MicroPython によるプログラミング教育の可能性

箕原 辰夫[*1]
Email: *minohara@cuc.ac.jp*

*1: 千葉商科大学政策情報学部政策情報学科

◎Key Words MicroPython, Python, micro:bit

1. はじめに

従来、マイクロコントローラでハードウェアを制御する授業を展開するのには、Arduino(1)が用いられてきました。そのプログラミング言語は、C 言語ベースの独自の言語が使われてきました。それに対して、最近いくつかのマイクロコントローラでは MicroPython(2)を標準のインタープリタ言語として稼働させることができるようになってきました。MicroPython は標準の Python(3)の Python 3 版のサブセットになっているため、通常のプログラミング言語として Python を学んだ学生は、そのまますぐにマイクロコントローラを制御させることができるようになります。プログラミングの初修言語として Python が多く用いられるようになってきましたので、高校や大学において、Python でマイクロコントローラを駆動させて周辺機器などの制御を扱う授業を展開することが可能となってきました。

Python でのプログラミング教育さえ受けてしまえば、小さなマイクロコントローラから大規模なスーパーコンピュータまでのプログラムを作成することができます。昨年はスーパーコンピュータ向けの数値計算について担当授業からの報告を致しましたが、今年はゼミナールの 3 年次と 4 年次の 2 名の学生に micro:bit(4)マイクロコントローラを与えて、モータやセンサなどと組み合わせて展開した補助授業の報告をします。

MicroPython を扱える CPU ボードとしては、Pyboard(5)などの ARM プロセッサベースのボードの他に、micro:bit などの小中高向けのボードでも稼働させることができます。また、Pyduino(6)と呼ばれる Arduino 風のボードでも MicroPython を稼働させる環境も発表されてきました。今回の報告では、昨年度の PC カンファレンスで紹介された micro:bit の CPU ボードを使って MicroPython を稼働させた授業を展開するときに遭遇した問題点も、Arduino で授業を展開したときと比較して記していきます。

2. micro:bit の仕様と問題点

2.1 CPU ボードの仕様と組込みセンサー

CPU ボードには、nRF51 という型番のついた ARM Cortex-M0 の 32bit のマイクロプロセッサが載せられています。動作周波数は、16MHz であり、当然ながら高速計算には向いていません。インターフェースは、Micro USB のインターフェースがあり、ここからもボードを駆動する電源を取ることができます。それ以外に、単体動作させるための、3V バッテリーへのコネクタがあります。外部ポートには、3.3V の直流電源を供給するための端子があり、実際に動作させてみて、周辺装置を駆動する際には、Micro USB とバッテリーの両方から電源を貰っている方が、外部への電源供給の電力も高いことがわかりました。また、Bluetooth 用の無線通信用のチップと Nordic Gazell 用の無線通信チップもインタフェースとして内蔵されています。ボードの組込みセンサーは、Freescale の 3 軸加速度センサー、方位計、温度計、光センサーがあります。また、簡易の入出力として、2 つのボタンや 25 個の LED マトリックスが用意されています。外部への入出力ポートとしては、GPIO およびデジタル入出力できる 19 ピンのポートがあり、用途によって、その中のいくつかを PWM 制御や、アナログ入力、I2C インターフェース、SPI 通信インターフェースなどに割り当てることができます(7)。

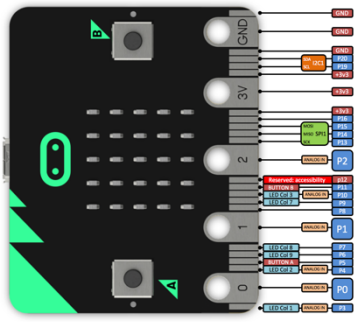

図 1 入出力ポート(8)

