

Max4Live Theremin Controller

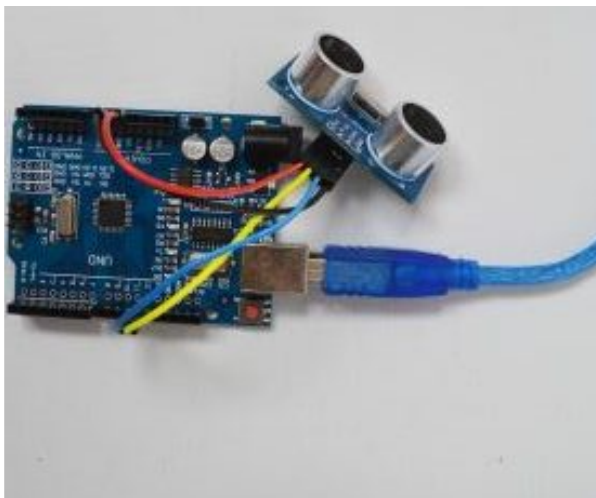
In this paper I'll outline my Proximity-Sensor Device for Max4Live. The purpose of this device is to provide continuous control of pitch and tone parameters, much like you would have with an analogue theremin.

The advantage of this over a traditional theremin is the flexibility that plugging into a DAW like Ableton can offer. Virtually any parameter can be controlled using the device leading to a near limitless set of possibilities for sound manipulation.

Having this level of flexibility is advantageous for adding a unique sonic quality to records, brainstorming and generating musical ideas that wouldn't have existed otherwise and creating unique samples similar to 'Foley' recordings made using a magnetic pickup.

I defined the brief to be a project that achieves the following:

- Continuous pitch and filter control
- A device that I'll use regularly
- Something that's inexpensive to build
- Something with pre-existing code readily available
- Easy to play



The device is currently in the prototype stage, hence the 'rugged' look.

Process

Kit List:

- Arduino Uno Microcontroller Board £4.69

- SR-04 Proximity Sensor £2.10
- M to F Jumper Cables £1.20

Total Spend: £7.99

I already owned Ableton Live Suite which comes with Max4Live, so incurred no further costs for that.

Kit Explained:

Board:

Arduino is a company that makes microcontroller boards which are commonly used for creative projects and prototyping electronic devices. They have also developed a high-level programming language which is designed to be very accessible to novice / DIY coders.

The Arduino Duo I opted for is the entry level model which has no operating system and can only run one program at a time.

Sensor:

The SR-04 is an “ultrasonic ranging module” which acts as a proximity sensor. It works by sending sound waves and measuring how long they take to bounce off an object and return to the device.



It has two cones, a transmitter on the left (the trigger) and a receiver on the right (the echo).

The four pins are as follows:

Trigger Pin - Device sends a trigger (high freq sound) which bounces off objects in front of it. When the sensor is connected to the microcontroller the trigger pin tells the arduino that this has happened.

Echo Pin - Once the sound bounces off an object, the reflected “echo wave” is received by the sensor, then the echo pin tells the Arduino and it’s possible to calculate distance by measuring the time between the trigger and the echo, using the speed of sound.

VCC Pin - 5V Power

GND - Ground

Pros and Cons:

This type of sensor can still be used in a darkly lit studio unlike a light or IR sensor which may encounter issues.

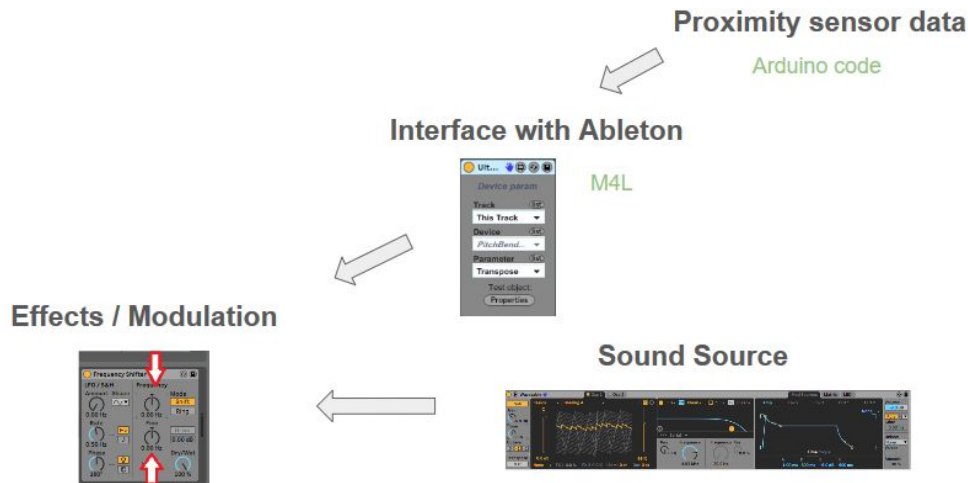
However it can often give sporadic readings when measuring small or uneven objects, as outlined in an instruction manual by Sparkfun (2019):

