ENGR 1
Summary Report

Spring 2018
Final Robot Project

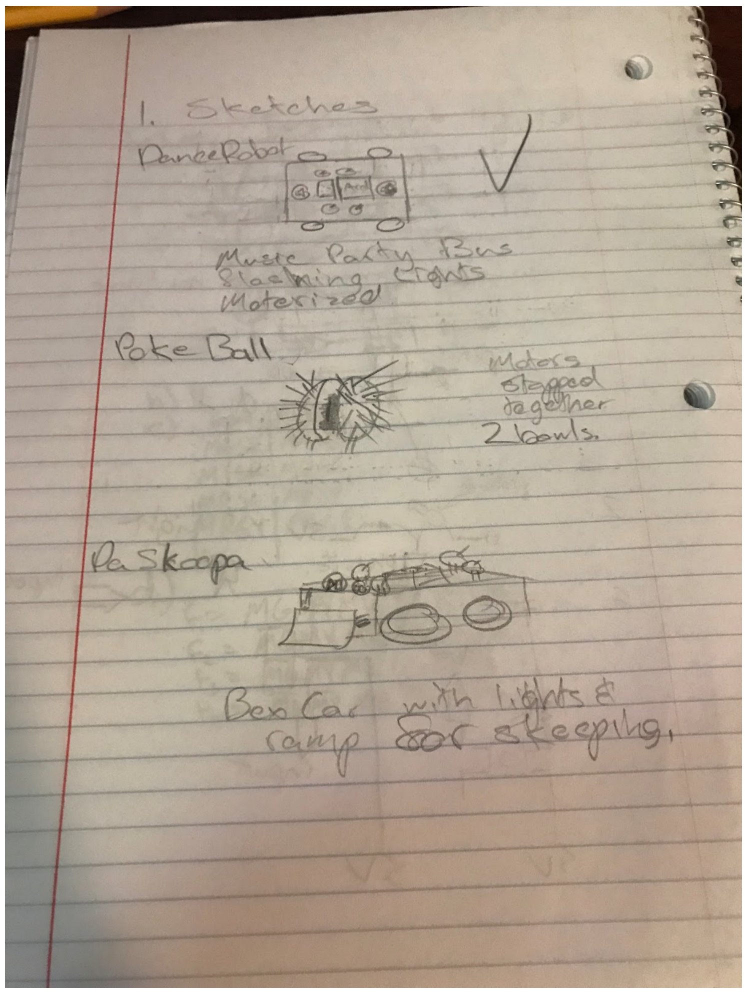
Team Name/Number: Team 14

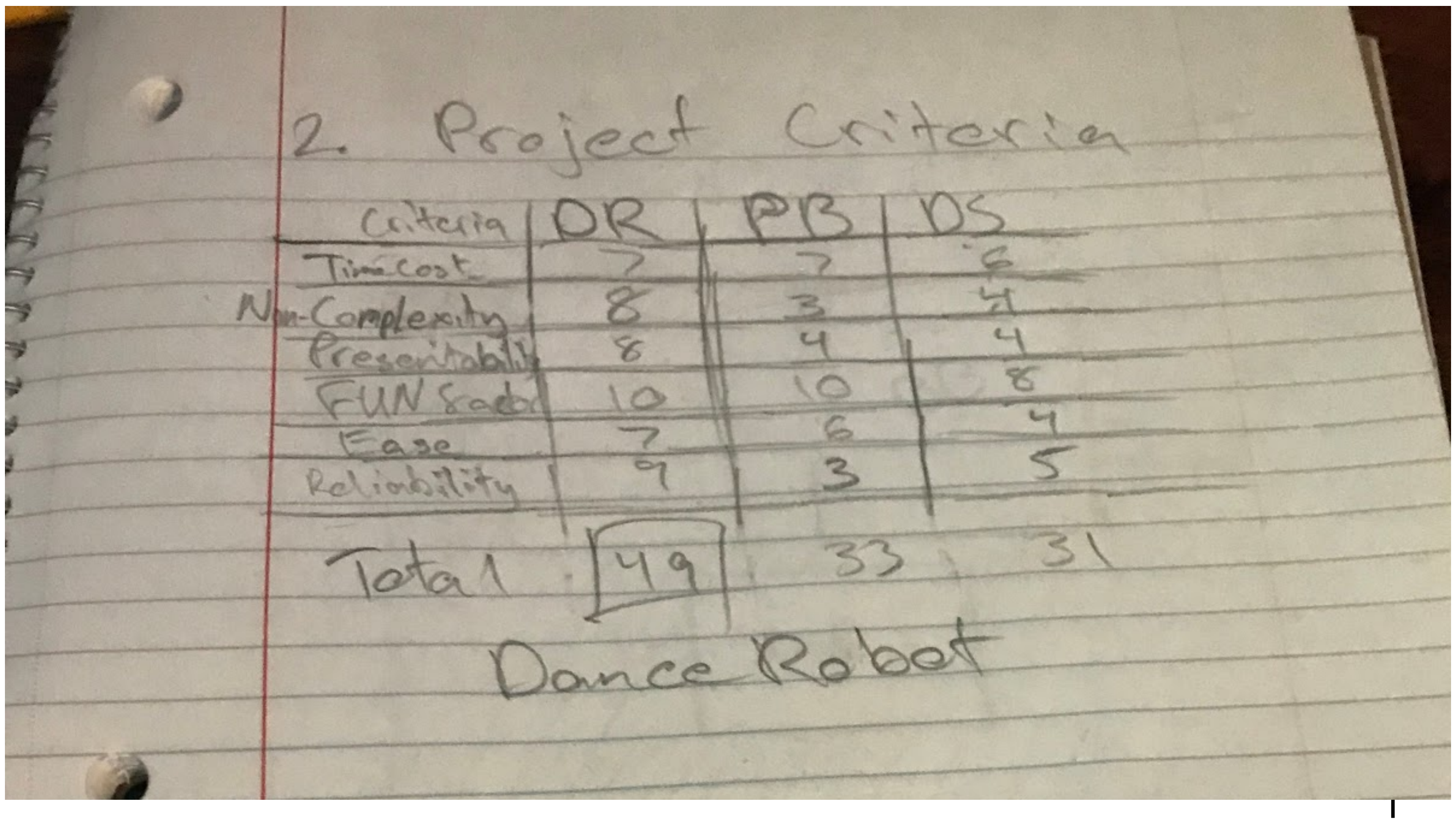
Team Member Names (Last, First): Walter, Steven ; Eide, Sterling ; Arrouye, Theo

Date of submission: 12 June 2018

Rick Roller

These diagrams to the right show our initial 3 design options. The first was a dance bot that would play music, flash LEDs, and dance. The second was a spike ball that could roll around using two rotating hemispheres and spikes that protruded from those hemispheres. Our final design was a scooping machine that used 4 wheels and a front scooper to drive around on command and scoop things up for the user.