2.2 外部機器との接続

外部機器の接続は、上記のポートの大きいもの (0～2 番のポートと 3V および GND) を鰐口クリップで挟む事例

などがホームページで紹介されていますが、特定の用途（スピーカーを鳴らす）以外では、あまり良い接続とは思えません。そのため、ブレッドボードと接続するためのコネクタがいくつかのメーカーから出ています。今回は、switch science 社の「micro:bit 用プロトタイピングセット」を用いました。これだけで、micro:bit と同額の出費を強いられるため、授業で外部機器を展開するときの予算を計上するときは余裕を持たせた方が良いでしょう。今回、0 番ピンを用いて、デジタル出力としてモーター駆動を PWM のプログラムを Python で組んで行ないましたが、0 番ピンは GPIO のアナログ出力も可能です。しかしながら、アナログ出力として使うと、電力が足りないようで、トランジスタで増幅させても、モーターを駆動させることができませんでした。また、複数のモーターや、複数のサーボモーターを駆動するためには、同じ switch science 社から出ているモータードライバ用のボードやサーボモータードライバ用のボードを利用した方が良いでしょう。これらのボードも micro:bit と同額の出費を強いられます。

また、switch science 社から出ている micro:bit 用のゲームパッド／コントローラ用キットを用いて、簡単なジョイスティック対応の制御を行なおうとしましたが、このようなキットは、外部ポート用のコネクタの 80 ピンを半田付けしなければならず、一般の授業の教材として準備するのは大変だということが判明しました。もちろん、学生と半田付けをワイワイとするのは、楽しいのですが、それでも個数が多くなれば、楽しさどころではなくなりそうです。

図２　80 ピンの半田付けに臨む学生

2.3　外部機器接続について、Arduino との比較

一番問題になるのは、外部電源供給だと思われます。Raspberry Pi も 3.3V ベースですが、micro:bit や Pyboard も 3.3V の電源供給しかできません。Arduino では、5V および 3.3V の両電圧による電源供給があるので、外部機器の駆動での電源問題が発生しません。しかし、3.3V しか持っていないとなると、5V 対応の機器は駆動できない可能性が大きく、正常に動作させるためには、電圧をレベル変換する回路を必要とします。5V 用の機器でも 3V で駆動できるという報告もあるのですが、micro:bit は外部機器を 3V 用のものだけに制限するという点においては、Arduino の方に軍杯があがります。また、Arduino や Pyboard では外部ポート用のピン（メス）が最初からついていることを考えれば、そのピンを持つ開発用のセットを購入しなければならない必要性を考えても、外部機器の制御をメインに考えている場合は、別の選択肢を持つべきでしょう。加えて、内部的に PWM を用いているアナログ出力があるのですが、接続の仕方が悪かったのかも知れませんが、外部機器を駆動させるに必要な電力を得られなかったことからも、信頼性が感じられませんでした。デジタル出力で、プログラムで PWM を行なった方が良いようです。

2.4　micro:bit の開発環境

Arduino のような専用のエディタという形ではなくて、基本的にはブラウザで稼働する Python エディタを利用します。ここから、ファイルに保存したり、micro:bit 転送用の Hex ファイルを作り出したりします。転送は、micro:bit を USB 接続すると、外部のドライブとしてマウントされますので、そのドライブに Hex ファイルをコピーすることによって、プログラムのダウンロードが行なわれます。コピーが終わると、プログラムが実行開始されます。ブラウザで稼働するエディタの他に、JavaScript ベースのタイルプログラミングの環境では、MakeCode と呼ばれるシミュレータ付きのエディタがあり、組込みのセンサーや LED マトリックス、ボタンなどを、予めエディタ上で動作を確かめてから、ダウンロードさせることができます。Python エディタでは、シミュレータがついていないのが大きな問題点となっています。Pyboard の場合は、コンソールから、ボード内部の MicroPython のインタープリタを直接起動できますので、Python エディタについても、今後、そのような機能が必要ではないでしょうか。

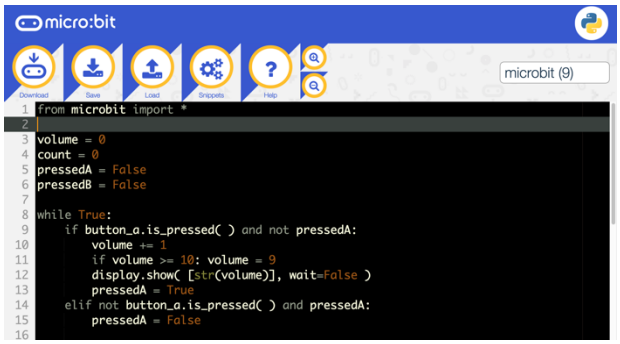

図３　Web ブラウザ上の Python エディタ[10]

