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10週

2024.06.21 松浦知也（matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com）

残りの授業のスケジュール

- 6/20(今日) 出力を考える①Arduinoで音を鳴らす
- 6/26(木、補講、出られる人だけ)14:00〜秋葉原遠足
- 6/27 課題制作相談会1
- 7/5 出力を考える②モーターを使ってみよう
- 7/12 課題制作相談会2
- 7/19 課題作品インストール作業
- 7/25(木) 講評&振り返り&撤収

残りの授業のスケジュール

- 部品の買い出し等の都合があると思うので、7/5の回含め、制作の時間に充てたい人はそれも可とする。
  - ただし、授業開始前までにClassroomのコメント、メール、口頭などで連絡をした上で、出席フォームも期限内に回答すること。
- Classroomの課題：プロジェクトシートを授業終了時まで継続して更新し、最終課題講評1週間後（8/1）までに提出を完了してください。
- ただし：展示のレイアウトの検討のため、作品サイズは7/5までに決定してください。
- 作品タイトルとキャプション/解説文(400~800字)は7/16（火）23:59の時点で記入されているものをこちらで出力します。間に合わない人はインストールまでに自分で用意してください。

6/27、7/12の授業の進行

- 6/27は17:00ごろまでオフィスアワー＋自由作業、後半は制作内容を2グループに分かれて一人ずつ発表
- 7/12は4,5限通しでオフィスアワー＋自由作業

課題（再掲）

あなたにとっての「パーソナルコンピューター」を作れるようになること

- 例えば：自動ドローイングマシン、アクセサリー、入力装置
- Or、授業の中で学んだ内容のうちのトピックのいずれかを教えるための道具作り
  - 例えば、Conditional Designの新しいルールを考えるとかもあり◎
- 自由枠：自分の課題制作等に授業内で学んだ内容を反映する

- 技術的な難易度よりも、なぜそれを作ったのか伝わるように

本日の残りスケジュール

- Arduino（Mozzi）で音を出すプログラムを作ろう

Arduino Uno R4で音を出す

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

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

  pinMode(sound_pin, OUTPUT);
}

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

tone関数を使うと3,11ピンをPWMとして使う時に干渉するので注意

tone_minimal.ino

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）

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

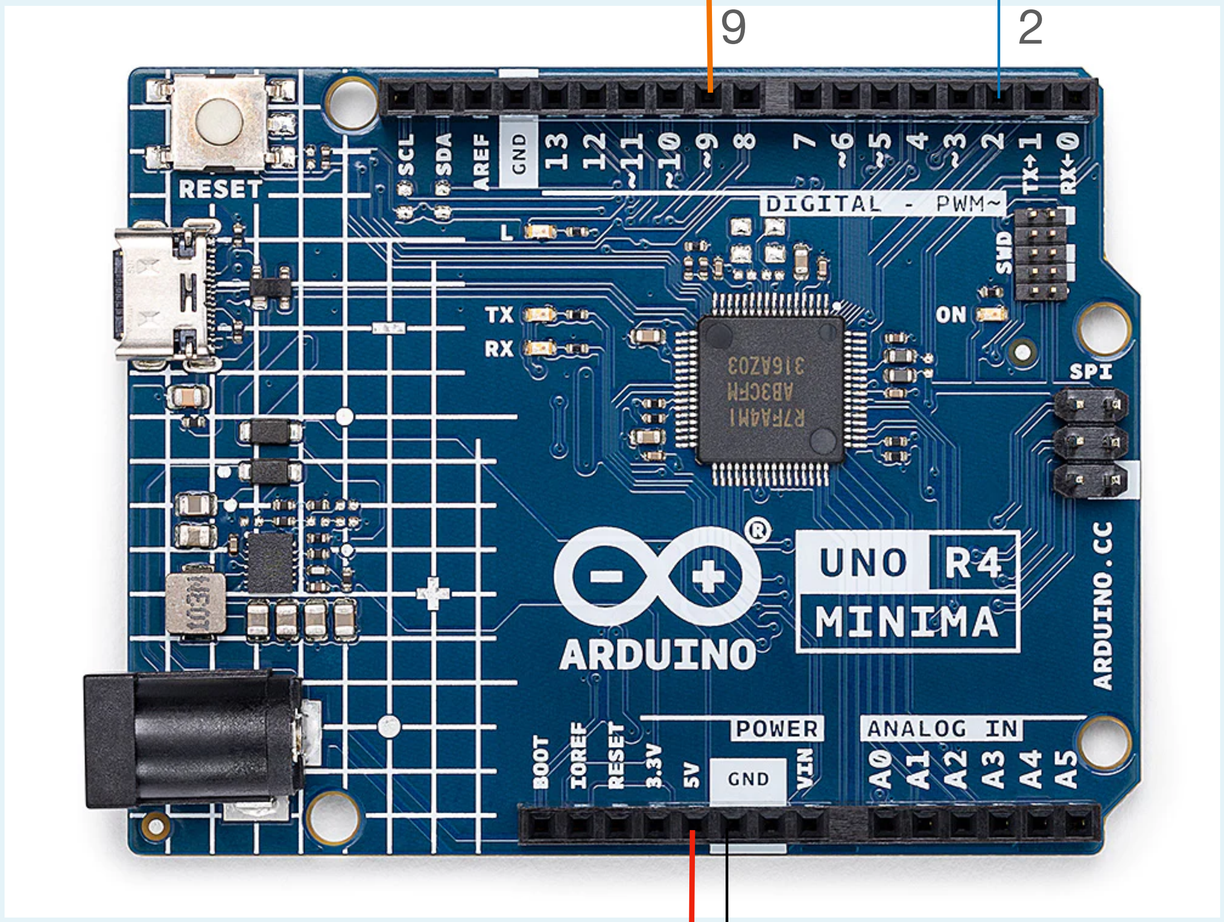
```ino
const int button_pin = 2;
const int sound_pin = 9;

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

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

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

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

  if(button_prev == LOW && button_state==HIGH){
    is_playing = !is_playing;
    freq = midiToFreq(random(40,100));
  }
  if(is_playing){
    tone(sound_pin,freq);
  }else{
    noTone(sound_pin);
  }

  delay(20);
  button_prev = button_state;
}
```

