CRMT250 Individual Report - Michael Freeman (Holmes) - 6th May 2024

Installation Title: "*Get Down Everybody*"

Overview and Aim

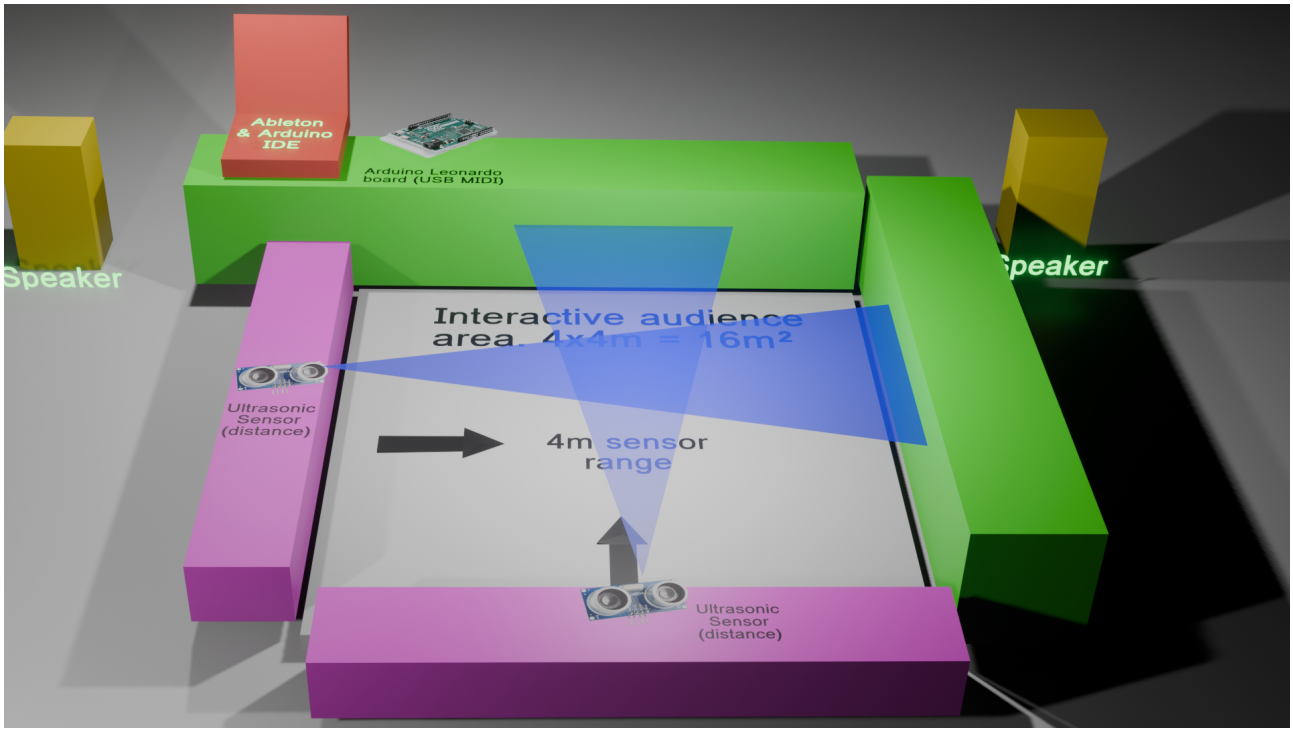

Fig 1: Installation plan/sketch.

I discovered the Arduino brand which can quickly prototype functionality (Severance 2014). I developed an idea to have a space where participants would alter an Ableton composition in real time based on their position in a 4x4 meter space (Fig. 1) (Freeman 2024). The audience member moves around or dances. This would be achieved by using ultrasonic sensors to measure the distance to a person across the space. The sensors are arranged in an X/Y configuration with some resemblance to an X/Y pad. This was once seen as a novel approach to controlling sound.

> *"The joystick idea was also marketed as a pressure plate, where the XY position of the fingertip sent out voltages."* (Metlay 1990: 82)

According to Paine (2007) this is now a common approach.

