SDP Team 34: Hand Surfer

University *of* Massachusetts Amherst

Team Members

Sidney Kim - Computer Engineering

Joshua Silva - Computer Engineering

Haozhe Shu - Computer Engineering

Problem Statement

In today's gaming market, rhythm games suffer from limitations in accessibility. Also, after the COVID pandemic, concerns about hygiene and public health have become more magnified than ever. Rhythm games are often confined to arcades and rely on non conventional controllers that can contribute to germ transmission. Our project, Hand Surfer will address these issues by providing a gaming experience with a non physical interface, eliminating the concern of germ spread.

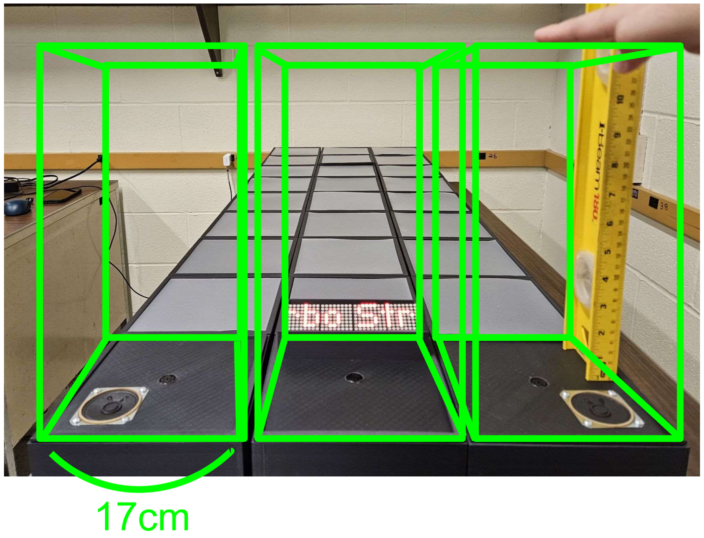
Project Goals

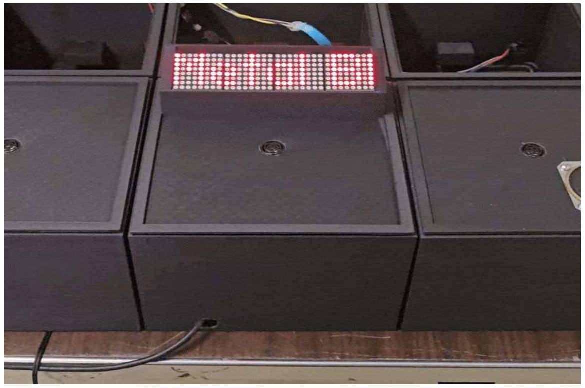
Our project will redefine the landscape of rhythm gaming, by transforming the way players interact and engage with the game. With this transformation, we look to accomplish these main goals:

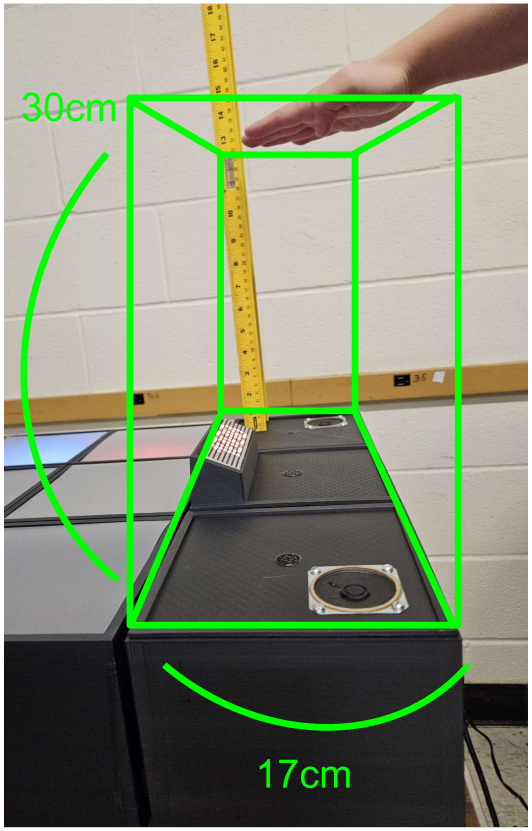
- Create a fun accessible rhythm game
- Eliminate the exposure to germs and bacteria
- Draw more players into rhythm gaming
- Meet conventional rhythm gameplay standards

User Interface

- On system startup, the device will enter the main song selection menu.
- The system will be able to detect the hand's position within 0-30cm vertical from the game board's position
  - Using the left and right sensor modules, user can navigate between songs
- Upon hovering over the middle sensor, the current selected song will be chosen and the game will begin

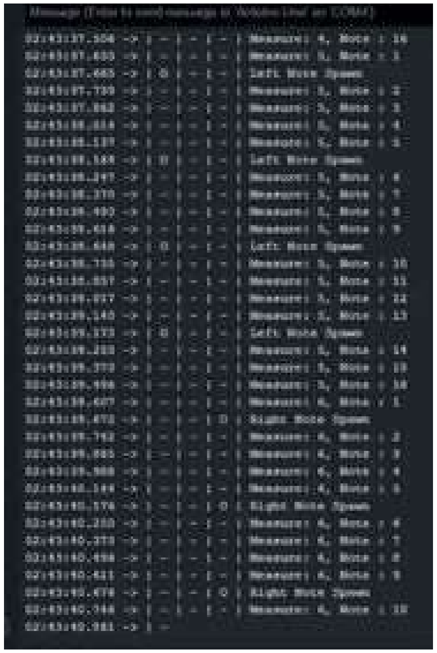




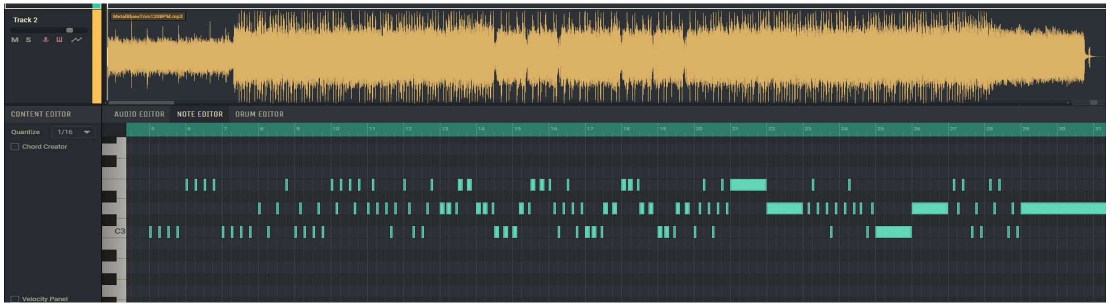


Song mechanics

- The played song will work from a premade map of notes.
  - Note map is hard coded in the source code, and coordinates note spawning with planned locations and timings.
  - Songs notes have a maximum spawn frequency of 125ms (the equivalent of one 16th note in a 4/4 time signature.
- Curated pattern of notes will play until the song’s completion (Playable song “Metal Blues” has a play time of approximately 1 minute and 6 seconds)

