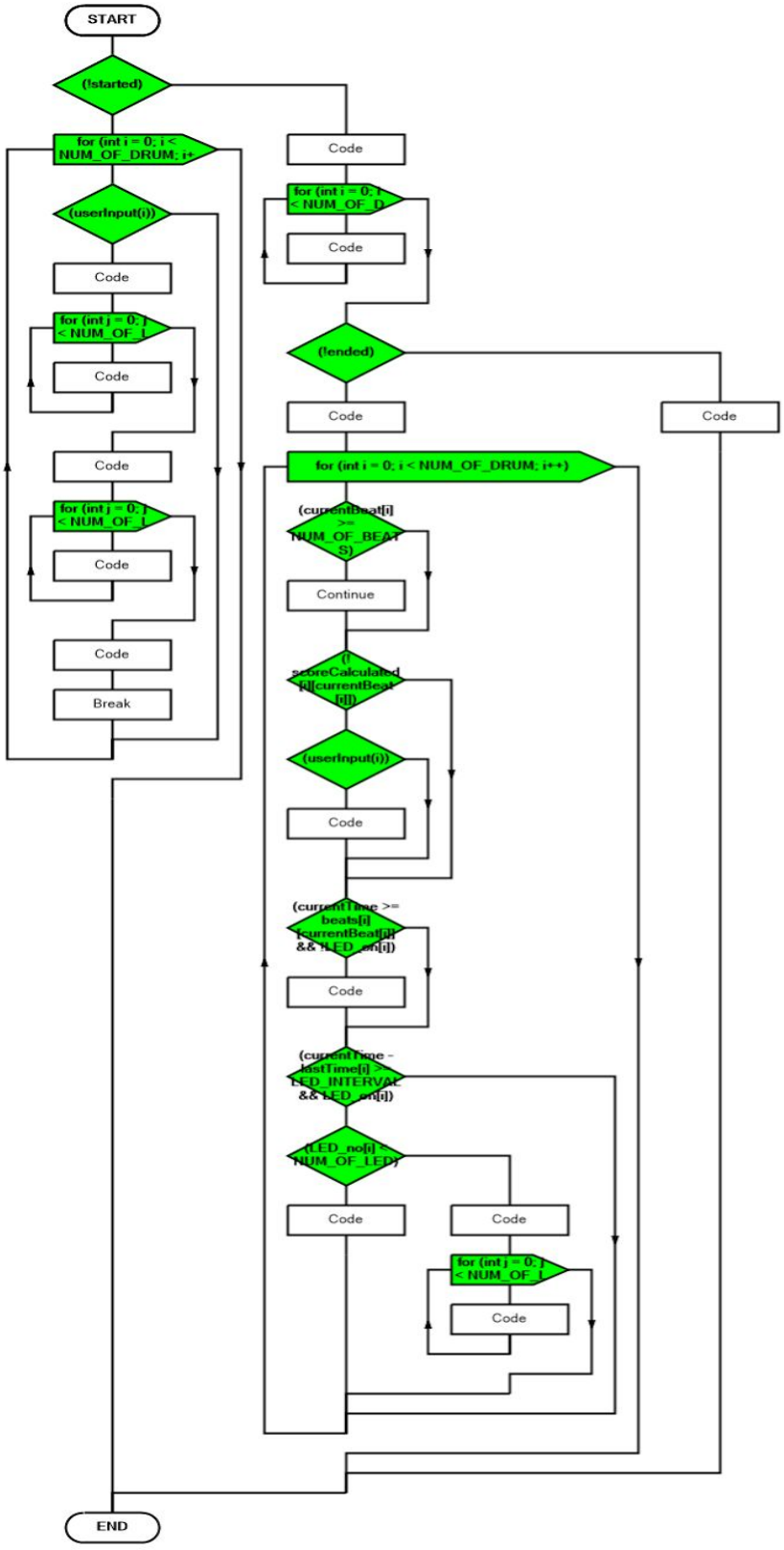


CIRCUIT DIAGRAM



PICTURE OF WHOLE PROJECT

FLOW CHART OF SOFTWARE



Code:

```
#define SEGMENT_A 22 // pin     4
#define SEGMENT_B 31 // pin 26
#define SEGMENT_C 27 // pin 22
#define SEGMENT_D 26 // pin     8
#define SEGMENT_E 25 // pin     7
#define SEGMENT_F 32 // pin 27
#define SEGMENT_G 30 // pin 25

#define DIGIT_1 24 // pin   6
#define DIGIT_2 23 // pin   5
#define DIGIT_3 28 // pin 23
#define DIGIT_4 29 // pin 24

/* global variables */

bool started = false;
bool ended = false;

/* clock display */

int LED_left[5];
int LED_right[5];

const int DIGIT_L = 4;
const int MAX_DIG = 3;

int digitPin[DIGIT_L] = {DIGIT_4, DIGIT_3, DIGIT_2, DIGIT_1};
int digit[DIGIT_L];

unsigned int score = 0;

/* end clock display */

/* sensor */

const int NUM_OF_DRUMS = 2;
```

```
int vibrationDigitalIn[NUM_OF_DRUMS] = {A0, A1};

unsigned long lastBeat[NUM_OF_DRUMS];
int onAnl[NUM_OF_DRUMS] = {550, 550};
bool isOn[NUM_OF_DRUMS];

int currentVol[NUM_OF_DRUMS];

/* end sensor */

/* LEDs */

const int NUM_OF_LED = 5;
const int NUM_OF_DRUM = 2;
const int NUM_OF_BEATS = 20;
const int LED_INTERVAL = 75;

int LED_pin[NUM_OF_DRUM][NUM_OF_LED];

int beats[NUM_OF_DRUM][NUM_OF_BEATS];
int currentBeat[NUM_OF_DRUM];
int LED_no[NUM_OF_DRUM];

bool LED_on[NUM_OF_DRUM];
bool scoreCalculated[NUM_OF_DRUM][NUM_OF_BEATS];

unsigned long startTime, currentTime, lastTime[NUM_OF_DRUM];

/* end LEDs */

/* global variables */

/* functions */

/* clock display */

void printOneDigit(int a) {
```

```
digitalWrite(SEGMENT_A, HIGH);
digitalWrite(SEGMENT_B, HIGH);
digitalWrite(SEGMENT_C, HIGH);
digitalWrite(SEGMENT_D, HIGH);
digitalWrite(SEGMENT_E, HIGH);
digitalWrite(SEGMENT_F, HIGH);
digitalWrite(SEGMENT_G, HIGH);

switch(a) {
        case 0:
                digitalWrite(SEGMENT_A, LOW);
                digitalWrite(SEGMENT_B, LOW);
                digitalWrite(SEGMENT_C, LOW);
                digitalWrite(SEGMENT_D, LOW);
                digitalWrite(SEGMENT_E, LOW);
                digitalWrite(SEGMENT_F, LOW);
                break;
        case 1:
                digitalWrite(SEGMENT_B, LOW);
                digitalWrite(SEGMENT_C, LOW);
                break;
        case 2:
                digitalWrite(SEGMENT_A, LOW);
                digitalWrite(SEGMENT_B, LOW);
                digitalWrite(SEGMENT_D, LOW);
                digitalWrite(SEGMENT_E, LOW);
                digitalWrite(SEGMENT_G, LOW);
                break;
        case 3:
                digitalWrite(SEGMENT_A, LOW);
                digitalWrite(SEGMENT_B, LOW);
                digitalWrite(SEGMENT_C, LOW);
                digitalWrite(SEGMENT_D, LOW);
                digitalWrite(SEGMENT_G, LOW);
                break;
