

コードとデザイン

東京藝術大学 芸術情報センター開設科目 金曜4-5限 第9週

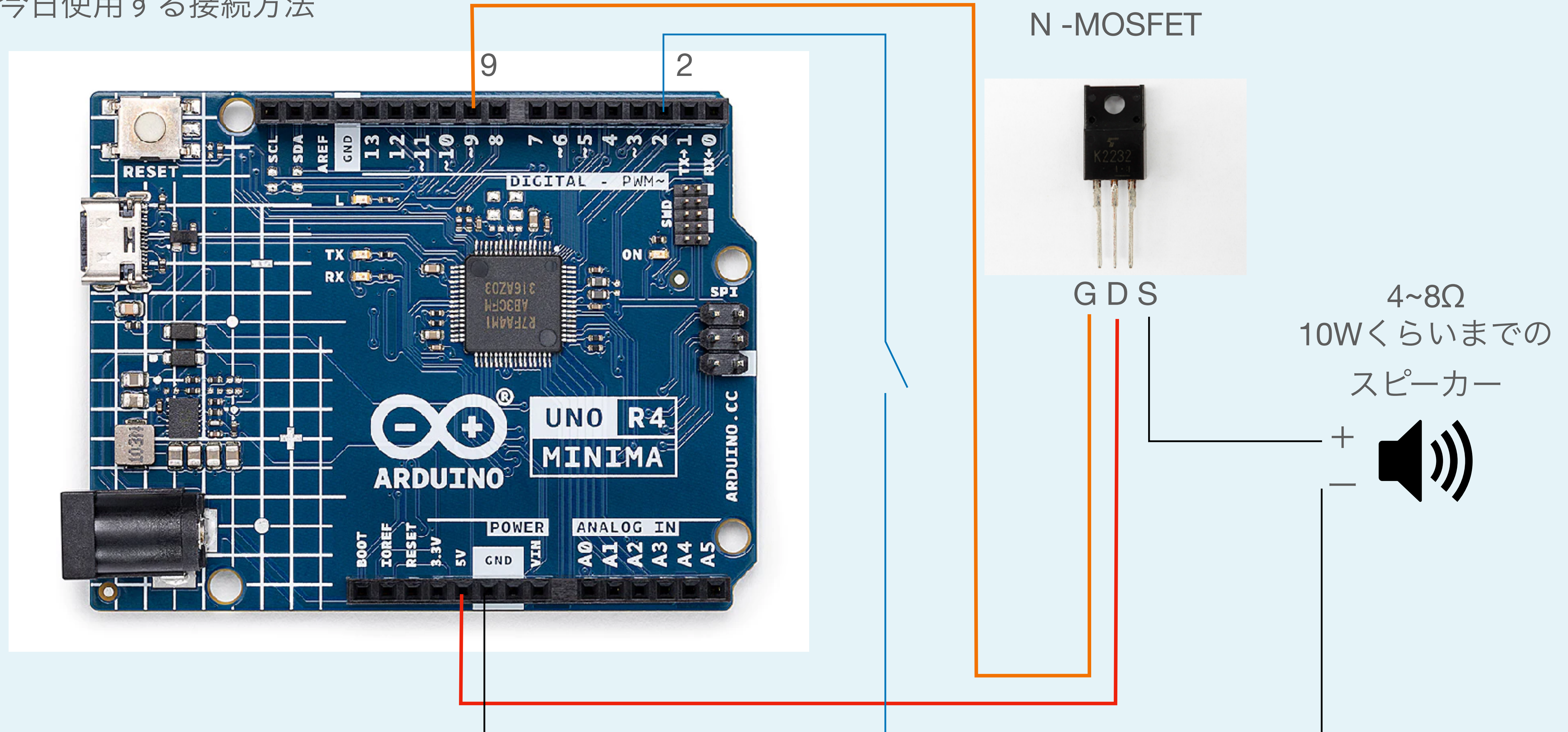
2025.06.06 松浦知也 (matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com)



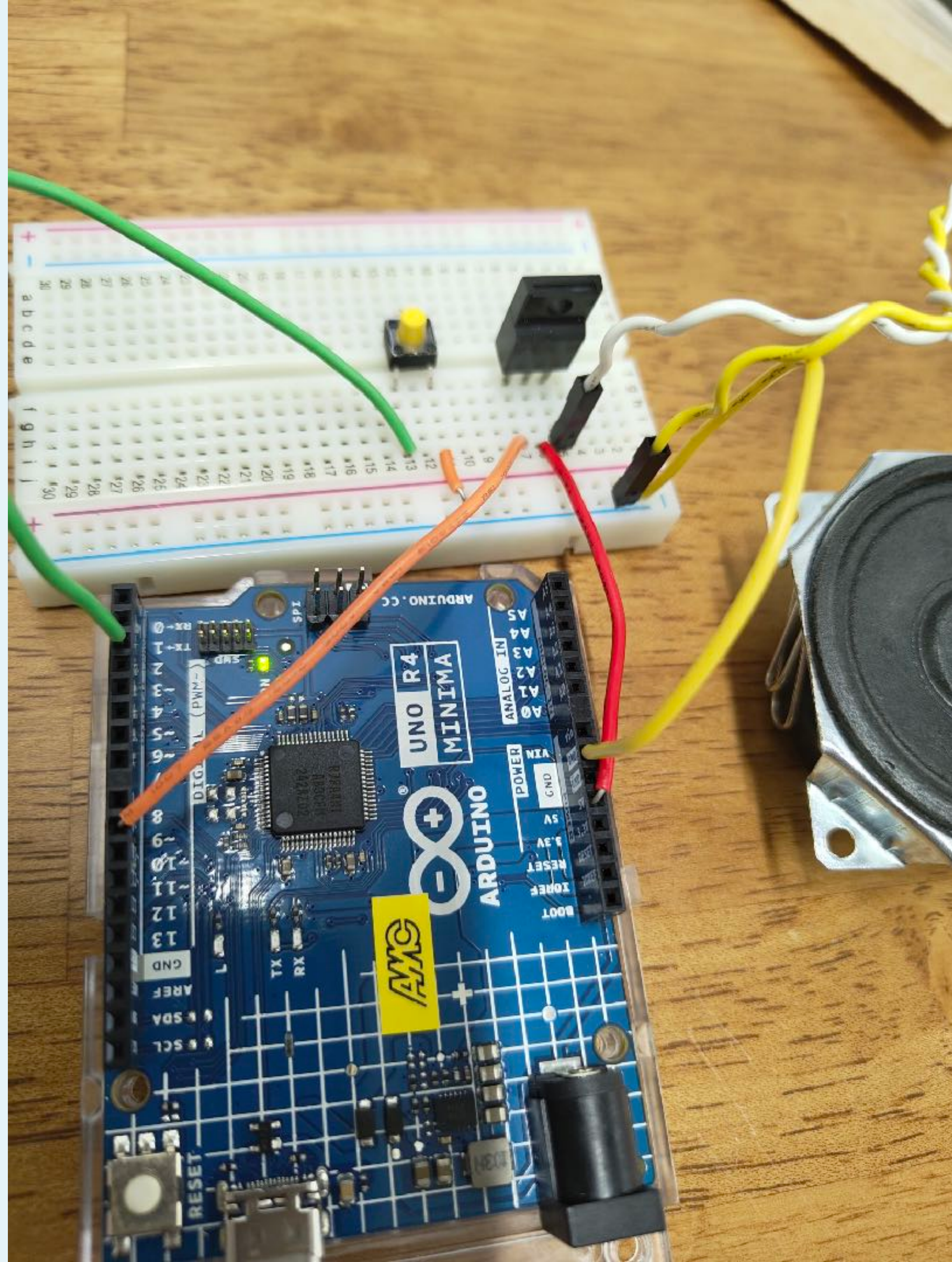
Arduino Uno R4で音を出す

1. 矩形波(1ビット)の波だけで
いい場合：PWMとtone()