> "*Interactive sound environments are commonly characterized by an interface that allows the triggering of pre-made sonic material according to an X-Y coordinate location of an interactive agent in an architectural space*" (Paine 2007)

The setup is intended to blur the boundary between passive audience member and the musical composition.

> "*Within these new environments, visitors were invited across the threshold and into the performance in a move that dissolved the physical, as well as aesthetic, boundaries between author (composer/artist), performer, and audience."* (Rogers 2013: 152)

The installation divides up the 4x4m space into "zones". When a zone is triggered by the audience member a MIDI note or CC control message is sent to Ableton that initiates a mixture of clip toggling, vocal sample manipulation and changing the position of a synth Filter Cutoff. This is done by sending the detected range from the sensor to a "*case*" statement in two functions "*sendMIDI*" and "*sendCC*" thusly (Fig. 2).

```
/**
    __ _          __                       _____          _        _ _       _   _
   / /(_)_   _____   / /  ___   ___  _ __   ___ _ __   \_   \_ __  ___| |_ __ _| | | __ _| |_(_) ___  _ __
  / / | \ \ / / _ \ / /  / _ \ / _ \| '_ \ / _ \ '__|   / /\/ '_ \/ __| __/ _` | | |/ _` | __| |/ _ \| '_ \
 / /__| |\ V /  __// /__| (_) | (_) | |_) |  __/ |   /\/ /_ | | | \__ \ || (_| | | | (_| | |_| | (_) | | | |
 \____/_| \_/ \___|\____/\___/ \___/| .__/ \___|_|   \____/ |_| |_|___/\__\__,_|_|_|\__,_|\__|_|\___/|_| |_|
                                    |_|

   Live Looper Installation for CRMT250 - Immersive Audio assessment performance by Michael Freeman (Holmes) 2024.

   ---- Board circuit design is based the "Measure Your Height by Ultrasonic Sensor" project ...               ----
   ---- ( https://projecthub.arduino.cc/kiroloskhairy/measure-your-height-by-ultrasonic-sensor-79b987 )         ----
   ---- Based on original code as supplied by Sparkfun sensor supplier ( https://www.sparkfun.com/products/15569 ) ----
   ---- Original Sparkfun comments are below  ...                                                               ----

    HC-SR04 Demo
    Demonstration of the HC-SR04 Ultrasonic Sensor
    Date: August 3, 2016

    Description:
     Connect the ultrasonic sensor to the Arduino as per the
     hardware connections below. Run the sketch and open a serial
     monitor. The distance read from the sensor will be displayed
     in centimeters and inches.

    Hardware Connections:
     Arduino | HC-SR04
     -------------------
       5V    |   VCC
       7     |   Trig
       8     |   Echo
       GND   |   GND

    License:
     Public Domain
  */

  #include <NewPing.h> // https://www.arduino.cc/reference/en/libraries/newping/
  #include "MIDIUSB.h"

  #define SONAR_NUM 2      // Number of sensors.
  #define MAX_DISTANCE 400 // Maximum distance (in cm) to ping.

  // For Serial Plotter - https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-plotter/
  int static_variable = 200;
  int sensor1cm = 0;
  int sensor2cm = 0;
  int sensorCm = 0;

  NewPing sonar[SONAR_NUM] = {   // Sensor object array.
    NewPing(7, 8, MAX_DISTANCE), // Each sensor's trigger pin, echo pin, and max distance to ping.
    NewPing(11, 12, MAX_DISTANCE),
    // NewPing(8, 9, MAX_DISTANCE)
  };

  int sendMIDI(int sensorCm){
    // https://www.arduino.cc/reference/en/language/functions/math/map/
    // Take sensor cm's reading and split it into 8 different zones in range 0cm to 400cm
    int range = map(sensorCm, 0, 400, 0, 8);
    switch (range) {
      case 0:  // your hand is on the sensor
        noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
        MidiUSB.flush();
        delay(250);
        noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
        MidiUSB.flush();
        break;
      case 1:  // your hand is close to the sensor
        noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
        MidiUSB.flush();
        delay(500);
        noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
        MidiUSB.flush();
        break;
      case 2:  // your hand is a few inches from the sensor
        noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
        MidiUSB.flush();
        delay(750);
        noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
        MidiUSB.flush();
```

```
      break;
    case 3:  // your hand is nowhere near the sensor
      noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(1000);
      noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 4:  // your hand is a few inches from the sensor
      noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(1250);
      noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 5:  // your hand is nowhere near the sensor
      noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(1500);
      noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 6:  // your hand is a few inches from the sensor
      noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(1750);
      noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 7:  // your hand is nowhere near the sensor
      noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(2000);
      noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 8:  // your hand is a few inches from the sensor
      noteOn(0, 48, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(2250);
      noteOff(0, 48, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
  }
}

int sendCC(int sensorCm){ // Also sends MIDI notes
  int range = map(sensorCm, 0, 400, 0, 8);
  switch (range) {
    case 0:
      controlChange(0, 1, 16);
      MidiUSB.flush();
      break;
    case 1:
      controlChange(0, 1, 32);
      MidiUSB.flush();
      //Toggle 909 clip
      noteOn(0, 49, 64);
      MidiUSB.flush();
      delay(500);
      noteOff(0, 49, 64);
      MidiUSB.flush();
      break;
    case 2:
      controlChange(0, 1, 48);
      MidiUSB.flush();
      // Toggle 808 clip
      noteOn(0, 50, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(500);
      noteOff(0, 50, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 3:
      controlChange(0, 1, 64);
      MidiUSB.flush();
      break;
    case 4:
      controlChange(0, 1, 80);
```

```
      MidiUSB.flush();
      // Toggle Bass clip
      noteOn(0, 51, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(500);
      noteOff(0, 51, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 5:
      controlChange(0, 1, 96);
      MidiUSB.flush();
      break;
    case 6:
      controlChange(0, 1, 112);
      MidiUSB.flush();
      // Toggle vocal sample clip
      noteOn(0, 52, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(500);
      noteOff(0, 52, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 7:
      controlChange(0, 1, 128);
      MidiUSB.flush();
      // Toggle synth clip
      noteOn(0, 53, 64);   // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      delay(500);
      noteOff(0, 53, 64);  // Channel 0, middle C, normal velocity
      MidiUSB.flush();
      break;
    case 8:
      controlChange(0, 1, 64);
      MidiUSB.flush();
      break;
  }
}

void noteOn(byte channel, byte pitch, byte velocity) {
  midiEventPacket_t noteOn = {0x09, 0x90 | channel, pitch, velocity};
  MidiUSB.sendMIDI(noteOn);
}

void noteOff(byte channel, byte pitch, byte velocity) {
  midiEventPacket_t noteOff = {0x08, 0x80 | channel, pitch, velocity};
  MidiUSB.sendMIDI(noteOff);
}

// First parameter is the event type (0x0B = control change).
// Second parameter is the event type, combined with the channel.
// Third parameter is the control number number (0-119).
// Fourth parameter is the control value (0-127).
// controlChange(0, 10, 65); // Set the value of controller 10 on channel 0 to 65

void controlChange(byte channel, byte control, byte value) {
  midiEventPacket_t event = {0x0B, 0xB0 | channel, control, value};
  MidiUSB.sendMIDI(event);
}

void setup() {
  Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}

void loop() {
  //for (uint8_t i = 0; i < SONAR_NUM; i++) { // Loop through each sensor and display results.
  // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
    delay(50);
    // Sensor 1 triggers MIDI note to Ableton Live vocal sample Track Activator control
    Serial.print("Sensor_0:");
    sensor1cm = sonar[0].ping_cm();
    // Make sure max range is 400 even if range is over maximum (which is returned as zero by NewPing library)
    if (sensor1cm == 0) sensor1cm = 400;
    // Trigger Ableton by MIDI note.
    sendMIDI(sensor1cm);
    Serial.print(sensor1cm);
    Serial.print(",");
    delay(50);
    Serial.print("Sensor_1:");
    sensor2cm = sonar[1].ping_cm();
    // Make sure max range is 400 even if range is over maximum (which is returned as zero by NewPing library)
    if (sensor2cm == 0) sensor2cm = 400;
    // Trigger Ableton by CC
    sendCC(sensor2cm);
    Serial.print(sensor2cm);
    Serial.print(",");
    Serial.println();
}
```

Fig 2: Arduino code (C++).

