Pyboard を用いた Python のプログラミング教育について

西塚想一[*1]・箕原辰夫[*2]

*1: 千葉商科大学大学院政策情報学研究科
*2: 千葉商科大学政策情報学部

◎Key Words Python, Pyboard, 組込みコンピュータ, プログラミング

1. はじめに

学部 4 年次生のときに、春学期に MicroPython(1) のプログラミングの正課外の特別授業を受け、そこでは micro:bit(2) を用いました。秋学期に Pyboard(3)を用いて MicroPython を再度プログラミングする特別授業を受けました。なお、学部 3 年次のときに、標準の Python 3 の授業を受けており、Python でのプログラミングには既に親しんでいました。

本報告では、Pyboard を用いたときに作成したプログラムの概要や組込みボードを用いた MicroPython を利用した教育についての方法について考察した結果を報告したいと思います。秋学期の特別授業においては、LCD やサーボモータの制御や、みちびき衛星を利用した GPS モジュールを使った緯度・経度計測なども行ないました。周辺機器の制御については、MicroPython で直接記述するのが優れている点も紹介したいと思います。

2. Pyboard の仕様と開発環境

2.1 Pyboard の仕様（micro:bit と比べて）

春学期に micro:bit を使ってプログラミングをしたときには、メモリの容量から、動かせないものがあったのですが、Pyboard の場合は、ライブラリとして用意されているプログラムを置いておいても稼働させることができました。表 1 に仕様の比較をだしておきます。

表 1 micro:bit と Pyboard の仕様比較

項目	micro:bit	pyboard
価格*	2,160 円	7,192 円
CPU	ARM Cortex M0	ARM Cortex M4 (FPU 付)
動作周波数	16MHz	168MHz
RAM	16KB	192KB
FLASH	256KB	1MB+SD Card
内部センサー	3 軸加速度・磁力	3 軸加速度
入出力	USB/Radio/GPIO/Bluetooth	USB/GPIO/I2C/CAN/SPI/ADC
ボタン類	2 ×汎用・リセット	1×汎用・リセット
LED	5x5=25 LED（赤）	4 LED（赤・青・緑・黄）
外部駆動電圧	3.3V	3.3V（入力は 3.6～16V）

*価格は、switch science を参考

プログラミングでは、外部機器の制御がメインだったので、動作速度の違いは感じられませんでしたが、Pyboard は 10 倍の速さがあり、かつ FPU が付属しているので、実数計算が速くなっていることがわかります。またフラッシュメモリの容量も、4 倍あり、SD-Card も利用できるので、大規模な用途にも使えることがわかります。内蔵のセンサーは少ないのですが、入出力に I2C や CAN などのバス型接続をサポートしているので、複数の周辺機器を識別して、同時に制御できる機能が充実しています。

2.2 Pyboard の開発環境（micro:bit と比べて）

どちらの開発ボードも micro USB 端子がついており、この端子とホスト PC 側の USB を接続すれば、ドライブとして見えますので、ファイルの移動・コピーなどの機能を用いて、プログラムを転送することができます。

micro:bit には、Web での開発環境がありますが、mu editor(4)というローカルで動くアプリケーションとしての統合開発環境があり、MicroPython のプログラミングでは通常 mu が使われています。Pyboard ではターミナルから直接 REPL（Read-Eval-Print Loop）とも呼ばれる MicroPython のインタープリタを起動することができ、そこでプログラムをロードして実行させることが可能になっています。例えば、Mac OS X では、ターミナルか

ら次のようなコマンド起動させることができます。下記の USB Modem の番号は、接続によって変わります。

```
$ screen /dev/cu.usbmodem1422
```

この REPL への直接アクセス以外に、mu と同様に Thonny[5] と呼ばれる統合開発環境が Pyboard には用意されていて、プログラムエディタ上でプログラミングしたものを、Pyboard の REPL 上で実行させることが可能になっています。

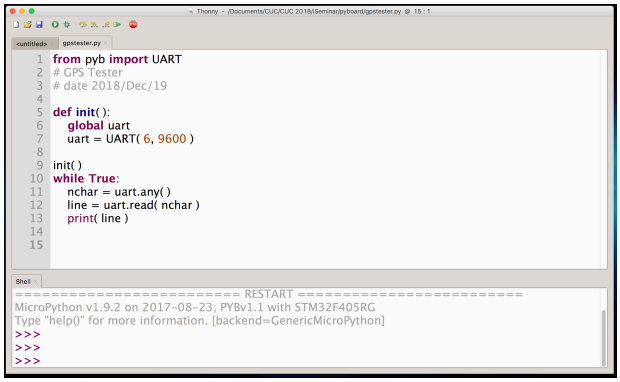

図 1　開発環境 Thonny

開発ボードへのファイルの転送ですが、Pyboard の場合は、micro:bit のような hex 形式のファイルに変換する必要はなく、プログラム（.py 拡張子を持つファイル）を直接転送すれば動くようになります。ライブラリは、micro:bit が独自の microbit パッケージを利用していたのに対して、Pyboard は MicroPython 標準の pyb パッケージを利用することになります。

