Micro Processors

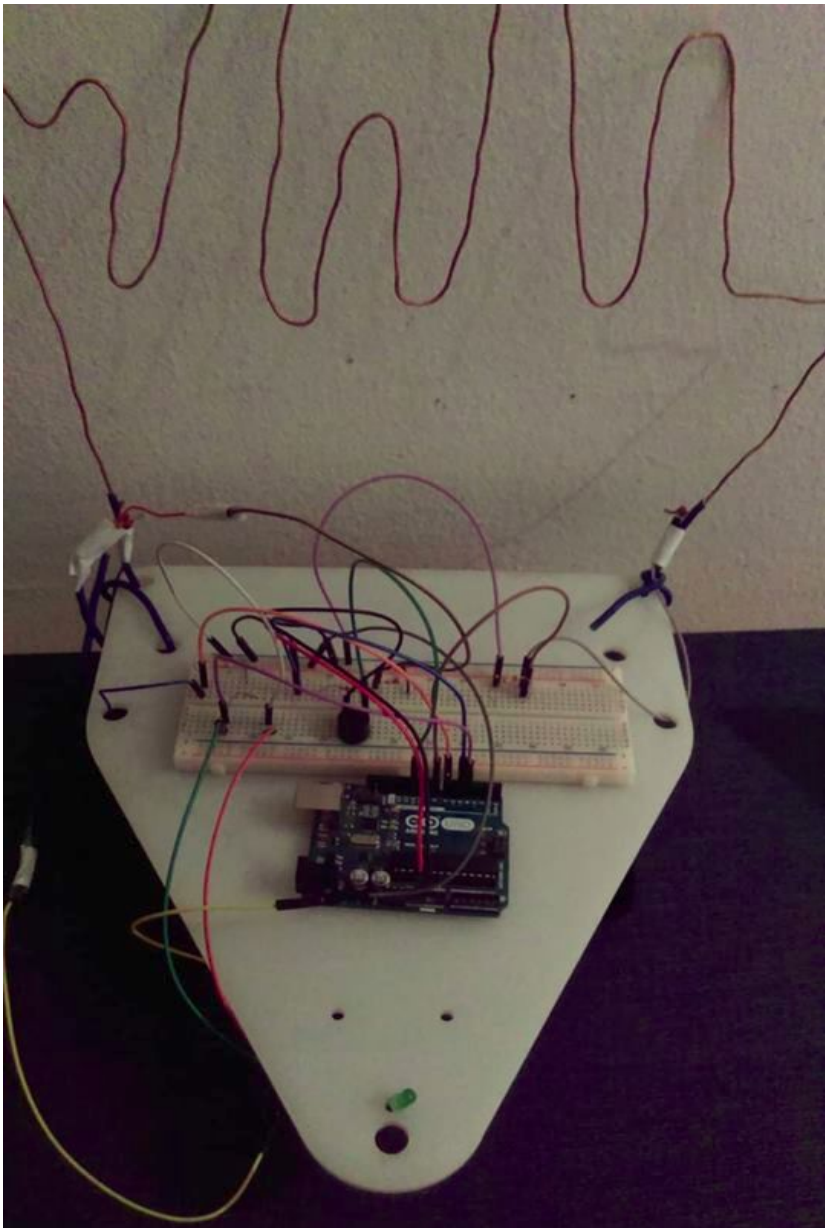
Arduino Project

Buzz-wire Game

Team Members

Name	ID
Alaa Shehab	1
Aya Ashraf	2
Bassant Ahmed	19
Sohayla Mohammed	32

Introduction



Description and Overview

Buzz-wire is a steady hand game that is well known to many as a table top amusement. Buzz wire is a challenging and competitive game where you are

