LittleBits Beat Sampler

Instruction Set

This document is a brief but detailed set of instructions and specifications to help you construct your own beat sampler and pitch converter using only littleBits™ components. Beat samplers are commonly used throughout the music industry. The purpose of the device is to record a sample beat and produce an output of the same beat pattern at a different pitch. The device is intended for a simple beat input such as the clapping of one's hands, but will work for most distinct impulse sounds. This set of instructions will walk you through the steps of constructing the device and writing the code associated with making the device function. The code will include comments that discuss how modifying certain values will produce customized results.

Background Information

LittleBits™

littleBits™ rapid prototyping technology is used for quickly assembling and disassembling circuits with interchangeable components. Each of these components typically have two magnetic sides: one positive and one negative that correspond to the input and output ends of each Bit. This design makes it easy to swap and add circuit components to improve the system. In this document, we will refer to each end of the littleBits as its input and output. Figure 1, found in the appendix, shows both the input and output ends of our example microphone bit. If the component facing you has writing you can read, the left end is the input and the right is the output.

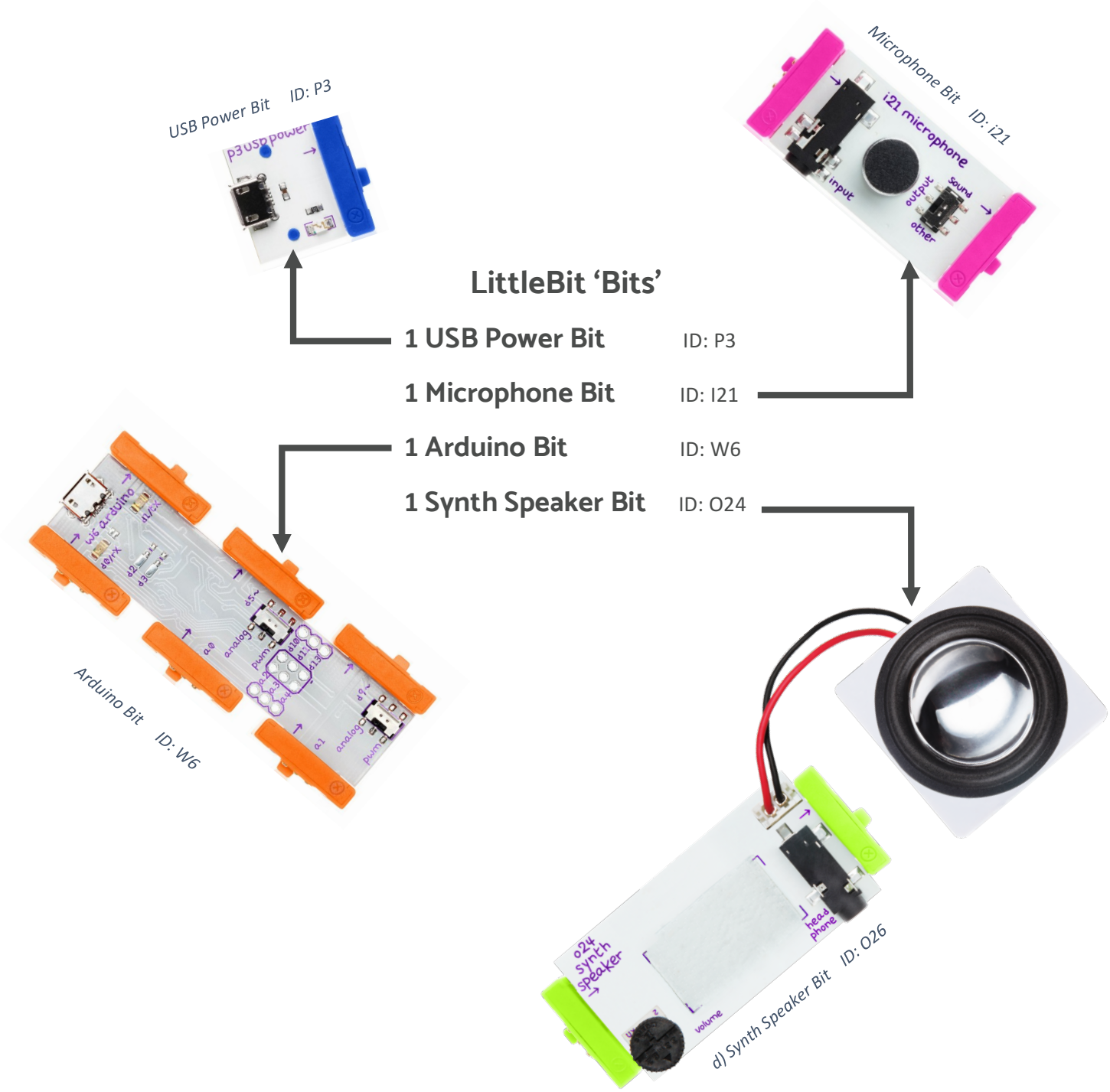
Arduino

Arduino is the other prominent technology used for this project. For this project, we will utilize both the Arduino programming language and an Arduino LittleBit component. The 'bit' itself is a device that can control and modify input and output signals. In coordination with the code, these signals can be modified in an endless number of ways to complete numerous objectives. The littleBit™ technology can coordinate with Arduino using USB connection and code can be uploaded to your littleBit™ circuit to modify the existing input and output signals.

Audience

This build is designed for anyone that can use a computer and has access to littleBits™ and Arduino products. The coding used for this project is intermediate level; however, the instructions will be written with comments and numbered so that even someone with no Arduino coding experience can build it.

Materials Needed

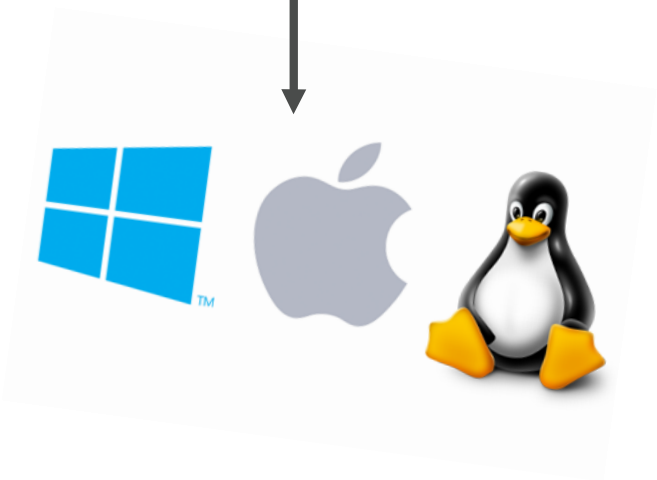


Power Accessories

2 USB Type A to USB Micro Cable

1 USB Power Adapter

1 Windows/Mac/Linux Computer



Instructions

1. Insert USB Cable into USB Adapter, and inset adapter into a wall outlet.

   Note: *The magnets that connect the bits together are not very strong. Make sure you have a solid connection and no debris between the bits.*