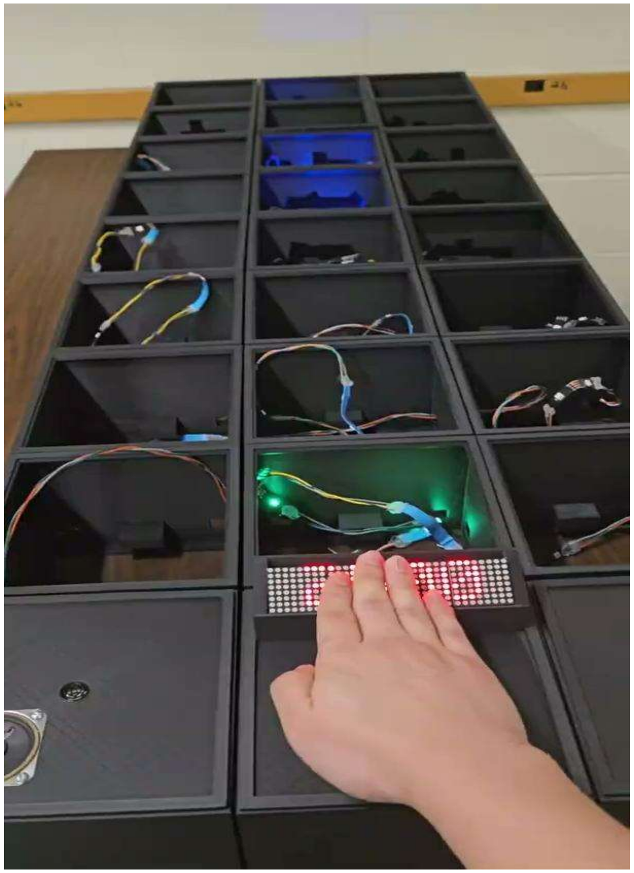


```
Message (Enter to send message to 'Arduino Uno' on 'COM4')
02:43:37.508 -> | - | - | - | Measure: 4, Note : 16
02:43:37.633 -> | - | - | - | Measure: 5, Note : 1
02:43:37.665 -> | O | - | - | Left Note Spawn
02:43:37.759 -> | - | - | - | Measure: 5, Note : 2
02:43:37.882 -> | - | - | - | Measure: 5, Note : 3
02:43:38.014 -> | - | - | - | Measure: 5, Note : 4
02:43:38.137 -> | - | - | - | Measure: 5, Note : 5
02:43:38.168 -> | O | - | - | Left Note Spawn
02:43:38.247 -> | - | - | - | Measure: 5, Note : 6
02:43:38.370 -> | - | - | - | Measure: 5, Note : 7
02:43:38.492 -> | - | - | - | Measure: 5, Note : 8
02:43:38.618 -> | - | - | - | Measure: 5, Note : 9
02:43:38.648 -> | O | - | - | Left Note Spawn
02:43:38.735 -> | - | - | - | Measure: 5, Note : 10
02:43:38.857 -> | - | - | - | Measure: 5, Note : 11
02:43:39.077 -> | - | - | - | Measure: 5, Note : 12
02:43:39.140 -> | - | - | - | Measure: 5, Note : 13
02:43:39.173 -> | O | - | - | Left Note Spawn
02:43:39.233 -> | - | - | - | Measure: 5, Note : 14
02:43:39.373 -> | - | - | - | Measure: 5, Note : 15
02:43:39.496 -> | - | - | - | Measure: 5, Note : 16
02:43:39.627 -> | - | - | - | Measure: 6, Note : 1
02:43:39.672 -> | - | - | O | Right Note Spawn
02:43:39.752 -> | - | - | - | Measure: 6, Note : 2
02:43:39.885 -> | - | - | - | Measure: 6, Note : 3
02:43:39.988 -> | - | - | - | Measure: 6, Note : 4
02:43:40.144 -> | - | - | - | Measure: 6, Note : 5
02:43:40.176 -> | - | - | O | Right Note Spawn
02:43:40.250 -> | - | - | - | Measure: 6, Note : 6
02:43:40.373 -> | - | - | - | Measure: 6, Note : 7
02:43:40.496 -> | - | - | - | Measure: 6, Note : 8
02:43:40.621 -> | - | - | - | Measure: 6, Note : 9
02:43:40.678 -> | - | - | O | Right Note Spawn
02:43:40.746 -> | - | - | - | Measure: 6, Note : 10
02:43:40.881 -> | -
```



