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

2023.06.09 松浦知也（matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com）

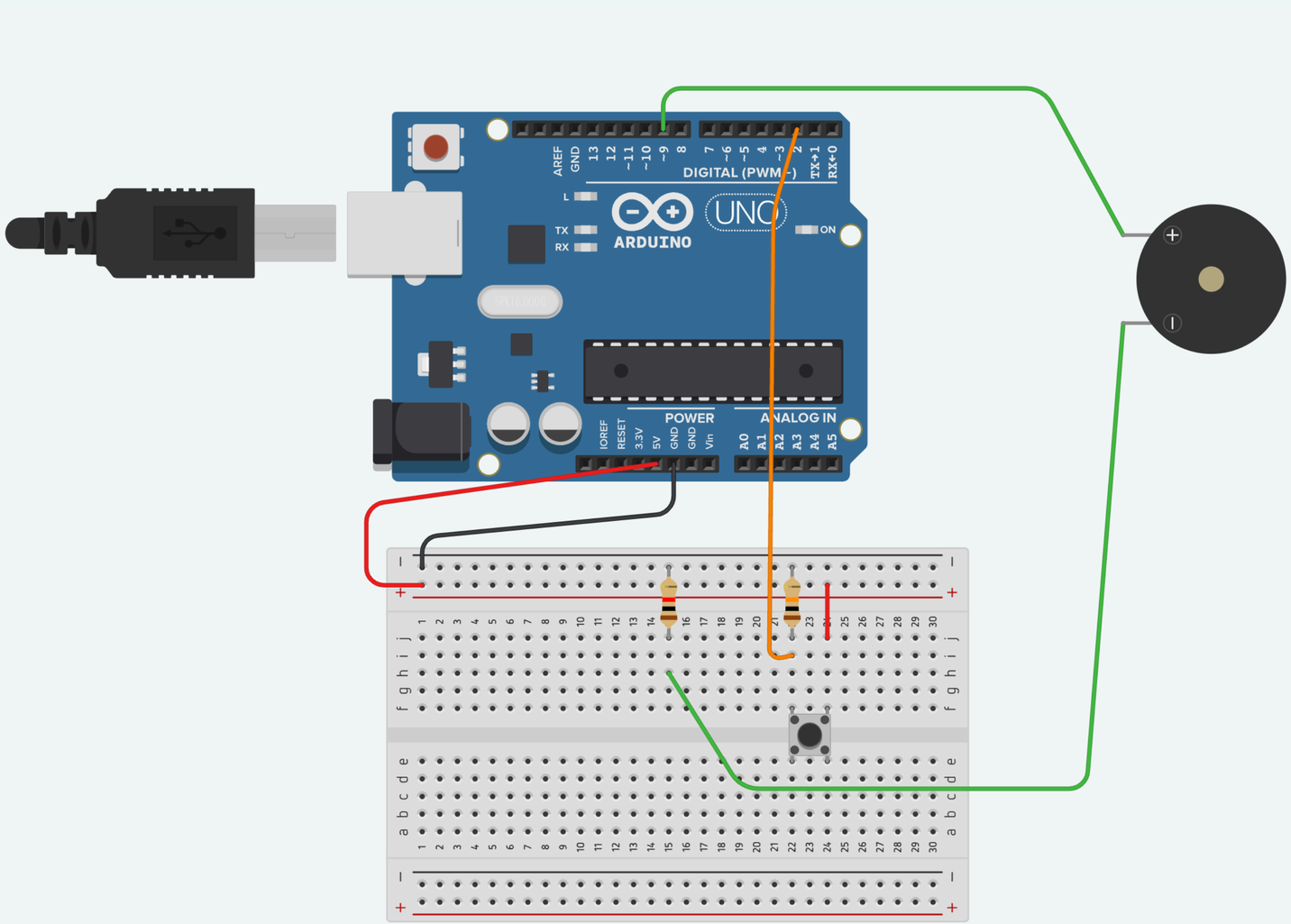
本日のスケジュール

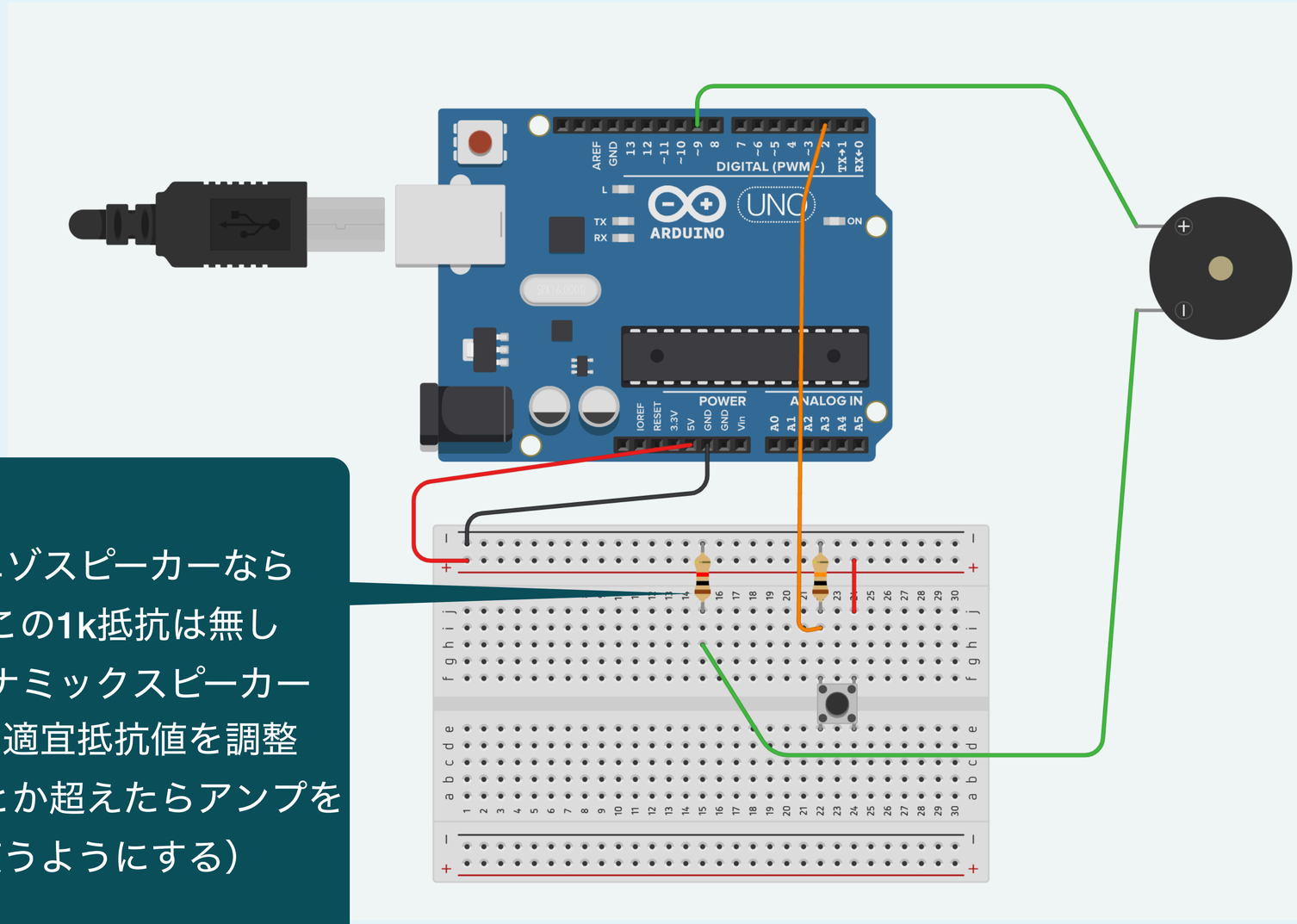
- Arduino（Mozzi）で音を出すプログラムを作ろう

質問コーナー

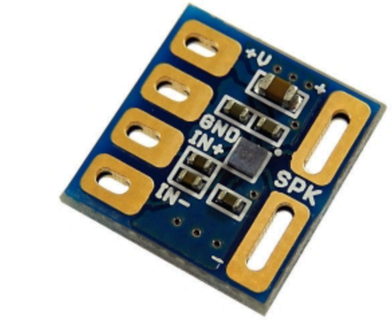
- 何も知らない状態からJava言語やC言語のようなプログラミング言語を勉強するのは用語集のようなものを扱って覚えて行くものなのでしょうか?
  - 言語によらず使いまわせる知識はある。（if文のような基本的な概念とか、アルゴリズムとか。）ここは一度学べばあまり陳腐化しない
  - 言語依存の知識は頻繁にアップデートされるので、調べて追いつく力の方が重要
- ググったりChatGPTに聞くための語彙を増やすためには根本的な知識がいる

とりあえずArduinoで音を出す


DIGITAL (PWM~)
ARDUINO
UNO
POWER
ANALOG IN



ピエゾスピーカーなら
ここの1k抵抗は無し
ダイナミックスピーカー
なら適宜抵抗値を調整
（5Wとか超えたらアンプを
使うようにする）







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
小型ボリューム(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。

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

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

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

void setup() {
  pinMode(button_pin, INPUT);
  //tone()を使うときは3,11以外
  pinMode(sound_pin, OUTPUT);
}