3. MicroPython による micro:bit のプログラムの記述

3.1 ライブラリを利用した内部機器の制御

最初に、LED マトリックスについて記述しますが、display.scroll( )という関数があり、これは引数の英数字の文字列を電光掲示板のようにLEDマトリックスに表示させるのですが、スクロールが流れてしまい、何が表示されているのかはまったくわかりません。それに対して、display.show( )という関数は、引数に文字列（1文字）や画像のマトリックスを取りますが、定常的に表示をしてくれますので、こちらを使う方が多かったです。また、display.show( )関数では、wait というオプション引数があり、これを False にすると、表示でプログラムの進行がブロックされません。実時間の事例を扱う場合は、この機能を利用する必要があります。加えて、LED の明るさを調節することもできます。MicroPython には用意されていませんが、JavaScript では、lightLevel( )関数を呼べば、LED マトリックスを用いて環境光の明るさを計ることも可能です。

２つのボタンについては、基本的に、押されているかどうかのチェックする button.is_pressed( )関数が用意されていますが、押し始めのときと離されたときをチェックするためには論理変数が別に必要となります。イベント駆動ではないので、イベントループをプログラムで作って、ポーリングする必要があります。イベントループでは、sleep( )関数を使って、引数のマイクロ秒を指定して、別のスレッドに制御を渡す機会を与える必要があります。

内部の組込み機器に関してのクラスライブラリは、良く整理されています。たとえば、0番ピンと GND に小型のスピーカーを繋げば、music.play( )関数を呼ぶことによって、音楽（単一音源）を鳴らすことができますし、speech.say( )関数を用いて、英単語の並びの音声出力も可能です。音声は、0番および1番ピンのアナログ出力を用いています。

３軸加速度については、accelerometor.get_values( )関数を用いると加速度のタプルを得ることができます。また、シェイクなどいくつかの動きをジェスチャーで得ることができます。方位計は、compass.calibrate( )関数を呼んで、キャリブレーションした後に、compass.heading( )関数を呼んで、北の角度を得ることができます。温度については、temperature( )関数で micro:bit 自体の温度を摂氏で得ることができます。

3.2 外部機器の制御・通信

外部の入出力ポートは、それぞれのピンに番号がついていて、そのピンをデジタル入出力か、アナログ入出力に使うのかの指定ができます。

pin 番号.read_digital( )…デジタル入力
pin 番号.write_digital(1 か 0)…デジタル出力
pin 番号.read_analog( )…0〜1023(3.3V)アナログ入力
pin 番号.write_analog(0〜1023)…PWM でアナログ出力

外部との有線での通信あるいは入出力には、SPI, UART, I2C のインタフェース用のモジュールが用意されていて、そのモジュールの関数を呼べば、これらのプロトコルを介しての有線でのデータの送受信が可能となります。無線通信については、Bluetooth については、MicroPython ではメモリが少ないので、サポートされていません。通常の無線通信については、radio モジュールが対応しています。

以下のプログラムは、デジタル出力を用いて、自前で PWM（Pulse Width Modulation：パルス幅変調）制御をして、トランジスタ回路を通して、小型のモータを制御したときのプログラムです。ボタン A を押すと、回転速度が増し、ボタン B を押すと回転速度が落ちます。0〜9 の間で速度を調節できるのですが、PWM で 50%以上にしないと、モータが回り始めないので、段階 1 の場合で、50%以上の比率でパルス幅が出力されるように調整しています。

```
from microbit import *

volume = 0
count = 0
pressedA = False
pressedB = False

while True:
    if button_a.is_pressed( ) and not pressedA:
        volume += 1
        if volume >= 10: volume = 9
        display.show( [str(volume)], wait=False )
        pressedA = True
    elif not button_a.is_pressed( ) and pressedA:
        pressedA = False

    if button_b.is_pressed( ) and not pressedB:
        volume -= 1
```

```
        if volume < 0: volume = 0
        display.show( [str(volume)], wait=False )
        pressedB = True
    elif not button_b.is_pressed( ) and pressedB:
        pressedB = False

    if volume == 0:
        pin0.write_digital( 0 )
    elif count % 20 <= volume+10:
        pin0.write_digital( 1 )
    else:
        pin0.write_digital( 0 )
    count = (count+1)%1000
    sleep(10)
```