Game mechanics

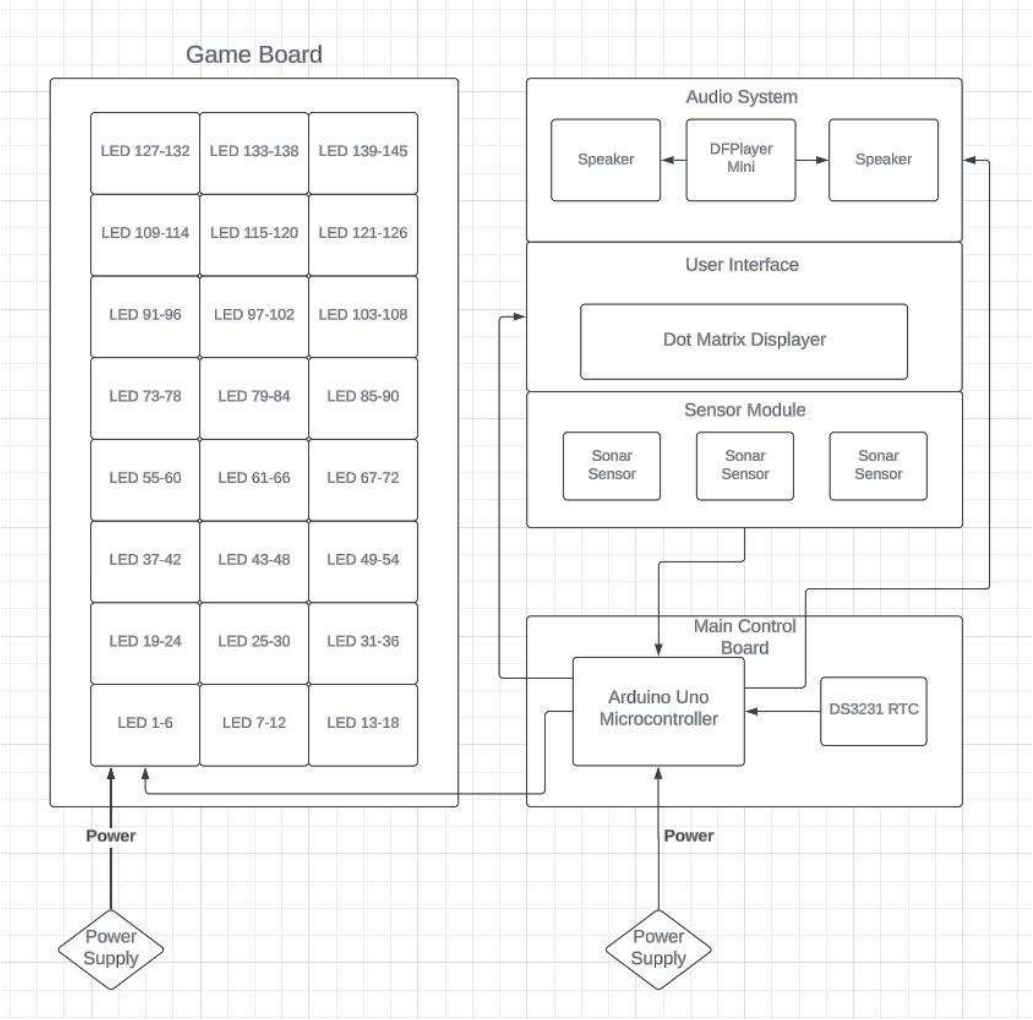
- When the game begins, the audio will be audible from a distance of 82 ± 5 cm.
- All LED positions will be checked, and will be turned off while turning on the LEDs of a subsequent row.
- The sonar sensor will determine which of the 3 different regions the hand is located in and record that position.
- System will record positions of LEDs on the bottom most row, and compare them to that of the hand position. Incrementing score displayed on dot matrix display.



Hardware Used

- **Microcontroller: Arduino Uno**
- **Sensors: MaxBotix LV-EZ1 Sonar range finder**
- **Audio:**
  - Speaker: GF0506
  - MP3 Module: DFPlayer Mini MP3 Player
- **Display: HiLetgo MAX7219 Dot Matrix**
- **RTC: Adafruit DS3231 Precision RTC Breakout**
- **Lighting: WS2812B Individual addressable RGB LED strip**

Hardware Block Diagram

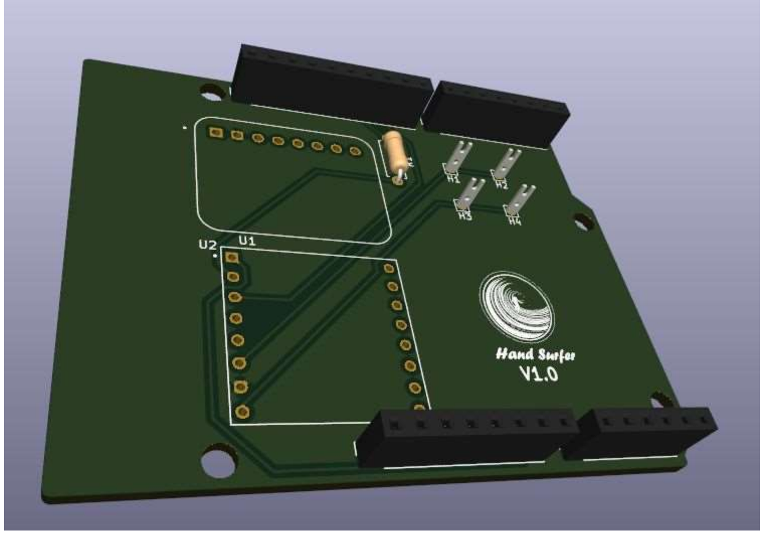


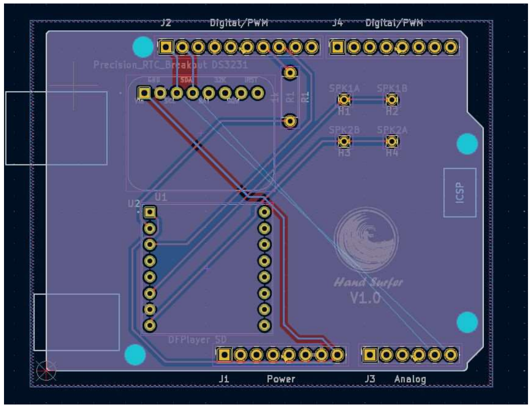
Game Board

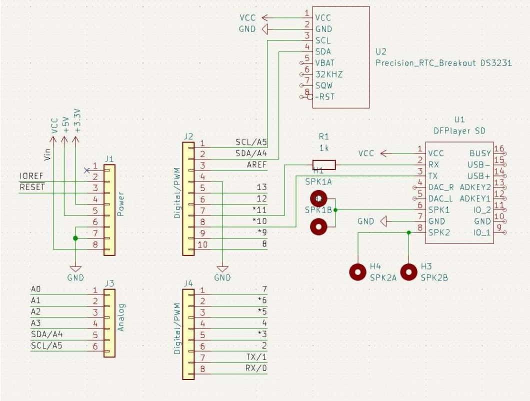
LED 127-132	LED 133-138	LED 139-145
LED 109-114	LED 115-120	LED 121-126
LED 91-96	LED 97-102	LED 103-108
LED 73-78	LED 79-84	LED 85-90
LED 55-60	LED 61-66	LED 67-72
LED 37-42	LED 43-48	LED 49-54
LED 19-24	LED 25-30	LED 31-36
LED 1-6	LED 7-12	LED 13-18

