MIDI 楽器製作班 中間活動報告書

立命館コンピュータクラブ 2017 年度前期プロジェクト活動

青木 雅典 *
吉田 享平 †
清水 麻也人 ‡

2017 年 8 月 6 日

概要

本活動では、ユーザーの直感的な入力を楽曲制作環境に反映させるため、鍵盤や弦といった既存の入力装置とは異なる新たなデバイスを開発することを目標とする。今回は、この目標を達成するための基礎技術となる「センシング」および「MIDI 信号の制御」を行い、ユーザー入力を MIDI データに変換して演奏に反映させるシステムを製作した。

* 理工学部電子情報工学科 1 回生
† 情報理工学部情報理工学科 1 回生
‡ 情報理工学部情報理工学科 1 回生

目次

1 はじめに 3

2 ハードウェアの製作 3

2.1 概要 3

2.2 動作の検出 3

2.3 BPM の算出 4

3 ソフトウェアの製作 5

3.1 概要 5

3.2 使用言語・API 5

3.3 仕様 5

3.4 Win32API による MIDI ファイルの制御 5

3.5 C#によるシリアルポート通信 5

3.6 Win32API と C#との齟齬 5

3.7 フローチャート 6

4 おわりに 6

1　はじめに

文責:青木 雅典

MIDI とは、Musical Instrument Digital Interface の略で、電子楽器の演奏データを転送するための世界共通規格である。MIDI のプロトコルを活用した楽器、すなわち MIDI 楽器を製作するにあたって重要となる基礎技術は、「ユーザー入力の取得」と「MIDI データの制御」である。そのため、今回の活動では、指揮棒を模したデバイスを製作し、その入力を MIDI のテンポ情報に反映させることを目標とした。

2　ハードウェアの製作

文責:青木 雅典

2.1　概要

ここでは、ユーザーの動作からテンポを算出するハードウェアを製作する。ユーザーは指揮棒[*1]を模したデバイスを振ることで BPM[*2]を表現する。そのため、このデバイスでは、「動作の検出」と「BPM の算出」の 2 つを行う必要がある。

2.2　動作の検出

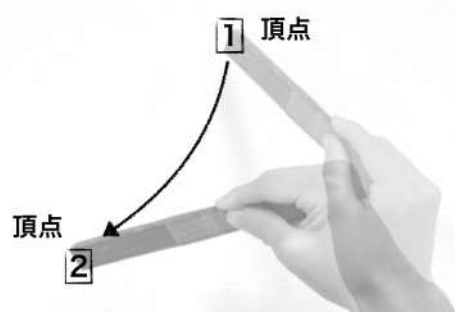

図 2.1　指揮棒が頂点間を移動する軌跡

指揮棒で曲の拍子やテンポを表現する際、指揮棒を持つ指揮者は一般に、各拍子で頂点を描くように棒を動かす。つまり、指揮棒の動きからテンポ情報を読み取るには、棒が頂点に到達するタイミングを検出し、頂点間を移動する時間を測定すればよい。また、頂点に到達する直前、棒は減速するため加速度が発生する。そこで、今回は加速度センサーを利用することでユーザーの動作を検出しようと考えた。
センサーには 9 軸センサーである「MPU-9150」を採用し、これを読み取るために「mbed LPC1768」と呼ばれるマイクロコンピュータ（以下マイコン）を利用した。これらは、現在流通する多数の部品の中でも情報が豊富であり、容易に扱えるからである。

[*1] 音楽で、指揮者が楽曲のリズム、拍子、テンポ、強弱、表情などを演奏者に知らせるために指揮台で振る棒（日本国語大辞典）
[*2] beats per minute

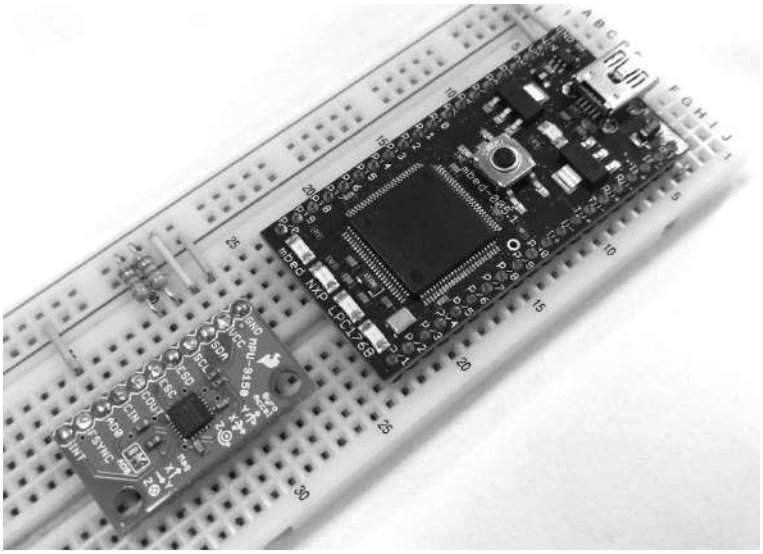

図 2.2　左 : MPU-9150 右 : mbed LPC1768

MPU-9150 と mbed の間は、I^2C と呼ばれる 2 線式通信で接続する。MPU-9150 は加速度、角速度、磁気の 3 種類の情報がそれぞれ 3 軸で検出可能なセンサーであり、これらを活用することで実空間上での位置や変化を詳細に捉えることができる。しかし今回は、複雑な演算や煩雑なプログラムを避け、単純なアルゴリズムでセンシングを行うため、加速度の情報のみを利用した。また、平静時でも重力加速度の影響を受けるため、通常使用時に鉛直方向となる Z 軸のデータは扱わないこととした。