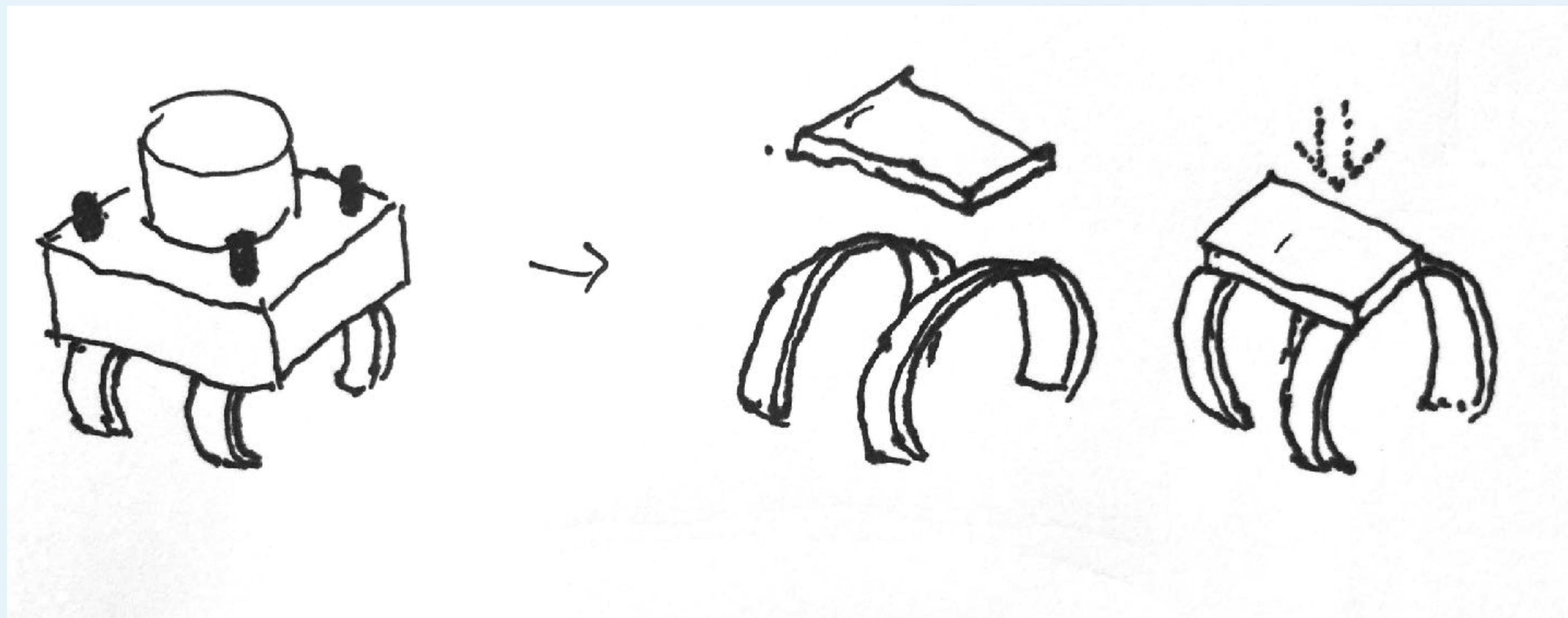
今日使用する接続方法



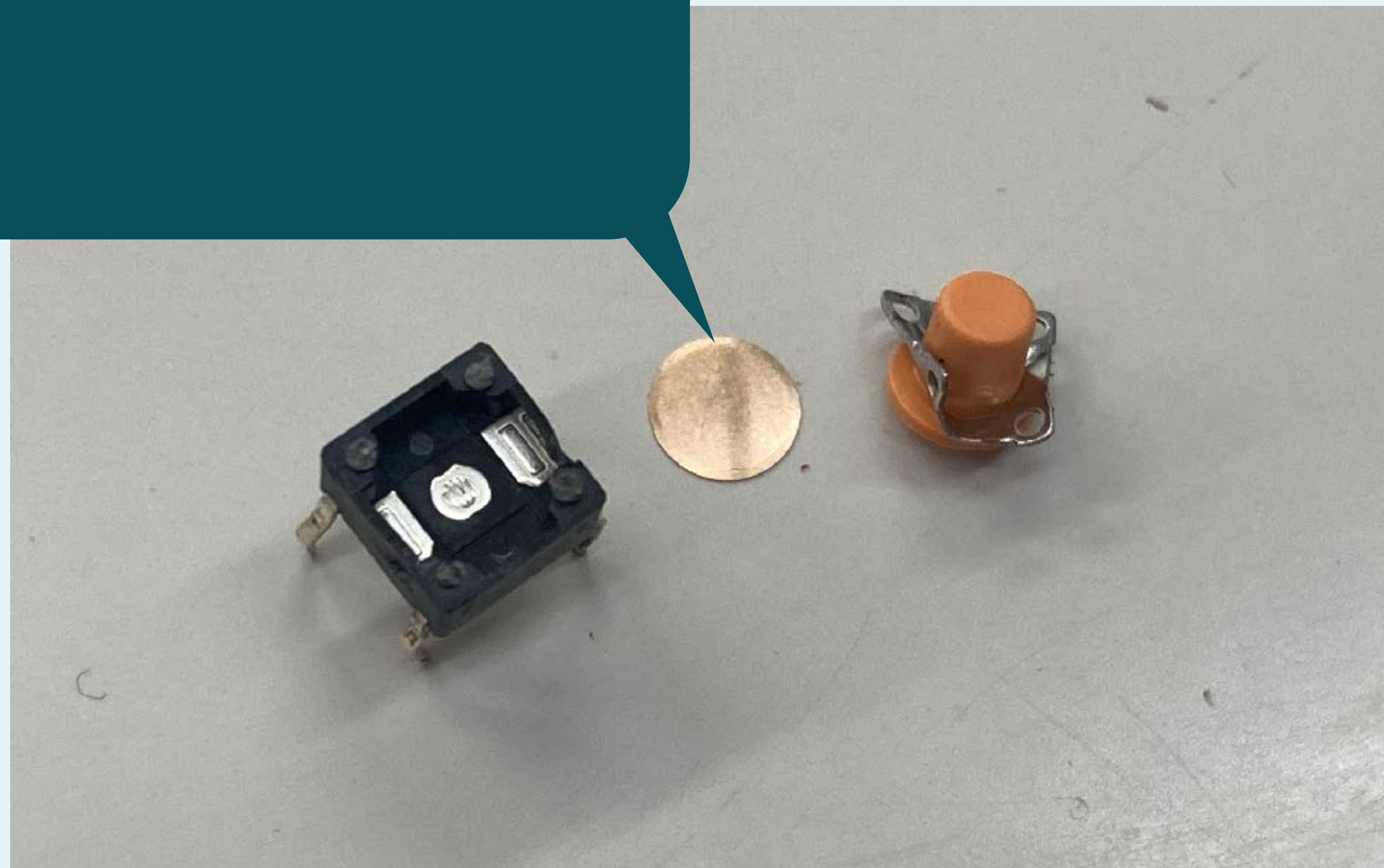
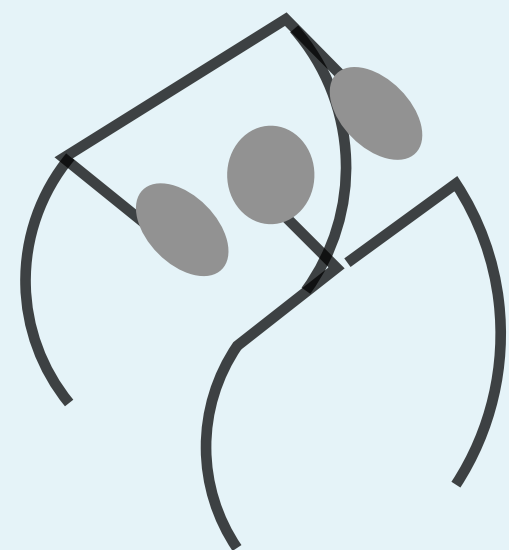
※簡易的な接続のため、直流成分が常にGNDに流れっぱなしになる。
FETが結構発熱するので注意。でかいスピーカーではやらない方がいい



タクトスイッチ



若干膨らんでいて、押すとペコッと凹む



```
const int button_pin = 2;
const int sound_pin = 9;

void setup() {
    pinMode(button_pin, INPUT_PULLUP);

    pinMode(sound_pin, OUTPUT);
}

void loop() {
    if (digitalRead(2) == LOW) {
        tone(sound_pin, 1000);
    } else {
        noTone(sound_pin);
    }
    delay(20);
}
```

tone_minimal_R4.ino

トップ

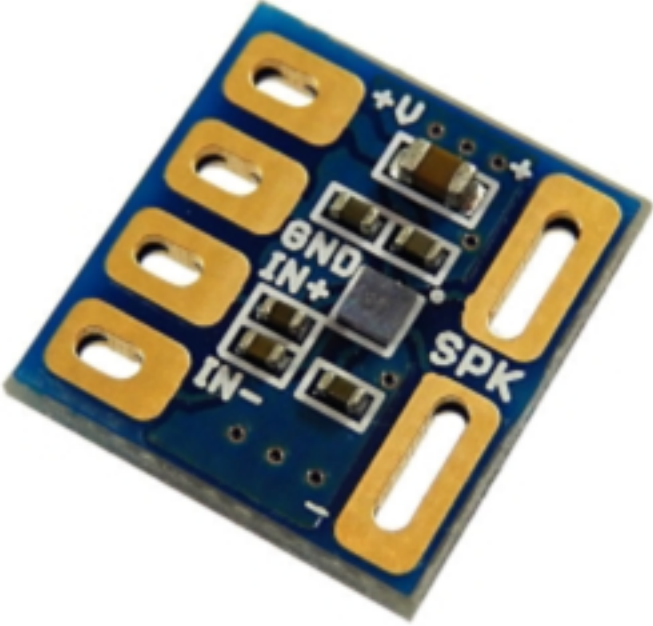
組立キット(モジュール)

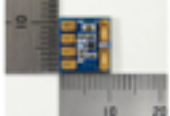
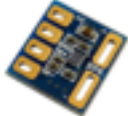
オーディオ関連

D級7Wキット

PAM8012使用2ワットD級アンプモジュール

AAA





この商品を友達に教える

お気に入りに追加する

店舗在庫情報

PAM8012使用2ワットD級アンプモジュール

[AE-PAM8012]

通販コード K-08217

発売日 2014/07/11

メーカーカテゴリ [株式会社秋月電子通商](#)

超小型ながら最大2ワットの出力が可能なD級アンプモジュールです。ピン取り付け穴が長穴加工となっていますので、電線のハンダ付けがし易くなっています。差動入力型ですが、IN-をGNDに接続することでシングルエンド入力としても使用できます。

■特長

・電源電圧:2.5V~5.5V

・出力:
8Ω負荷時 最大1.0W
4Ω負荷時 最大2.0W
(@5V、THD1%未満)

・低ノイズ、高電源リプル除去率(PSRR)

・自動復帰短絡保護、過熱保護機能を内蔵しています。

・超小型モジュールサイズ:10.5x11mm

PAMシリーズ アンプキット ⇒ [K-15698](#)

PAMシリーズ アンプ単品 ⇒ [I-14187](#) [I-13716](#) [I-13715](#) [I-09160](#)

 [取扱説明書](#)

 [PAM8012 PDFデータシート](#)

[オーディオアンプキット一覧](#)

[ダイナミックスピーカー一覧](#)

[小型ボリュウム\(VR\)一覧](#)

[ピッチ変換\(DIP化\)基板一覧](#)

● この商品の [よくある質問\(Q&A\)](#) が1件あります。商品選定・製作の参考にしてください。

ちゃんとした接続の場合はオーディオアンプICを使うのがおすすめ。1個300円ぐらい

<https://akizukidenshi.com/catalog/g/gK-08217/>

関連アイテム

📍 愛知県, 日本

Choice AliExpressの保証

 送料無料

配達予定日: **6月08日-13日**にお届け
 運送会社: ヤマト運輸 SAGAWA佐川急便 など

 配送保証

- ✓ 150円配達遅延クーポン
- ✓ 紛失補償
- ✓ 破損補償
- ✓ 30日までに届かない場合は返金

 90日以内返品無料

 安全&個人情報保護

- ✓ 安全なお支払い: お客様の同意なしに...
- ✓ 個人情報の保護: お客様の個人情報を...