PCB Logistics

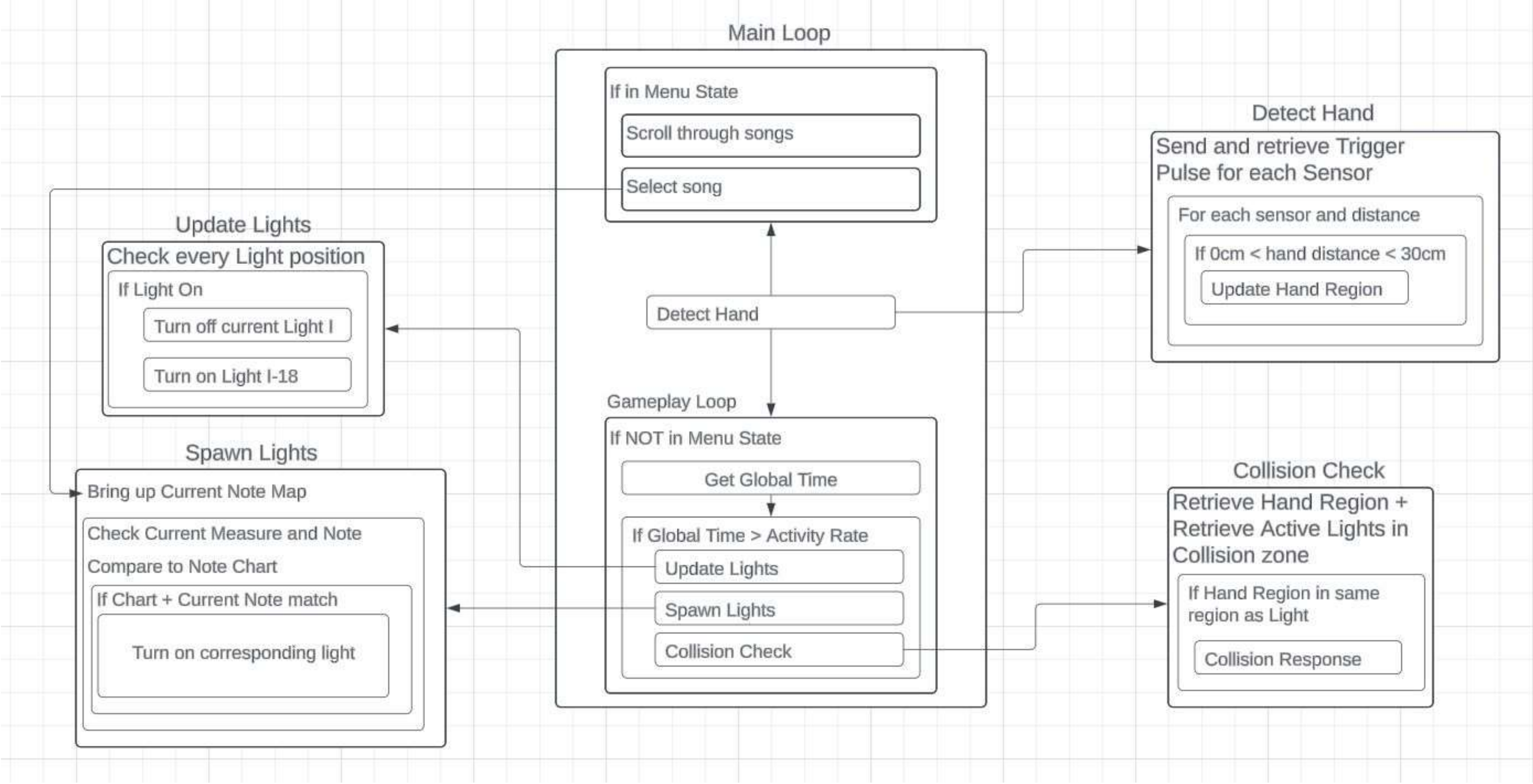
- Simple, streamlined and modular design
- Core static components, seperate from user interface components
- Other components (speakers, sensors, LEDs, etc.) housed separately on the gameboard
- Additional layer and modular shield allows ease of debugging and increased expandability







FPR Software Block Diagram



Main Loop

If in Menu State
- Scroll through songs
- Select song

Detect Hand

Gameplay Loop

If NOT in Menu State
- Get Global Time
- If Global Time > Activity Rate
  - Update Lights
  - Spawn Lights
  - Collision Check

Update Lights

Check every Light position
- If Light On
  - Turn off current Light I
  - Turn on Light I-18

Spawn Lights

Bring up Current Note Map
- Check Current Measure and Note
- Compare to Note Chart
- If Chart + Current Note match
  - Turn on corresponding light

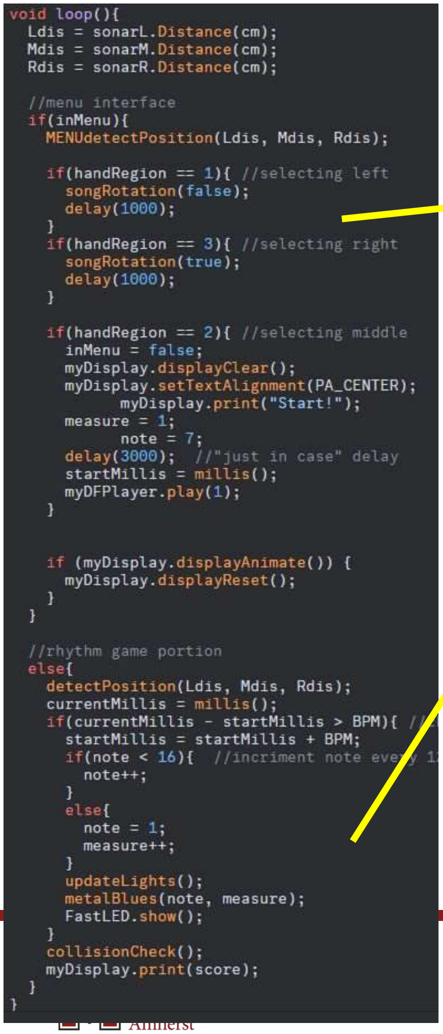
Detect Hand

Send and retrieve Trigger Pulse for each Sensor
- For each sensor and distance
  - If 0cm < hand distance < 30cm
    - Update Hand Region

