Drawdio のデジタル化

上田美穂　(指導教員: 粕川 正充)

1　はじめに

ピアノやリコーダーのように音階がはっきりとわかるもの、テルミンのように音階がはっきりしないもの、楽器にもさまざまなものが存在する。音楽を奏でる新しいインタフェースとして、描くことを音に変換し、直感的に音を鳴らすことができる Drawdio がある。アナログで動く Drawdio のデジタル化を試みる。

2　Drawdio とは

Drawdio とはマサチューセッツ工科大学 (MIT) の Jay Silver 氏が開発した電子楽器である [1]。例として鉛筆に搭載するものを用いる。Drawdio は、開いていた回路が導体を介して閉じることによりスピーカーから音が出力された。媒介対象の抵抗値によって音が変化する。Drawdio で音が鳴るしくみは以下の二段階からなる。

1. 鉛筆で線を描く。線の長さによって抵抗値が変化する。
2. 抵抗値によって発振する周波数が変化し、それをスピーカーから出力することで音が鳴る。

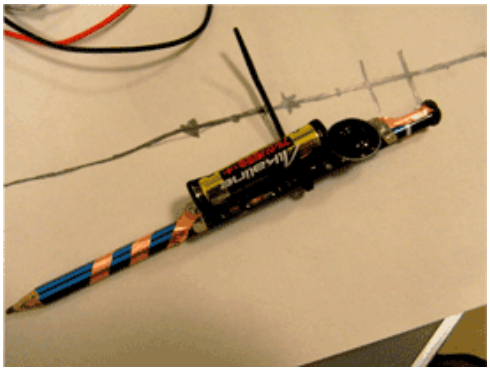

図 1：鉛筆に搭載した Drawdio

このままでも音は鳴るのだが、鳴らすだけでなく曲を演奏する場合には、次のような問題点がある。

- Drawdio はアナログデータを元に動いているため、同じ地点に鉛筆を押し付けても、押し付ける力の強弱で簡単に抵抗値が変化してしまい、同じ音を再現するのが難しい。
- 音が連続的に変化するため曖昧な音が無限に存在し、はっきりとした音階がない。

3　デジタル化への試み

3.1　デジタル化によるメリット

- 誤差範囲の抵抗値の揺れを平滑化することで音の再現性を高めることができる。
- 抵抗値の範囲を区切って音階を振り分けることで曖昧な音をなくすことができる。
- Drawdio は圧電スピーカから直接音を出すためブザーのような音しか鳴らないが、PC 上の音源を利用することで音色の変更ができる。
- 最低音、最高音の調整ができるので鳴らせる音の範囲を変更できる。
- 入力の変化に対応して音のオフや音色の変更など、挙動の制御ができる。

3.2　デジタル化に対応した音を鳴らす仕組み

Drawdio のデジタル化のため、音が鳴るしくみを以下のようにする。

1. **抵抗値の変化**　鉛筆で線を描く。線の長さによって抵抗値が変化する。これは Drawdio と同様。
2. **電圧への変換**　抵抗値の変化を計装アンプを用いて電圧の変化として出力する。
3. **文字列への変換**　出力される電圧値を Arduino を用いて数値を表す文字列に変換し、USB ポートを介して文字列を PC に送る。
4. **PC 上での処理**　文字列を PC のプログラムで処理し、音を鳴らす。