“When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.”

My own experience corroborated this as I'll outline later on.

Signal Chain

The proximity data has to pass from the sensor, through the Arduino board into the computer, where it then needs to be read by Ableton, in my case through a Max4Live plugin, before it can modulate parameters and affect the sound.



Arduino Code

I was reluctant to code the Arduino program from the ground up due to time constraints, as well as a desire to quickly begin experimenting with a working device. I found a fit-for-purpose sketch which required minimal tweaking and instead took some time to understand what each line is doing as outlined below in green:

```
#define echoPin 11  
#define trigPin 12
```

```
#define LEDPin 13
```

```
// Assigns the echo, trig and LED pins to inputs/outputs on the arduino. The LED is useful for testing, as you can see it flash on the arduino board when the sensor detects something.
```

```
int maximumRange = 70;  
int minimumRange = 0;  
long duration, distance;
```

```
// The code uses integer variables for the minimum & maximum range of the sensor, then we define a long variable for duration, which is then used to calculate distance.
```

```
// Int = a variable for 16 bit numbers (-32,768 to 32,767), long is for 32 bit numbers (-2,147,483,648 to 2,147,483,647)
```

```
void setup() {
```

```
//Void setup () runs when the board boots up and sets the initial values.
```

```
  Serial.begin(9600); //starts the serial communication via USB  
  pinMode(trigPin, OUTPUT);  
  pinMode(echoPin, INPUT);  
  pinMode(LEDPin, OUTPUT); //board LED for testing  
}
```

```
// In this case it's setting whether the echo, trig and LED pins are inputs or outputs, and begins the serial communication.
```

```
//Serial communication means sending data one bit at a time over a communication channel. Serial.begin(speed) sets the rate that data is transmitted in bits/second (baud), in this example I'm using 9600. Other suggested rates for communicating with a PC include 300, 600, 1200, 2400, 4800, 14400, 19200, 28800, 38400, 57600, or 115200.
```

```
void loop() {
```

```
//Void loop () runs in real time and therefore allows for control of the board in real time.
```

```
  digitalWrite(trigPin, LOW);  
  delayMicroseconds(2);
```

```
  digitalWrite(trigPin, HIGH);  
  delayMicroseconds(10);
```

```
  digitalWrite(trigPin, LOW);  
  duration = pulseIn(echoPin, HIGH);  
  distance = duration / 58.2;
```

```
//DigitalWrite(trigPin, LOW); makes the trigger pin emit a 'low' pulse, which basically means the pulse is off.
```

```
//DigitalWrite(trigPin, HIGH) fires sound waves at the object in front of the sensor, while delayMicroseconds(10) means the pulse lasts for 10 microseconds.
```

```
//Duration = pulseIn(echoPin, HIGH)" measures how long it takes before the echo pin detects the sound waves bouncing back off the object.
```

```
//Finally, distance = duration / 58.2 is used to calculate distance based on the speed of sound.
```

```
  if (distance >= maximumRange || distance <= minimumRange) {  
    Serial.println("-1"); //if nothing observed output -1  
    digitalWrite(LEDPin, HIGH); // LED will be on
```

```
// Now we're error handling
```

```
// If nothing is detected then the output will be -1 and the LED will be on

} else {
    Serial.println(distance);
    digitalWrite(LEDPin, LOW);
}

// Otherwise the distance will be displayed and the LED will switch off

delay(10);

// Apparently this is to not overload the serial port. I reduced it from 50 to 10 with no adverse effects.

}
```

If I experiment with different sketches in the future then my first port of call would be to explore the NewPing libraries, as from the community posts I've read it seems like the best place to start. A quote from the NewPing developer Tim Eckel (2019):

“When I first received an ultrasonic sensor I was not happy with how poorly it performed. I soon realized the problem wasn't the sensor, it was the available ping and ultrasonic libraries causing the problem. The NewPing library totally fixes these problems, adds many new features, and breathes new life into these very affordable distance sensors.”