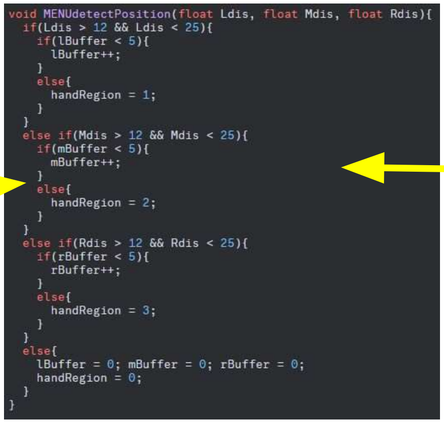
Collision Check

Retrieve Hand Region + Retrieve Active Lights in Collision zone
- If Hand Region in same region as Light
  - Collision Response

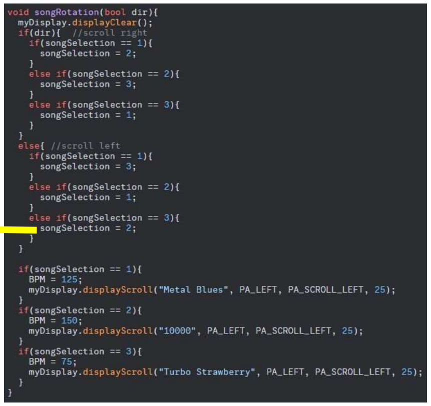
Software justifications



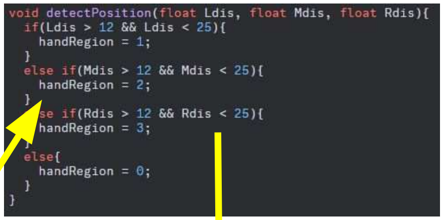
```
void loop(){
  Ldis = sonarL.Distance(cm);
  Mdis = sonarM.Distance(cm);
  Rdis = sonarR.Distance(cm);

  //menu interface
  if(inMenu){
    MENUdetectPosition(Ldis, Mdis, Rdis);

    if(handRegion == 1){ //selecting left
      songRotation(false);
      delay(1000);
    }
    if(handRegion == 3){ //selecting right
      songRotation(true);
      delay(1000);
    }

    if(handRegion == 2){ //selecting middle
      inMenu = false;
      myDisplay.displayClear();
      myDisplay.setTextAlignment(PA_CENTER);
         myDisplay.print("Start!");
      measure = 1;
         note = 7;
      delay(3000);  //"just in case" delay
      startMillis = millis();
      myDFPlayer.play(1);
    }


    if (myDisplay.displayAnimate()) {
      myDisplay.displayReset();
    }
  }

  //rhythm game portion
  else{
    detectPosition(Ldis, Mdis, Rdis);
    currentMillis = millis();
    if(currentMillis - startMillis > BPM){ //
      startMillis = startMillis + BPM;
      if(note < 16){  //incriment note every 1
        note++;
      }
      else{
        note = 1;
        measure++;
      }
      updateLights();
      metalBlues(note, measure);
      FastLED.show();
    }
    collisionCheck();
    myDisplay.print(score);
  }
}
```



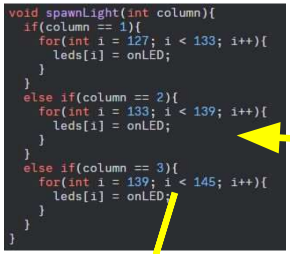
```
void MENUdetectPosition(float Ldis, float Mdis, float Rdis){
  if(Ldis > 12 && Ldis < 25){
    if(lBuffer < 5){
      lBuffer++;
    }
    else{
      handRegion = 1;
    }
  }
  else if(Mdis > 12 && Mdis < 25){
    if(mBuffer < 5){
      mBuffer++;
    }
    else{
      handRegion = 2;
    }
  }
  else if(Rdis > 12 && Rdis < 25){
    if(rBuffer < 5){
      rBuffer++;
    }
    else{
      handRegion = 3;
    }
  }
  else{
    lBuffer = 0; mBuffer = 0; rBuffer = 0;
    handRegion = 0;
  }
}
```



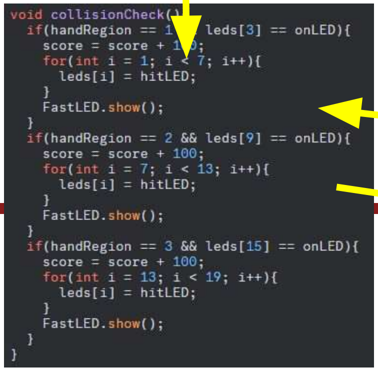
```
void songRotation(bool dir){
  myDisplay.displayClear();
  if(dir){  //scroll right
    if(songSelection == 1){
      songSelection = 2;
    }
    else if(songSelection == 2){
      songSelection = 3;
    }
    else if(songSelection == 3){
      songSelection = 1;
    }
  }
  else{ //scroll left
    if(songSelection == 1){
      songSelection = 3;
    }
    else if(songSelection == 2){
      songSelection = 1;
    }
    else if(songSelection == 3){
      songSelection = 2;
    }
  }

  if(songSelection == 1){
    BPM = 125;
    myDisplay.displayScroll("Metal Blues", PA_LEFT, PA_SCROLL_LEFT, 25);
  }
  if(songSelection == 2){
    BPM = 150;
    myDisplay.displayScroll("10000", PA_LEFT, PA_SCROLL_LEFT, 25);
  }
  if(songSelection == 3){
    BPM = 75;
    myDisplay.displayScroll("Turbo Strawberry", PA_LEFT, PA_SCROLL_LEFT, 25);
  }
}
```