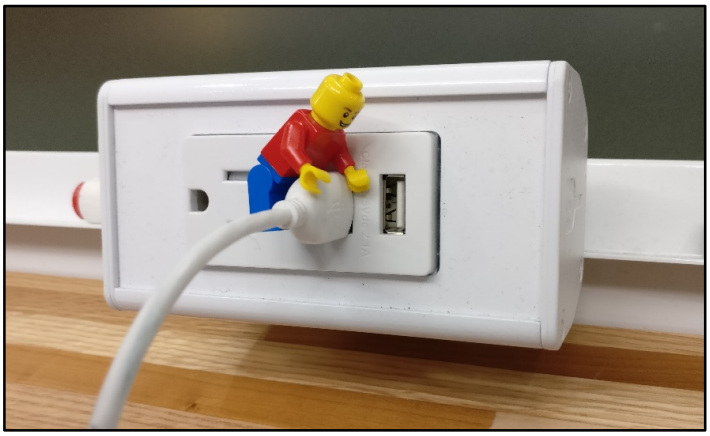


Figure 2: Insertion of the USB cable into USB enabled wall outlet

2. Connect the male Micro USB side of the cable used in step 1 to the micro USB female connector on the **Power Bit**.

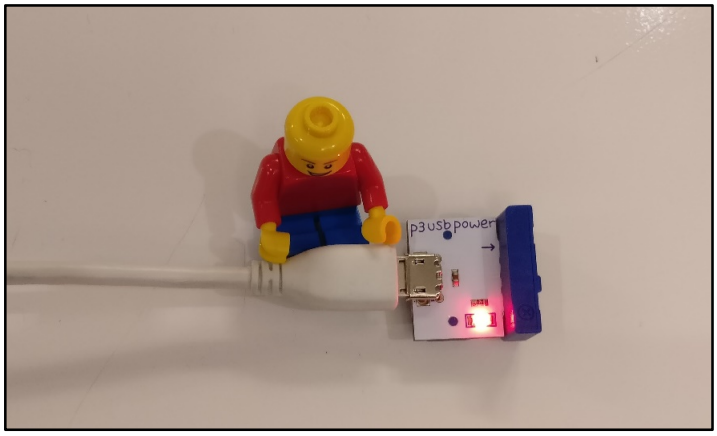


Figure 3: Insertion of the USB cable into the Power Bit

3. Magnetically connect the **Power Bit** output to the **Microphone Bit** input

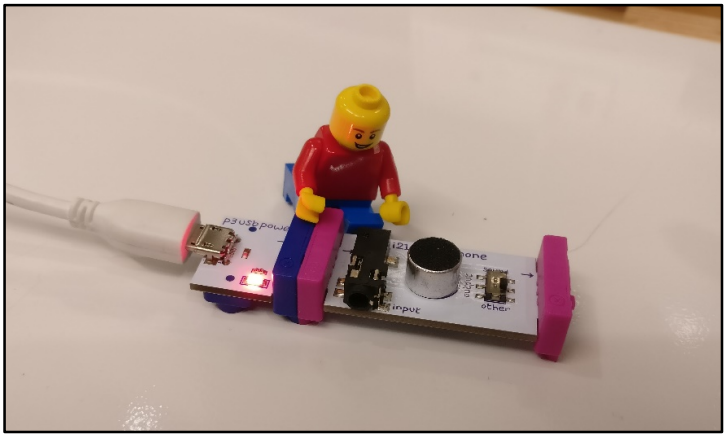


Figure 4: Connection of the Power Bit and the Microphone Bit

4. Magnetically connect the **Microphone Bit** output to the input port labeled *'a0'* on the **Arduino Bit**

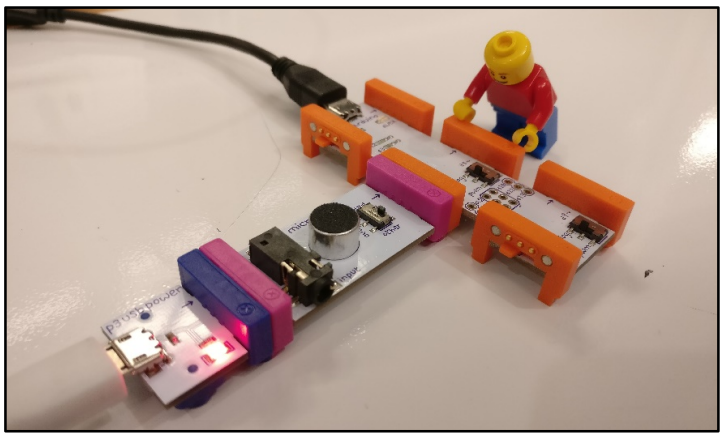


Figure 5: Connection of the Arduino Bit to the Microphone Bit

5. Magnetically connect the output port labeled *'d5'* on the **Arduino Bit** to the input of the **Synth Speaker Bit**