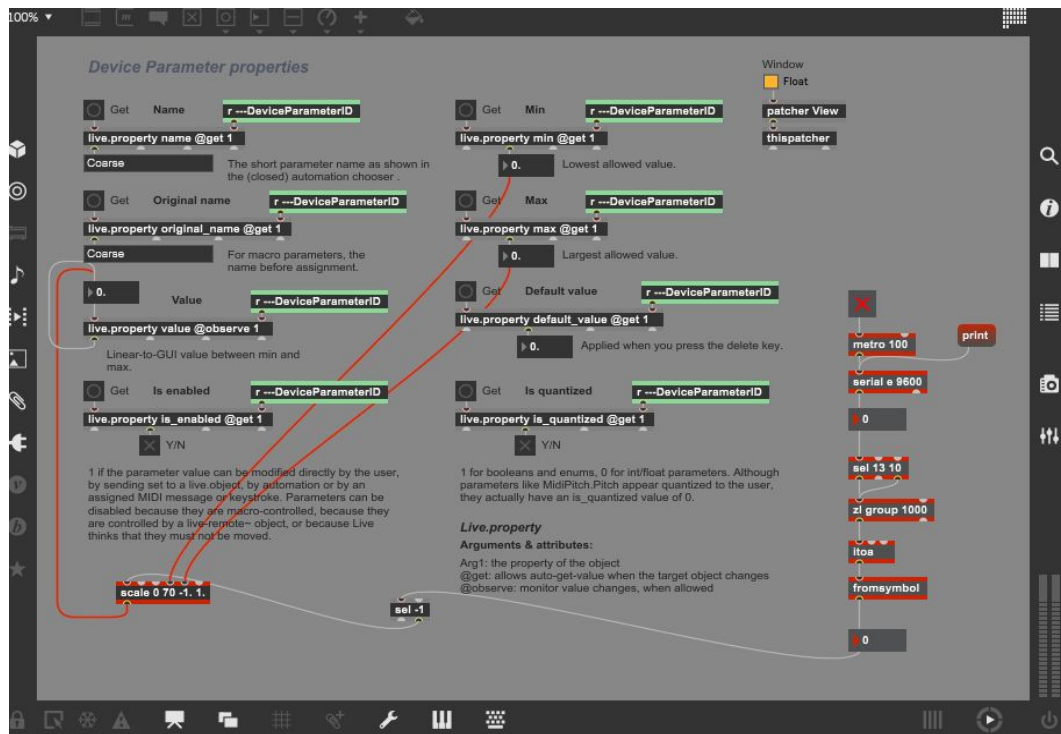
Max4Live Integration

There are many plugins available for getting Arduino and Ableton to talk to each other, including a connection kit from Ableton themselves. I had trouble getting the latter to work with the proximity sensor but later found a purpose-built Max4Live device which was a modification of the stock device Max.API.DeviceParam.

The front end:



Under the hood:



Everything in green is the original device and all the objects in red are modifications.

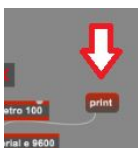
Problems Encountered

Max4Live patch didn't work

Initially I wasn't able to get the proximity sensor to affect variables in Live. Hours of troubleshooting, trial and error and 2kg of coffee later I found that one issue was that the serial object was set to the wrong port.



By pressing print the console displays which port the arduino is plugged into. This appears to refresh every time the Arduino is unplugged so it's necessary to perform this step every time. This issue will be addressed in the development roadmap section of this paper.



Once the port was changed data began to flow into ableton, but the console was still reporting errors en masse whenever the sensor didn't detect anything. The Arduino sketch contains code that returns a -1 value whenever there is no object in range. As it happens this confuses Ableton, causing it to down tools and not accept any values at all.

The solution was to add a sel object.



Had I used the left node at the bottom this would have filtered every result apart from the -1's, whereas the right node filters out -1's and lets everything else through. Sure enough this resulted in the ableton reading the serial data correctly.

It was now possible to observe continuous movement of values in the max4live patch when I moved my hand around above the sensor.