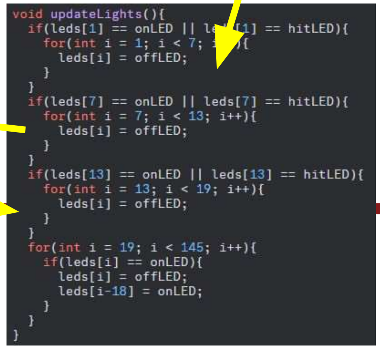
```
void detectPosition(float Ldis, float Mdis, float Rdis){
  if(Ldis > 12 && Ldis < 25){
    handRegion = 1;
  }
  else if(Mdis > 12 && Mdis < 25){
    handRegion = 2;
  }
  else if(Rdis > 12 && Rdis < 25){
    handRegion = 3;
  }
  else{
    handRegion = 0;
  }
}
```



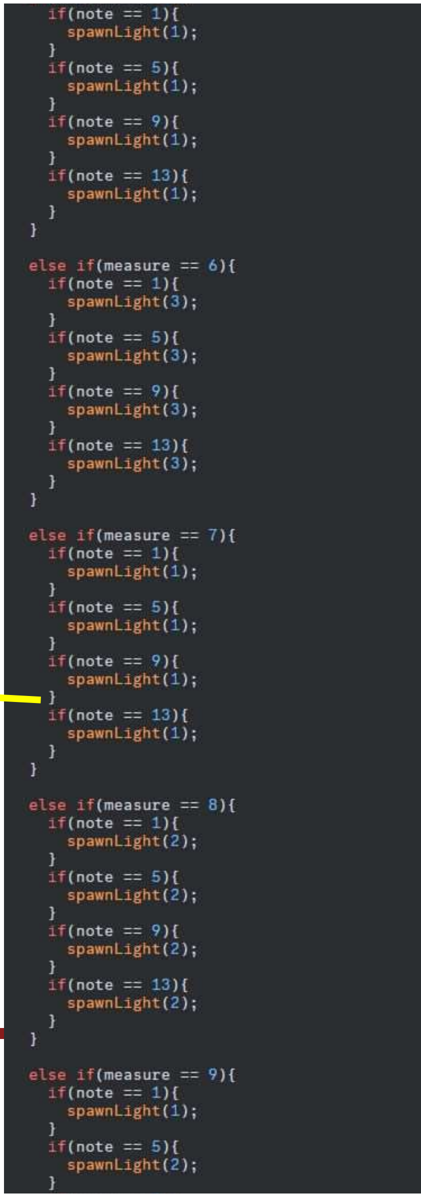
```
void spawnLight(int column){
  if(column == 1){
    for(int i = 127; i < 133; i++){
      leds[i] = onLED;
    }
  }
  else if(column == 2){
    for(int i = 133; i < 139; i++){
      leds[i] = onLED;
    }
  }
  else if(column == 3){
    for(int i = 139; i < 145; i++){
      leds[i] = onLED;
    }
  }
}
```



```
void collisionCheck(){
  if(handRegion == 1 && leds[3] == onLED){
    score = score + 100;
    for(int i = 1; i < 7; i++){
      leds[i] = hitLED;
    }
    FastLED.show();
  }
  if(handRegion == 2 && leds[9] == onLED){
    score = score + 100;
    for(int i = 7; i < 13; i++){
      leds[i] = hitLED;
    }
    FastLED.show();
  }
  if(handRegion == 3 && leds[15] == onLED){
    score = score + 100;
    for(int i = 13; i < 19; i++){
      leds[i] = hitLED;
    }
    FastLED.show();
  }
}
```