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

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）

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

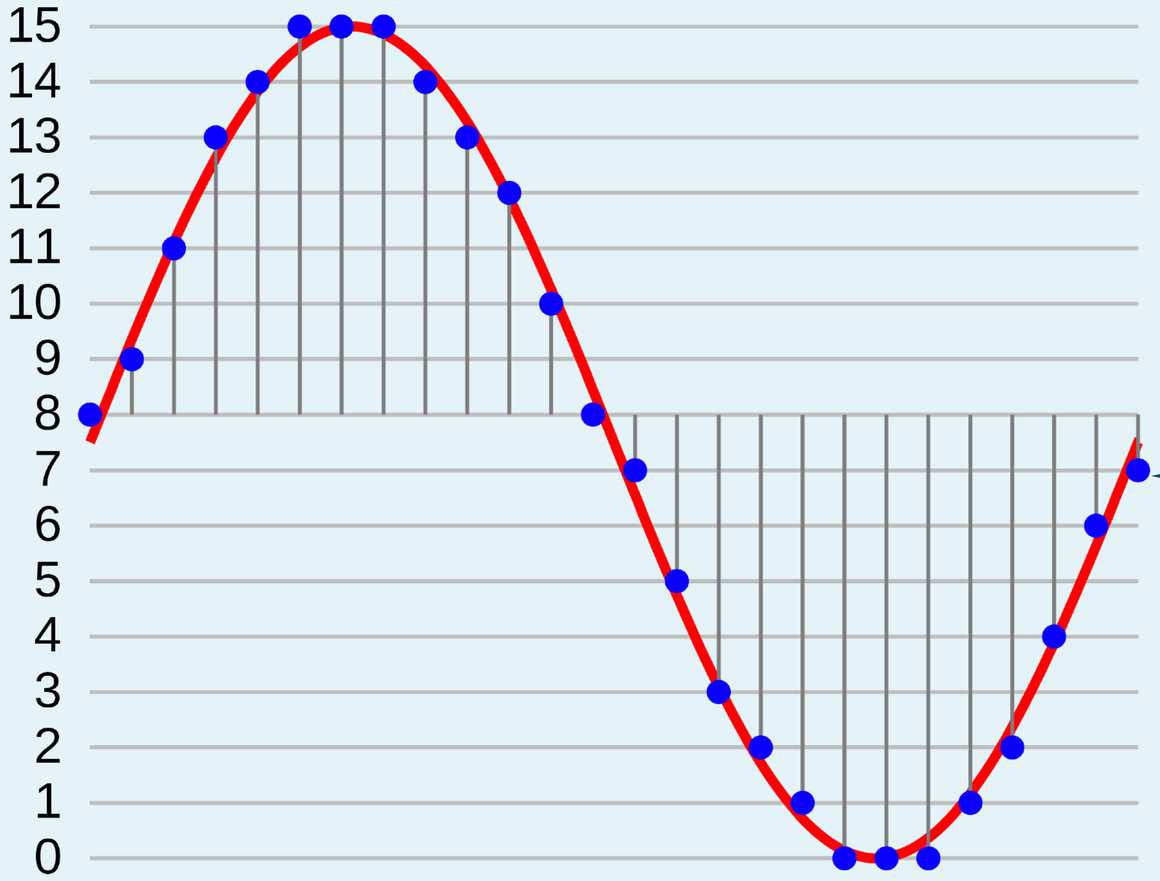
```ino
const int button_pin = 2;
const int sound_pin = 9;

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

Mozziで高度な音作り

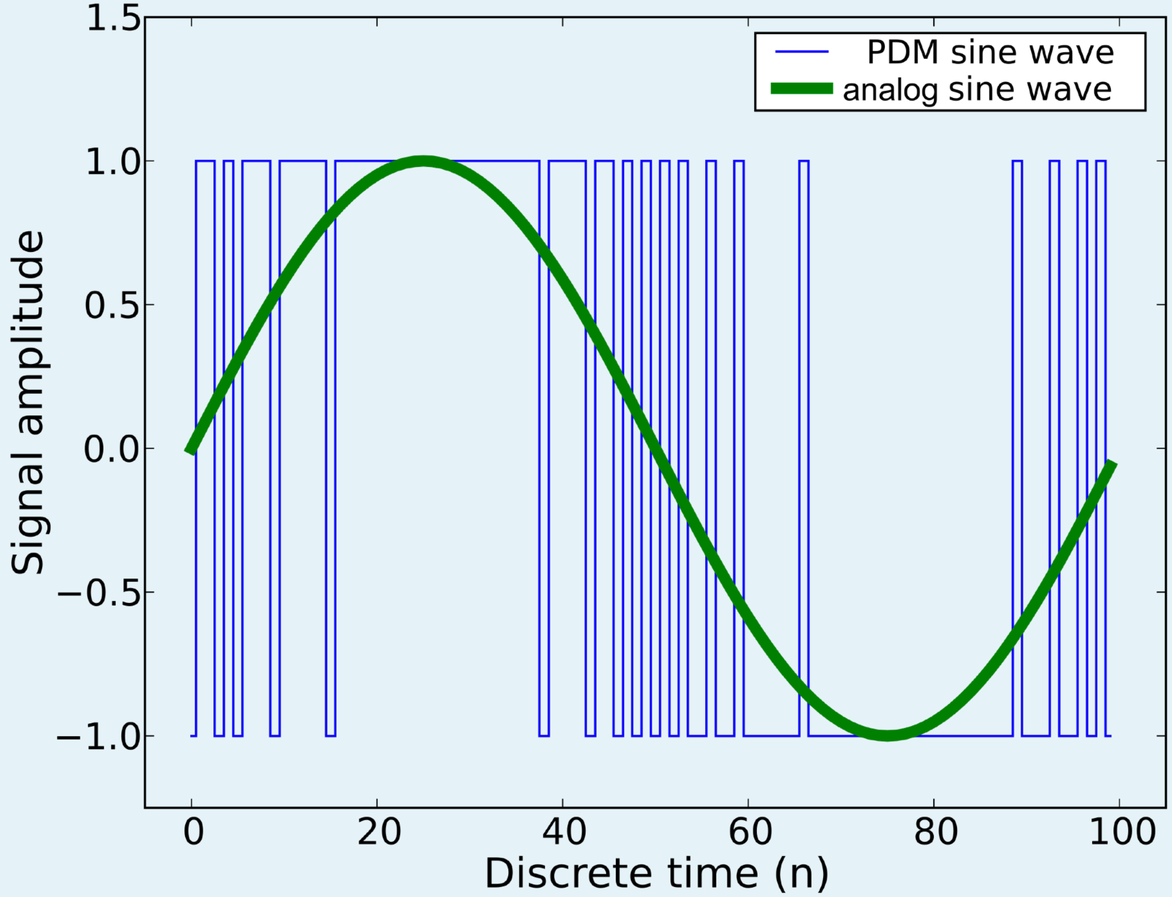
Pulse Code Modulation(PCM)



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

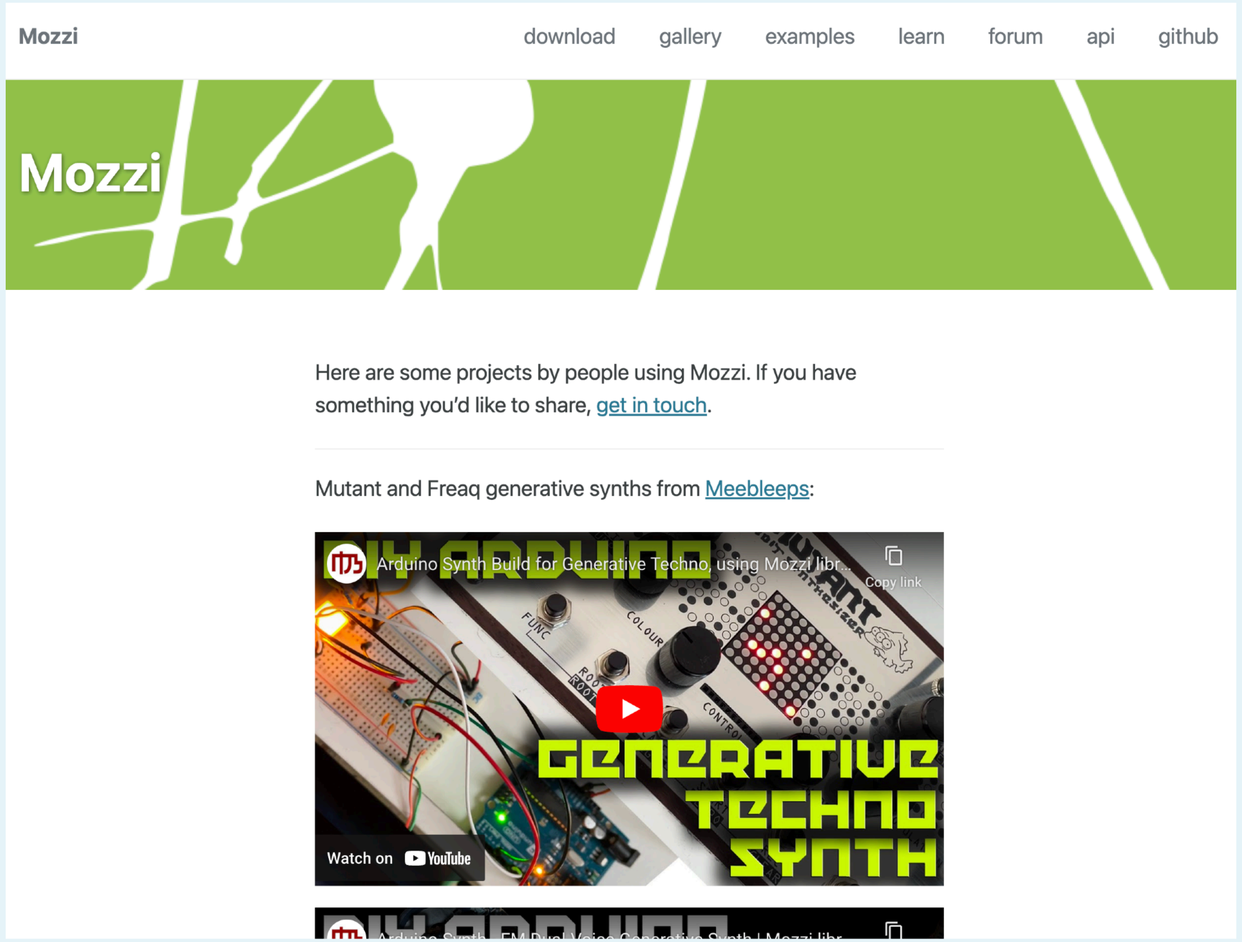
Pulse Density Modulation

縦軸が粗い代わりに横軸を細かくする



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



Arduinoのタイマーや PWM機能を最大限に活用して、DACなしで複雑な音声合成をするライブラリ


tone_random_flip | Arduino IDE 2.0.4
