

Hearing for All

Mukund Venkatakrishnan
duPont Manual High School



A black and white photograph of a massive crowd of people, heavily blurred to convey a sense of scale and movement. The individuals are packed closely together, filling the frame from the foreground to the background. Overlaid in the center of the image is the number 360,000,000 in a clean, white, sans-serif font.

360,000,000

A black and white photograph of a group of young children, likely of East Asian descent, looking towards the camera. The children are of various ages, mostly between 1 and 5 years old. The central child is a young boy with dark hair, wearing a light-colored collared shirt, looking directly at the camera with a neutral expression. To his left, another child is smiling slightly. To his right, a girl is looking off-camera. In the background, several other children are visible, some looking at the camera and others looking away. The lighting is soft, and the overall tone is somber due to the monochrome palette. The number '324,000,000' is overlaid in a large, white, sans-serif font in the center of the image.

324,000,000



Affordability & Accessibility

\$1,500



Sweden: 3.926

UK: 2.809

United States: 2.452

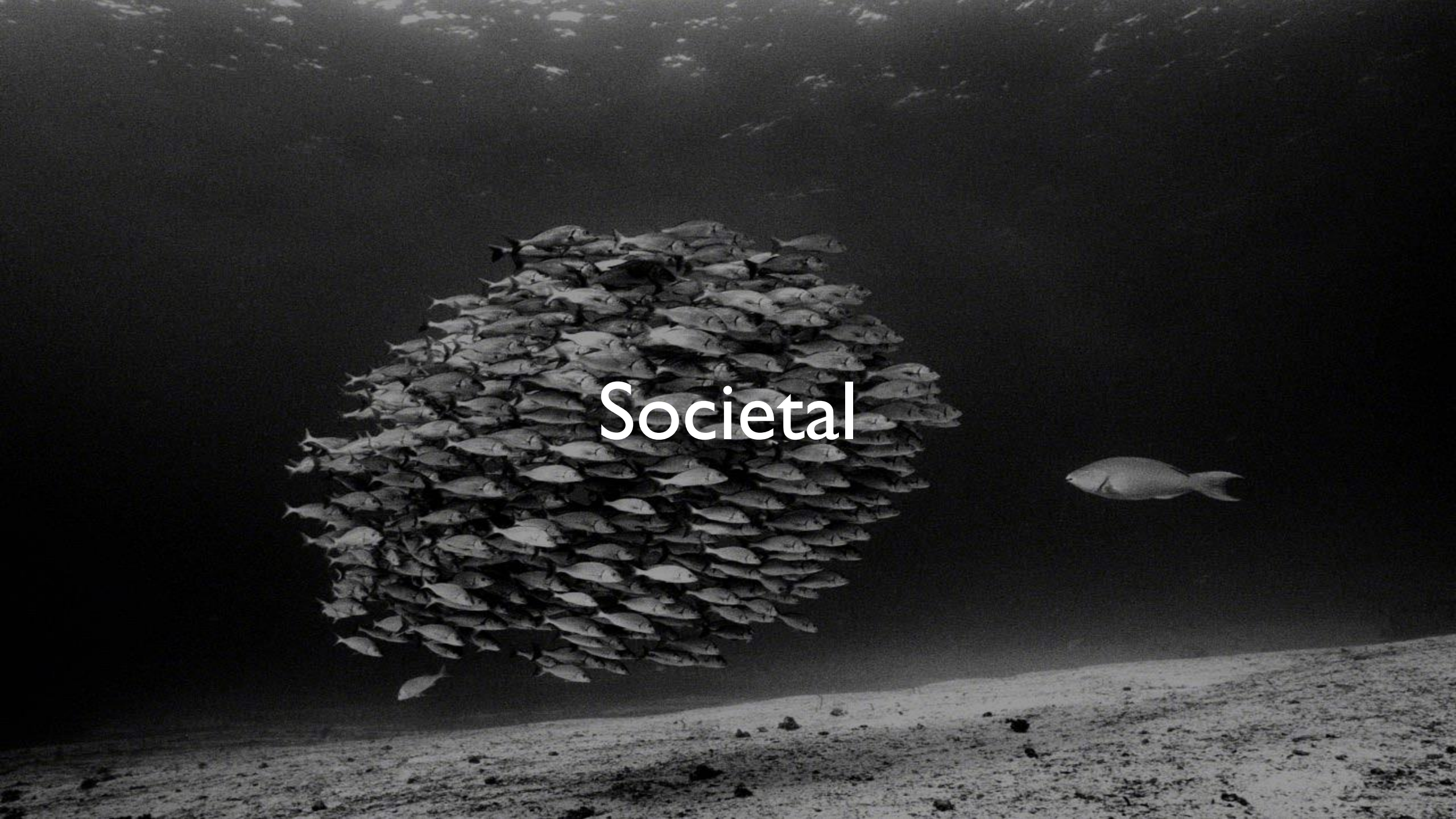
India: 0.702

Bolivia: 0.473

Liberia: 0.014



Functional



Societal



Economic

Measures hearing loss

Treats hearing loss

Solution?

Solution

Reduce price

Self-reliant

Solution

Hardware

Software

Solution

Signal
processor

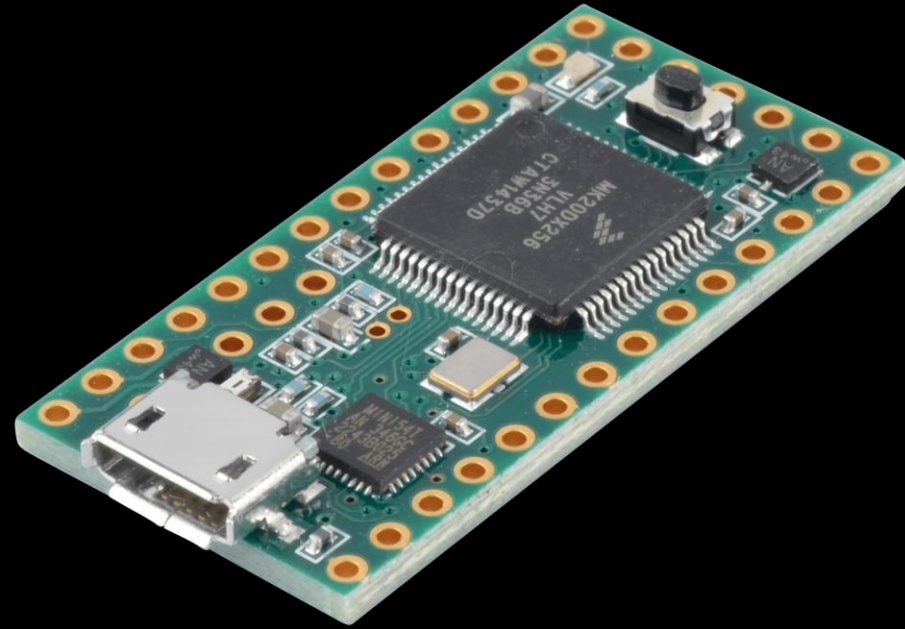
Code

How do they work?