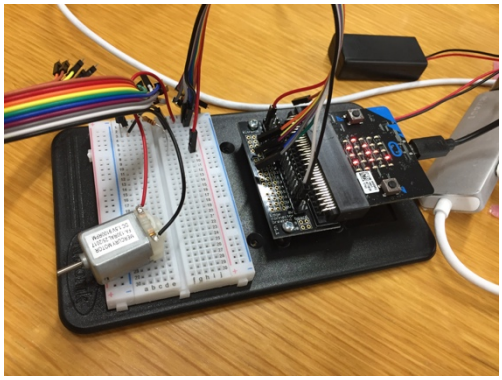

図４　モータ制御時のブレッドボードの配線

3.3　プログラミング環境を Arduino と比較して

micro:bit では、内部で MicroPython のインタープリタが動いていて、文法エラーや実行時例外について、それらが発生すると、まったく読めないスクロールメッセージとして LED マトリックスにエラーを表示してくれるのが興味深いのですが、これではエラーが起こっていることぐらいしかわかりません。Arduino は、C 言語をベースにしていますので、コンパイラに通す必要があります。そのため、事前に簡単な文法エラーがわかる仕組みになっています。micro:bit で MicroPython を使う際に、これと同様な支援を行なうためには、JavaScript に用意されているシミュレータが必要となり、シミュレータの実装が望まれる次第です。あるいは、Pyboard のように、ターミナルから直接内部の MicroPython インタープリタにアクセスできるようにする機能は、既に micro:bit にも備わっているのですが、これを Web ブラウザベースの Python エディタにも組み込むべきでしょう。

micro:bit では、Python や JavaScript がオブジェクト指向プログラミング言語なので、ライブラリが非常に充実しているのが特徴です。これは、Arduino にはなく、手軽にライブラリにあるモジュールの関数を利用して、音楽や音声などを試すことができるのは、プログラミング教育には欠かせないものだと思われます。また、LED マトリックスに何らかの情報を表示できるのも、状況のフィードバックを返すのに適しています。その点では、学習者が非常に取りつきやすい形で用意されているので、Arduino よりもお薦めできます。また、micro:bit の Web ブラウザベースでの開発環境は、ホストマシンのプラットフォームを選びませんし、ソフトウェアをインストールしたりアップデートしたりする必要がないので、教材準備に手間を取られることがなく、優れています。

4.　おわりに

micro:bit を用いた MicroPython による周辺装置のコントロールについては、micro:bit に組み込まれているセンサーやボタンからの測定値を入力として貰う作品や、組込みの LED マトリックスを表示する、あるいは音声や音楽を利用するような楽しいプログラムを、授業として作成する分には、Arduino と比べて micro:bit は非常に扱いやすい仕様になっています。反対に、外部のモーターやキャラクタ LCD などを扱う場合は、Arduino の方が、周辺機器の制御に特化されているので、一日の長があります。結論としては、中学校や高校において、マイクロコンピュータを MicroPython で動かす体験をさせる分には、micro:bit は手頃な教材として用いることができるでしょう。大学に入って、更にブレッドボードを使って、試作の回路で外部の機器を制御するときには、Arduino に移行するか、MicroPython を使い続けるのであれば、Pyboard を利用するべきなのではないかと考えられます。

参考文献

(1) Arduino: “Arduino” , https://www.arduino.cc/ (2018).
(2) Micropython.org: “MicroPython”, http://micropython.org/ (2018).
(3) python.org: “Python”, https://www.python.org/ (2018).
(4) micro:bit Educational Foundation: “micro:bit”, http://microbit.org/ja/ (2018).
(5) Micropython.org: “Pyboard”, http://docs.micropython.org/en/latest/pyboard/ (2018).
(6) Nitin: “Pyduino - Arduino the python way”, https://www.kickstarter.com/projects/707054749/pyduino-arduino-the-python-way (2016).
(7) micro:bit: developer community, “Hardware”, http://tech.microbit.org/hardware/ (2018).
(8) micro:bit, “BBC micro:bit Pins” , http://microbit.org/ja/guide/hardware/pins/ (2018).
(9) switch science, “BBC micro:bit 用プロトタイピングセット” https://www.switch-science.com/catalog/3179/ (2018).
(10) micro:bit, “Python Editor”, http://python.microbit.org/v/1 (2018).