<- Imagine this with numbers rapidly scrolling

Great almost there! Now it's just a case of freezing the patch and we can start making some phat noise. Except it wasn't that simple..

Max4Live Patch backend needs to be open

For whatever reason the plugin only modulates parameters in Ableton if I have the backend of the max4live device open. This appears to be a bug as nothing in the device seems to be designed to stop working once it's frozen and the window is hidden.

This issue will also be addressed in the development roadmap section of this paper. While it's not much of a barrier for my own personal use, if I were to make a commercial version of this device then it would warrant a complete rethink of my approach, where I'd likely bypass max4live altogether.

Proximity measurements keep jumping around sporadically

Holding my hand still above the proximity sensor resulted in a large amount of jumping between distance readings. This was the case both in Ableton live and in the Arduino app console where one can directly observe the readings in cm:

```
22
22
25
24
24
27
-1
-1
-1
-1
```

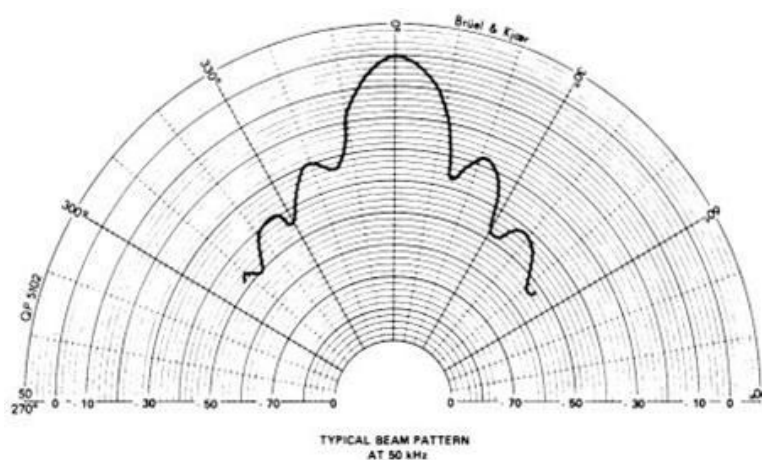
There are likely a number of reasons for this.

Firstly there is a margin of error that is inherent in the sensor I used. Based on research on community forums, anecdotally one can expect a margin of error of around 2-3cm although this depends on factors like distance from the object, temperature and so forth.

A community members study (Rigon2, 2014) found an absolute error of 0.035 cm per cm away from the sensor, while large anomalies were also observed in the data. For my use case these anomalies would potentially have resulted in large glitches in the sound produced.

The author then goes on to say *“Accuracy is rather volatile: I have measured average absolute errors ranging from 0.6 cm at 20 cm from the sensor to 5.7 cm at a distance of 180 cm.”* Perhaps there should be some kind of smoothing of the data to account for these jumps?

Secondly, the accuracy differs depending on the position of one's hand in relation to the sensor. Here is the beam pattern for the SR-04 sensor [Generation Robots, 2017]:



The sensor can detect objects within a 30 degree angle (which is wide compared to other types of sensor). However the beam appears to be concentrated around the center - evidently the readings will be much more accurate the closer to the center of the cone the object is. This is why these sensors often rotate when used in robotics in order to capture a range of measurements with the center of the cone.

The difficulty here for my project is twofold - firstly it's hard to tell if your hand is perfectly lined up or not, and secondly even if you could, it then requires a level of skill to keep it lined up while making expressive movements. The former issue will be addressed in the development roadmap. The latter just involves practise or a sensor with more concentrated frequencies at wider angles.

A third possibility I considered might have been be the time intervals between measurements as programmed in the Arduino sketch. However the findings of the aforementioned study (Rigon2, 2014) casts doubt on the effectiveness of this. *“With regard*

to the differences between measuring intervals, I have not noticed any difference of relevance between the intervals tested (1, 10, 50, 100 and 200 milliseconds)."

Another reason might be the need for a large and relatively flat object in order to observe steady readings. When pointing the sensor at the wall the readings were perfectly flat whereas they were sporadic when I used my hand. They improved when I used a piece of cardboard but this still wasn't rigid enough and it affected the playability of the instrument.