④ AliExpressのサステナビリティ

今すぐ購入

カートの追加

まとめ買いすると1個30円くらいのもある（ただし粗悪品もあるので自己責任）

<https://ja.aliexpress.com/item/1005007279115127.html>

音名と周波数

- 周波数のままでは音楽用に使いづらい
- 中心のラ（A4）=440Hz が一般的なチューニング
- 1オクターブ=周波数が2倍(A5=880Hz、A6=1760Hz)
- 半音上がるのを12回繰り返すと周波数が2倍になる(=半音で $2^{1/12}$ ずつ上昇)
- A4を69番として、音名に番号を振る（MIDIノート番号）

$$Frequency_{(Hz)} = 440 \cdot 2^{(midi-69/12)}$$

```

const int button_pin = 2;
const int sound_pin = 9;

bool is_playing = false;
int button_prev= HIGH;
float freq = 440;

float midiToFreq(int midi){
    return 440.*pow(2.,(midi-69.)/12.);
}

void setup() {
    pinMode(button_pin,INPUT_PULLUP);
    pinMode(sound_pin, OUTPUT);
}

void loop() {
    auto button_state = digitalRead(button_pin);

    if(button_prev == HIGH && button_state==LOW){
        is_playing = !is_playing;
        freq = midiToFreq(random(65,100));
    }
    if(is_playing){
        tone(sound_pin,freq);
    }else{
        noTone(sound_pin);
    }

    delay(20);
    button_prev = button_state;
}

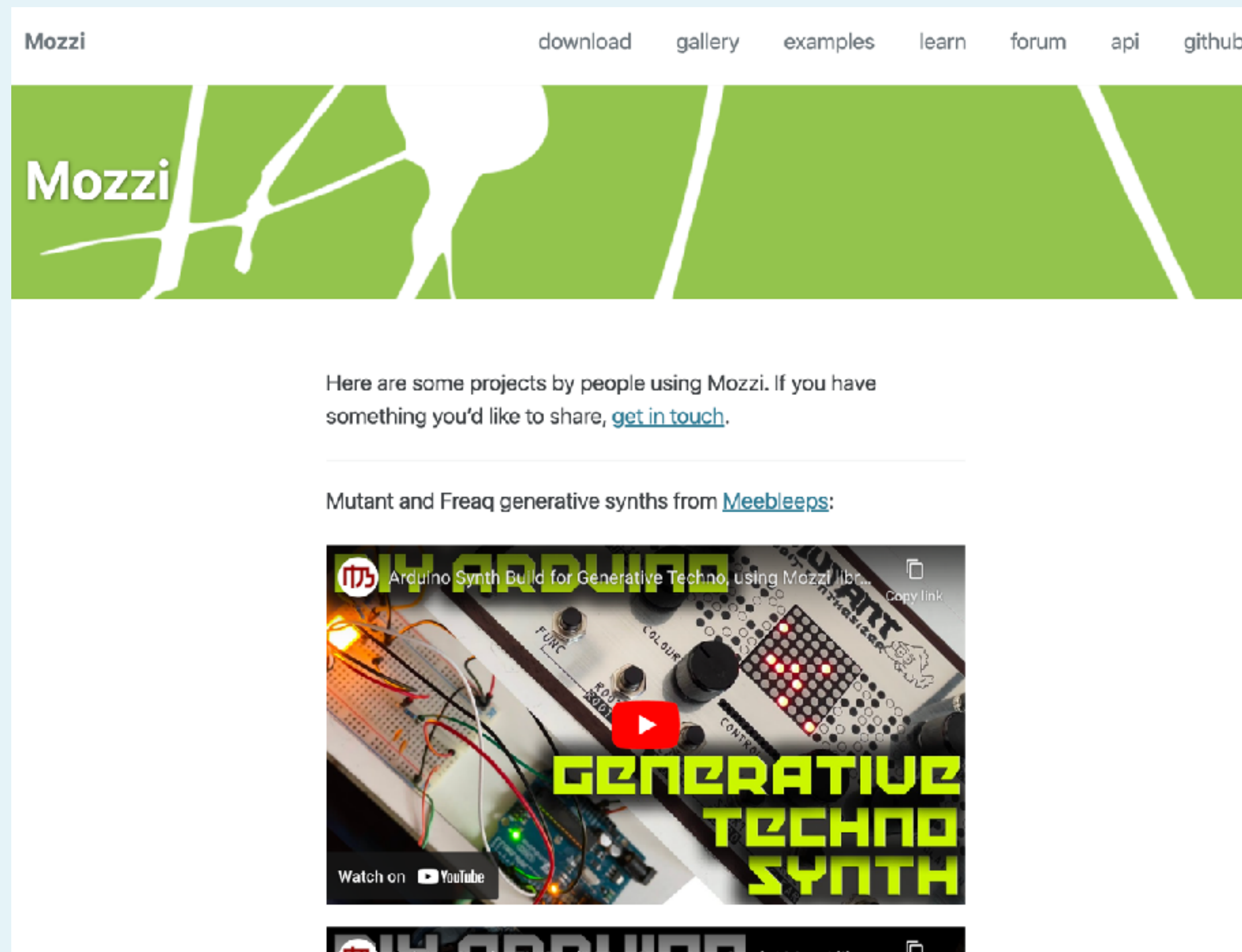
```