Ableton controls are mapped to the MIDI messages that the Arduino sends once this code is uploaded onto the Arduino board (Fig. 3).

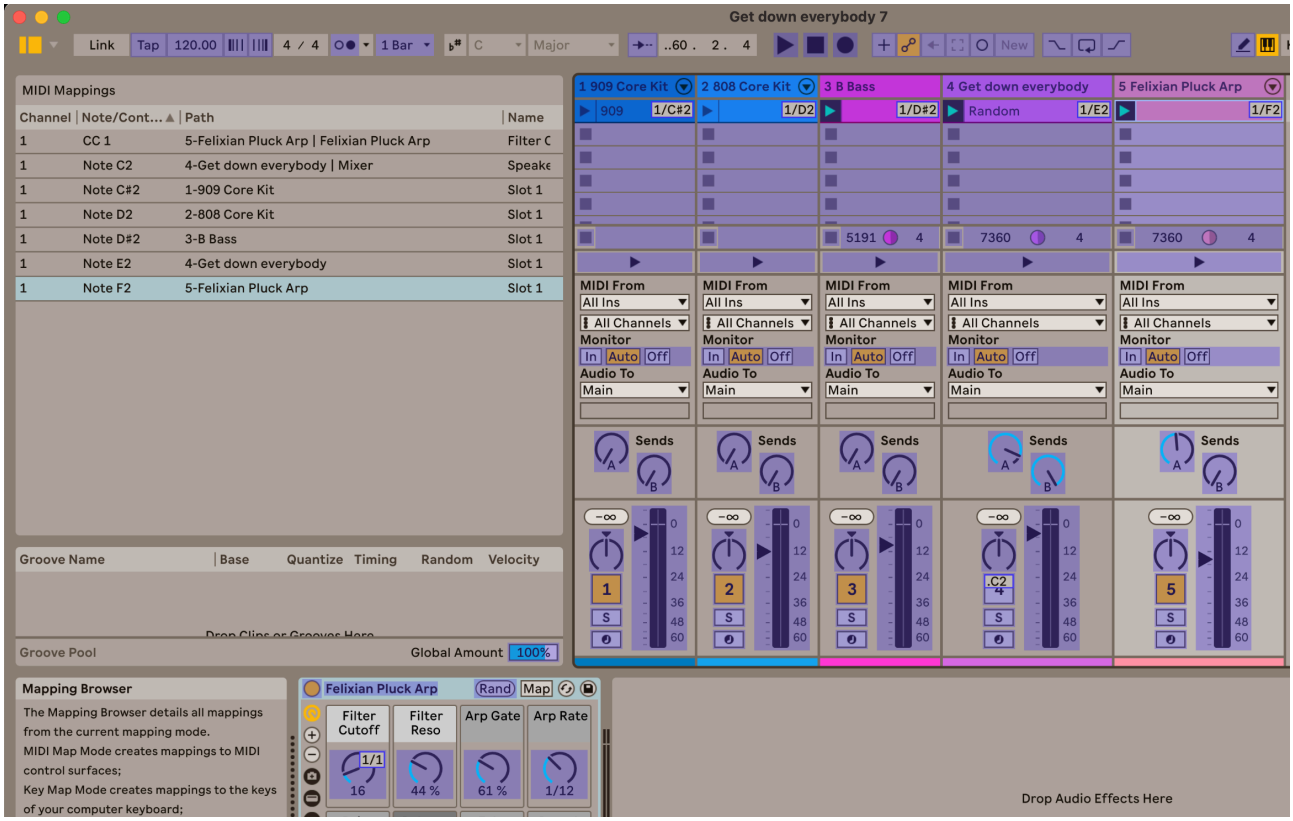

Fig 3: Ableton Live MIDI mappings.