Arduino Uno
ライブラリマネージャー
Mozzi
タイプ: 全て
トピック: 全て
Mozzi by Tim Barrass and contributors as documented in source, and at https://github.com/sensorium
バージョン
インストール済
1.1.0
With Mozzi, you can construct sounds using familiar synthesis units like oscillators, delays, filters and envelopes.
Sound synthesis library for Arduino
詳細情報
1.0.3
インストール
tone_random_flip.ino
1 const int button_pin = 2;
2 const int sound_pin = 9;
3
4 bool is_playing = false;
5 int button_prev= LOW;
6 float freq = 440;
7
8 float midiToFreq(int midi){
9 return 440.*pow(2.,(midi-69.)/12.);
10 }
11
12 void setup() {
13 pinMode(button_pin,INPUT);
14 pinMode(sound_pin, OUTPUT);
15 }
16
17 void loop() {
18 auto button_state = digitalRead(button_pin);
19
20 if(button_prev == LOW && button_state==HIGH){
21 is_playing = !is_playing;
22 freq = midiToFreq(random(40,100));
23 }
24 if(is_playing){
25 tone(sound_pin,freq);
26 }else{
行 21、列 30 Arduino Uno /dev/cu.usbmodem141301の

Mozziの注意点

- スピーカーへ出力するピンは9番で固定
- 他のPWMピン（analogWrite）も基本的に使えなくなる
- 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

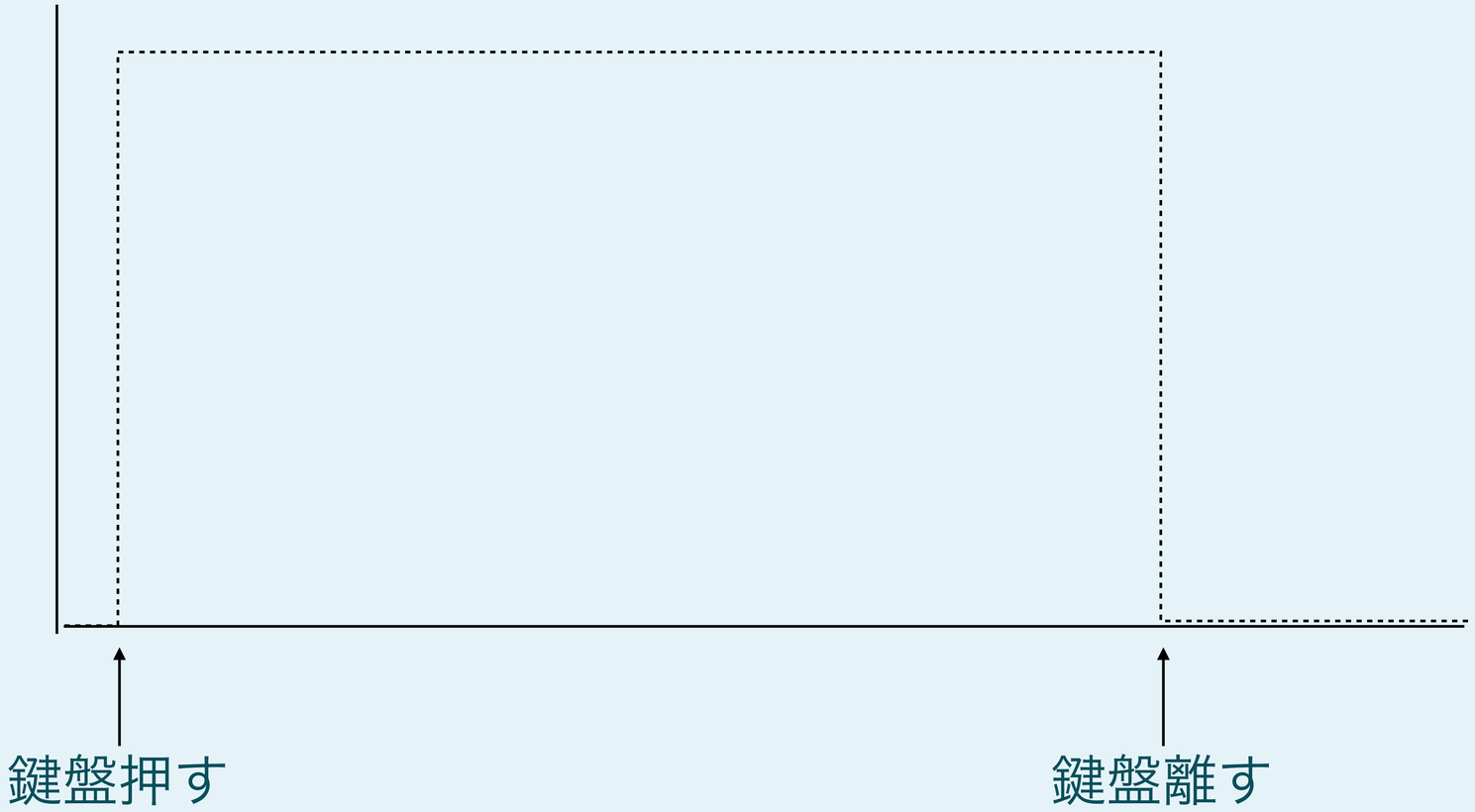
Mozziの使い方

- 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
  - この値の範囲は -244〜243でなければならない
- loopの中ではaudioHook()という関数だけを呼ぶ

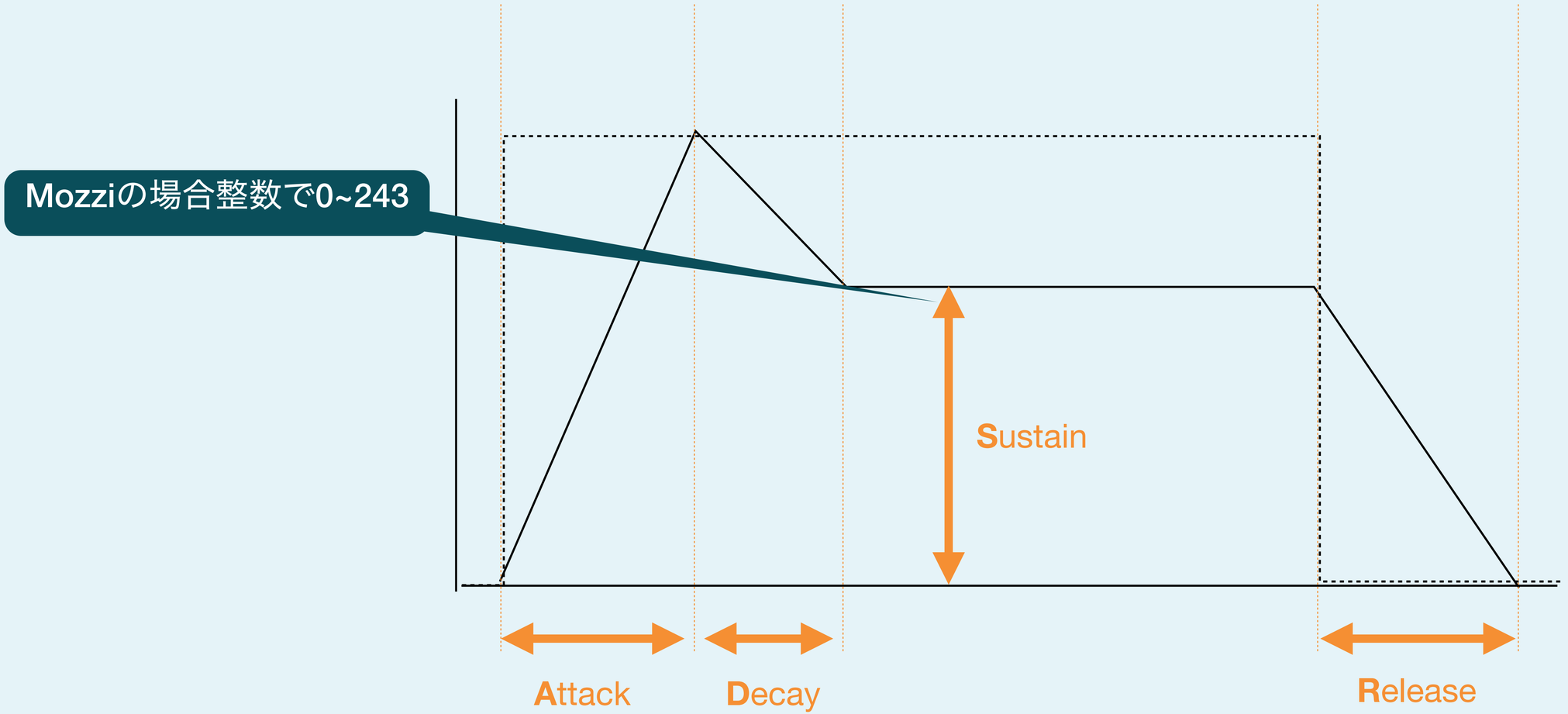
```ino
#include <MozziGuts.h>
#include <Oscil.h>
#include <tables/sin2048_int8.h>