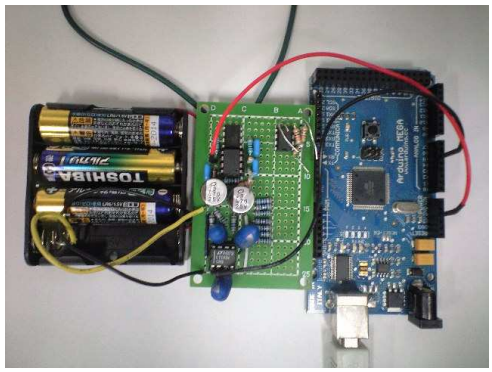

図 2：デジタル化 Drawdio の試作機

3.3　電圧への変換

抵抗値の変化を電圧の変化として出力するため、反転増幅回路を用いた。ここで主に使用する部品は以下のものである。

† **オペアンプ** LM358N [2]
　＋入力と-入力があり、その2 つの入力電位差を増幅して出力する。今回は計装アンプの補正用バッファとして用いる。

† **計装アンプ** INA128 [3]
　センサーの出力などノイズに弱い微弱な差動信号を増幅するために用いる。今回の回路では増幅度を 11 倍に設定している。

† **電圧コンバータ** LT1054CN8 [4]
　計装アンプには正負電源が必要である。今回は電圧コンバータを用いることで $+4.5V$ の単電源をチャージポンプし、$\pm 4.5V$ の正負電源をつくっている。

3.4　文字列への変換

Arduino は、単純な入出力を備えた基板と Processing/Wiring 言語を実装した開発環境から構成されるシステムである [5]。Arduino 上で以下のような処理を行った。

3.4.1　入力電圧の数値への変換

Arduino は AD コンバータを搭載しており、Arduino 基板上のアナログ入力端子に入力された 0 から 5 V の電圧を 0 から 1023 の数値として読み取ることができる。これを利用し、計装アンプの出力端子を Arduino のアナログ入力端子に接続、その入力電圧を読み取り数値に変換する。抵抗が大きくなるにつれて出力される数値も大きくなる。

3.4.2 数値の平滑化

入力電圧を数値に変換したものをそのまま用いるとノイズによる値の揺れや、誤差範囲のちょっとした値の変化がダイレクトにでてしまうので、値の平滑化を行う。今回は値を 1 つ読むごとに直近の 16 個の値も読み、その 17 個の値の中央値を出力する数値とした。

3.4.3 チェック記号の付加

エラー確認に使用するため、出力数値の末尾にチェック記号を付加する。出力数値の各桁の数字の合計値を ASCII コードの'@' 以降に対応するよう変換し、その文字をチェック記号とする。合計値が最小となる 0 なら'@'、1 なら'A'、最大となる 999 なら'[' がチェック記号である。

3.4.4 文字列の出力

「 入力電圧を変換した数値 + チェック記号」となる文字列が最終的な出力データである。この出力データを USB を通じてシリアル通信で PC に転送する。

3.5 PC 上での処理

Python で以下のような処理を行う約 100 行のプログラムを作成した。転送されたデータの数値が大きくなるほど音階が高くなるようにしている。また今回は MIDI 音源を利用しているので、プログラム上でどの楽器を使うか指定することで音色の変更が可能である。

3.5.1 転送されたデータのチェック

転送されたデータが音階変換に利用できるデータであるかどうかのチェックを行う。付加されているチェック記号が正しくない、空文字列であったり、チェック記号を除いたデータが数値を表す文字列でない場合はエラーと判断し、そのデータは無視して次のデータを読みにいくことにした。エラー訂正を行わない理由としては、訂正を行うとそのぶんタイムラグが生じ、リアルタイムな音の生成を阻害すると考えたからである。

3.5.2 値の音階への変換

生成する音のファイル形式として MIDI を使用する。MIDI では音階を表すのに低い音から順に 0 から 127 までのノートナンバーを呼ばれる番号が割り当てられている。ピアノ鍵盤上の中央の C が 60 にあたる。利用できるデータであると判断された場合、チェック記号を除いた数値を表す部分をノートナンバーに変換する。音名がないような曖昧な音にはノートナンバーがないので、ノートナンバーに変換することで曖昧な音が鳴ることはなくなる。計装アンプからの出力値は指数関数的に増加するため、AD 変換した値をそのまま用いてトーンナンバーに変換すると、抵抗値が大きくなるにつれて音階の変化が急に起こってしまう。それを防ぐため、値の対数関数をとり線形に増加するよう変換してから、欲しい音階の幅に合わせてノートナンバーへの変換式を調整する。今回は入力抵抗の範囲を最低 47Ω から最高 1MΩ と仮定して変換式を調整している。例えば、ノートナンバー 60 を中心に 2 オクターブ鳴らしたいのであれば、47Ω のときのノートナンバーを 48、1MΩ のときのノートナンバーを 72 になるように変換式を調整すればよい。