Immersive audio techniques: Problem Solving

I used the appropriate form of sensor for this project type according to Nussey (2013: 277).

> "*Infrared proximity sensors are not as accurate and have a much shorter range than ultrasonic range finders.*"

Basing my project approach on an *Arduino Project Hub* project to measure the height of your growing children (Khairy 2021), I setup a working ultrasonic sensor (Sparkfun 2017) to measure distance. However when I setup two of them on the same board my code appeared to develop a bug where one of the sensors reported erroneous readings. Swapping the sensors around gave the same result but eliminated any hardware error in the sensors. I decided to take the problem to a simulator (Fig. 4).

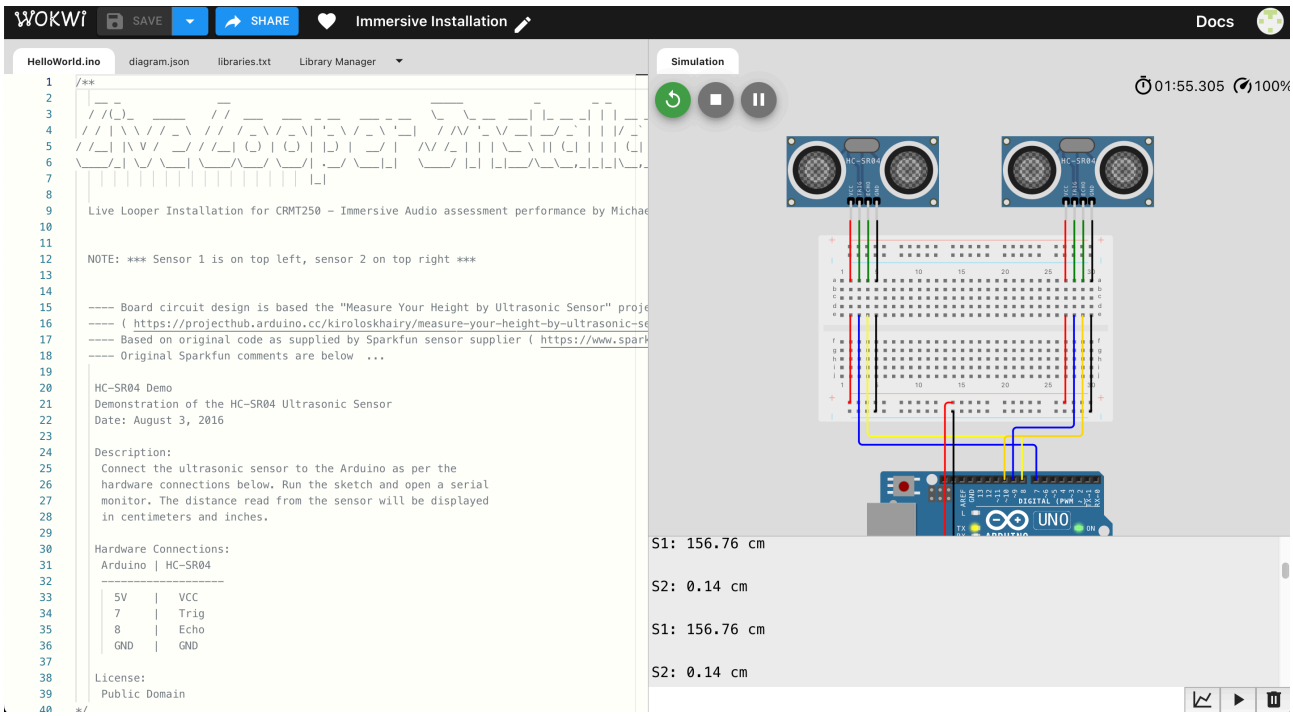

Fig 4: Simulator

The simulator (Wokwi 2024) showed the same result. The simulated distance is set to 156.76cm for both sensors. This led me to believe that there was some kind of bug in my code. I then noticed a forum post.