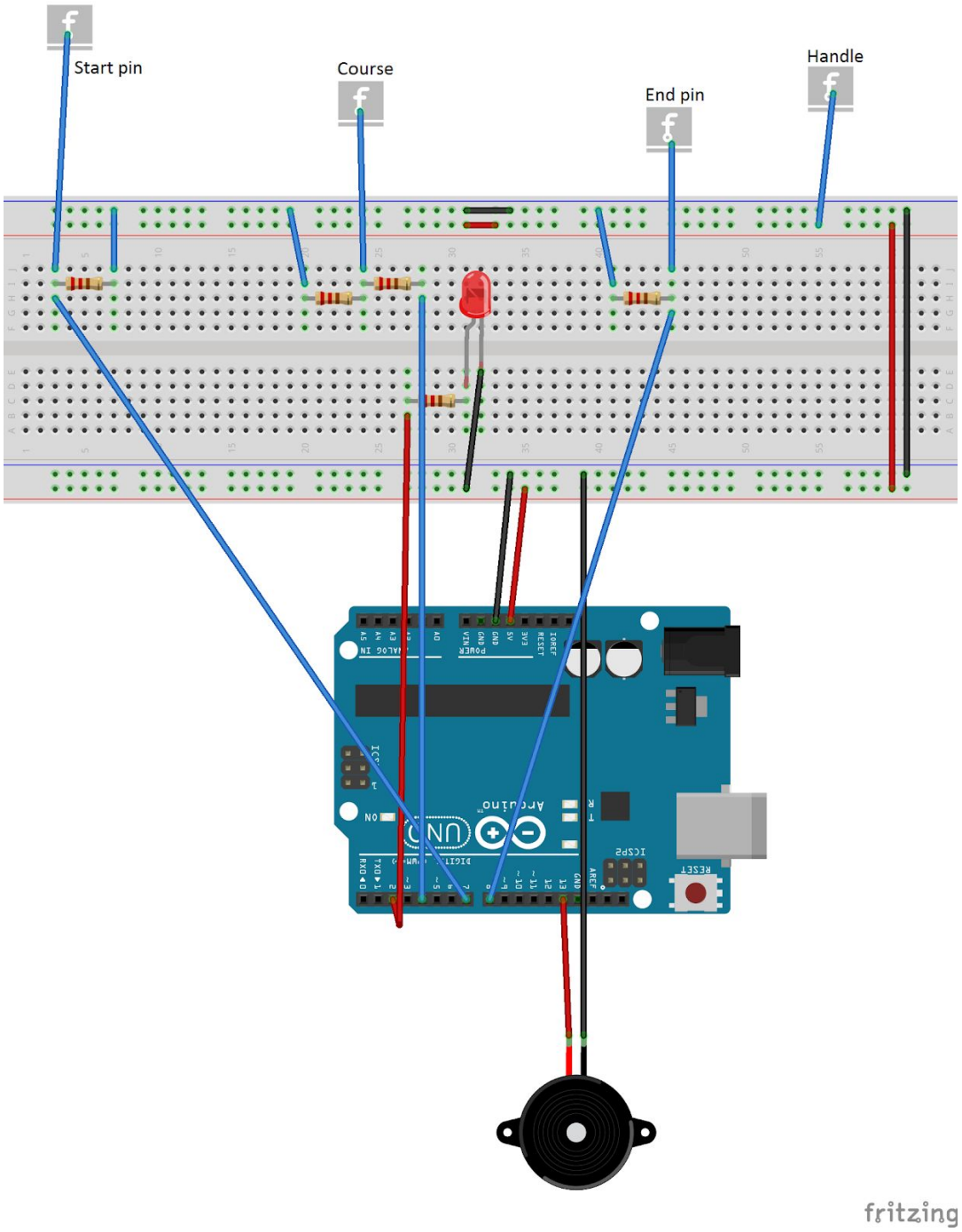
playing the number of touches against time. The player has to get the right balance between speed and skill in order to obtain the winning score.