$$Frequency_{(Hz)} = 440 \cdot 2^{(midi-69/12)}$$

tone_random_flip_R4.ino

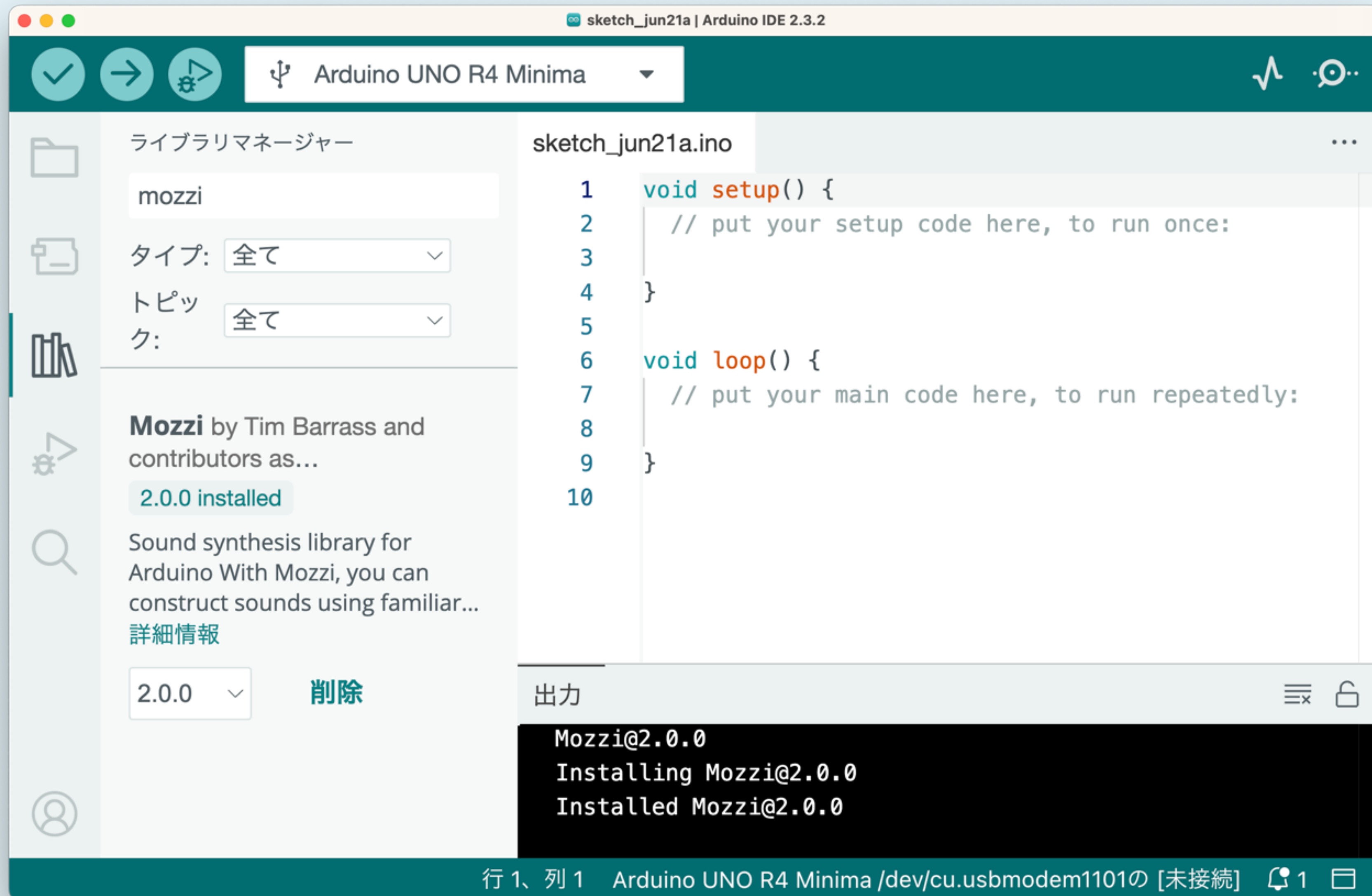
Mozziで高度な音作り

Mozzi

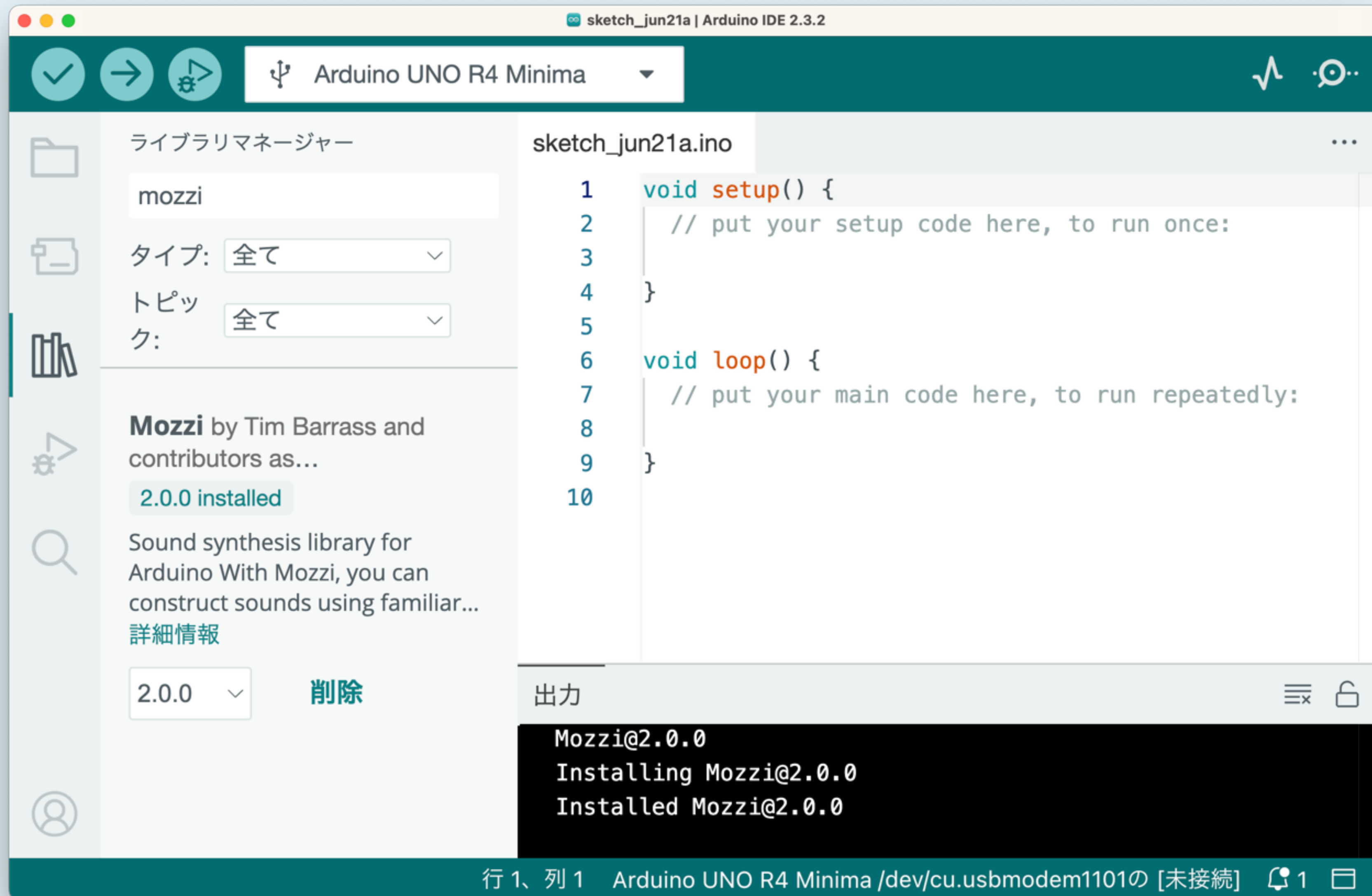


<https://sensorium.github.io/Mozzi/>

Arduinoで最もポピュラーな音声合成ライブラリ



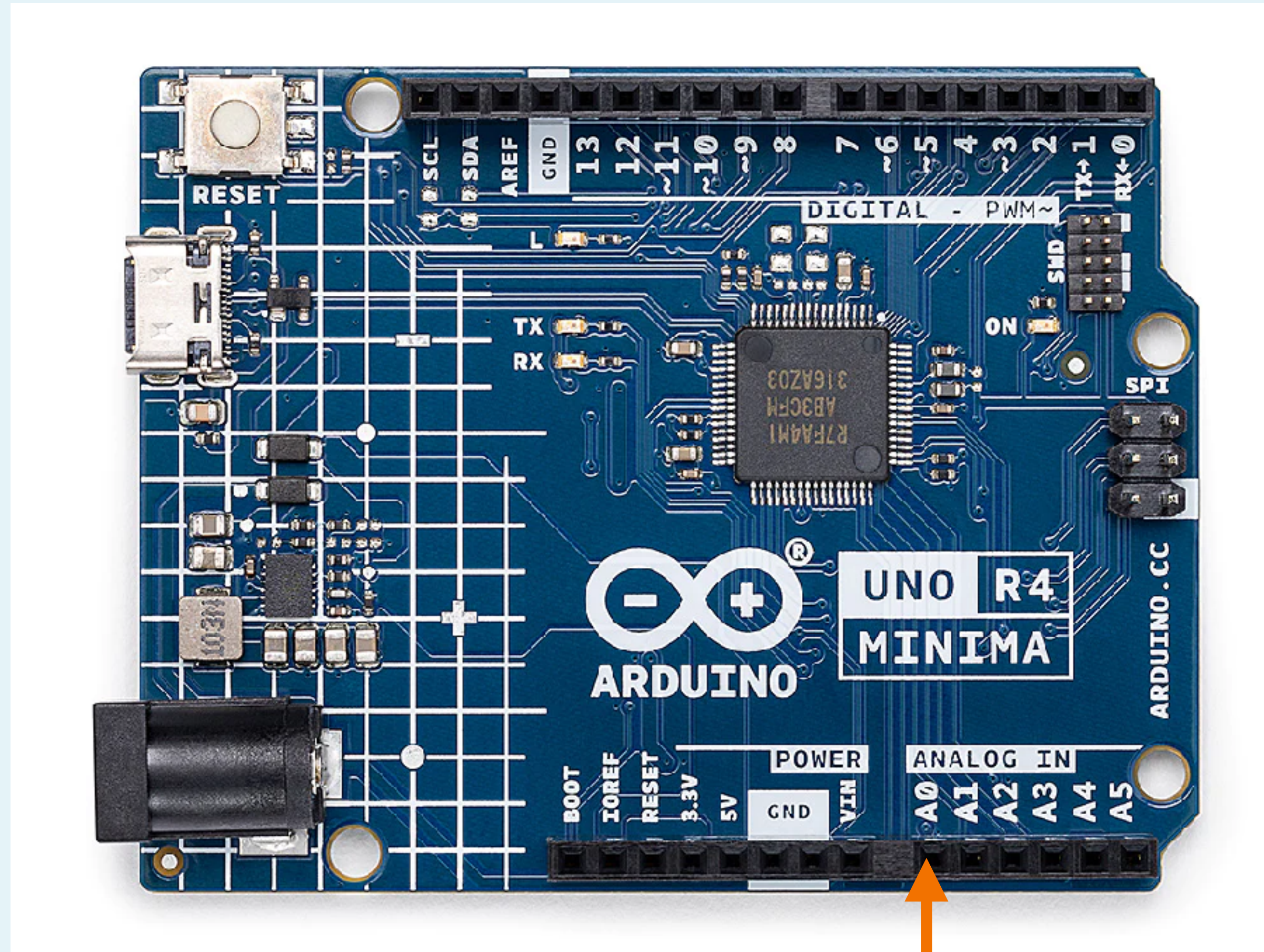
ライブラリマネージャーでMozziを検索→インストール



※今回は昨年リリースされたv2.0.2を使用します。

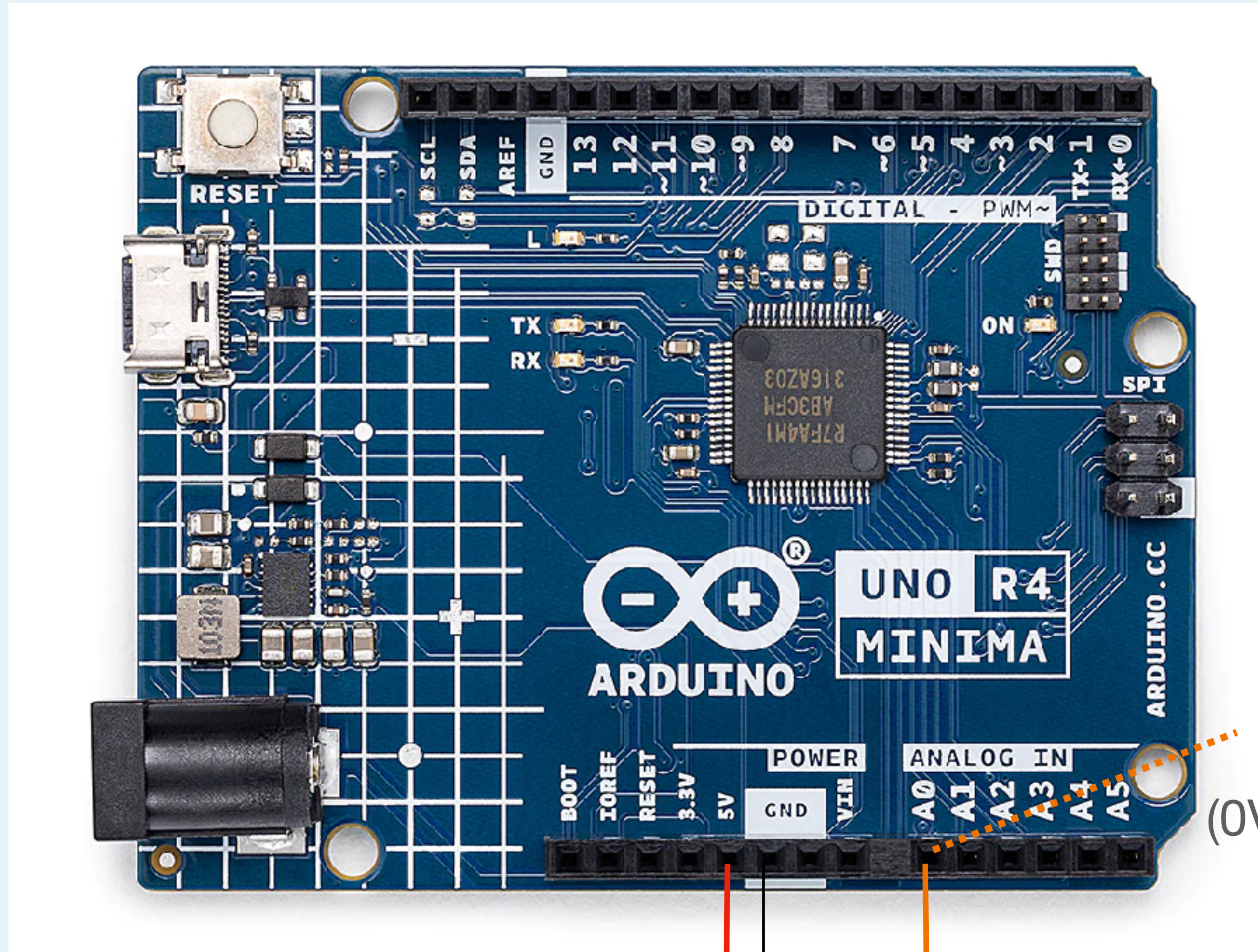
ネットに落ちている資料はまだほぼv1系ばかりで、書き方が変わっている部分もあるので要注意。