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

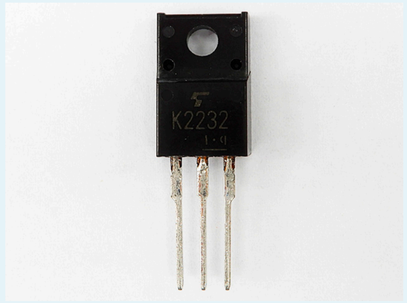
tone_random_flip.ino

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



N -MOSFET



G D S

4~8Ω
10Wくらいまでの
スピーカー

\+
—

※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

```ino
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

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/

AliExpress

IGMOPNRQ Component parts Store
97.6% 肯定的なフィードバック

+ フォロー
2423 フォロワー

私は...を買ってい | AliExpressにおいて | このストアにおいて

0

ストアホーム　商品　販売商品　トップセーリング　フィードバック

10個8403 PAM8403 3ワット * 2ステレオフィルタレスd級オーディオアンプsop-16新

さらに 2% オフ

★★★★★ 5.0 4 レビュー 17 Sold

￥423 / ロット (10 部分)

~~￥446~~ 5% オフ

数量: 1

追加 1% オフ (5 lots 以上)
662 lots ご利用可能

配送先 Japan

配送: ￥189

8月 09日に配送予定

10 日配送 お買い上げ金額 ￥709 以上で

Cainiao Super Economy Global 経由で China 発 Japan 着

納期保証

その他のオプション

今すぐ買います　カートに追加　♡ 625

75日バイヤープロテクション
返金の保証

お客様へのおすすめ

￥175

￥85

￥70

まとめ買いするともっと安いのもある（ただし粗悪品もあるので自己責任）

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）

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

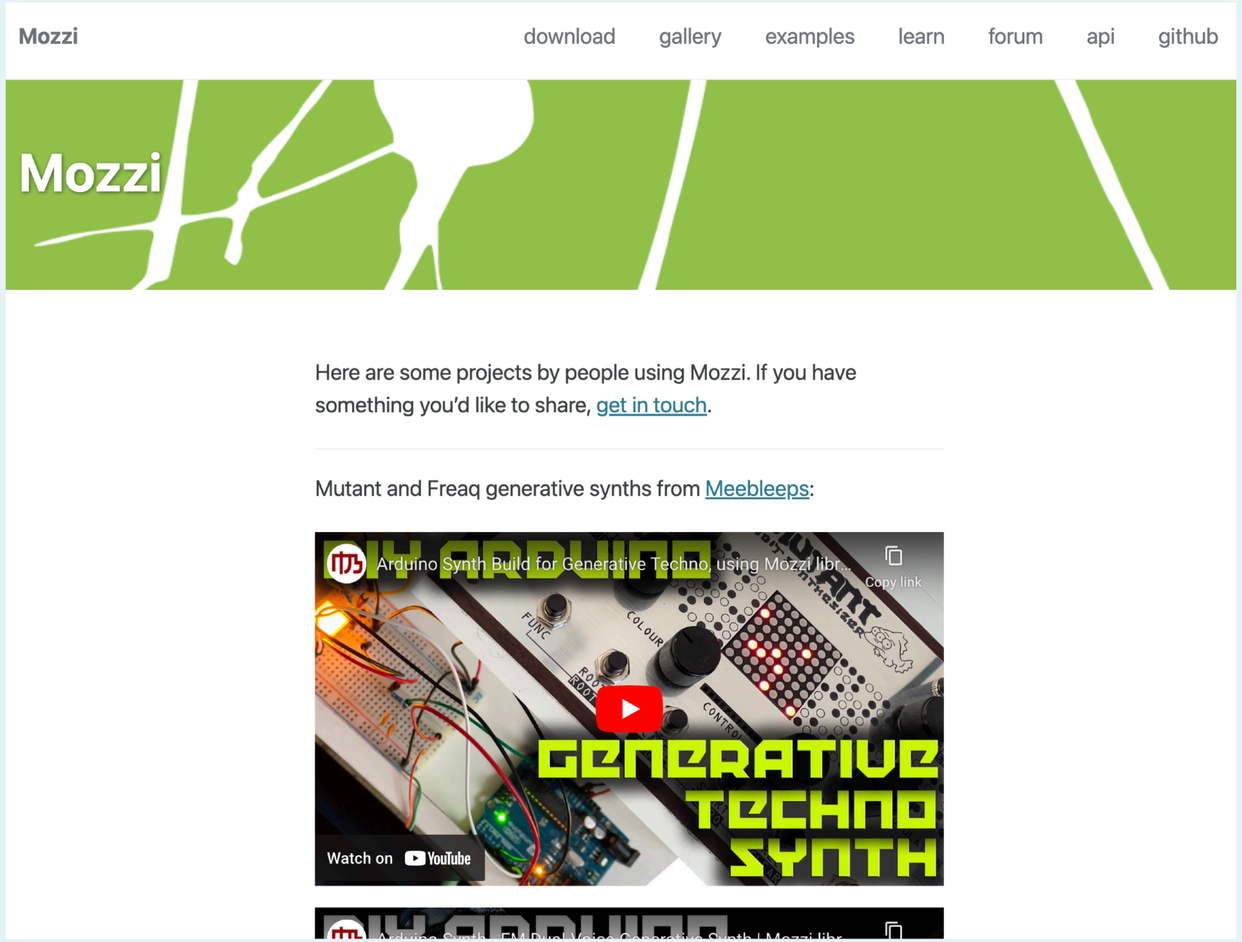
```ino
const int button_pin = 2;
const int sound_pin = 9;