2. Project Criteria

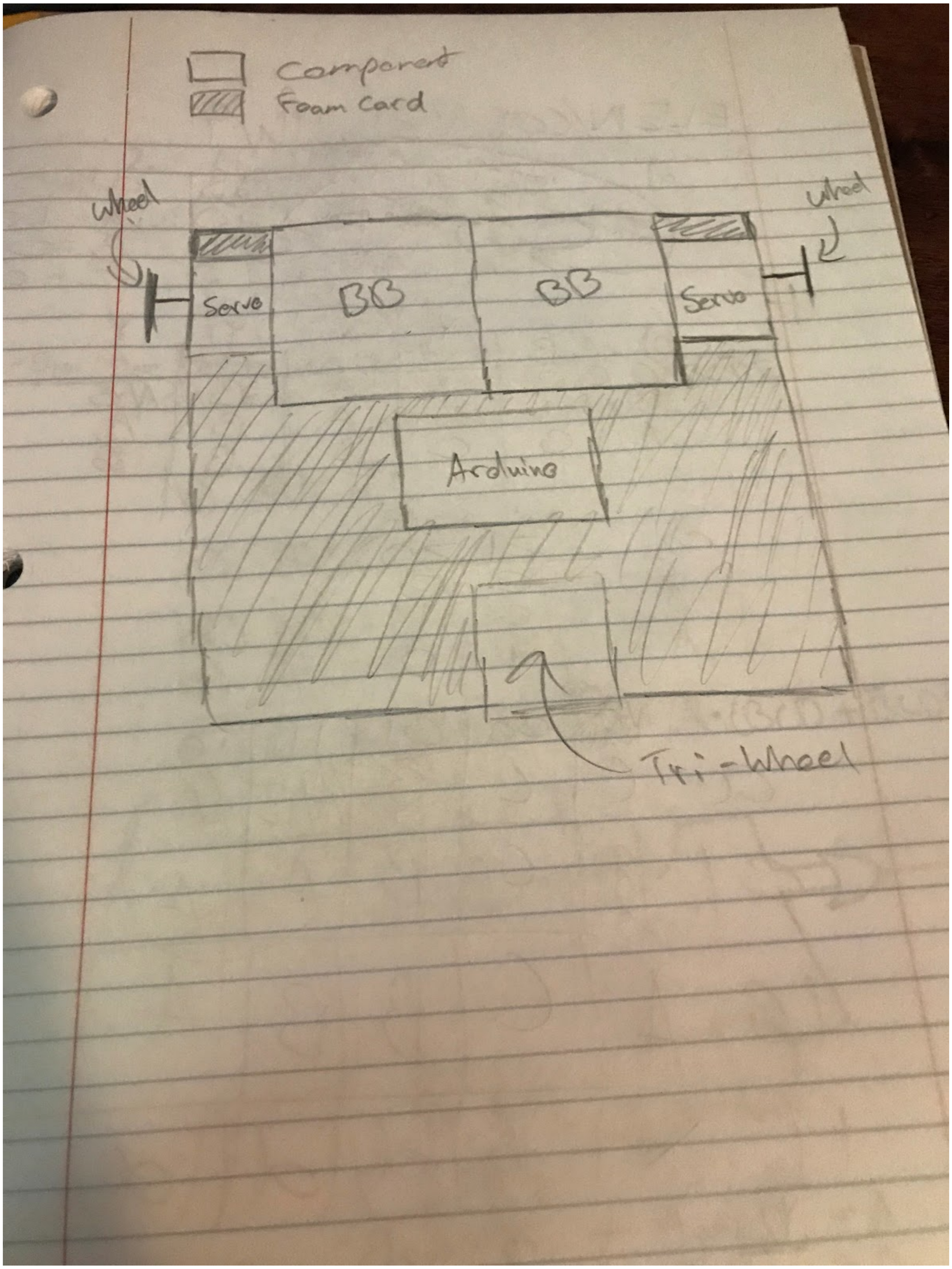
Criteria	DR	PB	DS
Time-cost	7	7	6
Non-Complexity	8	3	4
Presentability	8	4	4
FUN factor	10	10	8
Ease	7	6	4
Reliability	9	3	5
Total	$\boxed{49}$	33	31

Dance Robot

In choosing between our three options we considering 6 important criteria: Time Cost, Simplicity, Presentability, FUN FACTOR, Ease, Reliability.
The Dance machine was much simpler compared to the others as the only physical requirements were the wheels, so there would be little variation in its performance and would be quicker to build and test. With the music and lights and movement, the dance bot would be the flashiest of our three options, and it ranked high on the FUN ratings, because it was a party bus. The ease of using it was also much better than that of the other two options. It's simplicity also provided increased reliability, since only the wheels or the front stand could result in error.

The other designs failed on account of impracticality and simply being to unreliable. The Poke-Ball would have been impossible to get functioning correctly for movement seeing as it was a radical design choice that wouldn't hold up by use of cardboard and duct tape. The Scoop machine had a different problem where it just wasn't interesting compared to the Dance Machine, lacking lights and sound. It also had an extra physical component in the scooper, which would decrease reliability as the additional component would increase the chance of a physical malfunction.

With the Dance Machine chosen, scoring a 49 on our criteria a good 16 points higher than the closest other design, we had to optimize what it was to do and check the practicality of our goals:
How can we make a robot that inspires a fun and interactive atmosphere? The party bus.
This design included lights, multiple songs it could play, and dance routines that it knew. We eventually went away from this idea because we found it did not have enough user interaction to be fully enticing, as well providing 4 songs overcomplicated the required circuitry and sounded hectic. We modified out idea to still use lights and play a song, but the dance would be choreographed by the user.