Name of DSP	Ability to run complex filters	Ability to receive inputs	Ability to process sounds under .2 seconds
miniDSP	No	No	Not Tested
Arduino DSP Shield	No	Yes	Not Tested
PlainDSP	No	No	Not tested
Teensy 3.1	Yes	Yes	Yes

Teensy 3.1

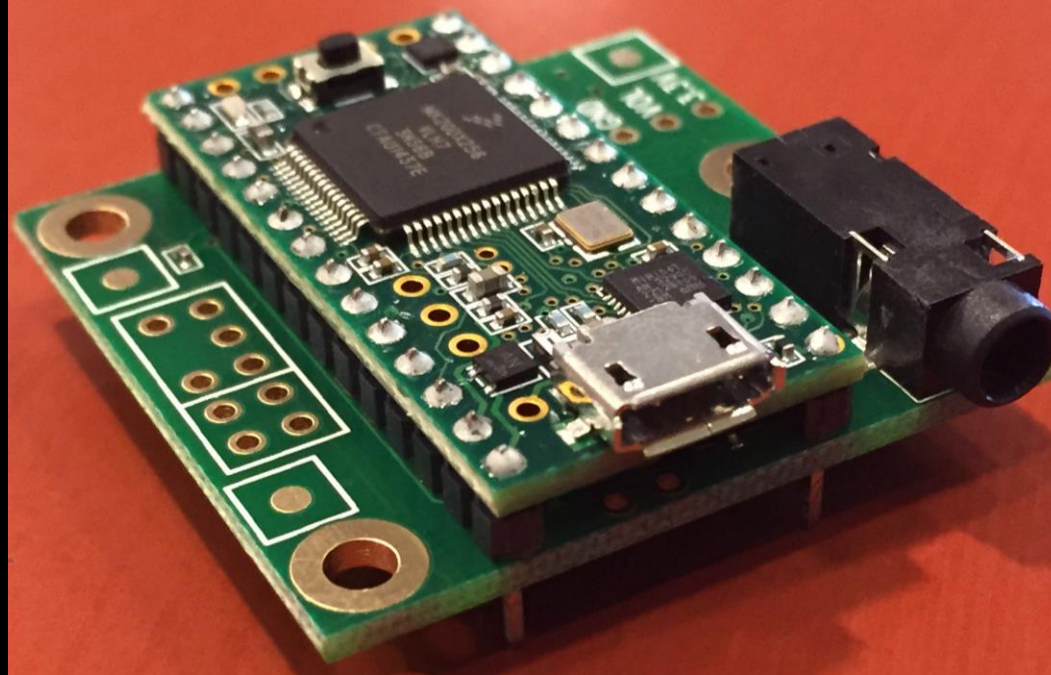
open-source



\$15

powerful processor

Hardware



Solution

DSP

Code

11-26V2

untitled

untitled

untitled

```

203
204 .....while(1){
205 .....Serial.println("Unable to access th
206 .....delay(500);
207 .....}
208
209 ..}
210
211 ..
212 ..calibration();
213 ..
214 ..delay(1000);
215 ..digitalWrite(ledPin, LOW);
216 ..int yesButtonState=digitalRead(yesButton
217 ..int noButtonState=digitalRead(noButton)
218 ..numberOfTests=0;
219 ..
220 ..while(numberOfTests<4){
221 ..
222 .....digitalWrite(ledPin, LOW);
223 .....int i=0;
224 .....int j=0;
225 .....
226 .....char* soundFiles[14][9]={
227 .....{"1250L.wav", "12510L.wav", "12520L
228 .....{"2500L.wav", "25010L.wav", "25020L
229 .....{"5000L.wav", "50010L.wav", "50020L
230 .....{"10000L.wav", "100010L.wav", "1000
231 .....{"20000L.wav", "200010L.wav", "2000
232 .....{"40000L.wav", "400010L.wav", "4000
233 .....{"80000L.wav", "800010L.wav", "8000
234 .....{"1250R.wav", "12510R.wav", "12520R
235 .....{"2500R.wav", "25010R.wav", "25020R
236 .....{"5000R.wav", "50010R.wav", "50020R
237 .....{"10000R.wav", "100010R.wav", "1000
238 .....{"20000R.wav", "200010R.wav", "2000
239 .....{"40000R.wav", "400010R.wav", "4000
240 .....{"80000R.wav", "800010R.wav", "8000
241 .....};
242 .....
243 .....while(i<=13){
244 .....digitalWrite(ledPin, LOW);
245 .....yesButtonState=digitalRead(yesButton
246 .....noButtonState=digitalRead(noButton)
247 .....j=0;
248 .....while(j<=8){
249 .....digitalWrite(ledPin, LOW);
250 .....if(i==0 && j==8){
251 .....digitalWrite(ledPin, HIGH);
252 .....originalResultsLeft[numberOfTes
253 .....i++;
254 .....j=0;
255 .....};
256 .....
257 .....if(i==7 && j==8){
258 .....digitalWrite(ledPin, HIGH);
259 .....i++;
260 .....j=0;
261 .....originalResultsRight[numberOfTes
262 .....};
263 .....
264 .....yesButtonState=digitalRead(yesButt
265 .....noButtonState=digitalRead(noButto
266 .....
267 .....playFile(soundFiles[i][j]);
268 .....Serial.print(i);
269 .....Serial.print(" ");
270 .....Serial.print(j);

```

```

1 AudioFilterBiquad biquad10;
2 AudioFilterBiquad biquad35;
3 AudioFilterBiquad biquad8;
4 AudioFilterBiquad biquad26;
5 AudioFilterBiquad biquad14;
6 AudioFilterBiquad biquad19;
7 AudioFilterBiquad biquad15;
8 AudioFilterBiquad biquad13;
9 AudioFilterBiquad biquad12;
10 AudioFilterBiquad biquad7;
11 AudioFilterBiquad biquad24;
12 AudioFilterBiquad biquad11;
13 AudioFilterBiquad biquad5;
14 AudioFilterBiquad biquad16;
15 AudioFilterBiquad biquad6;
16 AudioFilterBiquad biquad23;
17 AudioMixer4 mixer6;
18 AudioMixer4 mixer8;
19 AudioMixer4 mixer10;
20 AudioMixer4 mixer9;
21 AudioMixer4 mixer7;
22 AudioMixer4 mixer4;
23 AudioMixer4 mixer1;
24 AudioMixer4 mixer3;
25 AudioMixer4 mixer5;
26 AudioMixer4 mixer2;
27 AudioMixer4 mixer11;
28 AudioMixer4 mixer13;
29 AudioMixer4 mixer12;
30 AudioMixer4 mixer14;
31 AudioMixer4 mixer15;
32 AudioOutputI2S i2s;
33 AudioPlaySdWav playSdWav1;
34 AudioConnection patchCord01(i2s1,
35 AudioConnection patchCord02(i2s1,
36 AudioConnection patchCord03(i2s1,
37 AudioConnection patchCord04(i2s1,
38 AudioConnection patchCord05(i2s1,
39 AudioConnection patchCord06(i2s1,
40 AudioConnection patchCord07(i2s1,
41 AudioConnection patchCord08(i2s1,
42 AudioConnection patchCord09(i2s1,
43 AudioConnection patchCord10(i2s1,
44 AudioConnection patchCord11(i2s1,
45 AudioConnection patchCord12(i2s1,
46 AudioConnection patchCord13(i2s1,
47 AudioConnection patchCord14(i2s1,
48 AudioConnection patchCord15(i2s1,
49 AudioConnection patchCord16(i2s1,
50 AudioConnection patchCord17(i2s1,
51 AudioConnection patchCord18(i2s1,
52 AudioConnection patchCord19(i2s1,
53 AudioConnection patchCord20(i2s1,
54 AudioConnection patchCord21(i2s1,
55 AudioConnection patchCord22(i2s1,
56 AudioConnection patchCord23(i2s1,
57 AudioConnection patchCord24(i2s1,
58 AudioConnection patchCord25(biqua
59 AudioConnection patchCord26(biqua
60 AudioConnection patchCord27(biqua
61 AudioConnection patchCord28(biqua
62 AudioConnection patchCord29(biqua
63 AudioConnection patchCord30(biqua
64 AudioConnection patchCord31(biqua
65 AudioConnection patchCord32(biqua
66 AudioConnection patchCord33(biqua
67 AudioConnection patchCord34(biqua