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

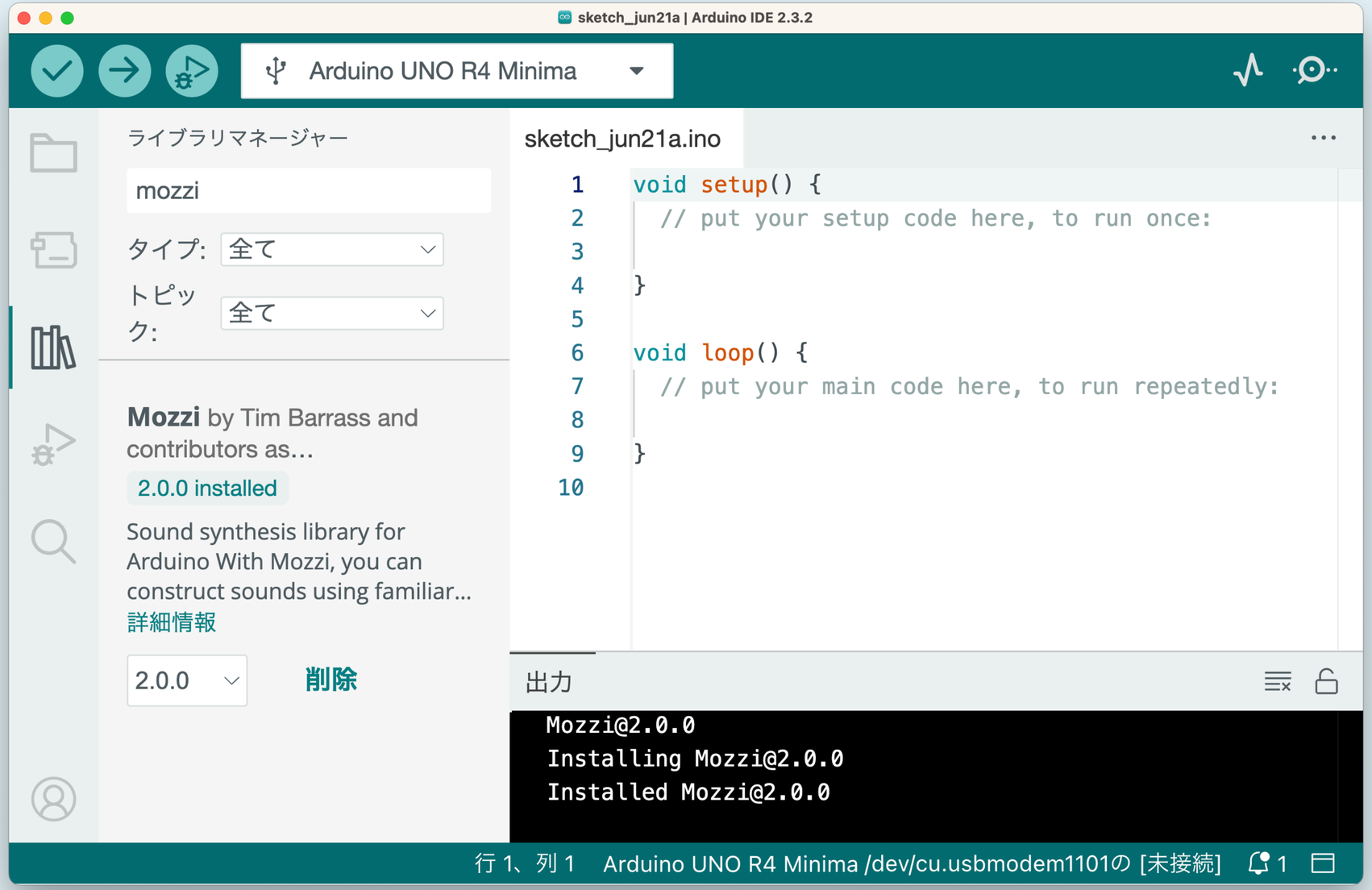
Mozziで高度な音作り

Mozzi



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

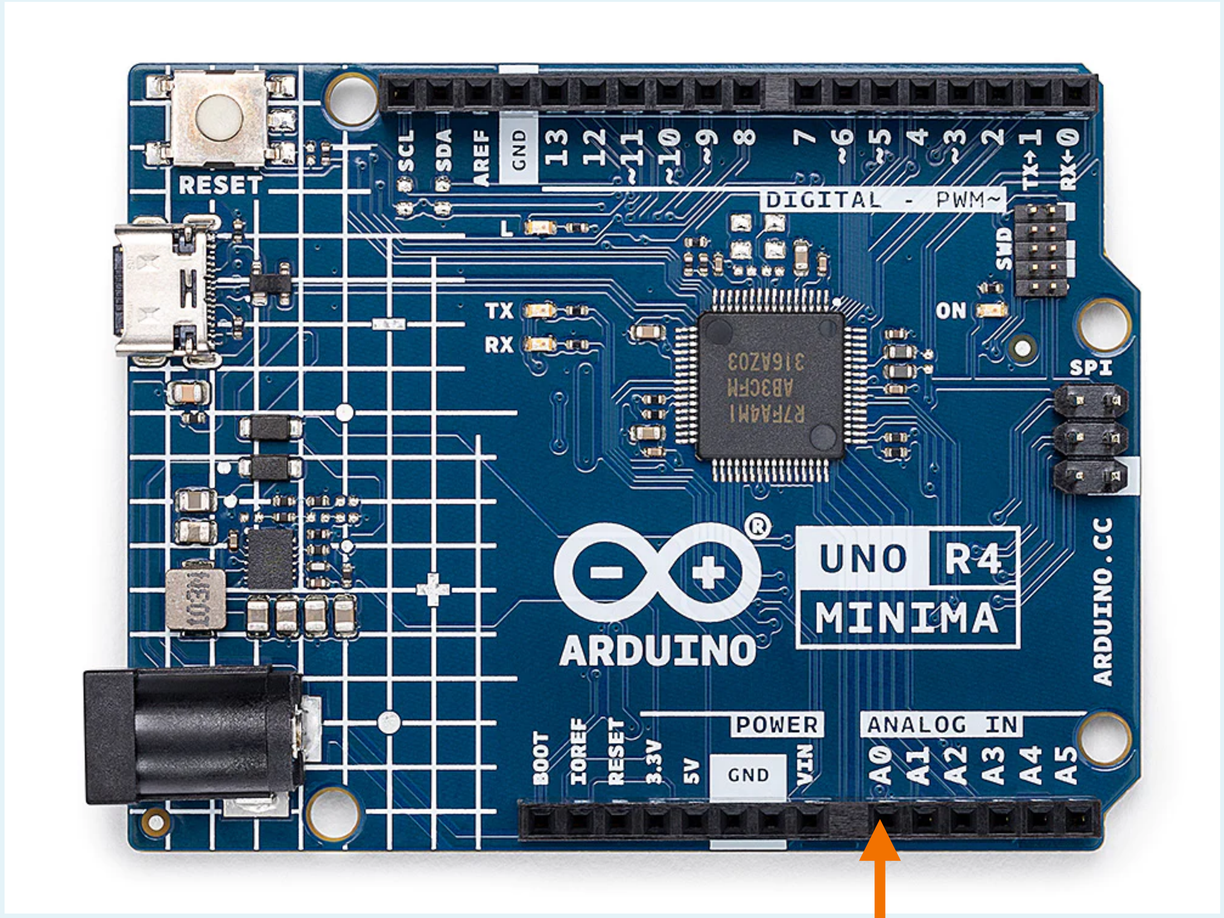
Arduinoで最もポピュラーな音声合成ライブラリ



※今回はつい最近リリースされたばかりのv2.0.0を使用します。

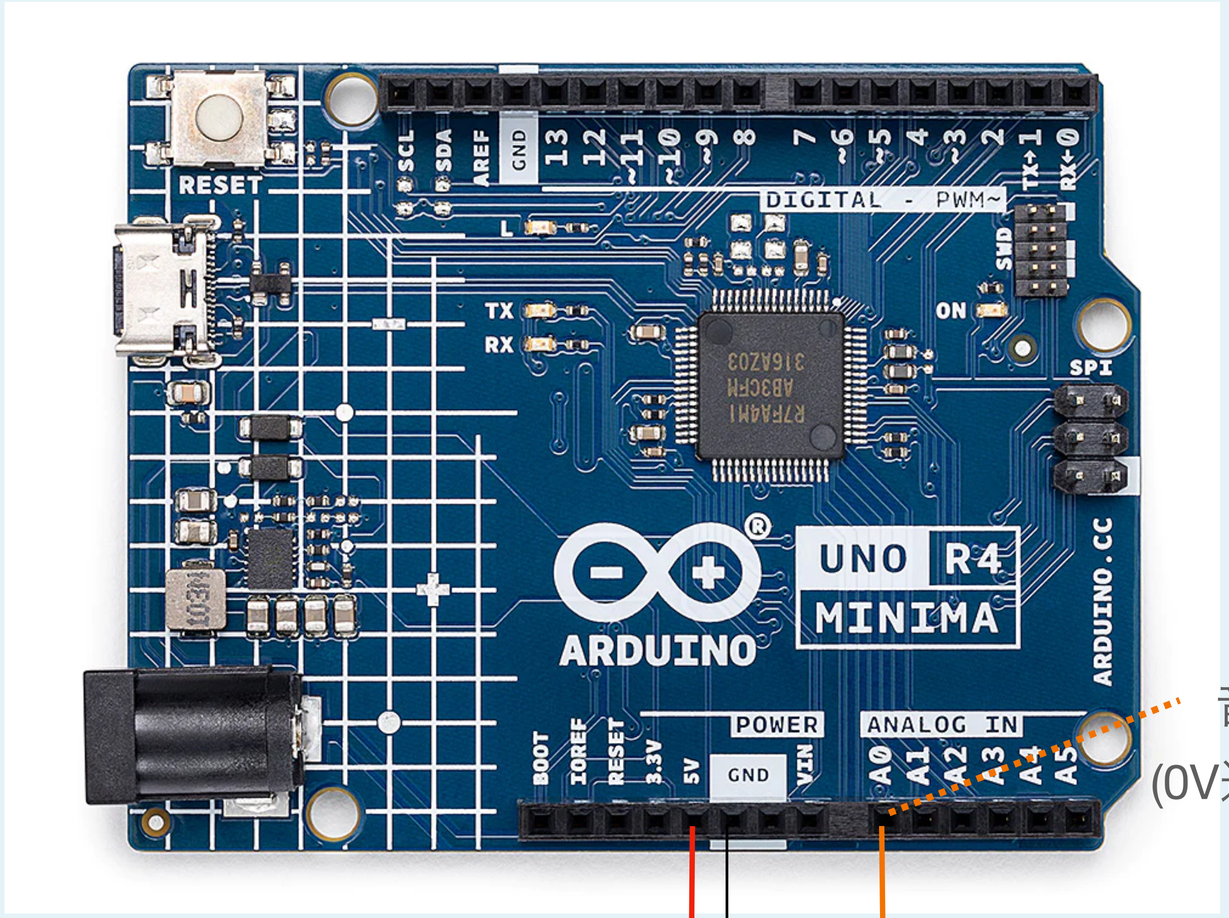
ネットに落ちている資料はまだほぼv1系ばかりで、書き方が変わっている部分もあるので要注意。