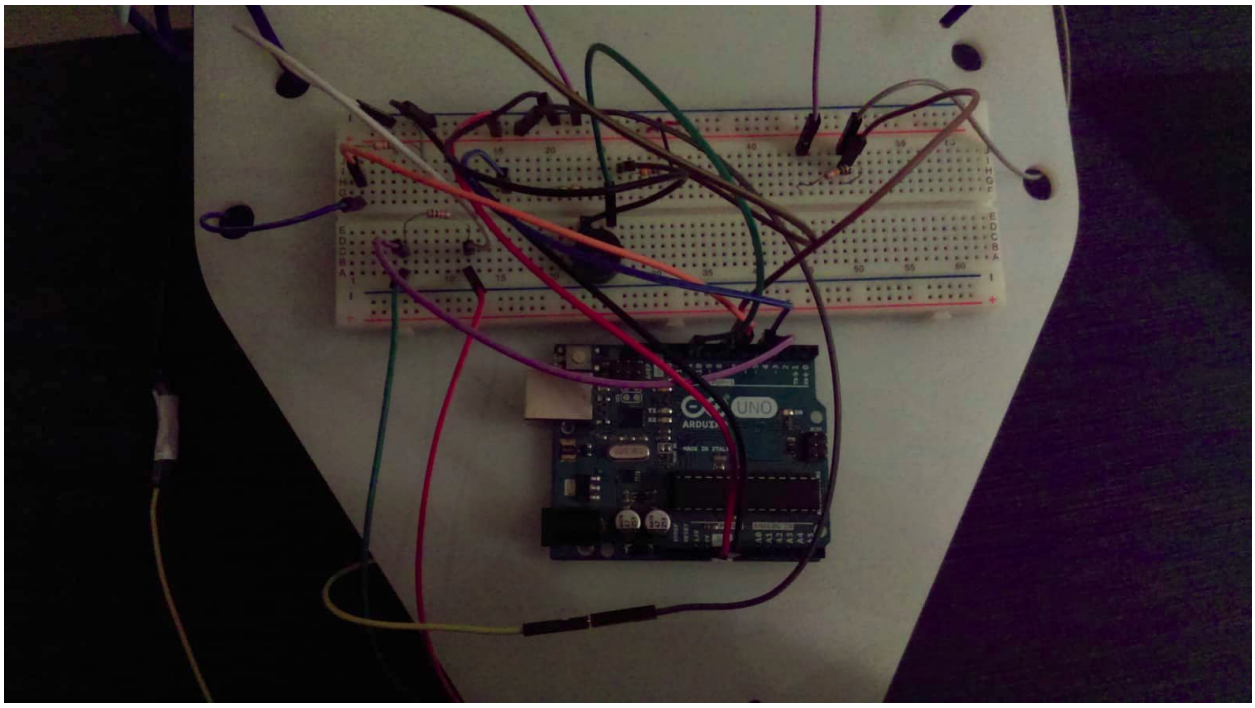
The core mechanic consists of a wire shape and a loop on a handle. The player has to guide the loop around the course without the two touching. If the two touch, the circuit is completed and the buzzer sounds.

Components

- 1 x Arduino UNO.
    - Acts as the microcontroller to control the game.
    - The game can be made without a microcontroller.
- Light pulp & piezo buzzer
    - Indicates whether or not the handle has touched the course.
- 2mm tube
    - To shape the course.
    - To mark the beginning and end of the game.
- 5-320 Ohm Resistors, connected to led, course, start and end wires.
- Breadboard
- Male-Male and Female-Male wires used for internal connections.
- Plastic stand, that acts as holder and base for the circuit and the course.