```

```

1 AudioConnection patchCord35(biqua
2 AudioConnection patchCord36(biqua
3 AudioConnection patchCord37(biqua
4 AudioConnection patchCord38(biqua
5 AudioConnection patchCord39(biqua
6 AudioConnection patchCord40(biqua
7 AudioConnection patchCord41(biqua
8 AudioConnection patchCord42(biqua
9 AudioConnection patchCord43(biqua
10 AudioConnection patchCord44(biqua
11 AudioConnection patchCord45(biqua
12 AudioConnection patchCord46(biqua
13 AudioConnection patchCord47(biqua
14 AudioConnection patchCord48(biqua
15 AudioConnection patchCord49(biqua
16 AudioConnection patchCord50(biqua
17 AudioConnection patchCord51(biqua
18 AudioConnection patchCord52(biqua
19 AudioConnection patchCord53(biqua
20 AudioConnection patchCord54(biqua
21 AudioConnection patchCord55(biqua
22 AudioConnection patchCord56(biqua
23 AudioConnection patchCord57(biqua
24 AudioConnection patchCord58(biqua
25 AudioConnection patchCord59(biqua
26 AudioConnection patchCord60(biqua
27 AudioConnection patchCord61(biqua
28 AudioConnection patchCord62(biqua
29 AudioConnection patchCord63(mixer
30 AudioConnection patchCord64(mixer
31 AudioConnection patchCord65(mixer
32 AudioConnection patchCord66(mixer
33 AudioConnection patchCord67(mixer
34 AudioConnection patchCord68(mixer
35 AudioConnection patchCord69(mixer
36 AudioConnection patchCord70(mixer
37 AudioConnection patchCord71(mixer
38 AudioConnection patchCord72(mixer
39 AudioConnection patchCord73(mixer
40 AudioConnection patchCord74(mixer
41 AudioConnection patchCord75(mixer
42 AudioConnection patchCord76(mixer
43 AudioConnection patchCord92(playS
44 AudioConnection patchCord93(playS
45 AudioConnection patchCord77(mixer
46 AudioConnection patchCord78(mixer
47 AudioControlSGTL5000 sgtl5000_1;
48 // GUItool: end automatically generated co
49
50 const int myInput = AUDIO_INPUT_LINEIN;
51
52 void setup() {
53 Serial.begin(9600);
54 pinMode(yesButton, INPUT);
55 pinMode(noButton, INPUT);
56 pinMode(ledPin, OUTPUT);
57
58 // Audio connections require memory to w
59 // detailed information, see the MemoryA
60 sgtl5000_1.enable();
61
62 sgtl5000_1.volume(0.5);
63 AudioMemory(150);
64 SPI.setMOSI(7);
65 SPI.setSCK(14);
66 if (!(SD.begin(10))) {
67

```

```

1 while(1){
2 Serial.println("Unable to access the
3 delay(500);
4 }
5
6 }
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```


11-26V2

untitled

untitled

untitled

```

484 mixer7.gain(1,mixGain(result250L, result250R));
485 mixer7.gain(2,mixGain(result500L, result500R));
486 mixer7.gain(3,mixGain(result1000L, result1000R));
487
488 mixer8.gain(0,mixGain(result500L, result500R));
489 mixer8.gain(1,mixGain(result1000L, result1000R));
490 mixer8.gain(2,mixGain(result2000L, result2000R));
491 mixer8.gain(3,mixGain(result4000L, result4000R));
492
493 mixer9.gain(0,mixGain(result2000L, result2000R));
494 mixer9.gain(1,mixGain(result4000L, result4000R));
495 mixer9.gain(2,mixGain(result8000L, result8000R));
496 mixer9.gain(3,mixGain(result16000L, result16000R));
497
498 mixer10.gain(0,mixGain(result4000L, result4000R));
499 mixer10.gain(1,mixGain(result8000L, result8000R));
500 mixer10.gain(2,mixGain(result16000L, result16000R));
501 mixer10.gain(3,mixGain(result32000L, result32000R));
502
503
504 double mixGain(int dB1, int dB2, double B
505 double mix = ((point*(dB2-dB1))/BW)+dB1
506 mix=pow(10, mix/20);
507 mix = (mix-0.1276)/1.7754;
508 Serial.println(mix);
509 return mix;
510 }
511
512 void calibration(){
513 int yesButtonState=digitalRead(yesButton);
514 int noButtonState=digitalRead(noButton);
515
516 while(yesButtonState==1 && noButtonState==0){
517 yesButtonState=digitalRead(yesButton);
518 noButtonState=digitalRead(noButton);
519
520 Serial.print("Calibrating...");
521 Serial.println("");
522
523 sgtl5000_1.volume(.9);
524 playFile("1234.WAV");
525 delay(1000);
526
527 Serial.println("");
528 Serial.print("Calibration complete.");
529 digitalWrite(ledPin, HIGH);
530 }
531
532 void playFile(char* filename)
533 {
534 Serial.print("Playing file: ");
535 Serial.println(filename);
536
537 // Start playing the file. This sketch
538 // run while the file plays.
539 playSdWav1.play(filename);
540
541 // A brief delay for the library read
542 delay(5);
543
544 // Simply wait for the file to finish p
545 while (playSdWav1.isPlaying()) {
546 float vol = analogRead(1);
547 vol = vol / 1024;
548 sgtl5000_1.volume(vol);
549 }
550 }
551

```