Component
Foam Card
Wheel
Wheel
Servo
BB
BB
Servo
Arduino
Tri-Wheel

Appendix:

Table of Minor Parts

Item	Quantity	Description	Cost
-1	47	Wires	N/A -ENGR1 loan
00	--	Cardboard/Glue/Tape	Found/Borrowed

Table of Parts

Item	Quantity	Description	Cost
01	2	Servo Motor	N/A - ENGR 1 loan
02	4	LEDs	N/A - ENGR 1 loan
03	2	Tone Buzzer	N/A - ENGR 1 loan
04	2	Breadboard	N/A - ENGR 1 loan
05	5	10k Ohm Resistor	N/A - ENGR 1 loan
06	1	Arduino	N/A - ENGR 1 loan
07	1	Battery Snap	N/A - ENGR 1 loan
08	5	Push Button	N/A - ENGR 1 loan
09	4	200 Ohm resistor	N/A - ENGR 1 loan
10	4	9V Battery	$12.29
		Sum of all parts- Total Cost	$12.29

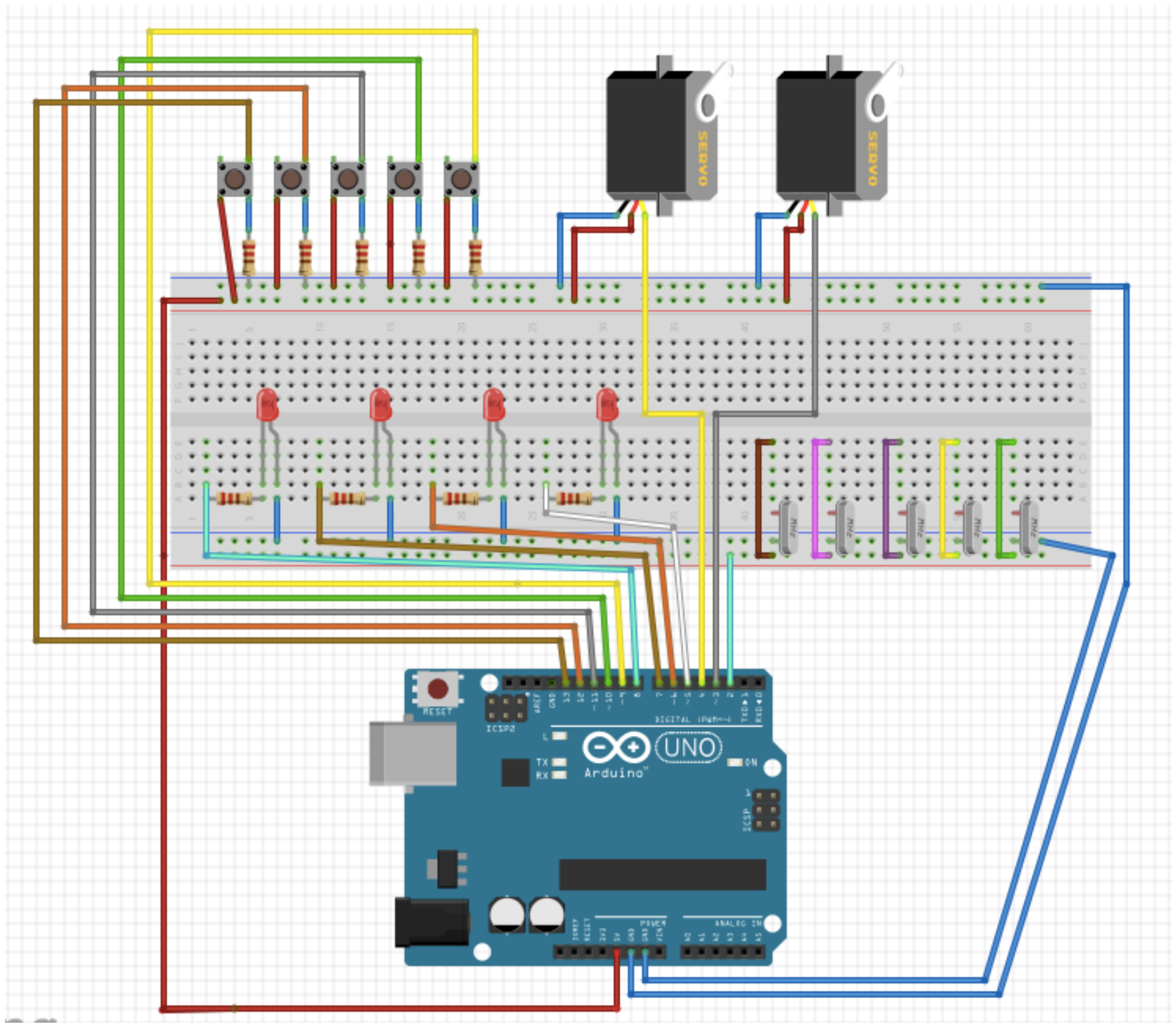


Fig _. Above is the circuit diagram for the Rick Roller created with Fritzing.All resistors are 220 Ω.

Table A2: Team member project time log

Team Member	Time spent on project (include individual and team time)	
	In-class (any lab periods attended)	Out-of-class (non-lab periods)
Sterling Eide	9 hrs	7 hrs.
Theo Arrouye	9 hrs	
Steven Walter	9 hrs	