サンプルスケッチを参考にするのが安牌

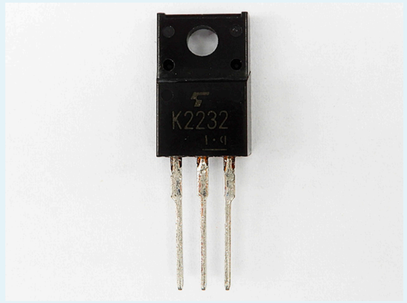


Arduino Uno R4はA0のピンにDAC(Digital-to-Analog Converter)がついている

今日使用する接続方法



N -MOSFET



G D S

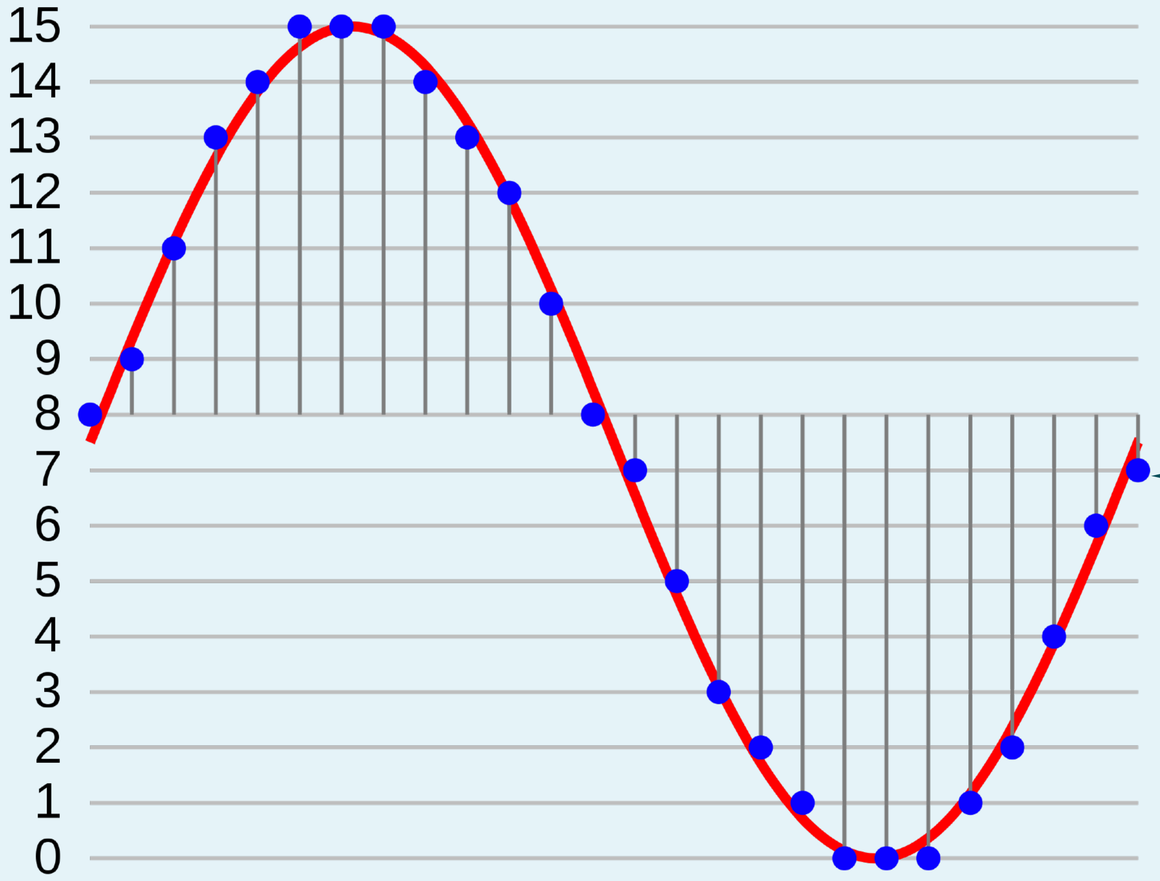
4~8Ω
10Wくらいまでの
スピーカー

＋
－

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

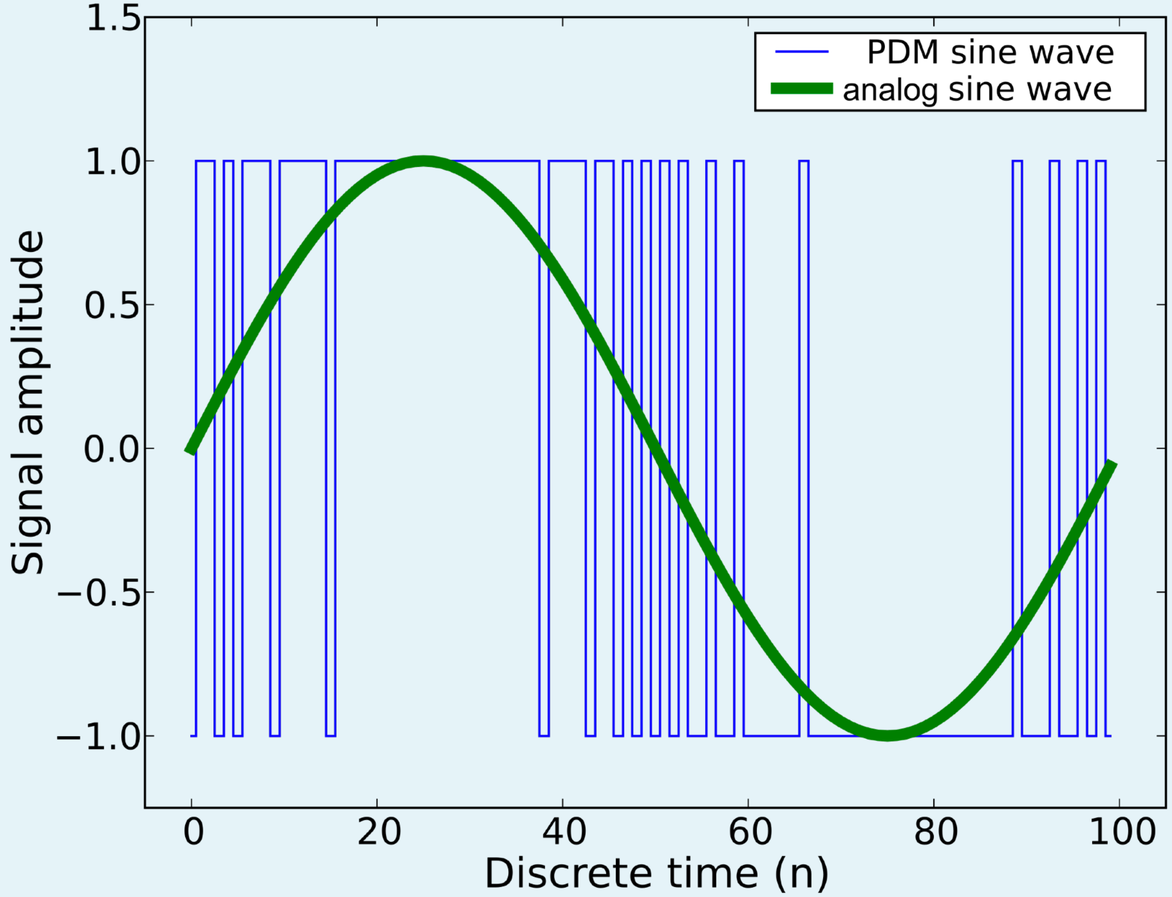
Pulse Code Modulation(PCM)



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