```

1 Serial.print("\n");
2
3 //Results being recorded//
4 if(yesButtonState==0 && noButtonState==1){
5 digitalWrite(ledPin, HIGH);
6 delay(1000);
7 //Results being recorded//
8 if(i<=6){
9 originalResultsLeft[numberOfTests+i]=result125L;
10 originalResultsLeft[numberOfTests+i]=result250L;
11 originalResultsLeft[numberOfTests+i]=result500L;
12 originalResultsLeft[numberOfTests+i]=result1000L;
13 originalResultsLeft[numberOfTests+i]=result2000L;
14 originalResultsLeft[numberOfTests+i]=result4000L;
15 originalResultsLeft[numberOfTests+i]=result8000L;
16 originalResultsLeft[numberOfTests+i]=result16000L;
17 }
18
19 if(yesButtonState==1 && noButtonState==0){
20 digitalWrite(ledPin, HIGH);
21 delay(1000);
22 //Results being recorded//
23 if(i<=6){
24 originalResultsRight[numberOfTests+i]=result125R;
25 originalResultsRight[numberOfTests+i]=result250R;
26 originalResultsRight[numberOfTests+i]=result500R;
27 originalResultsRight[numberOfTests+i]=result1000R;
28 originalResultsRight[numberOfTests+i]=result2000R;
29 originalResultsRight[numberOfTests+i]=result4000R;
30 originalResultsRight[numberOfTests+i]=result8000R;
31 originalResultsRight[numberOfTests+i]=result16000R;
32 }
33
34 if(j==0){
35 digitalWrite(ledPin, HIGH);
36 //Results being recorded//
37 if(i<=5){
38 originalResultsLeft[numberOfTests+i]=result125L;
39 originalResultsLeft[numberOfTests+i]=result250L;
40 originalResultsLeft[numberOfTests+i]=result500L;
41 originalResultsLeft[numberOfTests+i]=result1000L;
42 originalResultsLeft[numberOfTests+i]=result2000L;
43 originalResultsLeft[numberOfTests+i]=result4000L;
44 originalResultsLeft[numberOfTests+i]=result8000L;
45 originalResultsLeft[numberOfTests+i]=result16000L;
46 }
47
48 if(j==0){
49 digitalWrite(ledPin, HIGH);
50 //Results being recorded//
51 if(i<=5){
52 originalResultsRight[numberOfTests+i]=result125R;
53 originalResultsRight[numberOfTests+i]=result250R;
54 originalResultsRight[numberOfTests+i]=result500R;
55 originalResultsRight[numberOfTests+i]=result1000R;
56 originalResultsRight[numberOfTests+i]=result2000R;
57 originalResultsRight[numberOfTests+i]=result4000R;
58 originalResultsRight[numberOfTests+i]=result8000R;
59 originalResultsRight[numberOfTests+i]=result16000R;
60 }
61
62 //leftResultsFinal[i]=originalResultsLeft[i];
63 //rightResultsFinal[i]=originalResultsRight[i];
64
65 //leftResultsFinal[i]=originalResultsLeft[i];
66 //rightResultsFinal[i]=originalResultsRight[i];
67

```

```

1 //Outputs left side results to the user
2 Serial.print("Left side results: ");
3 for (int i = 0; i < 7; ++i){
4 Serial.print(leftResultsFinal[i]);
5 Serial.print(", ");
6 }
7
8 //Stores the values of the results to pass to the next stage
9 result125L = leftResultsFinal[0];
10 result250L = leftResultsFinal[1];
11 result500L = leftResultsFinal[2];
12 result1000L = leftResultsFinal[3];
13 result2000L = leftResultsFinal[4];
14 result4000L = leftResultsFinal[5];
15 result8000L = leftResultsFinal[6];
16 Serial.println("");
17
18 //Outputs right side results to the user
19 Serial.print("Right side results: ");
20 for (int i = 0; i < 7; ++i){
21 Serial.print(rightResultsFinal[i]);
22 Serial.print(", ");
23 }
24
25 //Stores the values of the results to pass to the next stage
26 result125R = rightResultsFinal[0];
27 result250R = rightResultsFinal[1];
28 result500R = rightResultsFinal[2];
29 result1000R = rightResultsFinal[3];
30 result2000R = rightResultsFinal[4];
31 result4000R = rightResultsFinal[5];
32 result8000R = rightResultsFinal[6];
33 Serial.println("");
34 Serial.println("");
35
36 Serial.print("TEST COMPLETE");
37 Serial.println("");
38
39 //format
40 //biquad.setBandpass(stage, center frequency, Q, gain);
41 //biquad1.setBandpass(0, 125, 125/28.9);
42 //biquad2.setBandpass(0, 157.5, 157.5/36.5);
43 //biquad3.setBandpass(0, 198.4, 198.4/45.9);
44 //biquad4.setBandpass(0, 250, 250/57.9);
45 //biquad5.setBandpass(0, 315, 315/73);
46 //biquad6.setBandpass(0, 396.9, 396.9/88.8);
47 //biquad7.setBandpass(0, 500, 500/115.8);
48 //biquad8.setBandpass(0, 630, 630/145.9);
49 //biquad9.setBandpass(0, 793.7, 793.7/183);
50 //biquad10.setBandpass(0, 1000, 1000/231.6);
51 //biquad11.setBandpass(0, 1259.9, 1259.9/2);
52 //biquad12.setBandpass(0, 1587.4, 1587.4/3);
53 //biquad13.setBandpass(0, 2000, 2000/463.1);
54 //biquad14.setBandpass(0, 2519.8, 2519.8/5);
55 //biquad15.setBandpass(0, 3174.8, 3174.8/7);
56 //biquad16.setBandpass(0, 4000, 4000/926.2);
57 //biquad17.setBandpass(0, 5039.7, 5039.7/1);
58 //biquad18.setBandpass(0, 6349.6, 6349.6/1);
59 //biquad19.setBandpass(0, 8000, 8000/1852);
60 //biquad20.setBandpass(0, 125, 125/28.9);
61 //biquad21.setBandpass(0, 157.5, 157.5/36.5);
62 //biquad22.setBandpass(0, 198.4, 198.4/45.9);
63 //biquad23.setBandpass(0, 250, 250/57.9);
64 //biquad24.setBandpass(0, 315, 315/73);
65 //biquad25.setBandpass(0, 396.9, 396.9/88.8);
66 //biquad26.setBandpass(0, 500, 500/115.8);
67 //biquad27.setBandpass(0, 630, 630/145.9);
68 //biquad28.setBandpass(0, 793.7, 793.7/183);
69 //biquad29.setBandpass(0, 1000, 1000/231.6);
70 //biquad30.setBandpass(0, 1259.9, 1259.9/2);
71 //biquad31.setBandpass(0, 1587.4, 1587.4/3);
72 //biquad32.setBandpass(0, 2000, 2000/463.1);
73 //biquad33.setBandpass(0, 2519.8, 2519.8/5);
74 //biquad34.setBandpass(0, 3174.8, 3174.8/7);
75 //biquad35.setBandpass(0, 4000, 4000/926.2);
76 //biquad36.setBandpass(0, 5039.7, 5039.7/1);
77 //biquad37.setBandpass(0, 6349.6, 6349.6/1);
78 //biquad38.setBandpass(0, 8000, 8000/1852);
79

```