Schematics





Code

```
#include "pitches.h" // include pitches


// music
int songState = 0;

int melody[] = {
 NOTE_F4, NOTE_E4, NOTE_D4, NOTE_CS4,
 NOTE_C4, NOTE_B3, NOTE_AS3, NOTE_A3,
 NOTE_G3, NOTE_A3, NOTE_AS3, NOTE_A3,
 NOTE_G3, NOTE_C4, 0,

 NOTE_C4, NOTE_A3, NOTE_A3, NOTE_A3,
 NOTE_GS3, NOTE_A3, NOTE_F4, NOTE_C4,
 NOTE_C4, NOTE_A3, NOTE_AS3, NOTE_AS3,
 NOTE_AS3, NOTE_C4, NOTE_D4, 0,

 NOTE_AS3, NOTE_G3, NOTE_G3, NOTE_G3,
 NOTE_FS3, NOTE_G3, NOTE_E4, NOTE_D4,
 NOTE_D4, NOTE_AS3, NOTE_A3, NOTE_A3,
 NOTE_A3, NOTE_AS3, NOTE_C4, 0,
```

```
 NOTE_C4, NOTE_A3, NOTE_A3, NOTE_A3,
 NOTE_GS3, NOTE_A3, NOTE_A4, NOTE_F4,
 NOTE_F4, NOTE_C4, NOTE_B3, NOTE_G4,
 NOTE_G4, NOTE_G4, NOTE_G4, 0,

 NOTE_G4, NOTE_E4, NOTE_G4, NOTE_G4,
 NOTE_FS4, NOTE_G4, NOTE_D4, NOTE_G4,
 NOTE_G4, NOTE_FS4, NOTE_G4, NOTE_C4,
 NOTE_B3, NOTE_C4, NOTE_B3, NOTE_C4, 0
};

int tempo[] = {
 8, 16, 8, 16,
 8, 16, 8, 16,
 16, 16, 16, 8,
 16, 8, 3,

 12, 16, 16, 16,
 8, 16, 8, 16,
 8, 16, 8, 16,
 8, 16, 4, 12,

 12, 16, 16, 16,
 8, 16, 8, 16,
 8, 16, 8, 16,
 8, 16, 4, 12,

 12, 16, 16, 16,
 8, 16, 8, 16,
 8, 16, 8, 16,
 8, 16, 4, 16,

 12, 17, 17, 17,
 8, 12, 17, 17,
 17, 8, 16, 8,
 16, 8, 16, 8, 1
};
//pins used
const int buzzer = 11;
const int startPin = 7;
const int endPin = 8;
const int touchPin = 5;
const int touchLed = 4;
// music
unsigned long previousMillis = 0; // time last changed
const long interval2 = 100; // interval between notes
```

```
//game states
int startGame = 0;
int endGame = 0;
int touchingNo = 0;
long timeTaken = 0;
int playing = 0;


void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 pinMode(touchPin, INPUT); // setup circuit
 pinMode(buzzer, OUTPUT); // setup buzzer 2
 pinMode(touchLed, OUTPUT);
 pinMode(startPin, INPUT); // setup button
 pinMode(endPin, INPUT); // setup button
digitalWrite(touchLed,LOW);

}

void loop() {

startGame = digitalRead(startPin);

endGame = digitalRead(endPin);
 // Serial.println(endGame);

if (startGame) {
  playing = 1;
  touchingNo = 0;
  timeTaken = millis();
          buzz(0, 1000/3);
          delay(75);

//  Serial.println("Started the game");
  //display start game
} else if (endGame) {
  playing = 0;
  buzz(0, 1000/3);
  //display score
   Serial.println(touchingNo);

}else if (playing){ //playing state
  //start sing
   if(digitalRead(touchPin) == HIGH) {
   delay(25);
```

```
  if(digitalRead(touchPin) == HIGH) {
        while(digitalRead(touchPin) == HIGH) {
        buzz(NOTE_B0, 1000/24);
        digitalWrite(touchLed,HIGH);
        touchingNo++;
        Serial.println(touchingNo);

        delay(25);
        }
  }
 } else if (playing) {
        digitalWrite(touchLed,LOW);
        sing();
 }
 //check
//   Serial.println("Playing");


}
}

void buzz(long frequency, long length) {

 long delayValue = 1000000/frequency/2; // calculate the delay value between transitions
 //// 1 second's worth of microseconds, divided by the frequency, then split in half since
 //// there are two phases to each cycle
 long numCycles = frequency * length/ 1000; // calculate the number of cycles for proper
timing
 //// multiply frequency, which is really cycles per second, by the number of seconds to
 //// get the total number of cycles to produce
 for (long i=0; i < numCycles; i++){ // for the calculated length of time...
  digitalWrite(buzzer,HIGH); // write the buzzer pin high to push out the diaphragm
  delayMicroseconds(delayValue); // wait for the calculated delay value
  digitalWrite(buzzer,LOW); // write the buzzer pin low to pull back the diaphragm
  delayMicroseconds(delayValue); // wait again for the calculated delay value
 }
}

void sing() {
 // play the song in a non blocking way
 unsigned long currentMillis = millis();

 if (currentMillis - previousMillis >= interval2) {
  previousMillis = currentMillis;
  int noteDuration = 1000 / tempo[songState];
  buzz(melody[songState], noteDuration);
  int pauseBetweenNotes = noteDuration;
```

```
  delay(pauseBetweenNotes);

  // stop the tone playing:
  buzz(0, noteDuration);

  ++songState;
  // start song again if finished
  if(songState > 79) {
  songState = 14; // skip intro
  }
 }
}
```

Debouncing

- We're using a debouncing code in keeping track of number of touches made.
- For a touch to be calculated, a time interval of 1 second has to pass between the last calculated touch and current time.