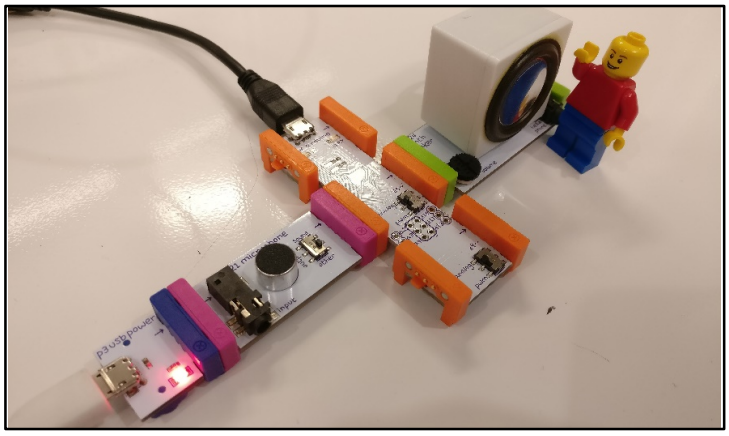


Figure 6: Connection of the Synth Speaker Bit and the Arduino Bit

6. Ensure that the switch underneath the *'d5'* text on the **Arduino Bit** is set to *'analog'* mode.

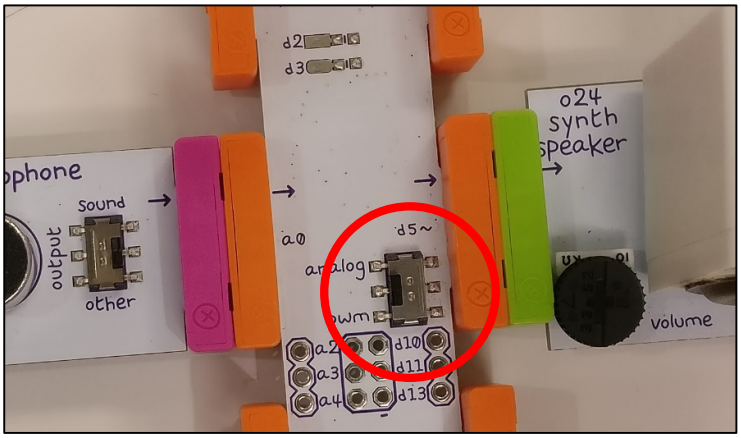


Figure 7: The analog mode switch

7. Connect the female micro-USB connector on the **Arduino Bit** to a <u>different</u> micro-USB cable, that then connects the computer.

Note: *Connecting the computer will not power this device, steps 1 & 2 must be completed.*

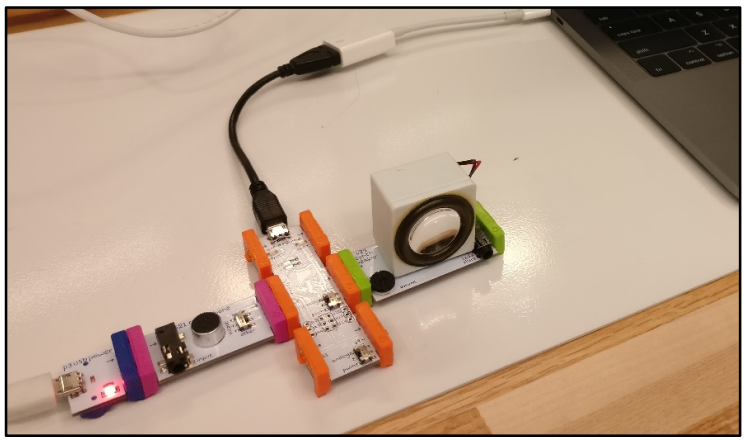


Figure 8: Connection of the Arduino bit to the computer with a 2nd USB cable

8. Go to the Arduino website and download the Arduino IDE for your operating system. Compatible operating systems include: Mac OS X, Windows, and Linux.