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ADCTouchのようなライブラリとの組み合わせも難しい
- delay()の代わりに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
- 遅い処理（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
- loopの中ではaudioHook()という関数だけを呼ぶ

```
#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();
}
```

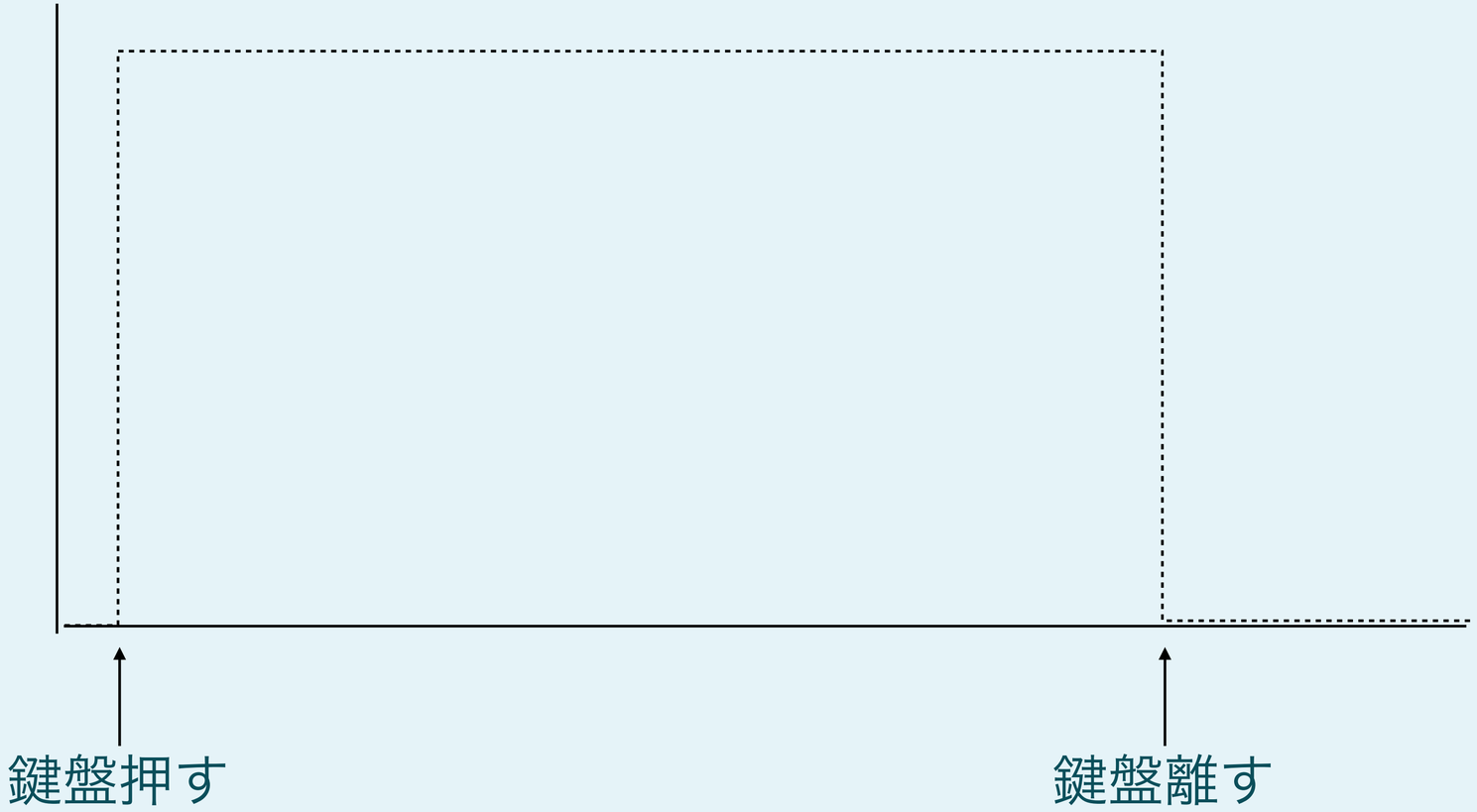
SineWave_R4.ino

```
#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();
}
```