2.3　BPM の算出

テンポ情報、すなわち BPM を算出するため、各頂点間の指揮棒の移動時間を計測する。
はじめに、頂点検出を行うため、一定間隔で加速度データを取得する。この時、閾値よりも大きな加速度を観測すれば頂点に到達したとする。次に、頂点間移動の時間を計測するため、mbed のタイマーをスタートさせる。最後に、再度一定間隔で加速度を取得し、閾値を超えたところでタイマーを止める。これを繰り返すことによって、加速度が閾値を超えた点を頂点とし、その間の時間を拍子間とした。
また、計測した拍子間の時間を BPM に変換するため、以下の計算を行なった。ただし、mbed による時間計測の単位はミリ秒である。

$$\mathrm{BPM} = \frac{60 \times 10^3}{\mathrm{MeasuredTime}}$$

算出した BPM は、UART[*3]を介してコンピュータへ送信した。

[*3] Universal Asynchronous Receiver/Transmitter の略。歩調同期方式によるシリアル通信である。

3　ソフトウェアの製作

文責:吉田 享平

3.1　概要

ここでは、ハードウェアから送られてきた BPM をもとに既存の MIDI ファイルを再生し，その上テンポをリアルタイムで変更するプログラムを実装した。

3.2　使用言語・API

以下に本アプリケーションで使用したプログラミング言語と API を示す。

- C#
- Win32API

3.3　仕様

以下に本アプリケーションの仕様を示す。

- コンソールアプリケーション
- .NET Framework 4.5.2 以上必須
- MIDI ファイルのファイルパスはソースコードの中に埋め込み
- MIDI ファイル再生時に若干の読み込み時間が生じる

3.4　Win32API による MIDI ファイルの制御

今回、MIDI ファイルの読み込みおよび再生を行うために Win32API を使用した。実装では winmm.dll をインポートし、mciSendString という関数を使用した。また、デバッグを補助する目的で mciGetErrorString という関数も使用した。mciSendString は 4 つの引数をもつ。しかし、このアプリケーションでは第 1 引数のみを使用している。これは、コマンド文字列を引数とし、MSDN で参照することができる。このコマンド文字列によって、MIDI ファイルの読み込み、再生、テンポの更新、停止を実現した。

3.5　C#によるシリアルポート通信

ハードウェアとの通信を行うために、C#に標準で用意されている SerialPort クラスを用いて通信を確立させた。データを受信するたびに BPM を受け取り、更新している。

3.6　Win32API と C#との齟齬

Win32API と C#は相性が悪く、単純な実装ではうまく行かなかった。原因は GUI（フォーム画面）プログラムとシリアルポートとの通信のプログラム（イベントハンドラ）が別スレッドで動作していたことだと考えられる。よって、これを回避するためにコンソールアプリケーションに変更した。

3.7　フローチャート

以下にアプリケーションの簡単なフローチャートを示す。

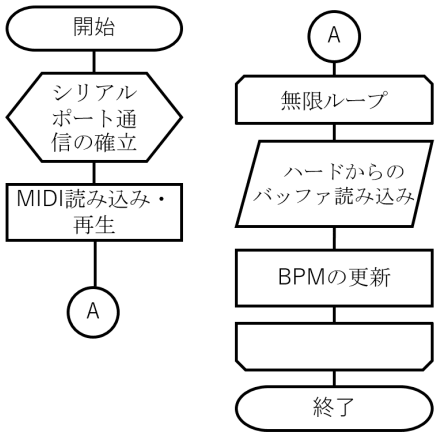

図 3.1　フローチャート

4　おわりに

文責:青木 雅典

今回の活動では、加速度センサーを用いてユーザーの動作を検出し、再生中の MIDI データにテンポ情報として反映することができた。しかし、実際の MIDI メッセージはテンポ以外に数多くの情報が含まれており、楽器開発を行う上ではより多くのメッセージを処理する必要がある。今後は、既存の MIDI ファイル再生だけでなく、MIDI 送信機としてメッセージを送ることにより、DAW などの作曲環境にもリアルタイムで利用できるようにしたい。

また、製作したセンシングデバイスは、精度が荒く、安定した BPM を取得することができなかった。今後は、センサーデータを正確に捉えることで、センシングデバイスの改善を目指したい。それに加え、加速度だけでなく他のセンサーと組み合わせることで、テンポ以外の MIDI 信号についてもユーザー入力を反映できるデバイスを開発していきたい。

参考文献

[1] 音楽電子事業協会, MIDI 1.0 規格書 (1998),〔http://amei.or.jp/midistandardcommittee/MIDI1.0.pdf〕(最終閲覧日：2017 年 8 月 6 日)

[2] Microsoft, MSDN MCI Reference,〔https://msdn.microsoft.com/ja-jp/library/windows/desktop/dd743458.aspx〕(最終閲覧日：2017 年 8 月 6 日)