```

1 biquad21.setBandpass(0, 157.5, 157.5/36.5);
2 biquad22.setBandpass(0, 198.4, 198.4/45.9);
3 biquad23.setBandpass(0, 250, 250/57.9);
4 biquad24.setBandpass(0, 315, 315/73);
5 biquad25.setBandpass(0, 396.9, 396.9/88.8);
6 biquad26.setBandpass(0, 500, 500/115.8);
7 biquad27.setBandpass(0, 630, 630/145.9);
8 biquad28.setBandpass(0, 793.7, 793.7/183);
9 biquad29.setBandpass(0, 1000, 1000/231.6);
10 biquad30.setBandpass(0, 1259.9, 1259.9/2);
11 biquad31.setBandpass(0, 1587.4, 1587.4/3);
12 biquad32.setBandpass(0, 2000, 2000/463.1);
13 biquad33.setBandpass(0, 2519.8, 2519.8/5);
14 biquad34.setBandpass(0, 3174.8, 3174.8/7);
15 biquad35.setBandpass(0, 4000, 4000/926.2);
16 biquad36.setBandpass(0, 5039.7, 5039.7/1);
17 biquad37.setBandpass(0, 6349.6, 6349.6/1);
18 biquad38.setBandpass(0, 8000, 8000/1852);
19
20 unsigned long last_time = millis();
21
22 void loop(){
23 sgtl5000_1.inputSelect(myInput);
24 if (millis() - last_time >= 2500) {
25 Serial.print("Proc = ");
26 Serial.print(AudioProcessorUsage());
27 Serial.print(" (");
28 Serial.print(AudioProcessorUsageMax());
29 Serial.print(", Mem = ");
30 Serial.print(AudioMemoryUsage());
31 Serial.print(" (");
32 Serial.print(AudioMemoryUsageMax());
33 Serial.println(")");
34 last_time = millis();
35 }
36
37 //mixGain(int dB1, int dB2, double BW, double dB1);
38 mixer1.gain(0,mixGain(0, result125L, result125R));
39 mixer1.gain(1,mixGain(result125L, result125R));
40 mixer1.gain(2,mixGain(result125L, result125R));
41 mixer1.gain(3,mixGain(result125L, result125R));
42
43 mixer2.gain(0,mixGain(result250L, result250R));
44 mixer2.gain(1,mixGain(result250L, result250R));
45 mixer2.gain(2,mixGain(result250L, result250R));
46 mixer2.gain(3,mixGain(result250L, result250R));
47
48 mixer3.gain(0,mixGain(result500L, result500R));
49 mixer3.gain(1,mixGain(result500L, result500R));
50 mixer3.gain(2,mixGain(result500L, result500R));
51 mixer3.gain(3,mixGain(result500L, result500R));
52
53 mixer4.gain(0,mixGain(result1000L, result1000R));
54 mixer4.gain(1,mixGain(result1000L, result1000R));
55 mixer4.gain(2,mixGain(result1000L, result1000R));
56 mixer4.gain(3,mixGain(result1000L, result1000R));
57
58 mixer5.gain(0,mixGain(result2000L, result2000R));
59 mixer5.gain(1,mixGain(result2000L, result2000R));
60 mixer5.gain(2,mixGain(result2000L, result2000R));
61 mixer5.gain(3,mixGain(result2000L, result2000R));
62
63 mixer6.gain(0,mixGain(result4000L, result4000R));
64 mixer6.gain(1,mixGain(result4000L, result4000R));
65 mixer6.gain(2,mixGain(result4000L, result4000R));
66 mixer6.gain(3,mixGain(result4000L, result4000R));
67
68 mixer7.gain(0,mixGain(result8000L, result8000R));
69 mixer7.gain(1,mixGain(result8000L, result8000R));
70 mixer7.gain(2,mixGain(result8000L, result8000R));
71 mixer7.gain(3,mixGain(result8000L, result8000R));
72

```

```
573 void playFile(char* filename)
574 {
575     Serial.print("Playing file: ");
576     Serial.println(filename);
577
578     // Start playing the file. This sketch continues to
579     // run while the file plays.
580     playSdWav1.play(filename);
581
582     // A brief delay for the library read WAV info
583     delay(5);
584
585     // Simply wait for the file to finish playing.
586     while (playSdWav1.isPlaying()) {
587         float vol = analogRead(1);
588         vol = vol / 1024;
589         sgtl5000_1.volume(vol);
590     }
591 }
```

```
553 void calibration(){
554     int yesButtonState=digitalRead(yesButton);
555     int noButtonState=digitalRead(noButton);
556
557     while(yesButtonState==1 && noButtonState==1){
558         yesButtonState=digitalRead(yesButton);
559         noButtonState=digitalRead(noButton);
560
561         Serial.print("Calibrating...");
562         Serial.print("\n");
563
564         sgtl5000_1.volume(.9);
565         playFile("1234.WAV");
566         delay(1000);
567     };
568     Serial.print("\n");
569     Serial.print("Calibration complete.");
570     digitalWrite(ledPin, HIGH);
571 }
```

```

520 double mixGain(int dB1, int dB2, double BW, double point){
521
522     //Calculating dB loss based on linear relationship
523     double mix = ((point*(dB2-dB1))/BW)+dB1;
524
525     //Kneepoints
526     if(mix<=10 && mix>=0){
527         mix=mix-20;
528     }
529     if(mix<=20 && mix>=10){
530         mix=mix-20;
531     }
532     if(mix<=30 && mix>=20){
533         mix=mix;
534     }
535     if(mix<=40 && mix>=30){
536         mix=mix-2.5;
537     }
538     if(mix<=50 && mix>=40){
539         mix=mix-10;
540     }
541     if(mix<=70 && mix>=50){
542         mix=mix-25;
543     }
544
545     mix=
546     //Conversion from dB to mixer coefficient
547     mix=0.4637*pow(2.71828, mix*0.1277);
548     Serial.println(mix);
549     return mix;
550     delay(2000);
551 }

```


11-26V2

untitled

untitled

untitled

```

203
204 .....while (1){
205 .....Serial.println("Unable to access th
206 .....delay(500);
207 .....}
208
209 ..}
210
211 ..
212 ..calibration();
213 ..
214 ..delay(1000);
215 ..digitalWrite(ledPin, LOW);
216 ..int yesButtonState=digitalRead(yesButton
217 ..int noButtonState=digitalRead(noButton)
218 ..numberOfTests=0;
219 ..
220 ..while(numberOfTests<4){
221 ..
222 .....digitalWrite(ledPin, LOW);
223 .....int i=0;
224 .....int j=0;
225 .....
226 .....char* soundFiles[14][9]={
227 .....{"1250L.wav", "12510L.wav", "12520L
228 .....{"2500L.wav", "25010L.wav", "25020L
229 .....{"5000L.wav", "50010L.wav", "50020L
230 .....{"10000L.wav", "100010L.wav", "1000
231 .....{"20000L.wav", "200010L.wav", "2000
232 .....{"40000L.wav", "400010L.wav", "4000
233 .....{"80000L.wav", "800010L.wav", "8000
234 .....{"1250R.wav", "12510R.wav", "12520R
235 .....{"2500R.wav", "25010R.wav", "25020R
236 .....{"5000R.wav", "50010R.wav", "50020R
237 .....{"10000R.wav", "100010R.wav", "1000
238 .....{"20000R.wav", "200010R.wav", "2000
239 .....{"40000R.wav", "400010R.wav", "4000
240 .....{"80000R.wav", "800010R.wav", "8000
241 .....};
242 .....
243 .....while(i<=13){
244 .....digitalWrite(ledPin, LOW);
245 .....yesButtonState=digitalRead(yesButton
246 .....noButtonState=digitalRead(noButton)
247 .....j=0;
248 .....while(j<=8){
249 .....digitalWrite(ledPin, LOW);
250 .....if(i==0 && j==8){
251 .....digitalWrite(ledPin, HIGH);
252 .....originalResultsLeft[numberOfTes
253 .....i++;
254 .....j=0;
255 .....};
256 .....
257 .....if(i==7 && j==8){
258 .....digitalWrite(ledPin, HIGH);
259 .....i++;
260 .....j=0;
261 .....originalResultsRight[numberOfTes
262 .....};
263 .....
264 .....yesButtonState=digitalRead(yesButt
265 .....noButtonState=digitalRead(noButto
266 .....
267 .....playFile(soundFiles[i][j]);
268 .....Serial.print(i);
269 .....Serial.print(" ");
270 .....Serial.print(j);

```