サンプルスケッチを参考にするのが安牌

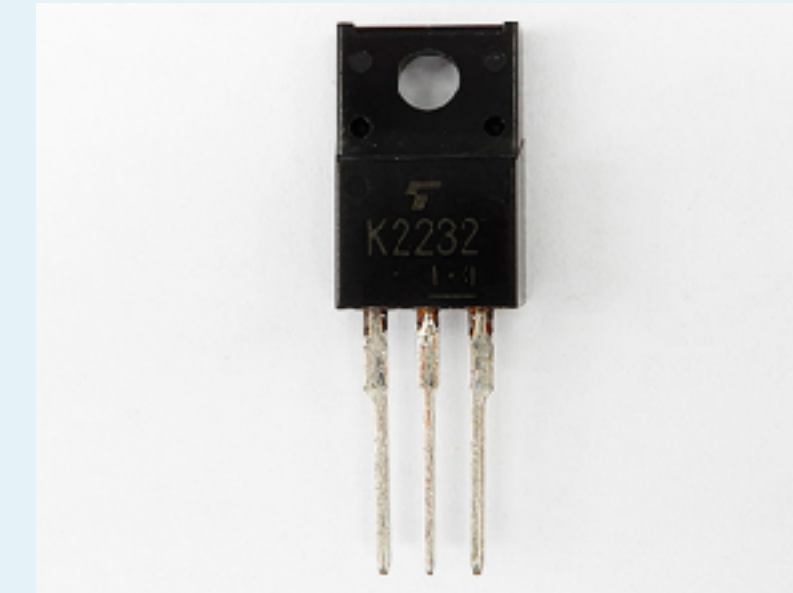


Arduino Uno R4はA0のピンにDAC(Digital-to-Analog Converter)がついている

今日使用する接続方法

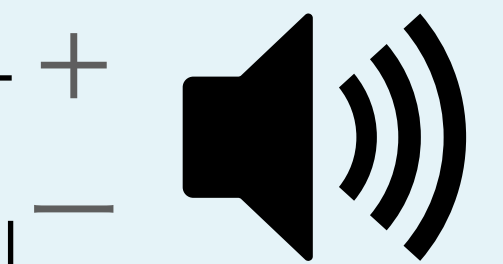


N-MOSFET



G D S

4~8Ω
10Wくらいまでの
スピーカー



1~5Vの範囲で
音声波形を送る
(0V近辺は波形が歪む)

※簡易的な接続のため、直流成分が常にGNDに流れっぱなしになる。
FETが結構発熱するので注意。でかいスピーカーではやらない方がいい

Pulse Code Modulation(PCM)



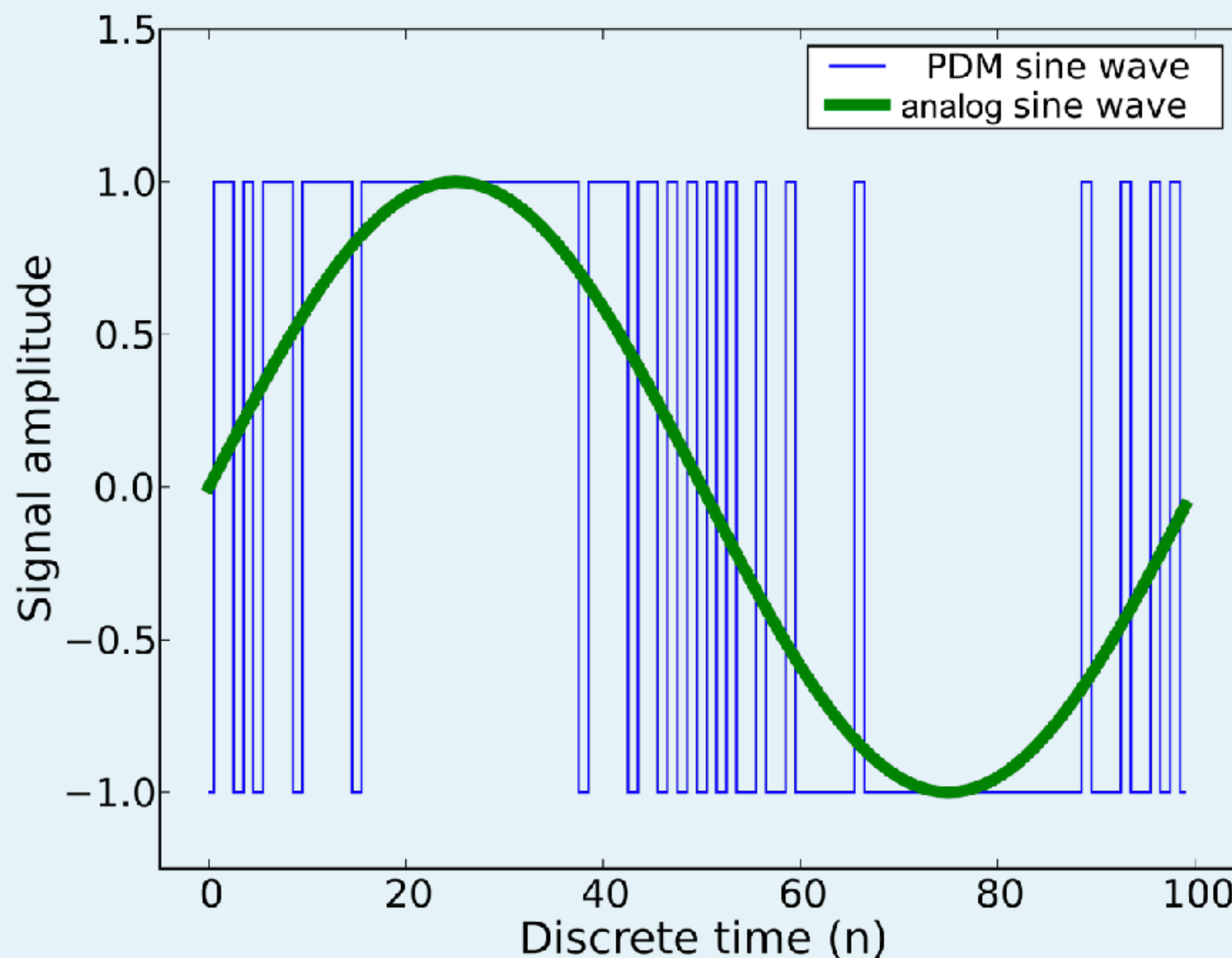
1サンプルごとのビット
(この例では4ビット) を
アナログ電圧に直す変換器
(DAコンバーター)が必要

R4ではA0のピンに12bitの
DACが内蔵

Pulse Density Modulation

縦軸が粗い代わりに横軸を細かくする

Uno R3などでのMozziはPWMピンを使用して擬似的なアナログ波形を実現している



ノイズは含まれているが、
高周波に偏っている。
コンデンサーで高周波を
取り除くだけで
そのまま使える。
(ノイズ成分が20kHz以上
で聞こえなければ
それすら不要)

Mozzi x Uno R4使用の注意点

- スピーカーへ出力するピンはA0で固定（内蔵DACを使用するため）
 - PWMを使った出力には現在非対応（なので、モノラル限定）
- センサーを使う時は、`analogRead()`の代わりに`mozziAnalogRead()`を使う必要がある
 - なので、内部的に`analogRead()`を呼ぶライブラリとの組み合わせも難しい
- `delay()`を使うコードはEventDelayクラスを使う必要がある

Mozziの使い方

- `setup()`と`loop()`に加えて、`void updateControl()`と`int updateAudio()`という関数を作る
- `loop()`の中では`audioHook()`という関数だけと呼ぶ
- 普通`loop()`に書くような処理（センサーの読み取りなど）は`updateControl()`に
- `updateAudio()`でreturnする値が1サンプルごとの電圧（音圧）になる
 - `MonoOutput::from8Bit(val)`なら0~255、`MonoOutput::from16Bit(val)`なら0~32767といった範囲

```

#define MOZZI_CONTROL_RATE 64
#include <Mozzi.h>
#include <Oscil.h> // oscillator template
#include <tables/sin2048_int8.h> // sine table for oscillator

Oscil <SIN2048_NUM_CELLS, MOZZI_AUDIO_RATE> aSin(SIN2048_DATA);

void setup(){
    startMozzi();
    aSin.setFreq(440);
}
void updateControl(){

}

AudioOutput updateAudio(){
    int16_t output = map((int16_t)aSin.next(),0,255,2000,INT16_MAX);
    return MonoOutput::from16Bit(output);
}

void loop(){
    audioHook();
}

```

SineWave_R4.ino

```

#define MOZZI_CONTROL_RATE 64
#include <Mozzi.h>
#include <Oscil.h> // oscillator template
#include <tables/sin2048_int8.h> // sine table for oscillator

Oscil <SIN2048_NUM_CELLS, MOZZI_AUDIO_RATE> aSin(SIN2048_DATA);

void setup(){
    startMozzi();
    aSin.setFreq(440);
}
void updateControl(){

}

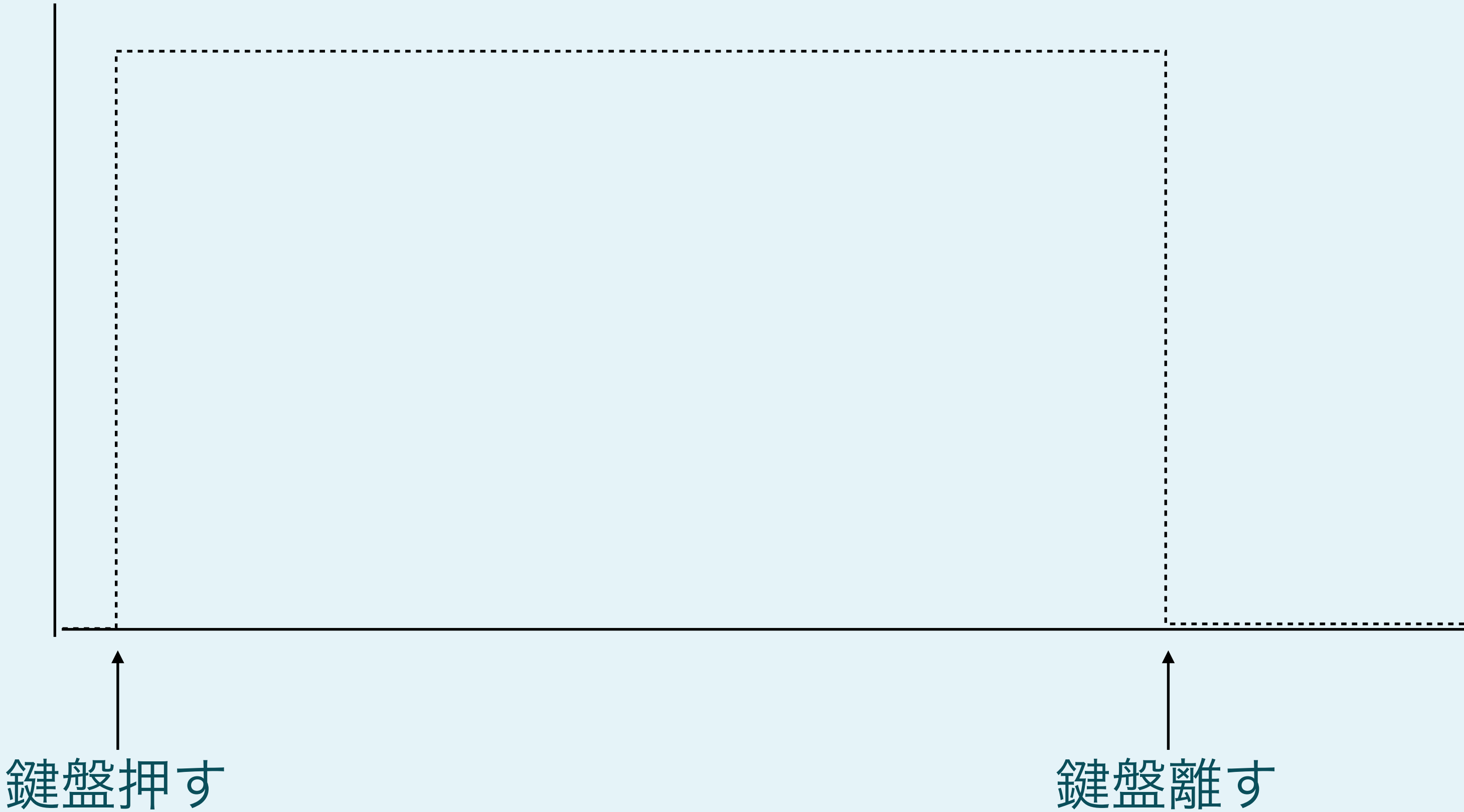
AudioOutput updateAudio(){
    int16_t output = map((int16_t)aSin.next(), 0, 255, 2000, 32767);
    return MonoOutput::from16Bit(output);
}

void loop(){
    audioHook();
}