> "*Note, however, that the sensors may pick up signals from one another*" (Arduino 2022)

Which led to what was a "*Doh!*" moment (*Simpsons*), part of an important creative and development insight experience.

> "*D'oh" moment (a positive reaction to having solved the goat riddle, followed by annoyance at it having been obvious)*" (Hill 2018)

The returning ultrasonic pulses are not isolated from each other (at different frequencies) so are picked up by both sensors. This leads to erroneous readings and is a common problem in electronics which is called *crosstalk*.

> "*Crosstalk is the unwanted coupling between signal paths.*" (Mazda 1993)

I coded in a delay between sensor activations to eliminate the crosstalk. I also revised my project sketch to include theatre blocks around the edge of the space to minimise scattering of the ultrasonic pulses which can also lead to crosstalk.

Immersive audio techniques: Escape from Abstraction to Audience Becoming "Artist"

With "Superstar" DJ's, the focus has moved away from the audience. The DJ, as well as musicians creating live performances of electronic dance music, used to be off to the side. The audience participant used to be the main show, not some elitist abstraction. The following quote is interesting in this respect.

> "*Let us give just two quotations, firstly Stockhausen's advice:*
>
> *I wish those musicians would not allow themselves any repetitions, and would go faster in developing their ideas or their findings, because I don't appreciate at all this permanent repetitive language […] I think it would be very helpful if he [Aphex Twin/Richard James] listens to my work Song of the Youth … Because he would then immediately stop with all these post-African repetitions and he would look for changing tempi and changing rhythms … (Stockhausen et al., 2004, p. 382).*

> *To which James replies (after listening to Song of the Youth [s]):*
>
> *Mental! I've heard that song before; I like it. I didn't agree with him. I thought he should listen to a couple of tracks of mine: "Didgeridoo", then he'd stop making abstract, random patterns you can't dance to. Do you reckon he can dance? You could dance to Song of the Youth, but it hasn't got a groove in it, there's no bass line (p. 383).*" (Cited in Emmerson 2017: 62)

In the original Wire article from 1995 (Stockhausen 1995) says "*I know that he wants to have a special effect in dancing bars, or wherever it is, on the public who like to dream away with such repetitions"*. It was this kind of attitude problem that inspired me to move away from purely abstract sound installations, to what is much more a mini recreation of a "rave". The DJ would normally respond to subtle cues from the audience and actively project the focus onto them, rather than basking in some limelight or spotlight. This is replaced by the ultrasonic sensors that place playful responsibility on the audience to change and guide the music through *their* own movements.

> "*In those murky, atmospheric clubs, the deejay booth was often tucked away in a corner rather than placed on a stage: dancers weren't meant to be looking in one direction, they were meant to get lost in music, in the collective intimacy of the dancefloor.*" (Reynolds 1998)

Conclusion

The installation successfully explored a space that put the audience participants back in control rather than having some elitist figure in control.

Evaluation

I did intend to put the sensors in boxes and to make the installation look more professional, rather than having exposed circuit boards and wiring. This could be an area of future study to explore, taking what is breadboard prototyping towards a fully soldered product. I think the musical composition could have been more extensive even though it was enough to demonstrate the interaction with the sensors.

LIST OF FIGURES

REFERENCES

ARDUINO. 2022. 'How Do You Use 2 Ultrasonic Sensors Simultaneously?' *Arduino Forum* [online]. Available at: https://forum.arduino.cc/t/how-do-you-use-2-ultrasonic-sensors-simultaneously/1064352/5?u=res1s7 [accessed 20 May 2024].

BLUM, Jeremy. 2013. *Exploring Arduino Tools and Techniques for Engineering Wizardry*. 1st edition. Indianapolis, Ind: Wiley.

ABDULLAH AL-KHATIB, Mutaz ARIF and Kenji SASAKI. 2015. 'Installation Using Arduino with Interlock Sensors: Introduction to Illumination and Sound Modules'. *2015 International Conference on Cyberworlds* (CW).

EMMERSON, Simon. 2017. *Living Electronic Music*. First edition. London: Taylor and Francis.