The most successful strategy I found was to modulate a parameter in Ableton with a small range of values. The glitches were very noticeable when I used a parameter like Frequency because one could cycle between any value from 20hz to 20,000hz. Hence a 2cm jump in the SR-04 reading resulted in a giant leap in the pitch of the sound in Ableton.

However if I used a modwheel plugin which can only bend one or two notes up or down, any errors / variations result in much smaller changes in pitch. Rather than being sporadic jumps, they result in a glitching effect which arguably adds a unique character to the sound.

Device In Action

The parameters I modulated included 'Pitchbend', using a bespoke plugin I found which is designed to emulate a modulation wheel on a keyboard. This delivered good results although the range of pitches is quite limited.

Sample:

<https://soundcloud.com/leoalexander/bass-sample-original/s-ujvNC>

In Context:

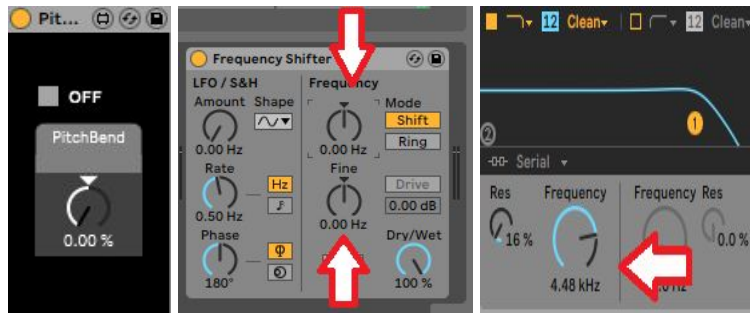
<https://soundcloud.com/leoalexander/bass-sample-full/s-cFFup?in=leoalexander/sets/midi-analogue-theremin-samples/s-NH7ex>

I also experimented with a frequency shifter although this behaved pretty erratically due to measurement errors and a wide range of values.

<https://soundcloud.com/leoalexander/high-sample-choppy/s-p3i7l?in=leoalexander/sets/midi-analogue-theremin-samples/s-NH7ex>

Mapping the device to a frequency cutoff knob provided interesting results, similar to a 'wah' pedal effect but with added artifacts due to the measurement errors. In this example I added distortion to a wavetable synth to create an industrial-style synth sound.

<https://soundcloud.com/leoalexander/filter-sample/s-41xGu?in=leoalexander/sets/midi-analogue-theremin-samples/s-NH7ex>



Suggested Use Case

The most effective use case I found for this device is for laying down a rhythmic pattern / vibe using the proximity sensor, then chopping up and pitching the recording to create musical sounding results.

By starting from an interesting place, the end result is usually unique and something that wouldn't have been possible without the device.

Another industrial example:

<https://soundcloud.com/leoalexander/second-song-sample/s-27S2y>

Market Research And User Testing

Focus Group Feedback

There were 11 people in the room, when asked the question "If you saw a friend using this would you buy one?" two people raised their hands and the general consensus was "It depends..."

Important factors identified in the session included:

- Ease of use
- Must look slick. It's currently difficult for potential customers to visualise the final product while it's a bare-bones prototype.
- Marketing collateral will be very important and this must include videos of the device being used impressively for composing music. Roli were mentioned as an example of a company which does this very well.
- The idea was also brought up of programming a robot to drive around walls / objects to manipulate the synth as a piece of viral marketing content.

User Testing

Sample size: 2. Before launching this as a commercial project I would use a sample closer to 30 and make this a mix of people I know and trust and people I've never met before.

Name: Jake

Background: Orchestral Composer

Relation: Housemate

Format: The workshop was performed on my laptop in Ableton, as his DAW (Cubase) is not currently supported. Hence he was limited to stock sounds that come with Ableton Suite rather than being able to use his orchestral libraries

Feedback: Enjoyed the ability to mess with samples and orchestral sounds. He also enjoyed how tactile it felt. Would consider buying something like this for his more industrial / experimental compositions.

This indicates that there is perhaps a niche market among more experimental producers and composers.

Name: Billy

Background: Songwriter, PR

Relation: Housemate

Format: See above

Feedback: Thought it was 'cool' and had fun trying it out. Probably wouldn't purchase it as he's not a producer. Enjoyed the samples/song ideas I'd created with it during the testing phase.