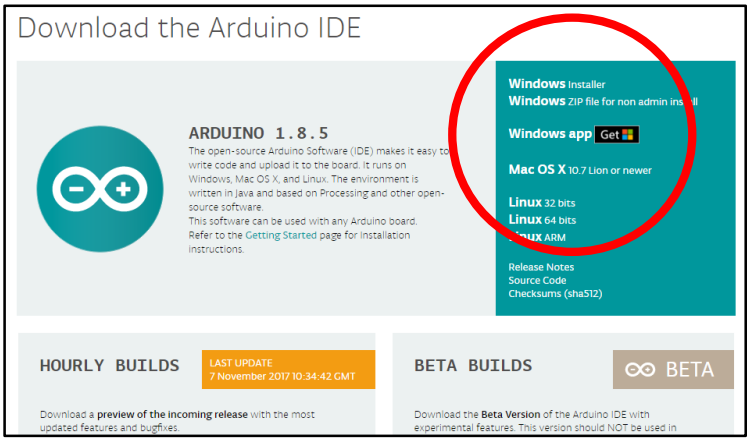


Figure 9: Arduino Software Webpage

9. Finish setting up the Arduino software by following this LittleBits™ official tutorial.

10. Copy and paste the code found in the Appendix section into a new Arduino sketch.

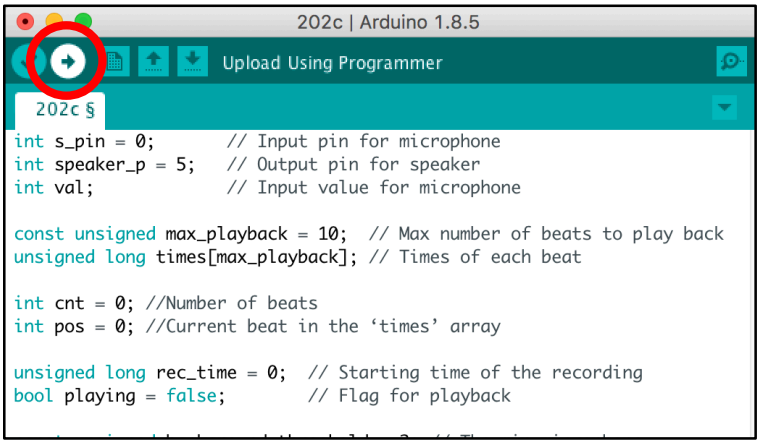


Figure 10: Arduino IDE Upload Button

11. Upload the sketch.

Specifications

General Description:

The device repeats user inputted beat patterns. Once powered, your recording will automatically start with the first distinct, loud sound you make. You can trigger the device by snapping your fingers, clapping your hands, or creating sounds any other way. After a set amount of time with no input, the device will enter into a playback mode. In this mode, your exact beat pattern will be played through the speaker, tuned to a desired frequency and volume.

Technical Description:

Size:

4.0'' x 4.25'' x 2.0''

Note: *For specific dimensions, see figure 10.*

Weight:

5.0 ounces

Note: *The weight specification does not include cable, computer, or power adapter weight.*

Volume:

0 – 70 dB

Note: *The volume can be adjusted by turning the dial on the* ***synth speaker bit****, shown in figure 12.*

Timing Accuracy:

1 Millisecond

Power Consumption:

.50 Watts (Standby)

.65 Watts (w/ Speaker Active)

Output Frequency Range:

20Hz – 20kHz (Human Hearing Range)