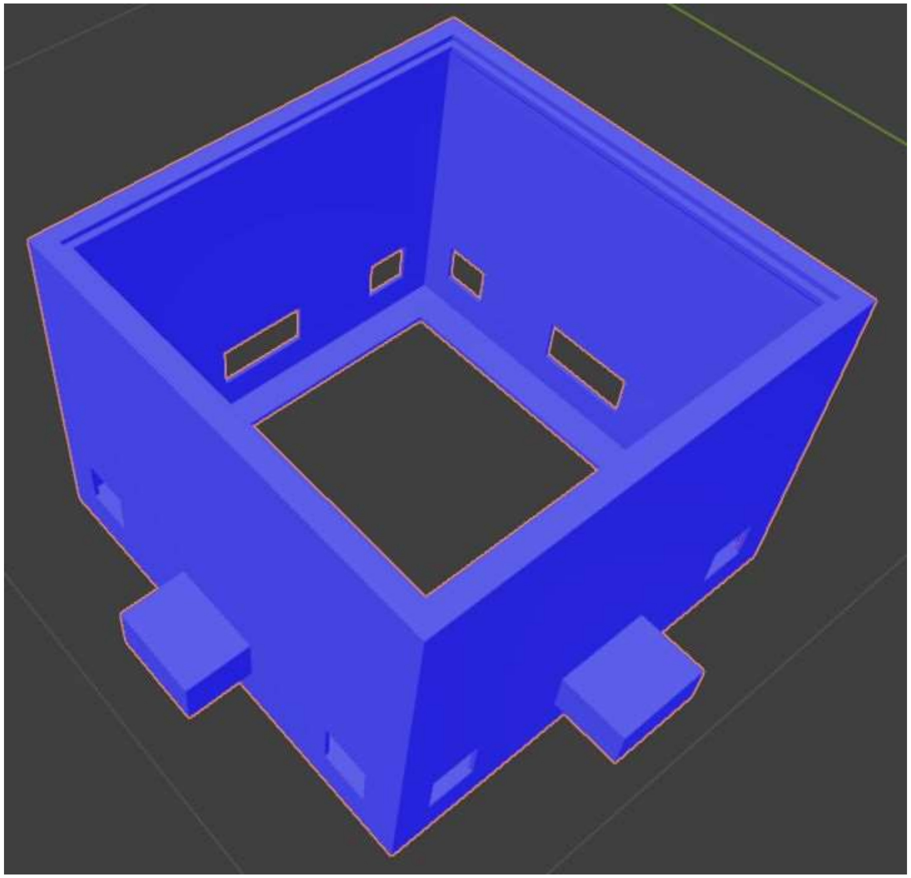


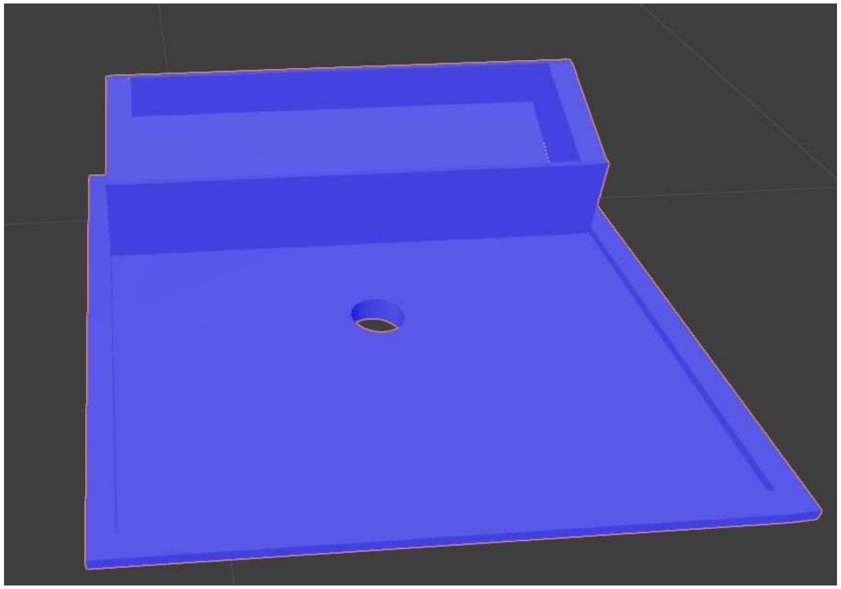
```
void updateLights(){
  if(leds[1] == onLED || leds[1] == hitLED){
    for(int i = 1; i < 7; i++){
      leds[i] = offLED;
    }
  }
  if(leds[7] == onLED || leds[7] == hitLED){
    for(int i = 7; i < 13; i++){
      leds[i] = offLED;
    }
  }
  if(leds[13] == onLED || leds[13] == hitLED){
    for(int i = 13; i < 19; i++){
      leds[i] = offLED;
    }
  }
  for(int i = 19; i < 145; i++){
    if(leds[i] == onLED){
      leds[i] = offLED;
      leds[i-18] = onLED;
    }
  }
}
```

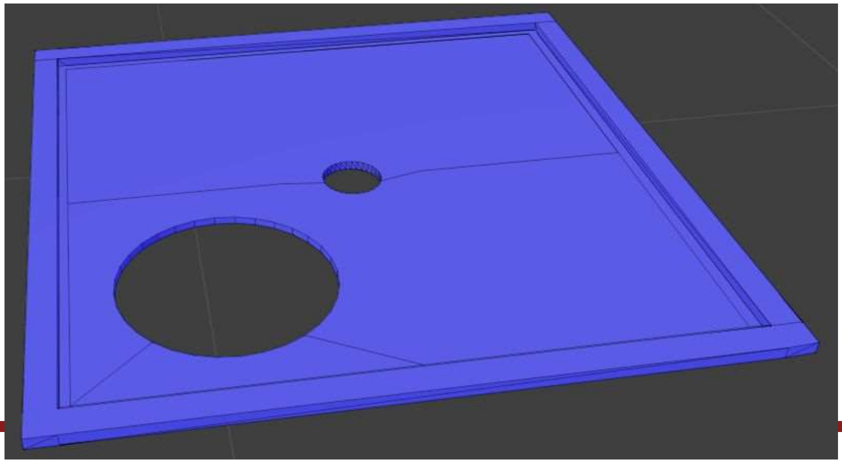


```
    if(note == 1){
      spawnLight(1);
    }
    if(note == 5){
      spawnLight(1);
    }
    if(note == 9){
      spawnLight(1);
    }
    if(note == 13){
      spawnLight(1);
    }
  }

  else if(measure == 6){
    if(note == 1){
      spawnLight(3);
    }
    if(note == 5){
      spawnLight(3);
    }
    if(note == 9){
      spawnLight(3);
    }
    if(note == 13){
      spawnLight(3);
    }
  }

  else if(measure == 7){
    if(note == 1){
      spawnLight(1);
    }
    if(note == 5){
      spawnLight(1);
    }
    if(note == 9){
      spawnLight(1);
    }
    if(note == 13){
      spawnLight(1);
    }
  }

  else if(measure == 8){
    if(note == 1){
      spawnLight(2);
    }
    if(note == 5){
      spawnLight(2);
    }
    if(note == 9){
      spawnLight(2);
    }
    if(note == 13){
      spawnLight(2);
    }
  }

  else if(measure == 9){
    if(note == 1){
      spawnLight(1);
    }
    if(note == 5){
      spawnLight(2);
    }
```

Final 3D Print Designs







FPR Deliverables

✅Updated Audio system Module with two speakers

❌At least three more hardcoded songs

✅ Finalized 3X8 3D printed board and updated sensor housing design

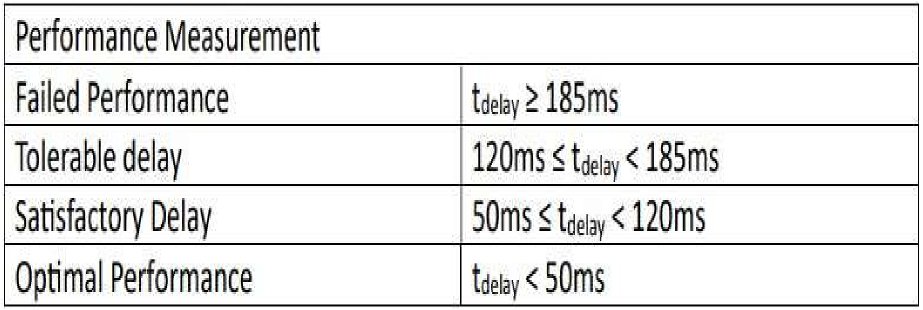
✅Updated User Interface, additional menu items