```

1 AudioFilterBiquad biquad10;
2 AudioFilterBiquad biquad35;
3 AudioFilterBiquad biquad8;
4 AudioFilterBiquad biquad26;
5 AudioFilterBiquad biquad14;
6 AudioFilterBiquad biquad19;
7 AudioFilterBiquad biquad15;
8 AudioFilterBiquad biquad13;
9 AudioFilterBiquad biquad12;
10 AudioFilterBiquad biquad7;
11 AudioFilterBiquad biquad24;
12 AudioFilterBiquad biquad11;
13 AudioFilterBiquad biquad5;
14 AudioFilterBiquad biquad16;
15 AudioFilterBiquad biquad6;
16 AudioFilterBiquad biquad23;
17 AudioMixer4 mixer6;
18 AudioMixer4 mixer8;
19 AudioMixer4 mixer10;
20 AudioMixer4 mixer9;
21 AudioMixer4 mixer7;
22 AudioMixer4 mixer4;
23 AudioMixer4 mixer1;
24 AudioMixer4 mixer3;
25 AudioMixer4 mixer5;
26 AudioMixer4 mixer2;
27 AudioMixer4 mixer11;
28 AudioMixer4 mixer13;
29 AudioMixer4 mixer12;
30 AudioMixer4 mixer14;
31 AudioMixer4 mixer15;
32 AudioOutputI2S i2s;
33 AudioPlaySdWav playSdWav1;
34 AudioConnection patchCord01(i2s1,
35 AudioConnection patchCord02(i2s1,
36 AudioConnection patchCord03(i2s1,
37 AudioConnection patchCord04(i2s1,
38 AudioConnection patchCord05(i2s1,
39 AudioConnection patchCord06(i2s1,
40 AudioConnection patchCord07(i2s1,
41 AudioConnection patchCord08(i2s1,
42 AudioConnection patchCord09(i2s1,
43 AudioConnection patchCord10(i2s1,
44 AudioConnection patchCord11(i2s1,
45 AudioConnection patchCord12(i2s1,
46 AudioConnection patchCord13(i2s1,
47 AudioConnection patchCord14(i2s1,
48 AudioConnection patchCord15(i2s1,
49 AudioConnection patchCord16(i2s1,
50 AudioConnection patchCord17(i2s1,
51 AudioConnection patchCord18(i2s1,
52 AudioConnection patchCord19(i2s1,
53 AudioConnection patchCord20(i2s1,
54 AudioConnection patchCord21(i2s1,
55 AudioConnection patchCord22(i2s1,
56 AudioConnection patchCord23(i2s1,
57 AudioConnection patchCord24(i2s1,
58 AudioConnection patchCord25(biqua
59 AudioConnection patchCord26(biqua
60 AudioConnection patchCord27(biqua
61 AudioConnection patchCord28(biqua
62 AudioConnection patchCord29(biqua
63 AudioConnection patchCord30(biqua
64 AudioConnection patchCord31(biqua
65 AudioConnection patchCord32(biqua
66 AudioConnection patchCord33(biqua
67 AudioConnection patchCord34(biqua

```

```

1 AudioConnection patchCord35(biqua
2 AudioConnection patchCord36(biqua
3 AudioConnection patchCord37(biqua
4 AudioConnection patchCord38(biqua
5 AudioConnection patchCord39(biqua
6 AudioConnection patchCord40(biqua
7 AudioConnection patchCord41(biqua
8 AudioConnection patchCord42(biqua
9 AudioConnection patchCord43(biqua
10 AudioConnection patchCord44(biqua
11 AudioConnection patchCord45(biqua
12 AudioConnection patchCord46(biqua
13 AudioConnection patchCord47(biqua
14 AudioConnection patchCord48(biqua
15 AudioConnection patchCord49(biqua
16 AudioConnection patchCord50(biqua
17 AudioConnection patchCord51(biqua
18 AudioConnection patchCord52(biqua
19 AudioConnection patchCord53(biqua
20 AudioConnection patchCord54(biqua
21 AudioConnection patchCord55(biqua
22 AudioConnection patchCord56(biqua
23 AudioConnection patchCord57(biqua
24 AudioConnection patchCord58(biqua
25 AudioConnection patchCord59(biqua
26 AudioConnection patchCord60(biqua
27 AudioConnection patchCord61(biqua
28 AudioConnection patchCord62(biqua
29 AudioConnection patchCord63(mixer
30 AudioConnection patchCord64(mixer
31 AudioConnection patchCord65(mixer
32 AudioConnection patchCord66(mixer
33 AudioConnection patchCord67(mixer
34 AudioConnection patchCord68(mixer
35 AudioConnection patchCord69(mixer
36 AudioConnection patchCord70(mixer
37 AudioConnection patchCord71(mixer
38 AudioConnection patchCord72(mixer
39 AudioConnection patchCord73(mixer
40 AudioConnection patchCord74(mixer
41 AudioConnection patchCord75(mixer
42 AudioConnection patchCord76(mixer
43 AudioConnection patchCord92(playS
44 AudioConnection patchCord93(playS
45 AudioConnection patchCord77(mixer
46 AudioConnection patchCord78(mixer
47 AudioControlSGTL5000 sgtl5000_1;
48 // GUItool: end automatically generated co
49
50 const int myInput = AUDIO_INPUT_LINEIN;
51
52 void setup() {
53 Serial.begin(9600);
54 pinMode(yesButton, INPUT);
55 pinMode(noButton, INPUT);
56 pinMode(ledPin, OUTPUT);
57
58 // Audio connections require memory to w
59 // detailed information, see the MemoryA
60 sgtl5000_1.enable();
61
62 sgtl5000_1.volume(0.5);
63 AudioMemory(150);
64 SPI.setMOSI(7);
65 SPI.setSCK(14);
66 if (!(SD.begin(10))) {
67

```

```

1 while (1) {
2 Serial.println("Unable to access the
3 delay(500);
4 }
5
6 }
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```