        case 4:
                digitalWrite(SEGMENT_B, LOW);
                digitalWrite(SEGMENT_C, LOW);
                digitalWrite(SEGMENT_F, LOW);
```

```
        digitalWrite(SEGMENT_G, LOW);
        break;
    case 5:
        digitalWrite(SEGMENT_A, LOW);
        digitalWrite(SEGMENT_C, LOW);
        digitalWrite(SEGMENT_D, LOW);
        digitalWrite(SEGMENT_F, LOW);
        digitalWrite(SEGMENT_G, LOW);
        break;
    case 6:
        digitalWrite(SEGMENT_A, LOW);
        digitalWrite(SEGMENT_C, LOW);
        digitalWrite(SEGMENT_D, LOW);
        digitalWrite(SEGMENT_E, LOW);
        digitalWrite(SEGMENT_F, LOW);
        digitalWrite(SEGMENT_G, LOW);
        break;
    case 7:
        digitalWrite(SEGMENT_A, LOW);
        digitalWrite(SEGMENT_B, LOW);
        digitalWrite(SEGMENT_C, LOW);
        break;
    case 8:
        digitalWrite(SEGMENT_A, LOW);
        digitalWrite(SEGMENT_B, LOW);
        digitalWrite(SEGMENT_C, LOW);
        digitalWrite(SEGMENT_D, LOW);
        digitalWrite(SEGMENT_E, LOW);
        digitalWrite(SEGMENT_F, LOW);
        digitalWrite(SEGMENT_G, LOW);
        break;
    case 9:
        digitalWrite(SEGMENT_A, LOW);
        digitalWrite(SEGMENT_B, LOW);
        digitalWrite(SEGMENT_C, LOW);
        digitalWrite(SEGMENT_D, LOW);
        digitalWrite(SEGMENT_F, LOW);
        digitalWrite(SEGMENT_G, LOW);
        break;
```

```
        }
}

void clockOff() {
        printOneDigit(-1);
        for (int i = 0; i < DIGIT_L; i++) {
                digitalWrite(digitPin[i], LOW);
        }
}

void printToClock() {
        for (int i = 0; i < MAX_DIG; i++) {
                digitalWrite(digitPin[i], HIGH);
                printOneDigit(digit[i]);
                clockOff();
        }
}

void updateClockData(int a) {
        for (int i = 0 ; i < 4; i++) digit[i] = -1;
        int i = 0;
        do {
                digit[i++] = a % 10;
                a /= 10;
                if (i == DIGIT_L) break;
        } while (a != 0);
}

/* end clock display */

/* sensor */

bool userInput(int i) {
        currentVol[i] = analogRead(vibrationDigitalIn[i]);
        if(currentVol[i] >= onAnl[i] && millis() - lastBeat[i] > 100 && !isOn[i]){
                lastBeat[i] = millis();
                isOn[i] = true;
                return true;
        } else {
```

```
                isOn[i] = false;
                return false;
        }
}

/* end sensor */

/* end functions */

void setup() {

        // setup clock display

        for (int i = 22; i <= 32; i++) {
                pinMode(i, OUTPUT);
        }
        clockOff();
        for (int i = 0; i < DIGIT_L; i++) {
                digit[i] = -1;
        }

        for (int i = 0; i < 10; i++) {
                pinMode(i, OUTPUT);