```

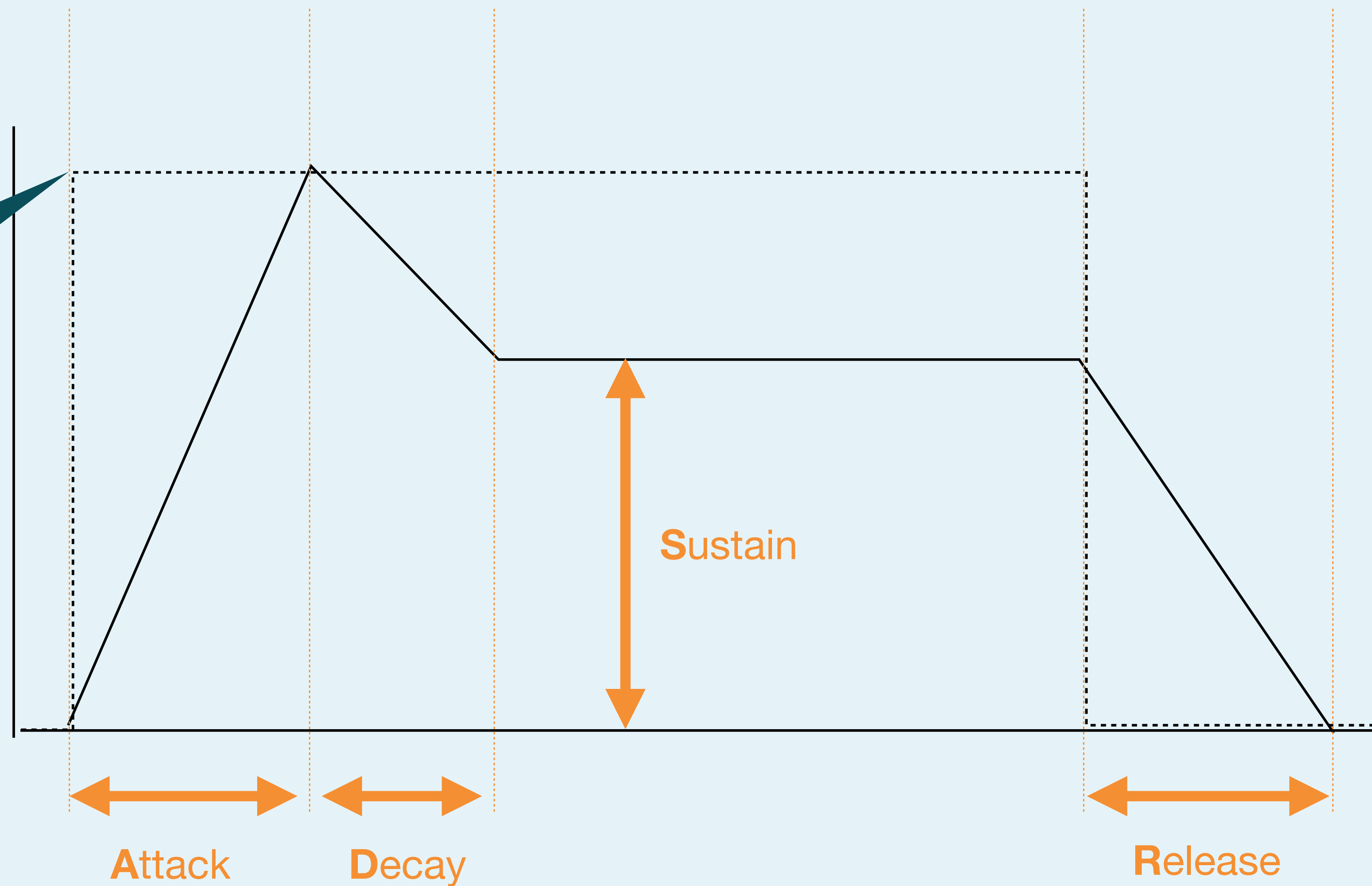
aSinの波形出力0~255を、
2000~32767にマッピング
(0~2000近辺は歪むので使わ
ない)

エンベロープを理解しよう



エンベロープを理解しよう

Mozziの場合整数で0~255



```

#define MOZZI_CONTROL_RATE 64
#include <Mozzi.h>
#include <Oscil.h>
#include <mozzi_midi.h>
#include <ADSR.h>
#include <tables/sin2048_int8.h>

Oscil <SIN2048_NUM_CELLS, MOZZI_AUDIO_RATE>
a0scil(SIN2048_DATA);

//エンベロープをかけるためのクラス
ADSR<MOZZI_AUDIO_RATE, MOZZI_AUDIO_RATE> envelope;

unsigned int Dur, Atk, Dec, Sus, Rel;
int button_prev = LOW;

void setup() {
  pinMode(2, INPUT_PULLUP);
  startMozzi();
  Atk = 10;
  Dec = 10;
  Sus = 50;
  Rel = 1000;
  envelope.setTimes(Atk, Dec, Sus, Rel);
  envelope.setADLevels(255, 128);
}

```

```

void updateControl() {
  auto button_state = digitalRead(2);
  if (button_prev == HIGH && button_state == LOW) {
    auto f = mtof((int)random(65, 100));
    a0scil.setFreq(f);
    envelope.noteOn();
  }
  if (button_prev == LOW && button_state == HIGH) {
    envelope.noteOff();
  }
  button_prev = button_state;
}

AudioOutput updateAudio() {
  envelope.update();
  int16_t output_raw = envelope.next() *
a0scil.next();
  int16_t output =
map(output_raw, 0, 65536, 5000, 32767);
  return MonoOutput::from16Bit(output);
}

void loop() {
  audioHook();
}

```

mozzi_envelope.ino

```

#define MOZZI_CONTROL_RATE 64
#include <Mozzi.h>
#include <Oscil.h>
#include <mozzi_midi.h>
#include <ADSR.h>
#include <tables/sin2048_int8.h>

```

```

Oscil <SIN2048_NUM_CELLS, MOZZI_AUDIO_RATE>
a0scil(SIN2048_DATA);

```

//エンベロープをかけるためのクラス

```
ADSR<MOZZI_AUDIO_RATE, MOZZI
```

```

unsigned int Dur, Atk, Dec,
int button_prev = LOW;

```

```

void setup() {
  pinMode(2, INPUT_PULLUP);
  startMozzi();
  Atk = 10;
  Dec = 10;
  Sus = 50;
  Rel = 1000;
  envelope.setTimes(Atk, Dec, Sus, Rel);
  envelope.setADLevels(255, 128);
}

```

mozzi_midi.hを読み込むと
midiToFreq()相当の関数を
mtof()で使える

値の範囲が
(0~255) * (0~255) =
0~65535 になる

```

id updateControl() {
  auto button_state = digitalRead(2);
  if (button_prev == HIGH && button_state == LOW) {
    auto f = mtof((int)random(65, 100));
    a0scil.setFreq(f);
    envelope.noteOn();
  }
  if (button_prev == LOW && button_state == HIGH) {
    envelope.noteOff();
  }
  button_prev = button_state;
}

```

```

AudioOutput updateAudio() {
  envelope.update();
  int16_t output_raw = envelope.next() *
  a0scil.next();
  int16_t output =
  map(output_raw, 0, 65536, 5000, 32767);
  return MonoOutput::from16Bit(output);
}

```

```

void loop() {
  audioHook();
}

```

複数のEventDelay

- アルゴリズムックな演奏をプログラムで表現することができる
- ライヒの「Piano Phase」を再現してみる

これによってアルゴリズムックなメロディをArduino内にプログラムすることができるようになります。
複数のEventDelayを平行して走らせることもできるので、独立した演奏プログラムを走らせたりすることもできます。下のスケッチでは、スティーヴ・ライヒの「Piano Phase」の序盤を再現してみました。譜面のパターンを配列変数に格納して、EventDelayが発火したタイミングで順番に鳴らしていています。1ミリ秒ごとにずれていく様子を聞くことができます。本家では途中から違う譜面になり終わり15分ほどで終了しますが、プログラムなので電源を供給する限り永遠に演奏が続きます。

```
#include <MozziGuts.h>
#include <mozzi_midi.h>
#include <Oscil.h>
#include <EventDelay.h>
#include <ADSR.h>
#include <tables/sin8192_int8.h>

Oscil <8192, AUDIO_RATE> aOscil(SIN8192_DATA); //元となる音色
Oscil <8192, AUDIO_RATE> aOscil2(SIN8192_DATA); //元となる音色
ADSR <AUDIO_RATE, AUDIO_RATE> envelope_A; //エンベロープをかけ
ADSR <AUDIO_RATE, AUDIO_RATE> envelope_B; //エンベロープをかけ
```