3.5.3 MIDI デバイスを通じての出力

ALSA(Advanced Linux Sound Architecture) とは Linux 用サウンドドライバである [6]。変換してもとめられたノートナンバーの音階を鳴らす ALSA イベントを、出力したいデバイスの MIDI ポートへ転送する。今回 ALSA MIDI デバイスとして TiMidity++ を利用している。TiMidity++ は MIDI ファイルをリアルタイムに出力することができるフリーソフトウェアである [7]。イベントに基づいた音階がスピーカーから出力される。

3.5.4 MIDI を選択した理由

python には WAV サウンドフォーマットへのインタフェースモジュールが標準ライブラリとしてあるため、最初は音の生成に WAV を用いることを検討していた。PC に転送されたデータを周波数に変換し、その音の WAV ファイルを出力するという方法である。この方法を実装してみたところ、WAV ファイルを 1 つ作成するごとに行われる open-close 処理に時間がかかって音が鳴るまでにタイムラグが生じてしまい、抵抗値の変化に対してリアルタイムに音を鳴らすことは不可能だった。WAV や mp3 は演奏時間をフォーマットに規定する必要があり、何秒鳴るかが動的に変化する今回のような場合にはこれらのファイル形式は不向きである。MIDI のデータフォーマットには実時間演奏があり、演奏時間を規定する必要がない。音をオンにする命令が送信されればその音はオフにする命令が送信されるまでずっと鳴り続ける。こちらのほうが動的な変化に対応できると考え、MIDI を選択した。

4 まとめと今後の課題

Drawdio のデジタル化を提案、新しく回路の設計、プログラムの作成を行い試作した。ノイズの除去や、計装アンプからの出力値が指数関数的に増加してしまうことについてはソフトで対処するにも限界がある。また、現在は音を鳴らせる抵抗値が最大で 1MΩ 程度に限られているため、例えば入力に鉛筆で描いた線を用いた場合、ある一定の長さ以上は使用することができず、鳴らせる音の範囲を広くしたときに 1 音に割り当てられる線の幅が狭くなるため演奏が難しくなると考えられる。これらを解決するために、計装アンプの増幅率の設定やオペアンプの使用が正しいかどうか、回路設計の見直しをしたい。さらにデジタル化 Drawdio 本体を自由に使用できる範囲を広げるため、またノイズの発生源が PC であることも考えられるので、Arduino を USB 通信から bluetooth 通信に切り替えることも考えられる。

参考文献

[1] Drawdio: A Pencil that Lets You Draw Music,http://drawdio.com/

[2] オペアンプ LM358N データシート,http://www.fairchildsemi.com/ds/LM/LM358.pdf

[3] 計装アンプ INA128 データシート,http://www.ti.com/jp/lit/gpn/ina128

[4] 電圧コンバータ LT1054 データシート,http://cds.linear.com/docs/Datasheet/1054lff.pdf

[5] Arduino,http://www.arduino.cc/

[6] Advanced Linux Sound Architecture (ALSA) project homepage,http://www.alsa-project.org/

[7] TiMidity++,http://timidity.sourceforge.net/

[8] 藤村安志,『 電気・電子回路入』, 誠文堂新光社,1991

[9] チャールズ・プラット ,『 Make:Electronics 作ってわかる電気と電子回路の基礎』, 鴨澤眞夫 訳, オライリー・ジャパン,2010