Oscil <SIN2048_NUM_CELLS, AUDIO_RATE> aSin(SIN2048_DATA);

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

void updateControl(){

}

int updateAudio(){
  return aSin.next();
}

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

mozzi_sine.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



```ino
#include <MozziGuts.h>
#include <mozzi_midi.h>
#include <Oscil.h>

#include <ADSR.h>
#include <tables/sin8192_int8.h>

Oscil<8192, AUDIO_RATE> aOscil(SIN8192_DATA);

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

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

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

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

int updateAudio() {
  envelope.update();
  return envelope.next() * aOscil.next() / 255;
}

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。

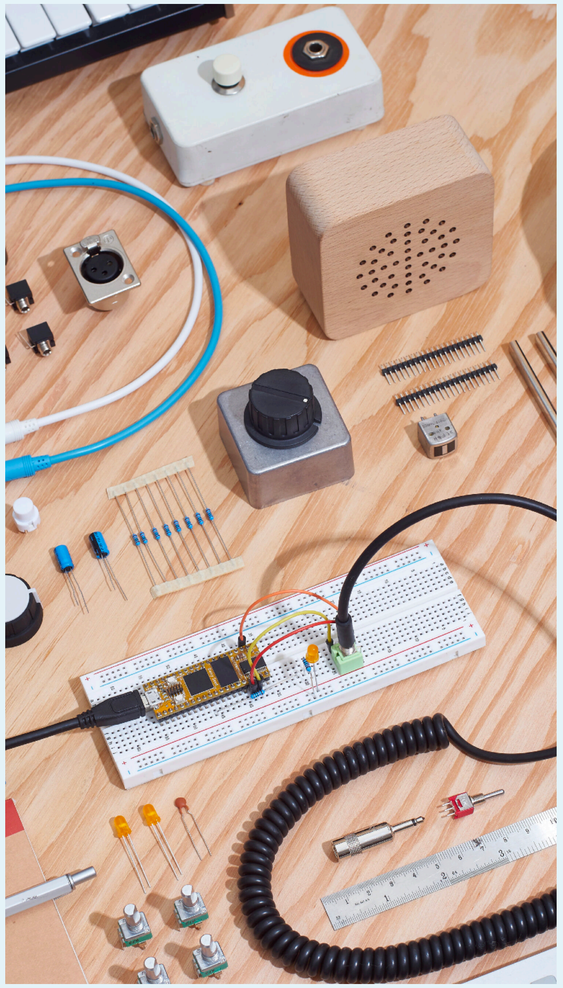
```
#include <MozziGuts.h>
#include <mozzi_midi.h>
#include <Oscil.h>
#include <EventDelay.h>
#include <ADSR.h>
#include <tables/sin8192_int8.h>