Commented computer code for RickRoller:

```
#include <Servo.h>
```

```
//defining music notes
#define  a3f    208     // 208 Hz
#define  b3f    233     // 233 Hz
#define  b3     247     // 247 Hz
#define  c4     261     // 261 Hz MIDDLE C
#define  c4s    277     // 277 Hz
#define  e4f    311     // 311 Hz
#define  f4     349     // 349 Hz
#define  a4f    415     // 415 Hz
#define  b4f    466     // 466 Hz
#define  b4     493     // 493 Hz
#define  c5     523     // 523 Hz
#define  c5s    554     // 554 Hz
#define  e5f    622     // 622 Hz
#define  f5     698     // 698 Hz
#define  f5s    740     // 740 Hz
#define  a5f    831     // 831 Hz

#define rest    -1

// buzzer
int piezo = 8;


volatile int beatlength = 100; // determines tempo
float beatseparationconstant = 0.3;

int b; // song index

int flag; // play/pause

//setting up melody for song
int song1_chorus_melody[] =
{ b4f, b4f, a4f, a4f,
  f5, f5, e5f, b4f, b4f, a4f, a4f, e5f, e5f, c5s, c5, b4f,
  c5s, c5s, c5s, c5s,
  c5s, e5f, c5, b4f, a4f, a4f, a4f, e5f, c5s,
  b4f, b4f, a4f, a4f,
  f5, f5, e5f, b4f, b4f, a4f, a4f, a5f, c5, c5s, c5, b4f,
  c5s, c5s, c5s, c5s,
  c5s, e5f, c5, b4f, a4f, rest, a4f, e5f, c5s, rest
};

//setting rhythm for song
int song1_chorus_rhythmn[] =
{ 1, 1, 1, 1,
  3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
  1, 1, 1, 1,
  3, 3, 3, 1, 2, 2, 2, 4, 8,
  1, 1, 1, 1,
  3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
  1, 1, 1, 1,
  3, 3, 3, 1, 2, 2, 2, 4, 8, 4, 5
};

//defining wheels
Servo LWheel;
Servo RWheel;

//setting wheel pins
int LWheelPosition = 5;
int RWheelPosition = 6;

//direction to turn Left wheel
int LFor = 0;
int LBack = 180;

//directions to turn right wheel
int RFor = 180;
int RBack = 0;

//directions to stop wheels
int StopL = 97;
int StopR = 95;

//pins for buttons
int button1 = 13;
int button2 = 12;
int button3 = 11;
int button4 = 10;
int button5 = 9;

//buttonstate initiation
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;
int buttonState5 = 0;

//a few counters
int i = 0;
int x = 0;
int z = 0;

//array to store commands
int pattern[11] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

//pins for LEDs
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 7;


void setup()
```

```
{
  //setting up buzzer
  pinMode(piezo, OUTPUT);

  //intiating some counters
  Serial.begin(9600);
  flag = 0;
  b = 0;

  //attaching wheels
  LWheel.attach(LWheelPosition);
  RWheel.attach(RWheelPosition);

  //setting up pin modes
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
  pinMode(button3, INPUT);
  pinMode(button4, INPUT);
  pinMode(button5, INPUT);

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);

  Serial.begin(9600);

  //stoping wheels when first turned on
  LWheel.write(StopL);
  RWheel.write(StopR);
}

void loop()
{
  //reading buttons
  buttonState1 = digitalRead(button1);
  buttonState2 = digitalRead(button2);
  buttonState3 = digitalRead(button3);
  buttonState4 = digitalRead(button4);
  buttonState5 = digitalRead(button5);

  //delay so 1 press isn't counted multiple times
  delay(200);

  //Serial.print("Beginning ");

  if(buttonState1 == HIGH && i != 10)
  {
    Serial.print("1 ");
    //setting variable on array to 1
    pattern[i] = 1;
    i = i + 1;
    //turning on LED
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);

  }
  if(buttonState2 == HIGH && i != 10)
  {
    Serial.print("2 ");
    //setting variable on array to 2