Music

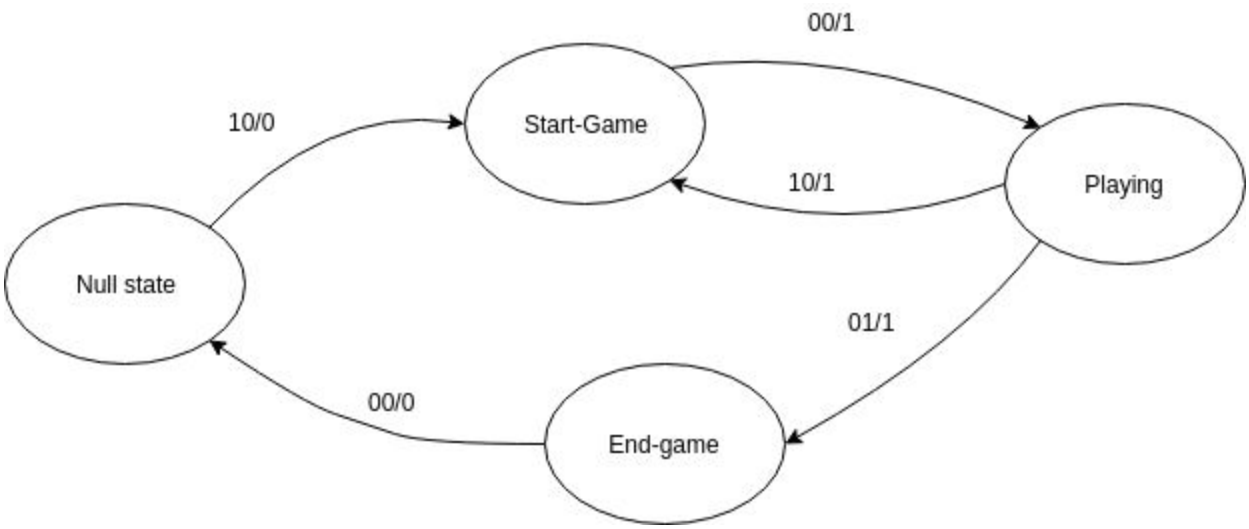
- Music is composed by giving different frequencies to the buzzer.
- This is done with the help of "pitch.h" library
    - Each music note is mapped to a different frequency.
    - Composing the music piece is done by storing its notes in a data structure, with an additional data structure to assign different tempos for each note
    - Notes are using the piezo buzzer by giving each note a duration, number of cycles and a different tempo.

Score

- Score is calculated as a function of the time taken and number of touches made during the game.
    - $Score = \frac{1}{Total\ Time} \times 1000 - 2 \times No.\ of\ touches\ made$

Game-states

States based on pins (start “a”, end ”b”), and variable “playing, c” -> ab/c



States

- NULL
    - The game is idle.
- Start-game
    - The game is reset and the player is ready to start the course traversing.
- Playing
    - The player is traversing the course.
    - Keeping track of time and number of touches.
- End-game
    - The player has finished traversing the game.
    - The score is calculated.

State transitions

- Null-Start
    - Start pin becomes high when touched, end pin remains low and player variable remains zero (10/0).
    - The game is reset and player is ready to start a new game.
- Start-Playing
    - Start pin becomes low, end pin remains low and playing variable becomes 1.

    - A new game has started.
- Playing-end
    - Start pin remains low, end pin becomes high and playing variable becomes 0.
    - Score is shown.
- End-null
    - Start remains low, end become low and playing remains 0.
    - Game is idle until player make the start pin goes high again.

Assumptions

- To start the game the handle must touch the starting mark to indicate that it's ready.
- Once the music started playing, it means that the game has started.
- To end, the handle must touch the the ending mark
- Once the music has stopped playing it indicates that the game has stopped.
- Interval between touches is 1 sec.

Challenges

- Compose the game song.
- Establishing how the game started and ended as we wanted to make it as interactive as possible.
- Isolating any external effects on the circuit.

Libraries

- Pitch.h
    - Maps each buzzer frequency to a melody.

User Guide

- Game starts on touching the start wire connected the left of the course at the very end.
- Whenever the handle touches the start-wire the game is restarted even if player was in the middle/end of the game and returned, and no score is calculated.
- Player must touch the end-wire - similar to start-wire but on the other side of the course-, where then the score is calculated.
- The serial shows the beginning and end of the game in addition to the score and time taken by player to finish.
- Both a buzzer and a led is used as an indicator to show when the handle has touched the course.

References

- Arduino - Home
- MakeUseOf - Technology, Simplified