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
```

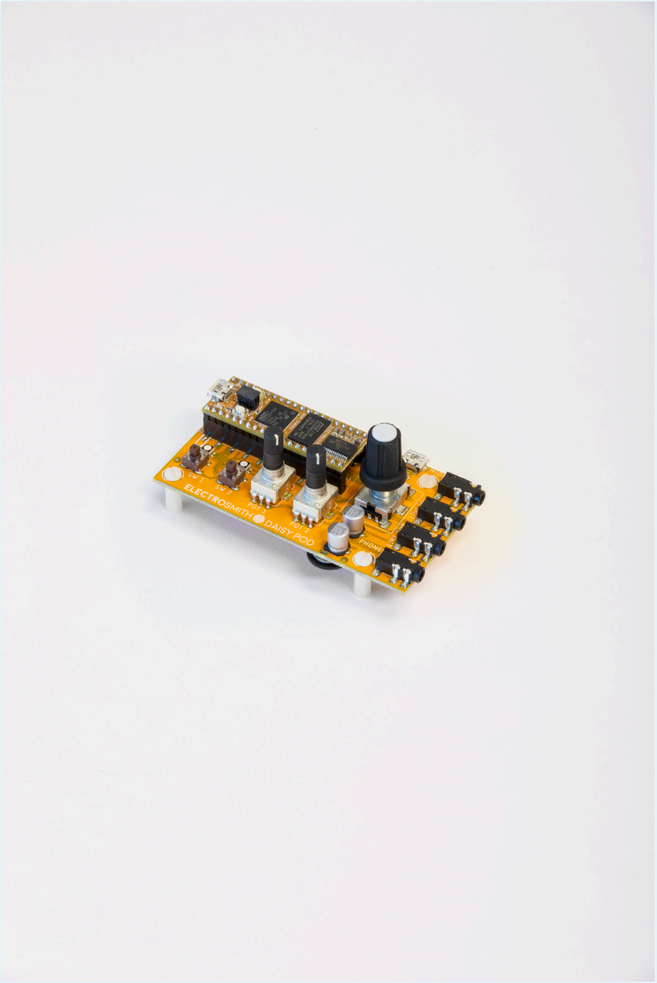
ひつじさんのサンプルより、Steve Reich「Piano Phase」の再現
http://sheep-me.me/2019/10/24/geidai_16/

```
#include <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);
Oscil <8192, AUDIO_RATE> aOscil2(SIN8192_DATA);
ADSR <AUDIO_RATE, AUDIO_RATE> envelope_A;
ADSR <AUDIO_RATE, 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();
  }
}

int updateAudio()
{
  envelope_A.update();
  envelope_B.update();
  int A = (envelope_A.next() * aOscil.next()) >> 8;
  int B = (envelope_B.next() * aOscil2.next()) >> 8;