                if (i % 2 == 0) {
                        LED_left[i / 2] = i;
                } else {
                        LED_right[i / 2] = i;
                }
        }
        updateClockData(score);

        // setup sensor

        for (int i = 0; i < NUM_OF_DRUMS; i++) {
                pinMode(vibrationDigitalIn[i], INPUT);
                lastBeat[i] = 0;
                isOn[i] = false;
        }
```

```
        // setup LEDs

        for (int i = 0; i < NUM_OF_DRUM; i++) {
                for (int j = 0; j < NUM_OF_LED; j++) {
                        LED_pin[i][j] = j * 2 + i;
                }
        }

        for (int i = 0; i < NUM_OF_DRUM; i++) {
                for (int j = 0; j < NUM_OF_LED; j++) {
                        pinMode(LED_pin[i][j], OUTPUT);
                        digitalWrite(LED_pin[i][j], LOW);
                }
        }

        for (int i = 0; i < NUM_OF_DRUM; i++) {
                for (int j = 0; j < NUM_OF_BEATS; j++) {
                        scoreCalculated[i][j] = false;
                }
        }

        beats[0][0] = 500;
        beats[1][0] = 1000;
        for (int i = 0; i < NUM_OF_DRUM; i++) {
                for (int j = 1; j < NUM_OF_BEATS; j++) {
                        beats[i][j] = beats[i][j - 1] + random(0, 10) * 250;
                }
        }

        for (int i = 0; i < NUM_OF_DRUM; i++) {
                currentBeat[i] = 0;
                LED_no[i] = 0;
                LED_on[i] = false;
                lastTime[i] = 0;
        }

}

void loop() {
```

```
if (!started) {
	for (int i = 0; i < NUM_OF_DRUM; i++) {
		if (userInput(i)) {
			started = true;
			for (int j = 0; j < NUM_OF_LED; j++) {
				digitalWrite(LED_pin[i][j], HIGH);
			}
			delay(LED_INTERVAL);
			for (int j = 0; j < NUM_OF_LED; j++) {
				digitalWrite(LED_pin[i][j], LOW);
			}
			startTime = millis();
			break;
		}
	}
} else {

	printToClock();

	ended = true;
	for (int i = 0; i < NUM_OF_DRUM; i++) {
		ended = ended && (currentBeat[i] >= NUM_OF_BEATS);
	}

	if (!ended) {

		currentTime = millis() - startTime;

		for (int i = 0; i < NUM_OF_DRUM; i++) {

			if (currentBeat[i] >= NUM_OF_BEATS) continue;

			if (!scoreCalculated[i][currentBeat[i]]) {
				if (userInput(i)) {
					score += LED_no[i] + 1;
					updateClockData(score);
					scoreCalculated[i][currentBeat[i]] = true;
					printToClock();
```

```
                                        }
                                    }

                                    if (currentTime >= beats[i][currentBeat[i]] &&
!LED_on[i]) {
                                        LED_on[i] = true;
                                    }
                                    if (currentTime - lastTime[i] >= LED_INTERVAL &&