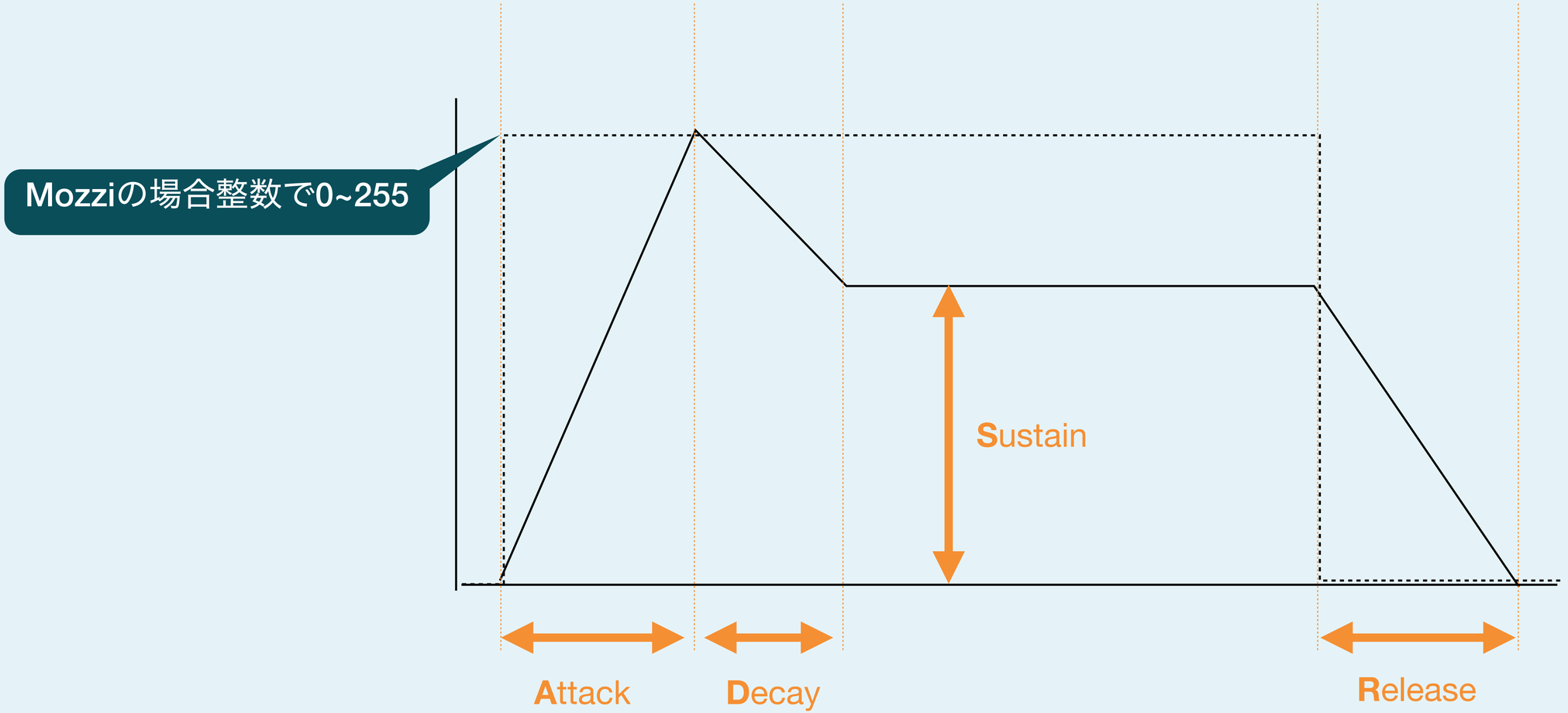
aSinの波形出力0~255を、2000~32767にマッピング(0~2000近辺は歪むので使わない)

SineWave_R4.ino

エンベロープを理解しよう



エンベロープを理解しよう



```
#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>
aOscil(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));
    aOscil.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() *
aOscil.next() ;
  int16_t output =
map(output_raw,0,65536,5000,65536);
  return MonoOutput::from16Bit(output);
}

void loop() {
  audioHook();
}
```

mozzi_envelope.ino

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

```
#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>
aOscil(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);
}
```

```
id updateControl() {
auto button_state = digitalRead(2);
if (button_prev == HIGH && button_state == LOW) {
  auto f = mtof((int)random(65, 100));
  aOscil.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() *
aOscil.next() ;
  int16_t output =
map(output_raw,0,65536,5000,32767);
  return MonoOutput::from16Bit(output);
}

void loop() {
  audioHook();
}
```

mozzi_envelope.ino

複数のEventDelay

- アルゴリズミ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ミックなメロディを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っています。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。

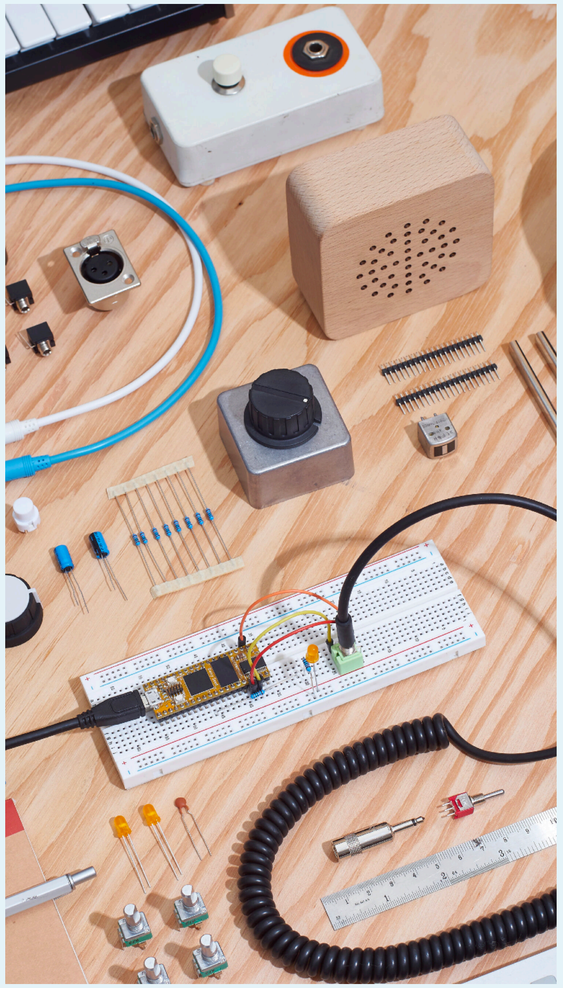
```
#include <MozziGuts.h>
#include <mozzi_midi.h>
#include <Oscil.h>
#include <EventDelay.h>
#include <ADSR.h>
#include <tables/sin8192_int8.h>

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
```