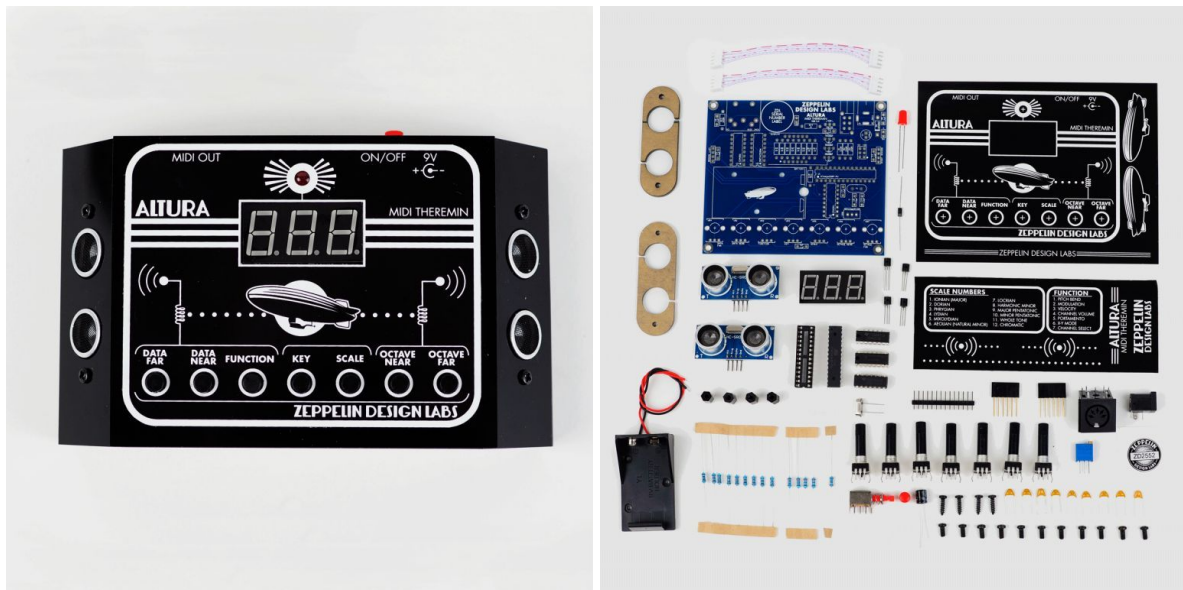
The device was already connected to Ableton when the tests occurred hence neither beta tester was asked to set the instrument up and get it working from scratch. This was due to the known issues around it which have already been documented above.

A different testing angle?

For a project like this, perhaps a more suitable testing process would be to get feedback on finished songs created using the device, as these are my most likely route to monetising it. If this feedback is positive then it could serve as a vehicle for promoting the device, but mainly it would serve to further my ambitions as a producer and cement the device as one of my USPs.

Closest Competitors

The closest commercial instrument to mine is probably the Altura Theremin Controller.



Price Point:

£57 for a kit or £95 for a fully assembled unit. This includes an outer shell, which I would also have to provide for my device if I were to sell it commercially. This would add to the current £7.99 build cost. However for personal use I'm not too fussed about the cosmetic appearance of my device.

Features:

The Altura has its own custom Arduino board with code pre-loaded. It uses the same motion sensors as my device but has two of them, as well as an LCD screen to display midi notes, and pentometers for modulating various parameters.

It's closer to an actual theremin in that the user can modulate two parameters at once, eg pitch and vibrato. To this end it has ~10 modes of operation whereby each hand sends a different parameter to the DAW and I imagine that each of these modes would have to be coded into the board using the Arduino language.

A key distinction between the two devices is that the Altura works as a plug-and-play midi controller for any DAW as it sends the proximity measurements as raw midi data. This is distinct from my device which is limited to Ableton Suite and requires a Max 4 Live plugin to interface with the Arduino and map the device to a parameter.

My device also requires the user to create a long note on the piano roll which can then be modulated, which seemingly isn't necessary with the Altura device.

Project Evaluation

As a commercial product, the Altura is currently a more viable proposition and serves as a good template for progressing my own design in the future.

However my current device scores very highly when compared with the success criteria defined at the beginning of the project:

Continuous pitch and filter control	✓	Achieved when mapping to mod wheel plugin
I'll use it regularly	✓	I will use it to add a unique flavour to productions, which will be noticeable even if the parts aren't prominent in the mix
Inexpensive to build	✓	Less than the price of a burger
Pre-existing code	✓	Yes, there seems to be code for almost anything out there..
Easy to play	✓	Once you get it working then yes

The ROI from unique-sounding productions is unclear at this stage and will difficult to accurately attribute directly to the device, but given the low cost of the investment there's a high probability that this device will make a profit.