2.3　周辺機器の駆動について

周辺機器に対応についてですが、イベントハンドリングが、micro:bit が自分でポーリングのためのループを組まなければいけなかったのに対して、Pyboard の場合は、基本的にイベントハンドリング用の関数（コールバック関数）を定義するだけでプログラミングができます。これは、Pyboard が制御する周辺機器の数の多さを考えてのことだと思われます。また、I2C/CAN/SPI などの周辺機器制御用のバス接続対応としても使えるピンが用意されており、これらのバスを利用して、複数の周辺機器を駆動することが可能になっています。

pyb のライブラリには、PWM 駆動、AD/DA 変換、サーボモータの角度を指定した駆動、I2C・CAN・SPI を利用しての周辺機器の制御、UART によるシリアル接続の制御、GPIO の利用などを簡単に行なえる関数などが含まれており、非常に簡単にプログラミングを行なうことが可能になっています。

3.　PyBoard を使っての MicroPython のプログラミングの実際

3.1　モータ・サーボモータの駆動

モータの駆動は、トランジスタでの増幅回路を間に置きました。また、PWM は、Pyboard の場合は、ライブラリに入っているので、それを用いることにしました。ボタンを用いて、回転を上げるプログラムは、次のようになります。

```
from pyb import *
ratio = 0

def escalator( ):
    global ratio
    if sw.value( ) == False:  return
    ratio += 10
    if ratio > 100: ratio = 0
    ch.pulse_width_percent( ratio )

p = Pin( "X1" ) # X1 has TIM2, CH1
timer = Timer(2, freq=1000)
ch = timer.channel( 1, Timer.PWM, pin=p )
ch.pulse_width_percent(ratio)
sw.callback( escalator )
```

サーボモータについても、Pyboard では、ライブラリに入っているため、それを用いることにしました。次のようなプログラムになります。用いたサーボモータは、フタバ製の S03T/2BBMG/F[6] になります。

3.2　LCD の駆動

LCD については、micro:bit のときはメモリ容量不足で失敗しました。今回用いたのは、HD44780 シリーズの互換の SC1602[7] です。これには、GitHub において、micropython-lcd[8] という MicroPython 用のライブラリが見つかりました。このライブラリを利用することにしました。このライブラリをしたプログラミングは、次のようになります。

```
import pyb
from lcd import HD44780

lcd = HD44780()
lcd.PINS = ['Y1','Y2','Y3','Y4','Y5','Y6']
lcd.init()  # Initialise display
```

```
lcd.set_line(0) # First line
lcd.set_string("ABCDEFGHIJKLMNOP")
lcd.set_line(1) # Second line
lcd.set_string("1234567890123456")
```

図 2 Pyboard から LCD への結線作業の様子

3.3　GPS モジュールの駆動

日本の準天頂衛星システム（QZSS）「みちびき」3機受信に対応したGPS受信キット[9]を用いてGPSモジュールを駆動することにしました。GPS についても、GitHub に micropyGPS[10]というライブラリがあり、これを利用することにしました。以下のようなプログラムで GPS からのデータを読み取るようにしました。

```
from pyb import UART
from micropyGPS import MicropyGPS

gps = MicropyGPS( )
uart = UART( 1, 9600 )
uart.init(9600, bits=8, parity=None, stop=1)
while True:
    nchar = uart.any( )
    if nchar > 0:
        line = (uart.readline( )).decode( )
        print( line )
        for x in line:
            stat = gps.update( x )
            if stat != None:
                print( gps.latitude, gps.longitude )
                print( gps.altitude, gps.date )
```