11-26V2

untitled

untitled

untitled

```

484 mixer7.gain(1,mixGain(result250L, result250R));
485 mixer7.gain(2,mixGain(result500L, result500R));
486 mixer7.gain(3,mixGain(result1000L, result1000R));
487
488 mixer8.gain(0,mixGain(result500L, result500R));
489 mixer8.gain(1,mixGain(result1000L, result1000R));
490 mixer8.gain(2,mixGain(result2000L, result2000R));
491 mixer8.gain(3,mixGain(result4000L, result4000R));
492
493 mixer9.gain(0,mixGain(result2000L, result2000R));
494 mixer9.gain(1,mixGain(result4000L, result4000R));
495 mixer9.gain(2,mixGain(result8000L, result8000R));
496 mixer9.gain(3,mixGain(result16000L, result16000R));
497
498 mixer10.gain(0,mixGain(result4000L, result4000R));
499 mixer10.gain(1,mixGain(result8000L, result8000R));
500 mixer10.gain(2,mixGain(result16000L, result16000R));
501 mixer10.gain(3,mixGain(result32000L, result32000R));
502
503
504 double mixGain(int dB1, int dB2, double B
505 double mix = ((point*(dB2-dB1))/BW)+dB1
506 mix=pow(10, mix/20);
507 mix = (mix-0.1276)/1.7754;
508 Serial.println(mix);
509 return mix;
510 }
511
512 void calibration(){
513 int yesButtonState=digitalRead(yesButton);
514 int noButtonState=digitalRead(noButton);
515
516 while(yesButtonState==1 && noButtonState==0){
517 yesButtonState=digitalRead(yesButton);
518 noButtonState=digitalRead(noButton);
519
520 Serial.print("Calibrating...");
521 Serial.println("");
522
523 sgtl5000_1.volume(.9);
524 playFile("1234.WAV");
525 delay(1000);
526
527 Serial.println("");
528 Serial.print("Calibration complete.");
529 digitalWrite(ledPin, HIGH);
530 }
531
532 void playFile(char* filename)
533 {
534 Serial.print("Playing file: ");
535 Serial.println(filename);
536
537 // Start playing the file. This sketch
538 // run while the file plays.
539 playSdWav1.play(filename);
540
541 // A brief delay for the library read
542 delay(5);
543
544 // Simply wait for the file to finish p
545 while (playSdWav1.isPlaying()) {
546 float vol = analogRead(1);
547 vol = vol / 1024;
548 sgtl5000_1.volume(vol);
549 }
550 }
551

```

```

1 Serial.print("\n");
2
3 //Results being recorded//
4 if(yesButtonState==0 && noButtonState==1){
5 digitalWrite(ledPin, HIGH);
6 delay(1000);
7 //Results being recorded//
8 if(i<=6){
9 originalResultsLeft[numberOfTests+i]=result125L;
10 originalResultsLeft[numberOfTests+i]=result250L;
11 originalResultsLeft[numberOfTests+i]=result500L;
12 originalResultsLeft[numberOfTests+i]=result1000L;
13 originalResultsLeft[numberOfTests+i]=result2000L;
14 originalResultsLeft[numberOfTests+i]=result4000L;
15 originalResultsLeft[numberOfTests+i]=result8000L;
16 originalResultsLeft[numberOfTests+i]=result16000L;
17 }
18
19 if(yesButtonState==1 && noButtonState==0){
20 digitalWrite(ledPin, HIGH);
21 delay(1000);
22 //Results being recorded//
23 if(i<=6){
24 originalResultsRight[numberOfTests+i]=result125R;
25 originalResultsRight[numberOfTests+i]=result250R;
26 originalResultsRight[numberOfTests+i]=result500R;
27 originalResultsRight[numberOfTests+i]=result1000R;
28 originalResultsRight[numberOfTests+i]=result2000R;
29 originalResultsRight[numberOfTests+i]=result4000R;
30 originalResultsRight[numberOfTests+i]=result8000R;
31 originalResultsRight[numberOfTests+i]=result16000R;
32 }
33
34 if(j==0){
35 digitalWrite(ledPin, HIGH);
36 //Results being recorded//
37 if(i<=5){
38 originalResultsLeft[numberOfTests+i]=result125L;
39 originalResultsLeft[numberOfTests+i]=result250L;
40 originalResultsLeft[numberOfTests+i]=result500L;
41 originalResultsLeft[numberOfTests+i]=result1000L;
42 originalResultsLeft[numberOfTests+i]=result2000L;
43 originalResultsLeft[numberOfTests+i]=result4000L;
44 originalResultsLeft[numberOfTests+i]=result8000L;
45 originalResultsLeft[numberOfTests+i]=result16000L;
46 }
47
48 if(j==0){
49 digitalWrite(ledPin, HIGH);
50 //Results being recorded//
51 if(i<=5){
52 originalResultsRight[numberOfTests+i]=result125R;
53 originalResultsRight[numberOfTests+i]=result250R;
54 originalResultsRight[numberOfTests+i]=result500R;
55 originalResultsRight[numberOfTests+i]=result1000R;
56 originalResultsRight[numberOfTests+i]=result2000R;
57 originalResultsRight[numberOfTests+i]=result4000R;
58 originalResultsRight[numberOfTests+i]=result8000R;
59 originalResultsRight[numberOfTests+i]=result16000R;
60 }
61
62 //leftResultsFinal[i]=originalResultsLeft[i];
63 //rightResultsFinal[i]=originalResultsRight[i];
64
65 //leftResultsFinal[i]=originalResultsLeft[i];
66 //rightResultsFinal[i]=originalResultsRight[i];
67