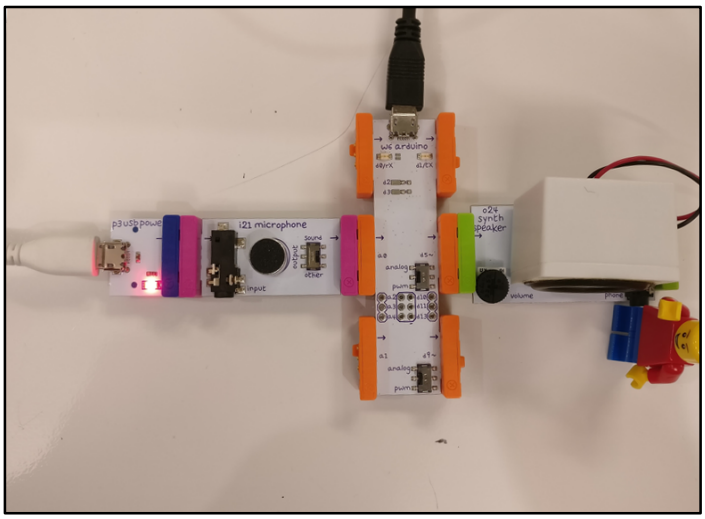


Figure 11: Device Dimensions

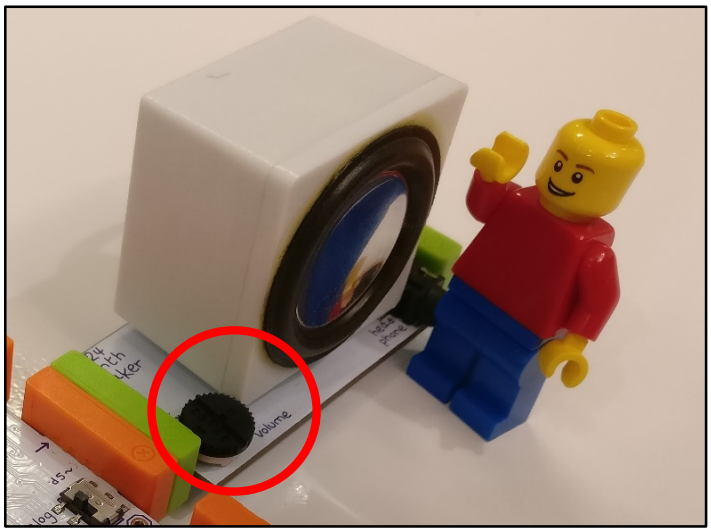


Figure 12: Volume Adjustment Knob

Program Operation:

The program is designed to have three states: listening, recording, and repeating.

It starts off in the listening state, and waits for a microphone input. Once it receives an input over a certain threshold defined by **background_threshold**, the program switches to the recording state.

During the recording state, the program reads microphone inputs over the **background_threshold** and stores the inputs in an array along with the time when the inputs were received. Once a certain amount of time has passed after the last input, defined by **timeout**, the program switches to the repeating state.

In the repeating state, the program plays a beat, and then it reads the time of the next beat and the time of the current beat, subtracting to get the time in-between the beats. It then waits this amount of time before playing the next beat. Beats are played at a user-defined **frequency**, for the **duration** in milliseconds.

Once the program has played all of the recorded beats, it returns to the listening state.

Note: *The Bolded parameters described above correspond to variables in the code. Tune them to your preferences.*

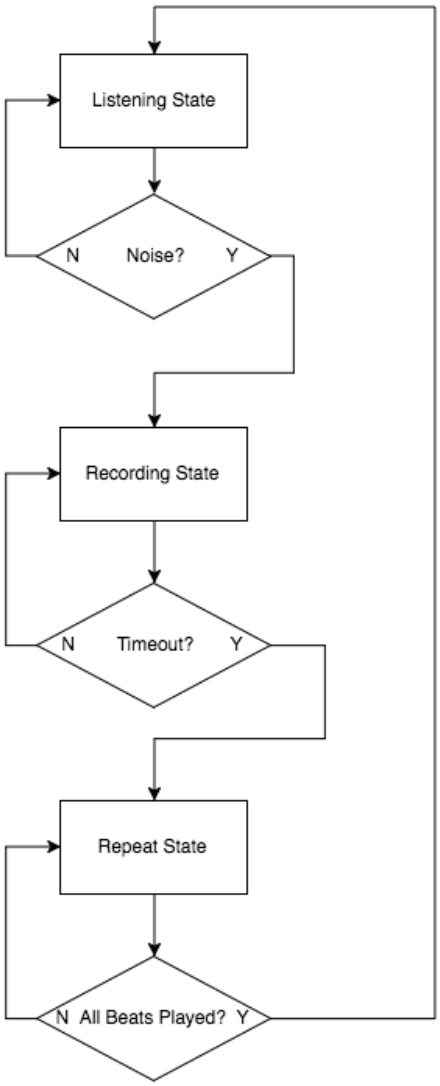


Figure 13: Program Flow Chart

Troubleshooting

	Problem	Solution
Extra Outputs	Extra outputs occur when background noises register as inputs.	increase the value of the variable `background_threshold` to a value between the current value and 255 which does not exceed the value of the measured input during a clap and the measured input of the background noise. To find an appropriate value, you can either print measured input values over time to the serial port and pick a threshold value based on your findings or randomly select a less.
No Playback	No playback sequence occurs after a recording sequence.	Make sure you have properly connected each component as described in the **Step by Step** section. The pin number used in the arduino code is printed on the board (`s_pin = 0 ; /* a0 */ speaker_p = 5; /* d5 */`). Additionally, ensure the value of the variable `background_threshold` does not exceed the measured value of the intended input. One final thing to check is to ensure that the speaker works and the volume is not set to off.
Undesirable Pitch	The pitch of the playback is either too high or too low	Modify the `frequency` variable. Increase this value for higher pitches and lower this value for lower pitches.
Undesirable Tone Duration	The Length of the playback tone is too long or too short.	Alter the `duration` variable. Increase this value for a longer playback tone lengths and decrease this value for shorter playback tone lengths.