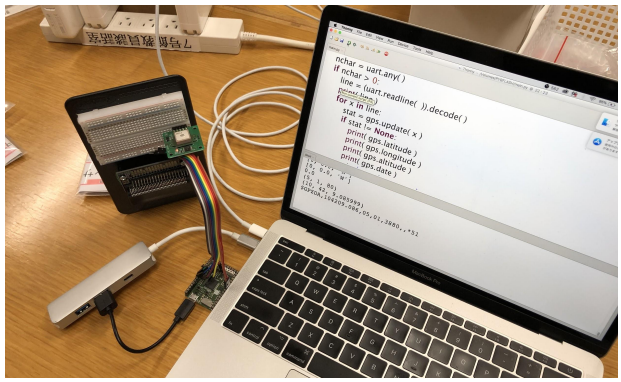

図 3　GPS 計測のための機器構成

GPS 計測は、屋外に出て行ないました。受信開始までには、かなり長い時間を要しました。

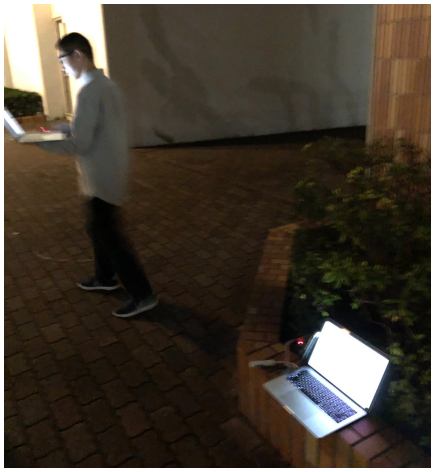

図 4　GPS 計測の様子

3.4　9 軸センサーの駆動

9 軸センサーとして利用したのは、BMX055[11]です。このデバイスは、加速度 3 軸、ジャイロによる角速度 3 軸、および磁気コンパスによる姿勢の傾きの 3 軸を測定することが可能になっています。また、各センサーの読取りに I2C の制御バスを利用するようにプログラムしなければなりませんでした。以下に加速度センサーの初期化の部分のプログラムを載せておきます。

```
from pyb import *
# 各センサーの I2C アドレス
Accl, Gyro, Mag = 0x19, 0x69, 0x13

# I2C の初期化
i2c = I2C( 1 )
i2c = I2C( 1, I2C.MASTER)
i2c.init( I2C.MASTER, baudrate=9600 )

# 加速度計の初期化
data = bytearray( 2 )
data[ 0 ] = 0x0F  # Select PMU_Range register
```

```
data[ 1 ] = 0x03  # Range = +/- 2g
i2c.send( data, Addr_Accl )
```

4. MicroPython を使った教育について

高校の授業においても、micro:bit 上の MicroPython を利用しての周辺機器の制御の教育を行なうことは可能だと思います。高校では時間数が限られているため、1 つの周辺機器を扱うことしかできないと思われます。むしろ、micro:bit の方が相応しい場合が多いのではないでしょうか。しかしながら、大学における周辺機器の制御の授業においては、複数の周辺機器を組み合わせたプログラミングを行なうことができるようになることが到達目標であるように思えます。そのときに、micro:bit では役不足になるでしょう。5x5 の LED のディスプレイがついていることは、デバッグなどにも役立つと思いますが、それよりも、I2C/CAN/SPI のようなバス接続の周辺機器を簡単に取り扱える Pyboard の方がむしろ使いやすい場合が多いと思われます。

今回の正課外の特別授業では、以下のようなプログラミングを各回に行ないました。全 6 回のうち、5 回目では各周辺機器の制御になっています。しかしながら、最後の第 6 回では、9 軸センサーから姿勢の傾きの値を読み取って、その値に応じてサーボモータを駆動させるというようなプログラミングについて扱いました。ロボットの制御などでは、このようなプログラミングが通常行なわれています。ある程度、ロボットのような形に組み立てる必要性はありますが、そのような工学系の到達目標があるときには、Pyboard は最適な学習教材になると思われます。

表 2　各回の授業内容

授業回	説明
第 1 回	内蔵の LED 制御と 3 軸加速度センサーからの値の読取り
第 2 回	モータの制御
第 3 回	2 行 LCD への表示
第 4 回	GPS モジュールからの値読取り
第 5 回	9 軸センサーからの読取り
第 6 回	センサーとサーボモータの駆動

5. おわりに

Pyboard は、micro:bit と較べてメモリ容量も多く、既存のライブラリをロードしても充分に稼働します。プログラム体験を主眼におく、高校までの教材としては、micro:bit でも充分でしょうが、大学で本格的に MicroPython を用いて、比較的大きなプロジェクトの教育教材として利用するには、Pyboard の方が良いことがわかりました。

参考文献

(1) Damien P. George, George Lobotics, Limited, MicroPython, https://micropython.org
(2) Micro:bit Education Foundation, micro:bit, https://microbit.org/ja/
(3) Damien P. George, pyboard, http://docs.micropython.org/en/latest/pyboard/quickref.html
(4) Nicholas H. Tollervey, Code with Mu: a simple Python editor for beginner programmers, https://codewith.mu
(5) Aivar Annamaa, Thonny, Python IDE for beginners, https://thonny.org
(6) Grand Wing Servo-Tech Co. Ltd., Standard Series, https://gwsus.com/gws_com_tw_www/english/product/servo/standard.htm
(7) Sunlike Display Tech Corp., SC1602, http://akizukidenshi.com/catalog/g/gP-04794/
(8) wjdb, micropython-lcd, https://github.com/wjdp/micropython-lcd
(9) 秋月電子通商, GPS 受信機キット AE-GYSFDMAXB, http://akizukidenshi.com/catalog/g/gK-09991/
(10) inmcm, micropyGPS, https://github.com/inmcm/micropyGPS
(11) Bosch Sensor Tech, BMX055, https://www.bosch-sensortec.com/bst/products/all_products/bmx055