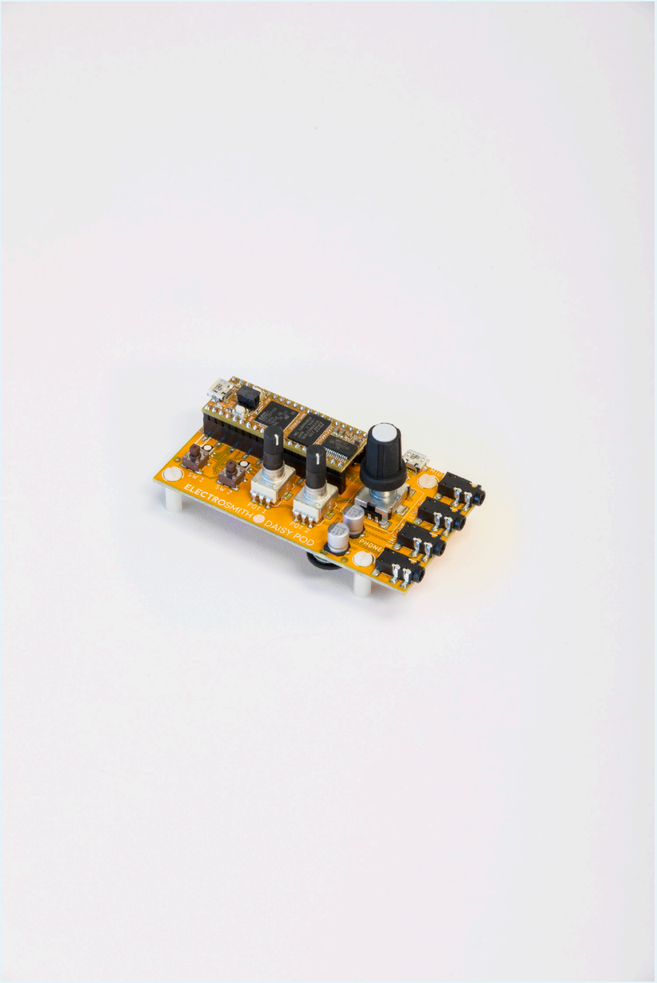
ひつじさんのサンプルより、Steve Reich「Piano Phase」の再現
http://sheep-me.me/2019/10/24/geidai_16/

```
#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();
}
```

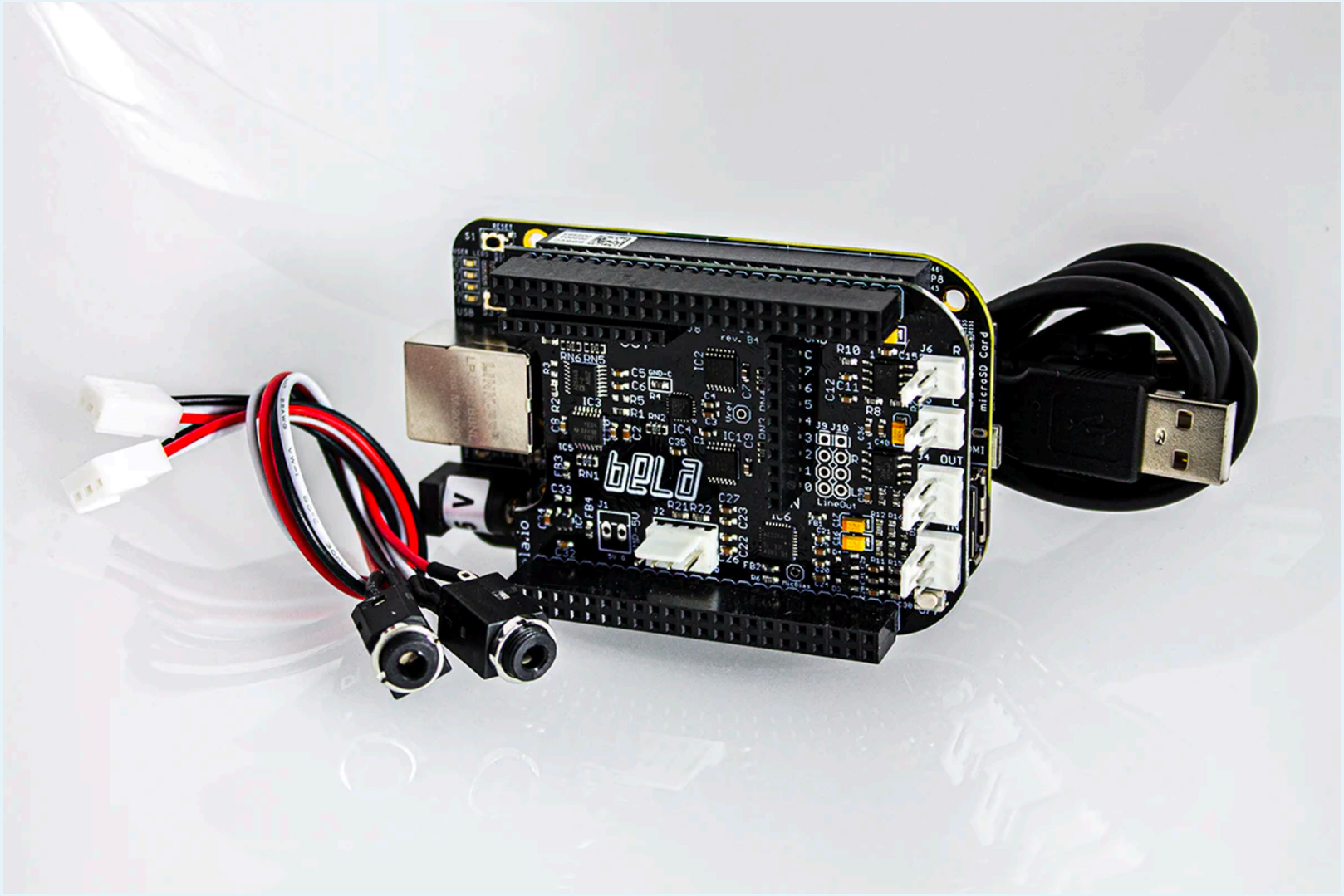
mozzi2_reich_R4.ino

より高度なことがやりたい人は





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



https://bela.io/

参考資料

kusamura

ひつじ(日辻)のWebページ

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

> 電子工作創作表現(2019/10/18)