Input Output Mismatch	The number of inputted beats are not reflected in the output.	Modify the `max_playback` variable to the maximum number of tones for a given playback sequence.
Missing Log Statements	The Serial print statements are not appearing on the serial terminal window.	Make sure the baud rate is matched to what the Arduino is programmed to use and that the serial port being used is the correct port. The Arduino typically programmed using a baud rate of 9600 bits/sec. Many serial terminals are defaulted to this value but can be modified if needed. If this is not the case, determine the baud rate your terminal uses and set that in line 25, replacing the existing value: 9600. If this does not fix the issue, ensure you have connected the USB cable to both your computer and the Arduino board.

Operating Conditions

For a fully functional and safe device,
only operate in the following conditions:

- A dry environment
- Temperatures less than 200° F
- A power supply (USB adapter) rated for 5v
- A vibration free environment
- A shock free environment

When using the littleBits™, adhere to all warnings listed on the littleBits™ website.

This project makes loud sounds. Please be mindful of your surroundings and lower the volume of the Synth Speaker bit before use.

Appendix

```
1   int s_pin = 0;        // Input pin for microphone
2   int speaker_p = 5;    // Output pin for speaker
3   int val;              // Input value for microphone
4
5   const unsigned max_playback = 10;  // Max number of beats to
6   play back
7   unsigned long times[max_playback]; // Times of each beat
8
9   int cnt = 0; //Number of beats
10  int pos = 0; //Current beat in the 'times' array
11
12  unsigned long rec_time = 0;  // Starting time of the recording
13  bool playing = false;        // Flag for playback
14
15  const unsigned background_threshold = 2; // The min microphone
16                                           // value to pick up
17                                           // as a sound
18
19  const unsigned timeout = 3000;  // The amount of time before
20                                  // stopping recording
21                                  // and playing back
22
23  const unsigned frequency = 700; // The frequency of the speaker
24  const unsigned duration = 100;  // The duration of a beat
25
26  void setup() {
27    pinMode(s_pin, INPUT);      // Set the input pin
28    pinMode(speaker_p, OUTPUT); // Set the output pin
29    rec_time = millis();        // Set the output pin
30  }
31
32  void loop() {
33    Serial.begin(9600);
34    val = analogRead(s_pin); // Read input from the microphone
35
36    // Check if the value of the input is above the threshold
37    // required, and not currently playing back, start recording
38    //
39    // Otherwise, check if 'timeout' amount of time has passed
40    // after the previous input, and if there are recorded values
41    // to play back
```

```
42    if (val > background_threshold && playing == false) {
43      rec_time = millis();   // Current time as the start time
44      times[cnt] = millis(); // Save the time of the beat
45      cnt++;                 // Increase the counter for
46                             // the number of beats recorded
47
48      delay(50); // Wait 50ms before trying to read input again
49    }
50    else if ((millis() - rec_time) > timeout && cnt != 0 || cnt ==
51  max_playback) {
52      playing = true; // Set playback to true
53      int delta = 200 // Set the default delta time to wait
54                      // in the case of one beat;
55
56      // Calculate the time between beats
57      // if there are at least two beats to analyze
58      if(pos + 1 < cnt)
59        delta = times[pos + 1] - times[pos];
60
61      tone(speaker_p, frequency, duration); // Play the beat
62                                            // from the speaker
63
64      delay(delta); // Wait the amount of time to the next beat
65
66      // Check if the current beat is the last
67      if (pos >= cnt - 1) {
68        pos = 0; // Reset the variables used for the next
69  recording
70        playing = false;
71        cnt = 0;
72      } else
73         pos ++; // Go to the next beat
74    }
75  }
```

Figure 14: Arduino Code

Figure 1: Littlebit Output & Input Distinction

Self-Assessment:

The greatest strength of our instruction/documentation set was the design. I think we did a good job with font variety, readability, text justification and format consistency.

In terms of improvement, I think the pictures we took of the device could have been lit better. Though the design is consistent, the drab pictures effect the presentation.

Zoe Schwartz, Alexandra Schneider, and Bryce has very constructive and informative feedback.