  return (A + B) / 2;
}

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

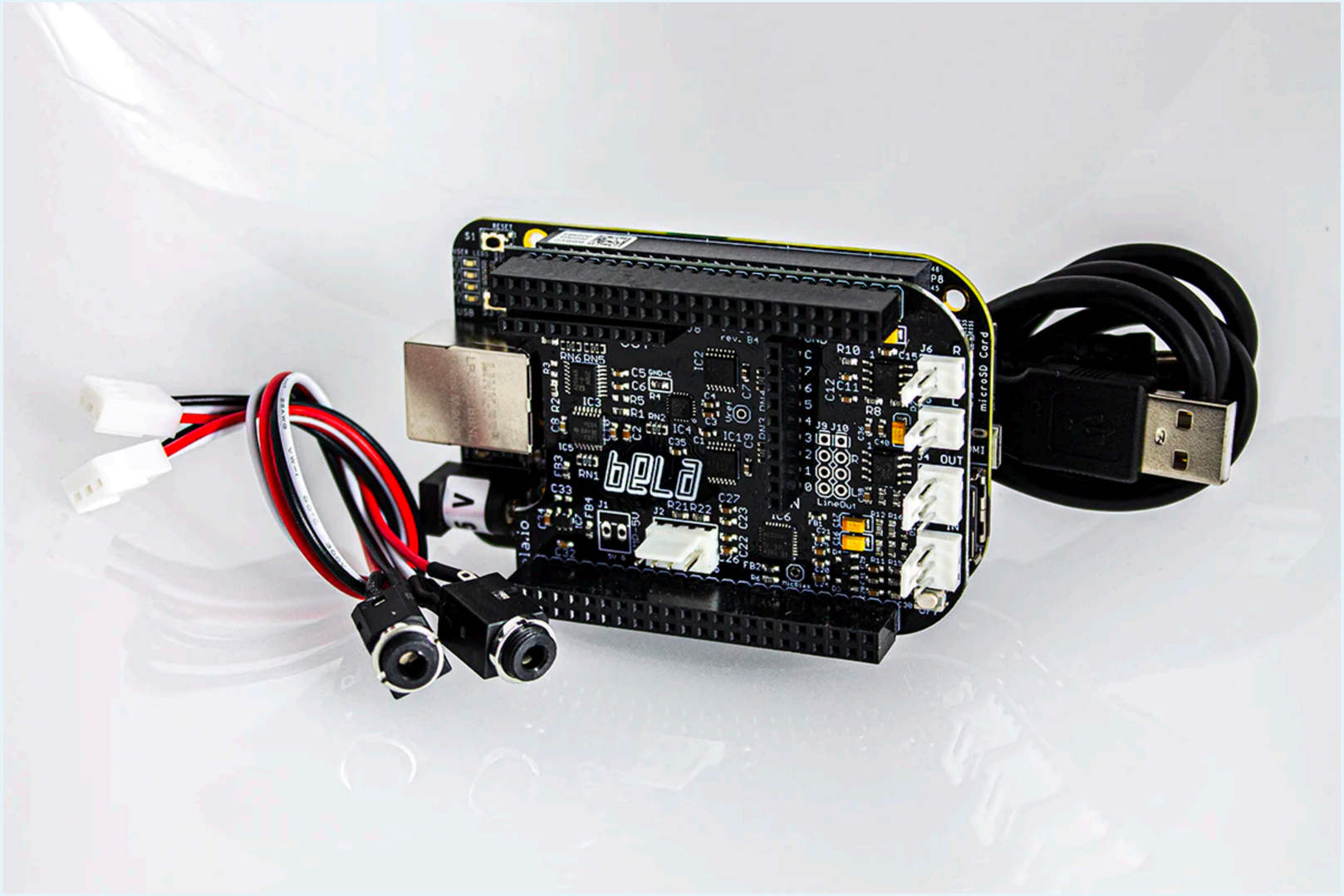
mozzi_reich.ino

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

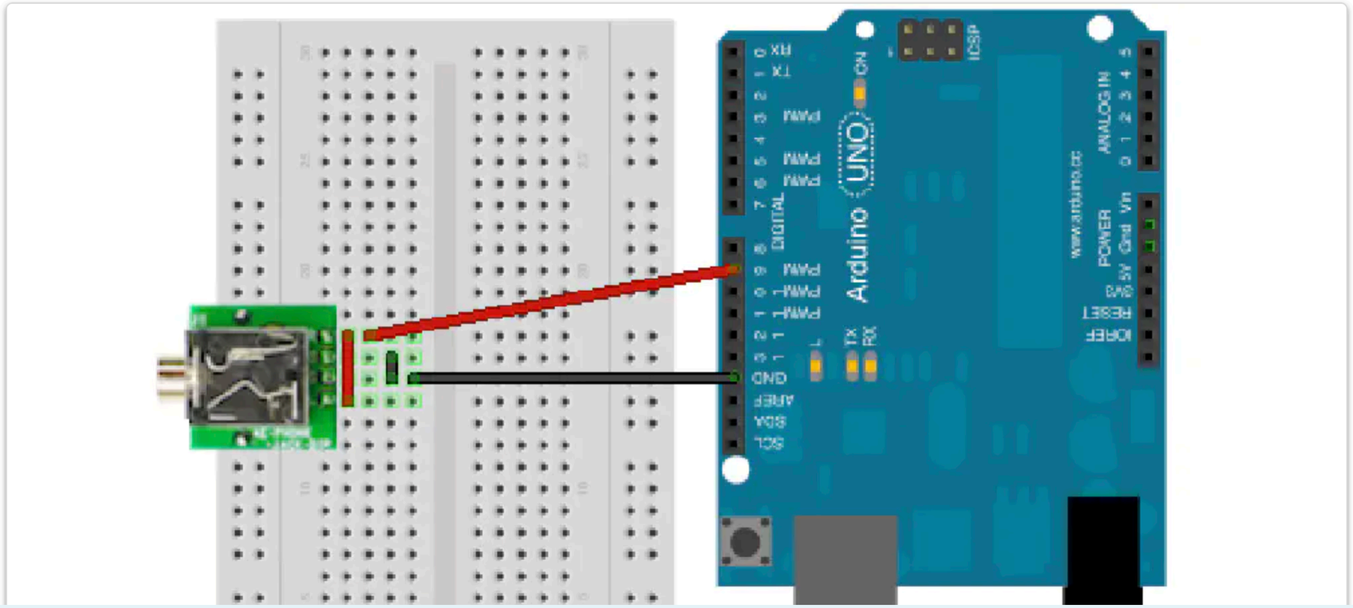
Arduino単体で音響合成を行う

- 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

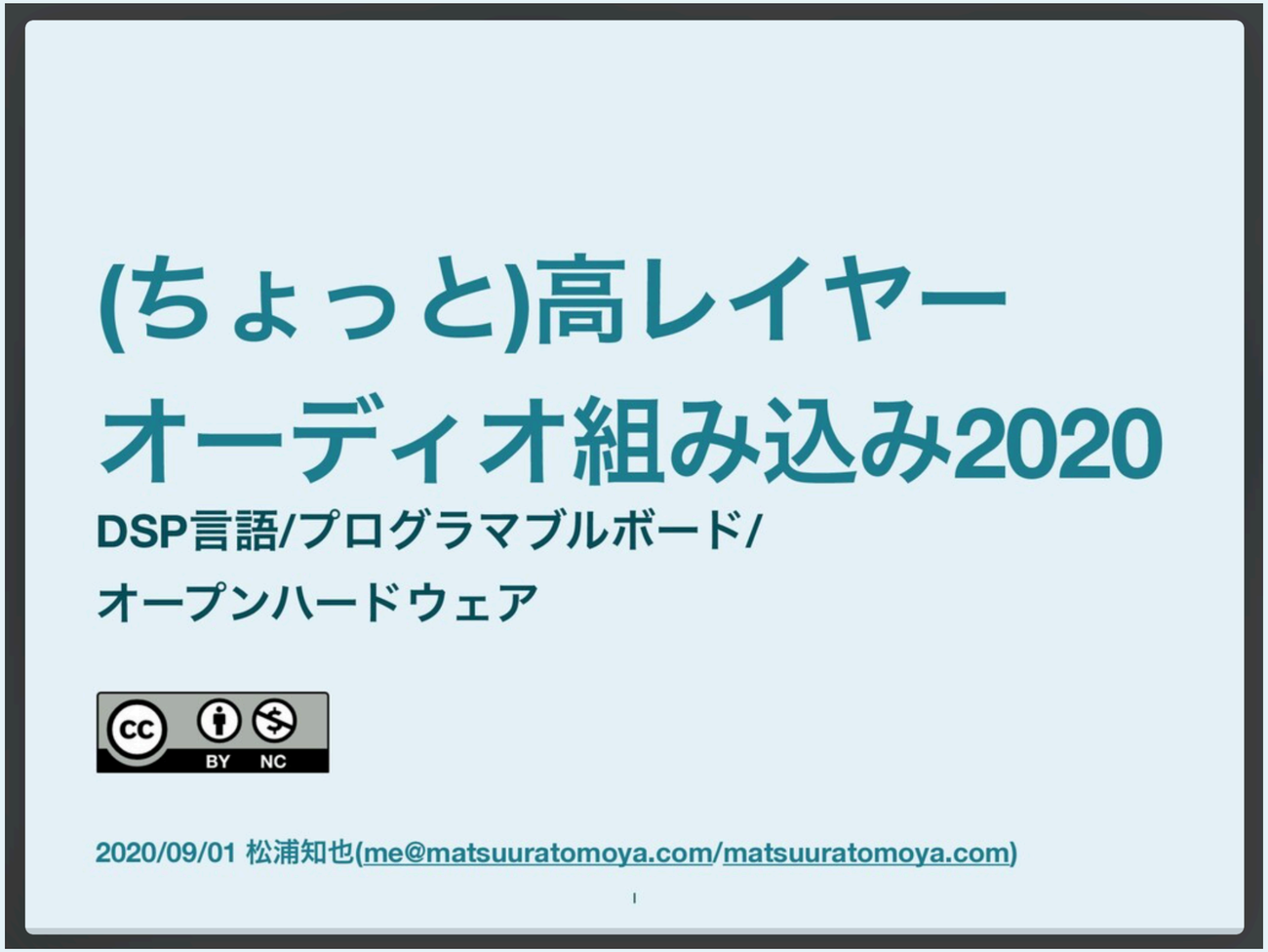
この授業について

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

残りの授業のスケジュール

- 6/16 課題制作展示相談会1
- 6/23 デジタルファブリケーション①(銅箔コイルでスピーカー制作)
- 6/30 課題制作展示相談会2
- 7/07 デジタルファブリケーション②(レーザーカッターで可変抵抗制作)
- 7/14 課題作品インストール作業
- 7/21 講評&振り返り&撤収