Development Roadmap

The development roadmap can be split into two parts - what I can do to my current design and what my approach would be for a complete redesign.

Current design

LED light / laser



This would assist with the user knowing whether their hand is lined up with the sensor. This is especially important given the beam pattern illustrated earlier, where accuracy of the sensor is greatest at the center.

Modify Arduino Code - Newping Libraries

```

#define echoPin 11 // The number of the digital pin connected to the echo input
#define trigPin 12 // The number of the digital pin connected to the trigger output
#define LEDPin 13 // This enables the LED feedback for testing

int maximumRange = 70; // The max distance observed from the sensor
int minimumRange = 0;
long duration, distance; //duration used to calculate distance

void setup() {
  Serial.begin(9600); //starts the serial communication via USB
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LEDPin, OUTPUT); //Board LED for testing
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2); // pulse off

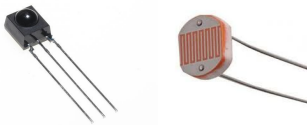
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10); // pulse for 10 microseconds

  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // echo pin listens / receives

```

As mentioned earlier in the paper, the inaccuracy of the readings might actually be down to the limitations of the Arduino code itself, hence it would be worth experimenting with purpose built Newping libraries which were designed to improve proximity sensor performance.

Test other sensors



Perhaps different sensors would improve playability or give different characteristics to the sound. Other options include an IR or Light Sensor.

It would also be worth researching ultrasonic sensors at a higher price point to see if their performance would be significantly better than the ones currently fitted.

Hack Modwheel Plugin



The modwheel provided smooth note modulation control due to the small range of values but is limited by only bending one step up or down. If it was possible to increase this range to 2 or 3 steps this would provide a lot more flexibility for making melodies, although it would increase the noticeability of any glitches.

It is possible that this is a limitation in Ableton as the 'modulation' midi parameter is designed to match the modwheel of a midi keyboard, which only bends a step up or down.

Complete Redesign

This would likely be necessary for a commercial release as only making the device available to Ableton Suite users would eliminate most of the market.

I would need to incorporate code for converting proximity sensor data to midi and make sure that both a PC and a MAC would recognise it as a midi controller device. The ttapa library (Github, 2019) would perhaps be a good place to start as it's purpose built for turning the Arduino into a midi controller.

There is also an app called Hairless MIDI which converts the serial communication data into MIDI and helps to route it to other applications.

Furthermore there are several instructables tutorials with different approaches to handling serial arduino data which I can peruse through for inspiration.

Conclusion - What's Next?

The time investment that would be required to make my device a commercial product makes it prohibitive, given the extent that Altura appears to have cornered the market from a technical standpoint. That being said, if I got to the point where the product was ready for sale then I could potentially compete with them via a superior viral marketing strategy.

Overall I believe there are other engineering projects I should prioritise (as previously discussed in class and in my tutor's 1-2-1s) which present a larger commercial opportunity as there is a clear gap in the market to be exploited. One of these would require learning Python from scratch, but the potential prize is much larger than if I was to invest those hours in optimising this product, to compete with an inexpensive offering from Altura which sounds like it already does a very good job.

Having said that, I will inevitably consider upgrading this device more and more as I get more use out of it, in order to improve my own user experience and the quality musical samples it creates.

References

Eckel, T. 2019. Arduinocc. [Online]. [4 April 2019]. Available from:
<https://playground.arduino.cc/Code/NewPing/>

Generation Robots. 2017. Génération Robots - Blog. [Online]. [4 April 2019]. Available from:
<https://www.generationrobots.com/blog/en/ultrasound-sensor-high-quality-ultrasound-sensor-s-available-now/>

Githubcom. 2019. GitHub. [Online]. [5 April 2019]. Available from:
https://github.com/ttapa/MIDI_controller

rigonz2. 2014. Boxcom. [Online]. [4 April 2019]. Available from:
<https://app.box.com/s/sj7du1n32in2777rcoi2>

Sparkfun.com. 2019. Sparkfun.com. [Online]. [4 April 2019]. Available from:
<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>