```

```

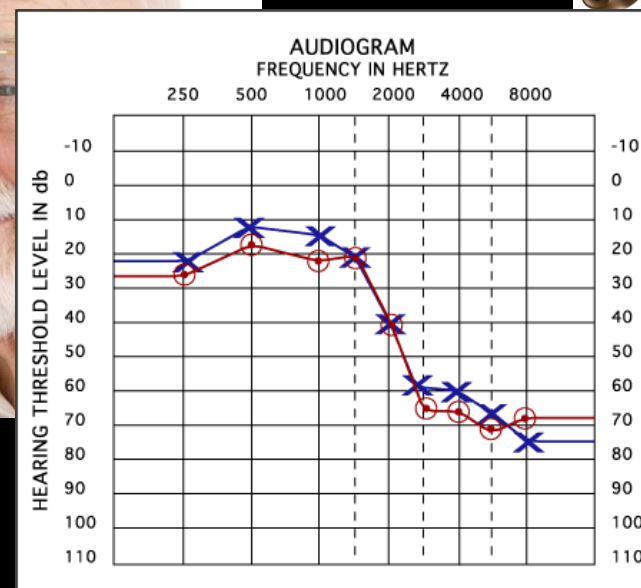
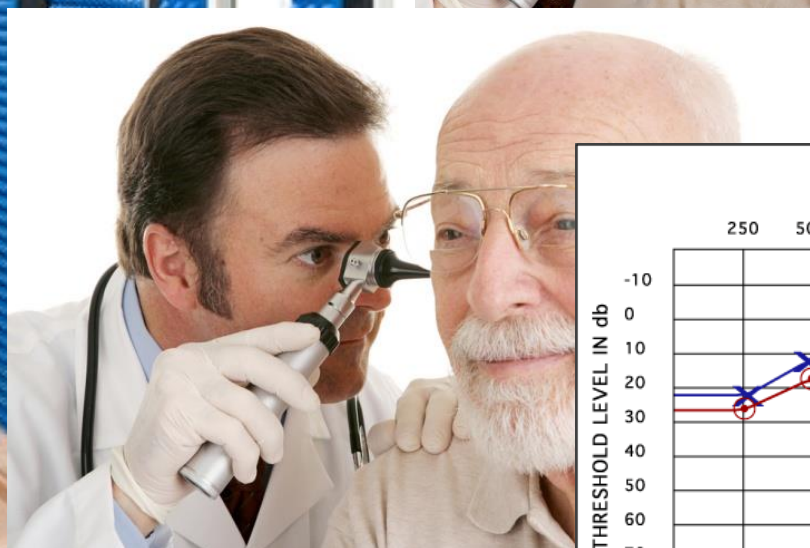
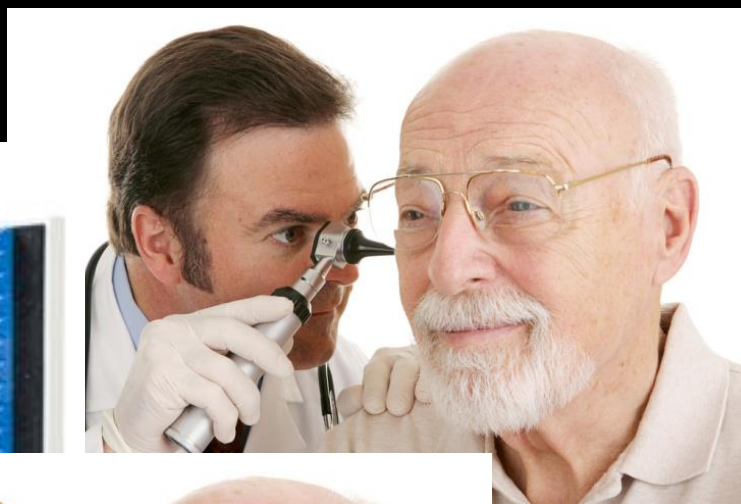
1 //Outputs left side results to the user
2 Serial.print("Left side results: ");
3 for (int i = 0; i < 7; ++i){
4 Serial.print(leftResultsFinal[i]);
5 Serial.print(", ");
6 }
7
8 //Stores the values of the results to pass to the next stage
9 result125L = leftResultsFinal[0];
10 result250L = leftResultsFinal[1];
11 result500L = leftResultsFinal[2];
12 result1000L = leftResultsFinal[3];
13 result2000L = leftResultsFinal[4];
14 result4000L = leftResultsFinal[5];
15 result8000L = leftResultsFinal[6];
16 Serial.println("");
17
18 //Outputs right side results to the user
19 Serial.print("Right side results: ");
20 for (int i = 0; i < 7; ++i){
21 Serial.print(rightResultsFinal[i]);
22 Serial.print(", ");
23 }
24
25 //Stores the values of the results to pass to the next stage
26 result125R = rightResultsFinal[0];
27 result250R = rightResultsFinal[1];
28 result500R = rightResultsFinal[2];
29 result1000R = rightResultsFinal[3];
30 result2000R = rightResultsFinal[4];
31 result4000R = rightResultsFinal[5];
32 result8000R = rightResultsFinal[6];
33 Serial.println("");
34
35 Serial.print("TEST COMPLETE");
36 Serial.println("");
37
38 //format
39 //biquad.setBandpass(stage, center frequency, Q, gain);
40 biquad1.setBandpass(0, 125, 125/28.9);
41 biquad2.setBandpass(0, 157.5, 157.5/36.5);
42 biquad3.setBandpass(0, 198.4, 198.4/45.9);
43 biquad4.setBandpass(0, 250, 250/57.9);
44
45 biquad5.setBandpass(0, 315, 315/73);
46 biquad6.setBandpass(0, 396.9, 396.9/88.8);
47 biquad7.setBandpass(0, 500, 500/115.8);
48 biquad8.setBandpass(0, 630, 630/145.9);
49
50 biquad9.setBandpass(0, 793.7, 793.7/183);
51 biquad10.setBandpass(0, 1000, 1000/231.6);
52 biquad11.setBandpass(0, 1259.9, 1259.9/2);
53 biquad12.setBandpass(0, 1587.4, 1587.4/3);
54
55 biquad13.setBandpass(0, 2000, 2000/463.1);
56 biquad14.setBandpass(0, 2519.8, 2519.8/5);
57 biquad15.setBandpass(0, 3174.8, 3174.8/7);
58 biquad16.setBandpass(0, 4000, 4000/926.2);
59
60 biquad17.setBandpass(0, 5039.7, 5039.7/1);
61 biquad18.setBandpass(0, 6349.6, 6349.6/1);
62 biquad19.setBandpass(0, 8000, 8000/1852);
63
64 //format
65 biquad20.setBandpass(0, 125, 125/28.9);
66

```

```

1 biquad21.setBandpass(0, 157.5, 157.5/36.5);
2 biquad22.setBandpass(0, 198.4, 198.4/45.9);
3 biquad23.setBandpass(0, 250, 250/57.9);
4 biquad24.setBandpass(0, 315, 315/73);
5
6 biquad25.setBandpass(0, 396.9, 396.9/88.8);
7 biquad26.setBandpass(0, 500, 500/115.8);
8 biquad27.setBandpass(0, 630, 630/145.9);
9 biquad28.setBandpass(0, 793.7, 793.7/183);
10
11 biquad29.setBandpass(0, 1000, 1000/231.6);
12 biquad30.setBandpass(0, 1259.9, 1259.9/2);
13 biquad31.setBandpass(0, 1587.4, 1587.4/3);
14 biquad32.setBandpass(0, 2000, 2000/463.1);
15
16 biquad33.setBandpass(0, 2519.8, 2519.8/5);
17 biquad34.setBandpass(0, 3174.8, 3174.8/7);
18 biquad35.setBandpass(0, 4000, 4000/926.2);
19 biquad36.setBandpass(0, 5039.7, 5039.7/1);
20
21 biquad37.setBandpass(0, 6349.6, 6349.6/1);
22 biquad38.setBandpass(0, 8000, 8000/1852);
23
24 }
25
26 unsigned long last_time = millis();
27
28 void loop(){
29 sgtl5000_1.inputSelect(myInput);
30 if (millis() - last_time >= 2500) {
31 Serial.print("Proc = ");
32 Serial.print(AudioProcessorUsage());
33 Serial.print(" (");
34 Serial.print(AudioProcessorUsageMax());
35 Serial.print(" Mem = ");
36 Serial.print(AudioMemoryUsage());
37 Serial.print(" (");
38 Serial.print(AudioMemoryUsageMax());
39 Serial.println(")");
40 last_time = millis();
41 }
42
43 //mixGain(int dB1, int dB2, double BW, double gain)
44 mixer1.gain(0,mixGain(0, result125L, result125R));
45 mixer1.gain(1,mixGain(result125L, result125R));
46 mixer1.gain(2,mixGain(result125L, result125R));
47 mixer1.gain(3,mixGain(result125L, result125R));
48
49 mixer2.gain(0,mixGain(result250L, result250R));
50 mixer2.gain(1,mixGain(result250L, result250R));
51 mixer2.gain(2,mixGain(result250L, result250R));
52 mixer2.gain(3,mixGain(result250L, result250R));
53
54 mixer3.gain(0,mixGain(result500L, result500R));
55 mixer3.gain(1,mixGain(result500L, result500R));
56 mixer3.gain(2,mixGain(result500L, result500R));
57 mixer3.gain(3,mixGain(result500L, result500R));
58
59 mixer4.gain(0,mixGain(result1000L, result1000R));
60 mixer4.gain(1,mixGain(result1000L, result1000R));
61 mixer4.gain(2,mixGain(result1000L, result1000R));
62 mixer4.gain(3,mixGain(result1000L, result1000R));
63
64 mixer5.gain(0,mixGain(result2000L, result2000R));
65 mixer5.gain(1,mixGain(result2000L, result2000R));
66 mixer5.gain(2,mixGain(result2000L, result2000R));
67 mixer5.gain(3,mixGain(result2000L, result2000R));
68

```





low
cost

integrated
device

open
source

So what?

YOU CAN DO
ANYTHING WITH THE
INTERNET



Learn how to program





MAKE SOMETHING



Amplify life.