ひつじさんのサンプルより、Steve Reich 「Piano Phase」 の再現
http://sheep-me.me/2019/10/24/geidai_16/

```

#include <Mozzi.h>
#include <mozzi_midi.h>
#include <Oscil.h>
#include <EventDelay.h>
#include <ADSR.h>
#include <tables/sin8192_int8.h>

Oscil<8192, MOZZI_AUDIO_RATE> aOscil(SIN8192_DATA);
Oscil<8192, MOZZI_AUDIO_RATE> aOscil2(SIN8192_DATA);
ADSR<MOZZI_AUDIO_RATE, MOZZI_AUDIO_RATE> envelope_A;
ADSR<MOZZI_AUDIO_RATE, MOZZI_AUDIO_RATE> envelope_B;
unsigned int Dur, Atk, Dec, Sus, Rel;

//piano phaseのパターン
unsigned int pattern[] = { 64, 66, 71, 73, 74, 66,
64, 73, 71, 66, 74, 73 };

//二つのタイマーを用意
int phase_A = 0;
int phase_B = 0;
EventDelay timer_B;
EventDelay timer_A;

void setup() {
  startMozzi(1024);
  Atk = 10;
  Dec = 10;
  Sus = 50;
  Rel = 100;
  envelope_A.setTimes(Atk, Dec, Sus, Rel);
  envelope_A.setADLevels(255, 128);
  envelope_B.setTimes(Atk, Dec, Sus, Rel);
  envelope_B.setADLevels(255, 128);

  timer_A.set(150);
  timer_B.set(151);
  timer_B.start();
  timer_A.start();
}

```

```

void updateControl() {
  if (timer_A.ready()) {
    int note = pattern[phase_A];
    phase_A = (phase_A + 1) % 12;
    aOscil.setFreq(mtof(note));
    envelope_A.noteOn();
    timer_A.start();
  }

  if (timer_B.ready()) {
    int note = pattern[phase_B];
    phase_B = (phase_B + 1) % 12;
    aOscil2.setFreq(mtof(note));
    envelope_B.noteOn();
    timer_B.start();
  }
}

AudioOutput updateAudio() {
  envelope_A.update();
  envelope_B.update();
  int16_t A = (envelope_A.next() * aOscil.next());
  int16_t B = (envelope_B.next() * aOscil2.next());
  int16_t output = map((A + B) / 2, 0, 32767, 2000,
32767);
  return MonoOutput::from16Bit(output);
}

void loop() {
  audioHook();
}

```

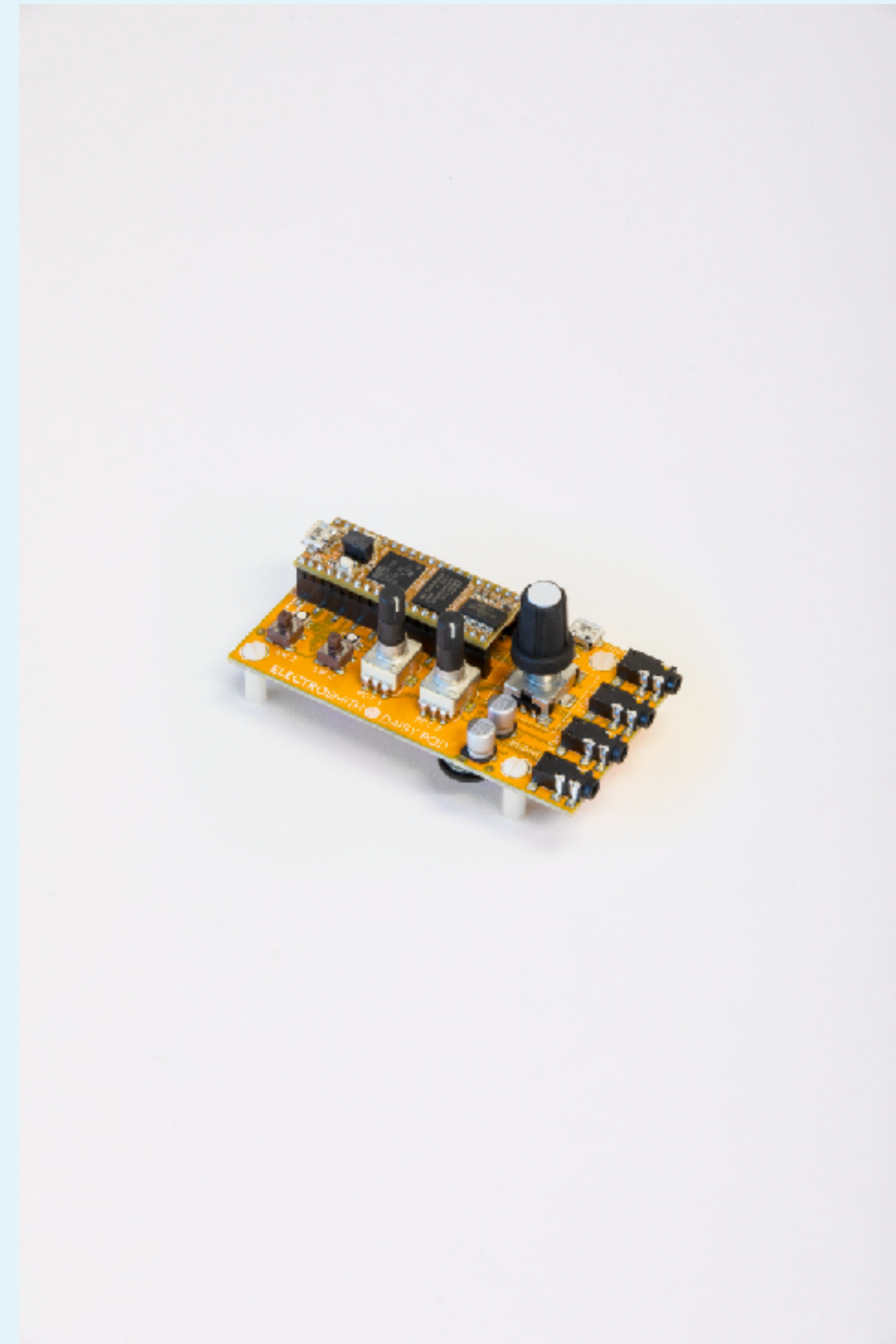
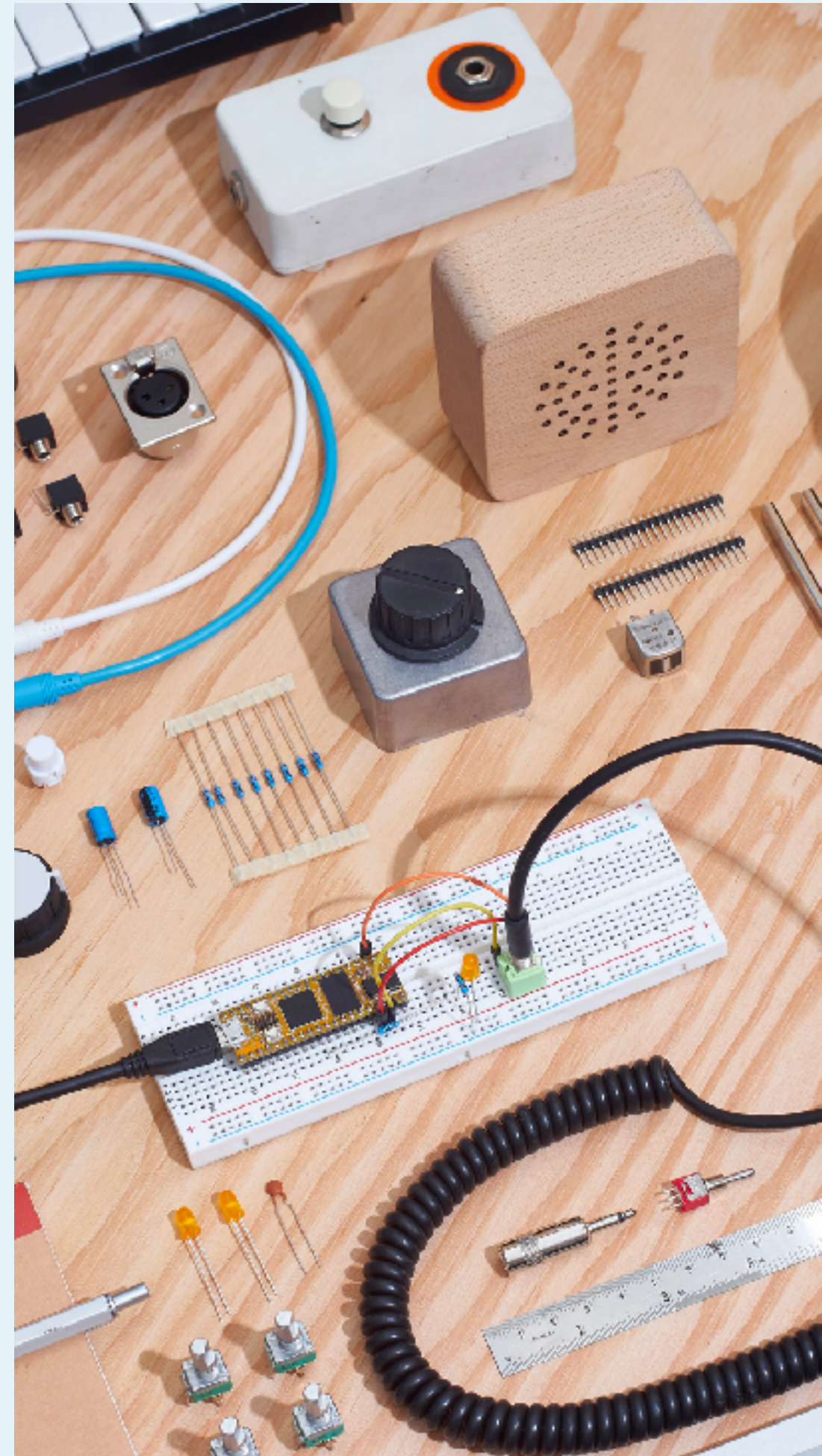
mozzi2_reich_R4.ino