＋、7月に電子基板発注WSを助手の川田さんと
一緒にやるかもしれません

課題（再掲）

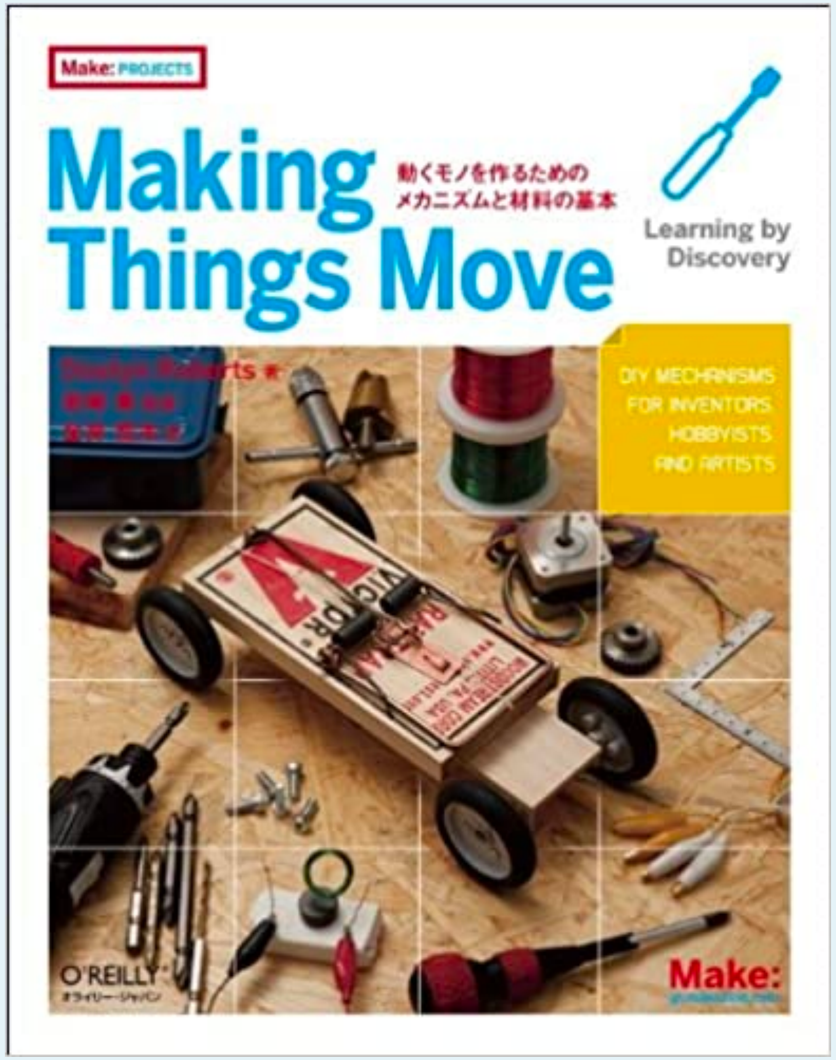
あなたにとっての「パーソナルコンピューター」を作れるようになること

- 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

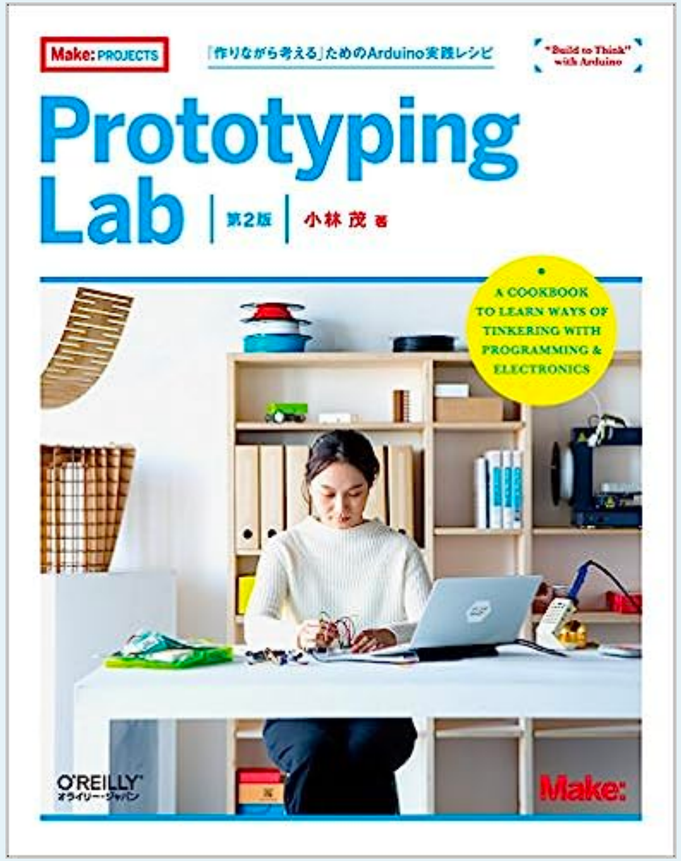
- 技術的な難易度よりも、なぜそれを作ったのか伝わるように

課題のためのルール

- 最終課題の製作に必要な材料等は原則自身で用意すること
  - Arduinoは授業終了時まで借りていっても良い。（無くさないように）
- ラボ内で半田ごて等の作業道具が必要な場合は松浦かラボスタッフに声をかけること。



Making Things Move —動くモノを作るためのメカニズムと材料の基本 (Make: PROJECTS) (2012), Dustyn Roberts (著), 岩崎 修 (監修), 金井 哲夫 (翻訳)



Prototyping Lab 第2版 — 「作りながら考える」ためのArduino実践レシピ (Make: PROJECTS),小林 茂 (2017)

Arduinoをはじめよう 第4版 (Make: PROJECTS) (2023),Massimo Banzi (著), Michael Shiloh (著), 船田 巧 (翻訳)