    pattern[i] = 2;
    i = i + 1;
    //turning on LED
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);

  }
  if(buttonState3 == HIGH && i != 10)
  {
    Serial.print("3 ");
    //setting variable on array to 3
    pattern[i] = 3;
    i = i + 1;
    //turning on LED
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
  }
  if(buttonState4 == HIGH && i != 10)
  {
    Serial.print("4 ");
    //setting variable on array to 4
    pattern[i] = 4;
    i = i + 1;
    //turning on LED
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, HIGH);
  }
  if(buttonState5 == HIGH || i == 10)
  {
    Serial.print("5 ");
    z = 0;
    x = 0;

    while( z == 0){
    int notelength;
      //inputing next movemet command if on a specific note so that each movement is ~ 1 sec
    if(b == 0 || b == 6 || b == 11 || b == 15 || b == 21 || b == 26 || b == 29 || b == 35 || b == 40 || b == 44 || b ==
  59)
    {
      //checking for what move command to do
      if(pattern[x] == 1)
      {
        //driving forward
        LWheel.write(LFor);
        RWheel.write(RFor);
        //turning on LED
        digitalWrite(LED1, HIGH);
        digitalWrite(LED2, LOW);
        digitalWrite(LED3, LOW);
        digitalWrite(LED4, LOW);
        //incrementing to next move command
        x++;

      }
      else if(pattern[x] == 2)
      {
        //driving backward
        LWheel.write(LBack);
        RWheel.write(RBack);
        //incrementing to next move command
        x++;
        //turning on LED
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, HIGH);
        digitalWrite(LED3, LOW);
        digitalWrite(LED4, LOW);

      }

      else if(pattern[x] == 3)
      {
        //turning left
        LWheel.write(LBack);
        RWheel.write(RFor);
        //incrementing to next move command
        x++;
        //turning on LED
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, LOW);
        digitalWrite(LED3, HIGH);
        digitalWrite(LED4, LOW);
      }

      else if(pattern[x] == 4)
      {
        //turning right
        LWheel.write(LFor);
        RWheel.write(RBack);
        //incrementing to next move command
```

```
        x++;
        //turning on LED
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, LOW);
        digitalWrite(LED3, LOW);
        digitalWrite(LED4, HIGH);

      }
      else
      {
        //stopping
        LWheel.write(StopL);
        RWheel.write(StopR);
        //incrementing to next move command
        x++;
        //turning off LED
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, LOW);
        digitalWrite(LED3, LOW);
        digitalWrite(LED4, LOW);
      }
    }

    //playing chorus
    notelength = beatlength * song1_chorus_rhythmn[b]; // fining not length based on rhythm
    //if max number of commands is done
    if (x > 10) {
      //stop wheels
      LWheel.write(StopL);
      RWheel.write(StopR);
      //get out of while loop
      z = 1;
      // set movement array to all 0s
      for( int k = 0; k < 11; k++)
      {
        pattern[k] = 0;
      }

    }
    else
    {
      //play note
      tone(piezo, song1_chorus_melody[b], notelength);
    }
    //incrment note
    b++;

  delay(notelength); // necessary because piezo is on independent timer
  noTone(piezo);
  delay(notelength * beatseparationconstant); // create separation between notes
}
//reset counters
i = 0;
```

```
  x = 0;
  b = 0;
  }
}
```