✅Adding the additional components necessary to enable another column of gameplay

✅Provide more interactive gameplay

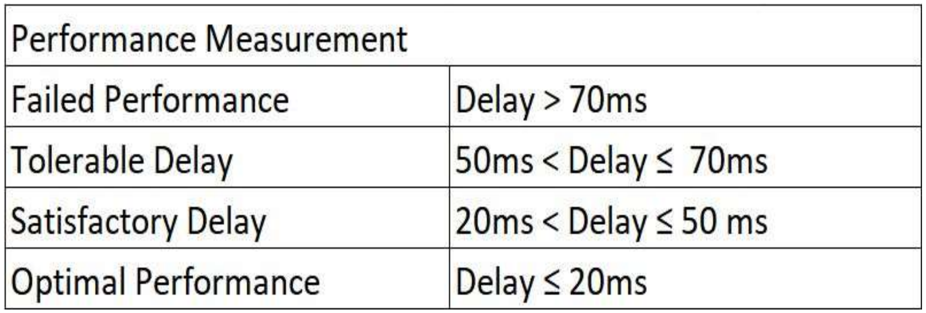
Testing Plan

1. **Audio & Visual Synchronization Module**
   a. Find the time delay between the audio notes and LED activation by implementing software time stamps



Performance Measurement	
Failed Performance	$t_{delay} \geq 185ms$
Tolerable delay	$120ms \leq t_{delay} < 185ms$
Satisfactory Delay	$50ms \leq t_{delay} < 120ms$
Optimal Performance	$t_{delay} < 50ms$

2. **Collision Timing Delay**
   a. Find the time delay from hand placement in a region to sensor recognition



Performance Measurement	
Failed Performance	Delay > 70ms
Tolerable Delay	50ms < Delay ≤ 70ms
Satisfactory Delay	20ms < Delay ≤ 50 ms
Optimal Performance	Delay ≤ 20ms

Test Results

Audio & Visual Synchronization Performance Measurement	
Failed Performance	Delay ≥ 185ms
Tolerable Delay	120ms ≤ t_Delay < 185ms
Satisfactory Delay	50ms ≤ Delay < 120 ms
Optimal Performance	Delay < 50ms

Collision Timing Delay Performance Measurement	
Failed Performance	Delay > 70ms
Tolerable Delay	50ms < Delay ≤ 70ms
Satisfactory Delay	20ms < Delay ≤ 50 ms
Optimal Performance	Delay ≤ 20ms

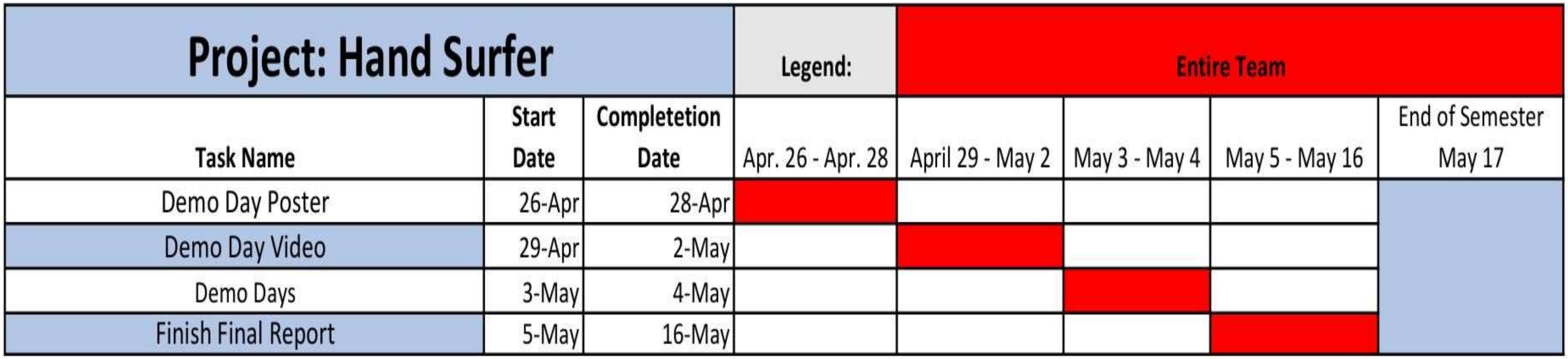
Expenditures

Item	quantity	costs
MDR PCB	1	9.8
CDR PCB	1	41.9
FDR PCB	1	3.1
Adafruit DS3231 Precision RTC Breakout	2	53.6
HiLetgo MAX7219 Dot Matrix Module for Arduino Microcontroller 4 in 1 Display with 5pin Line	1	8.99
MAX7219 Dot Matrix Module DIY kit	1	11
DFPlayer - A Mini MP3 Player	2	19.8
60 Pixels WS2812B Individual Addressable LED Strip	1	9.99
Maxbotix Ultrasonic Rangefinder - LV-EZ0 - LV-EZ0	3	99.19
Polymaker PLA+ 3D Printer Filament 1.75mm	9	197.91
Light filter	2	19.98
150 Pixels WS2812B Individual Addressable LED strip	1	28.99
Total	504.25	
Remaining	-4.25	

Final Product Cost

Categories	Item	Quantity	Cost Each	Cost Total
Main Control	Arduino Uno	1	23.99	23.99
	Adafruit DS3231 Precision RTC Breakout	1	26.8	26.8
	PCB	1	3.1	3.1
Audio	GF0506 Speaker	2	3.06	6.12
	DFPlayer Mini MP3 Player	1	9.9	9.9
UI	Maxbotic Ultrasonic RangeFinder	3	29.95	99.19
	WS2812B Individual Addressable LED strip	1	28.99	28.99
	HiLetgo MAX7219 Dot Matrix Module	1	8.99	8.99
Housing	Polymaker PLA+ 3D Printer Filament 1.75mm	9	21.99	197.91
	Light Filter Sheets	2	9.9	19.8
Total		427.99		

Gantt Chart



Project: Hand Surfer			Legend:	🟥 Entire Team				
Task Name	Start Date	Completetion Date	Apr. 26 - Apr. 28	April 29 - May 2	May 3 - May 4	May 5 - May 16	End of Semester May 17
Demo Day Poster	26-Apr	28-Apr	🟥				
Demo Day Video	29-Apr	2-May		🟥			
Demo Days	3-May	4-May			🟥		
Finish Final Report	5-May	16-May				🟥	

Conclusion

- We’ve created a fun rhythm game that is visually appealing and has a unique style of gameplay
- Produced the rhythm game that we originally envisioned when we first found this idea

Thank you for taking the time to review our project and provide constructive criticism over these past semesters

University of Massachusetts Amherst