Freeman, M. (2024). *Sketch / Planning Visualisation for Installation*. [online] www.youtube.com. Available at: https://youtu.be/MzAICLuOb7U [Accessed 11 May 2024].

HILL, Gillian and Shelly M KEMP. 2018. ‘Uh-Oh! What Have We Missed? A Qualitative Investigation into Everyday Insight Experience’. *The Journal of creative behavior* 52(3), 201–11.

KHAIRY, Kirolos. 2021. ‘Measure Your Height by Ultrasonic Sensor’. *projecthub.arduino.cc* [online]. Available at: https://projecthub.arduino.cc/kiroloskhairy/measure-your-height-by-ultrasonic-sensor-79b987.

MAZDA, Fraidoon F. 1993. *Telecommunications Engineer’s Reference Book*. Oxford: Butterworth-Heinemann.

METLAY, Michael P. 1990. ‘The Musician-Machine Interface to MIDI’. *Computer Music Journal* 14(2), 73–83.

NUSSEY, John. 2013. *Arduino for Dummies*. Chichester: Wiley.

REYNOLDS, Simon. 1998. *Energy Flash:: A Journey through Rave Music and Dance Culture*. London: Picador.

ROGERS, Holly. 2013. *Sounding the Gallery : Video and the Rise of Art-Music*. New York: Oxford University Press.

SEVERANCE, Charles. 2014. ‘Massimo Banzi: Building Arduino’. *Computer (Long Beach, Calif.)* 47(1), 11–2.

SIMPSONS. Fox [TV series].

SPARKFUN. 2017. ‘Ultrasonic Distance Sensor - HC-SR04 - SEN-15569 - SparkFun Electronics’. *Sparkfun.com* [online]. Available at: https://www.sparkfun.com/products/15569

Stockhausen Meets the Technocrats: The German Composer Swaps Opinions with Aphex Twin, Scanner and Daniel Pemberton. 1995. Wire (London, England). London: London : C. Parker.

WANG, Brian. 2018. *Reverse Engineering SpaceX to Learn How to Speed Up Technological Development. Next Big Future* [BLOG]. Moffett Field: Newstex.

WOKWI. 2024. ‘Immersive Installation - Wokwi ESP32, STM32, Arduino Simulator’. *wokwi.com* [online]. Available at: https://wokwi.com/projects/396596908989045761 [accessed 20 May 2024].

BIBLIOGRAPHY

BANZI, Massimo and Michael SHILOH. 2022. *Getting Started with Arduino*. 4th edition / Massimo Banzi and Michael Shiloh. Santa Rosa, CA: Make.

BURKE, Jeff. 2022. ‘The SpaceX Effect - the Culture behind SpaceX’. *jeffburke.substack.com* [online]. Available at: https://jeffburke.substack.com/p/the-spacex-effect-the-culture-behind [accessed 11 May 2024].

GEDDES, Mark. 2016. *The Arduino Project Handbook : 25 Practical Projects to Get You Started*. San Francisco: No Starch Press.

KARVINEN, Kimmo and Tero KARVINEN. 2014. *Getting Started with Sensors*. Sebastopol, CA: Maker Media.

LANGBRIDGE, James A. 2015. *Arduino Sketches : Tools and Techniques for Programming Wizardry*. 1st edition. Indianapolis, Indiana: John Wiley & Sons.

PAINE, G. 2007. Sonic immersion: Interactive engagement in real-time immersive environments. *SCAN Journal of Media Arts and Culture*, 4(1), pp.1-13.

STEWART, Becky. 2015. *ß*. 1st ed. Hoboken: Wiley.

Tömösvári, I. (2022). *Applying Elon Musk’s engineering principles to coding*. [online] TMSVR - Dev Blog. Available at: https://tmsvr.com/elon-musks-engineering-principles-to-coding/ [Accessed 12 May 2024].

WILCHER, Don. 2015. *Arduino Electronics Blueprints : Make Common Electronic Devices Interact with an Arduino Board to Build Amazing out-of-the-Box Projects*. Birmingham, England ; Packt Publishing.