LED_on[i]) {
                                        if (LED_no[i] < NUM_OF_LED) {
                                            lastTime[i] = currentTime;
                                            digitalWrite(LED_pin[i][LED_no[i]],
HIGH);
                                            LED_no[i]++;
                                        } else {
                                            LED_no[i] = 0;
                                            LED_on[i] = false;
                                            currentBeat[i]++;
                                            for (int j = 0; j < NUM_OF_LED; j++) {
                                                digitalWrite(LED_pin[i][j], LOW);
                                            }
                                        }
                                    }
                                } // end for i

                            } else {
                                digitalWrite(LED_pin[0][0], HIGH);
                            } // end if ended

                        } // end if !started

                    } // end loop
```

Results:

Our game ended up functioning quite successfully. When turned on, the game is ready to be played, and only needs to be started by hitting one of the drums. Once the game begins, the string of five LED lights light up one by one at random intervals, and when you hit the correct corresponding drum, a score is outputted to the LCD display.

Future Work:

Going forward, there are various improvements we could make to our game. For one, we could implement more drums. This would be relatively simple. All we would need to do is recreate the circuit for the additional drum, and then add another string of LEDs. The code was written in such a way that we can change the number of LED sets in the system by simply changing one number in the code. Another fun edition would be to add a bonus points function to the game. This would involve a period at the end of the game when the user simply hits the drums as fast as he/she can, and the more times he/she hits the drum, the more points he/she gets. We could also potentially implement sound into our game, so a sound is created when one hits the drum.

Conclusion:

At the end of the day, our game was a complete success. Everything we set out to accomplish was accomplished in the final design. However, as mentioned above, there is always room for improvement for our design. This project taught us a lot about Arduino, and the code associated with it, as well as working with LCD displays. We also gained valuable exposure to the Piezo vibration sensor, and after playing around with it, we learned how to decrease its sensitivity and get it to operate at an ideal level.