Mozziの使い方2

- 今回は使わなかったが、analogReadを使うコードの場合mozziAnalogRead()に置き換える必要あり
- R4のanalogReadの結果はデフォルトで他のボードと同じ0-1024（10bit）の範囲になっているが、実はハードウェア的には14bitの精度がある
- analogReadResolution(14)とすると14bitで取得できる
- mozziAnalogRead()は標準で14bit（0-16383）の値を返すので注意

より高度なことがやりたい人は

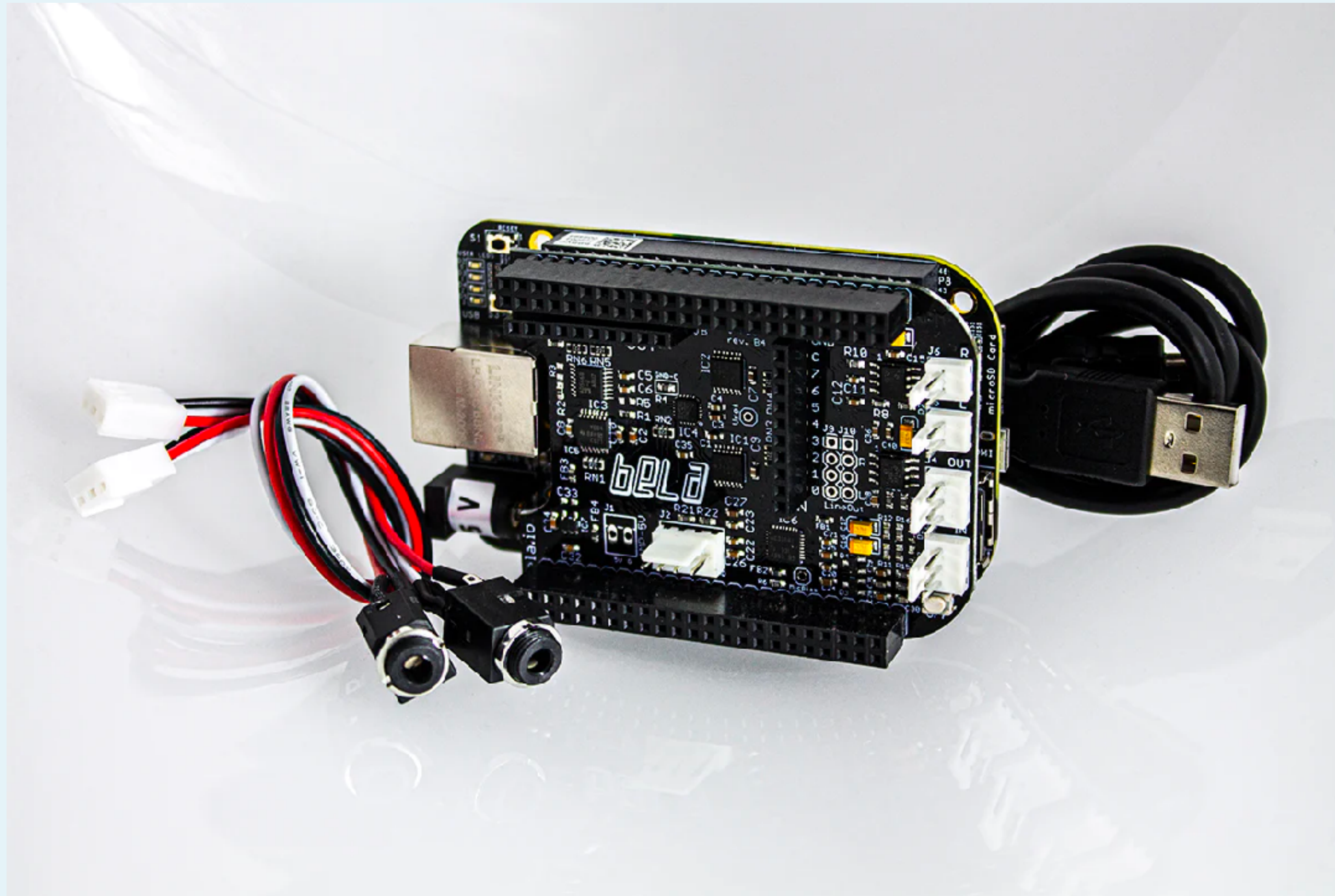
Daisy



<https://www.electro-smith.com/daisy>

Arduino IDE使用可 高音質ステレオDACあり 拡張ボード多数 国内だとFive Gとかで買える

Bela



<https://bela.io/>

Linuxが動く。専用のWeb IDEで開発可能

参考資料

※Mozziのサンプルはいずれもv1の情報なので参考までに



kusamura

ひつじ(日辻)のWebページ

Index

Biography

Artwork

講義資料



第十五項 音響合成ライブラリMozzi

第十五項 音響合成ライブラリMozzi

目次

1. 第十五項 音響合成ライブラリMozzi

1.1. スライドPDF

1.2. Arduino単体で音響合成を行う

1.3. Mozziとは？

1.3.1. 何故こんなことができるのか

1.4. インストール

1.4.1. Arduinoから線を出す

1.4.2. サインウェーブを鳴らす

1.4.3. Oscilオブジェクト

1.4.4. updateControlでパラメータを変更する

1.5. 足して和音を作ってみる

電子工作創作表現(2019/10/18)

スライドPDF

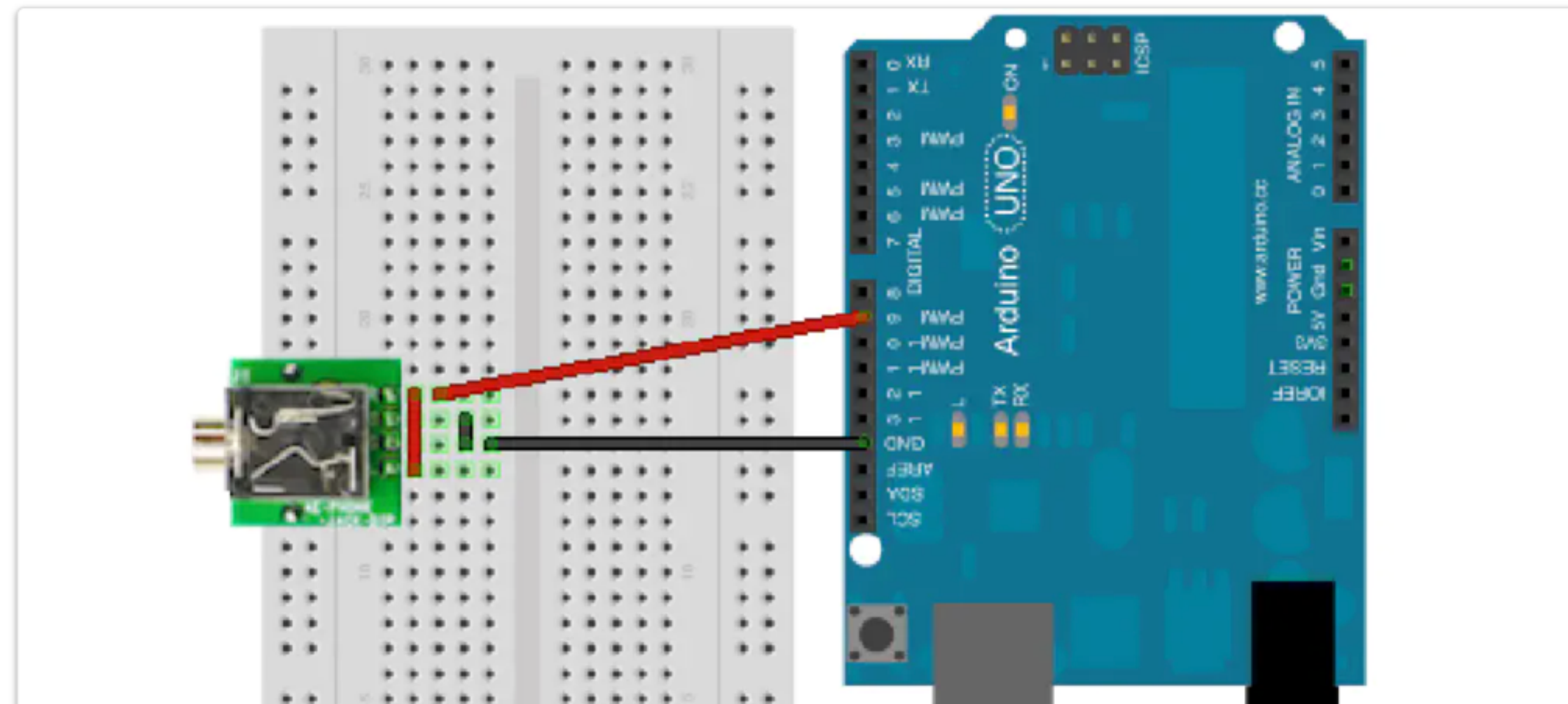
Arduino単体で音響合成を行う

- Arduino単体の機能で作れる音には限界がある
- 「Mozzi」を使うとかなり幅が広がる

電子工作創作表現 / 音楽音響創造特殊研究 講義資料(2019), ひつじ
<http://sheep-me.me/講義資料/>

MozziWS

[HOME](#) [NEWS](#) [BIOGRAPHY](#) [PROJECTS](#) [CLIENTWORKS](#) [WORKSHOPS](#) [CONTACT](#)



MozziWS, 中西宣人

<https://yoshihito-nakanishi.com/workshops/mozzi>

SSAW14 – サウンド&ソフトウェアアート 2014

- 多摩美術大学メディア芸術コース
- 対象：2年 (選択必修)
- 前期・第1クォーター
- 火曜、3～4限
- 206教室

この授業について

安価な電子部品や、分解した電気製品・玩具を利用(ハック)してその可能性を最大限に発揮させたり、Arduinoという汎用のデバイスにセンサーを接続したりプログラムを書くことで、創造的なインターフェイス・デザインとしての自作電子楽器や映像装置を制作します。最終発表会では、創作した楽器を用いたパフォーマンスを行って、身体と電子デバイスが呼応したハイブリッドな音楽制作の可能性を探求します。

講義ノート

- [Mozziを使ってみる](#)
- [センサーでMozziをコントロール](#)
- [Mozziでサンプリング&プレイバック](#)
- [PdとMozziを連携する](#)

(ちょっと)高レイヤー オーディオ組み込み2020

DSP言語/プログラマブルボード/
オープンハードウェア



2020/09/01 松浦知也(me@matsuuratomoya.com/matsuuratomoya.com)

1