スライドPDF

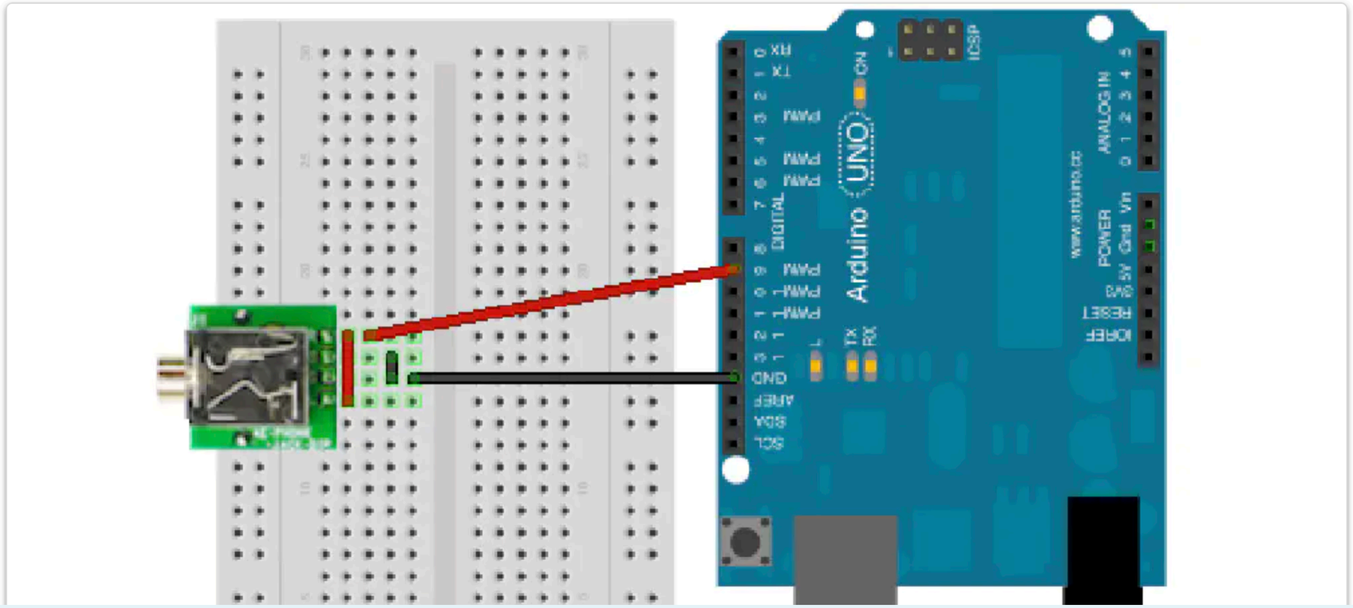
Arduino単体で音響合成を行う

- 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

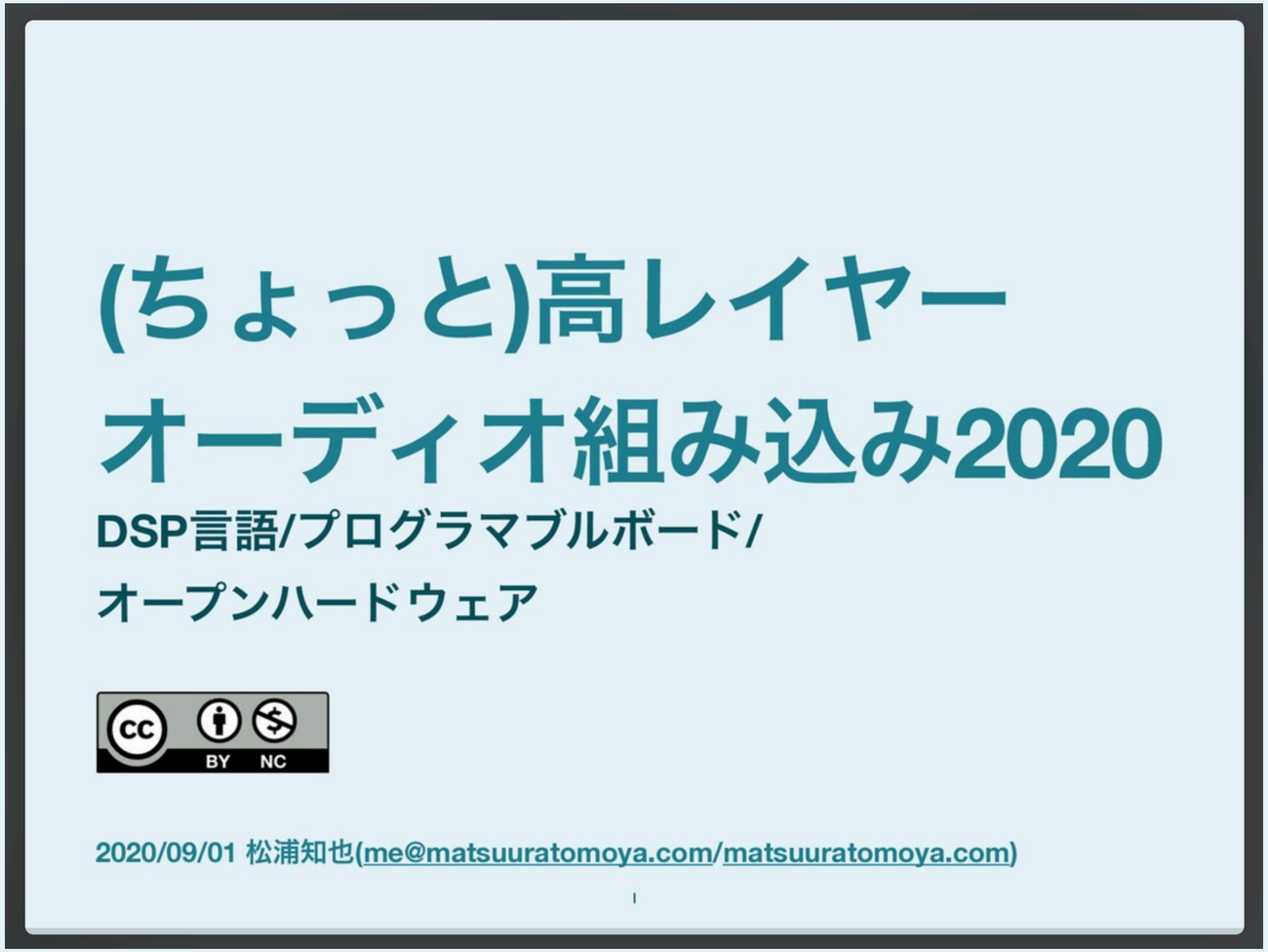
この授業について

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

SSAW14 – サウンド&ソフトウェアアート 2014, 田所淳
https://yoppa.org/ssaw14/



https://speakerdeck.com/tomoyanonymous/a-bit-high-layer-audio-embedded-platforms-in-2020