教育研究

# Raspberry Pi による画像処理を題材とした プログラミング教材の開発

宇都木　修一

## Development of programming teaching materials based on image processing using Raspberry Pi

Shuichi Utsugi

As represented by the term IoT, the use of information technology is essential not only for software development but also for hardware development. This article describes the development of an introductory programming material that can learn about hardware control with software. Raspberry Pi was adopted as a computer environment for learning how to control hardware with software. The Raspberry Pi is small PC but it can control the hardware easily, so it is suitable as an introduction material. However, if hardware control is realized too easily with RaspberryPi, which is not suitable for a teaching material. It is necessary to maintain an appropriate level of difficulty as an engineering programming education. The programming material presented in this article designed to allow students to experience the control of hardware while learning basic programming elements such as repeat statement and branches.

## １．はじめに

我が国において IT 教育の重要性が取り沙汰されて 20 年以上経過し，ついには，小学生にプログラミング教育の必修化に至った．現代の社会においてますます重要となる，人工知能を含めたソフトウェア技術を維持発展させていくためには，継続的な情報技術者の育成が欠かせない．さらには，モノのインターネット（IoT）という言葉に代表されるように，もはや情報技術はソフトウェアエンジニアだけでなく，ハードウエアエンジニアにとっても欠かせない技術となっており，ソフトウェア，ハードウェア双方に精通する技術者の養成が不可欠となっている．

そのような流れの中では，プログラミング，あるいは，機械設計や回路設計と分野ごとに明確に切り分けられた入門教材だけではなく，それぞれを融合した教材が必要になる．以上の考えに基づき，筆者は Raspberry PI によって簡単な画像処理とモータ制御を簡単なプログラミングによって体験できる入門用教材を作成した．具体的にはカメラに映った特定の色の物体を追跡するようにカメラの向きをコントロールするプログラムの作成を通して，プログラミングを学ぶというものである．

画像処理は一つ一つのピクセル（画素）に対して同様の処理を連続しておこなったり，状況に応じて処理を変化させたりするなど, 初歩的な「繰り返し」や「分岐」の習得に適している．さらに，出力結果を画像で表示させることで，視覚的に理解を促すこともできる．

また, Raspberry Pi は教育用として企図された廉価なシングルボードコンピューターであるが，OS として Linux を搭載し, PC としての一通りの機能を有したうえで，さらに，GPIO（汎用入出力）も有しており，プログラミングによる外部ハードウェアの制御を学ぶことができる．ハードウェアをプログラムによって制御できるという点については，ワンボードマイコンの Arduino であっても目的に適うが，画像処理，そして人工知能など複雑かつ計算量の多いプログラムを組み込んだ制御への発展を考えると，Raspberry Pi がより好適であるといえる．

カメラに映った特定の物体を，モータを制御して追跡する Raspberry Pi のプログラムの作成については，その方法を解説している文献[1)]や web ページが既にある．しかしながら，これらの文献や web ページは単にプログラムやハードウェアを動かす方法に


図 1 本教材のインターフェース
(Raspberry Pi 本体はディスプレイの裏に固定)

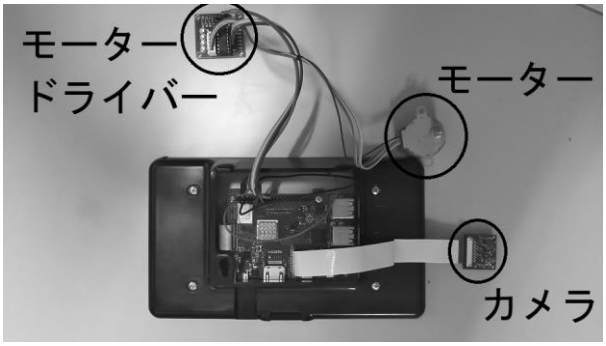


図 2 本教材の裏側
(Raspberry Pi 本体とその他の部品)

表 1 本教材のハードウェア構成

| | |
|---|---|
| PC | Raspberry Pi 3 b+ |
| カメラ | Raspberry Pi<br>カメラモジュール |
| ステッピングモータ | 28byj-48 |
| モータドライバー | ULN2003<br>ドライバボード |

ついて解説しているのみであり，プログラム教育を意識とした演習教材にはなっていない．本稿で紹介する教材は，あくまでプログラミングの演習教材として成立するよう構成されている．

## 2．ハードウェア構成

ハードウェアの構成は可搬性を考慮して表 1 のとおりとした. PC 本体には Raspberry Pi3 b+を使用し，ディスプレイは Raspberry Pi 公式 7 インチディスプレイとして，さらに専用のケースを用いることでディスプレイと本体と一体化した．また，折りたたみ式キーボード，マウスも用意し，これらは Bluetooth で Raspberry Pi と接続するようにした.

さらに, 電源も モバイルバッテリーを利用すれば, ケーブルレスで Raspberry Pi を稼働させることができ，一般の教室でも演習が可能になる.

カメラには Raspberry Pi 専用のカメラモジュールを用いた．また，モータはステッピングモータ（28BYJ-48）[2)]，モータドライバーはダーリントン・トランジスタ・アレイ ULN2003 を用いた（図 2). これらはカメラも含めて電子工作向きの廉価な部品である.

## 3．画像処理

画像処理およびモータ制御をおこなうためのプログラミング言語として Python を用いる．また，画像処理用のライブラリには Intel によって開発された OpenCV を用いる． ただし，OpenCV は強力な画像処理ライブラリであるがゆえに，教育目的で使用する際には注意を要する．大抵の画像処理が関数呼び出しだけで簡単に行われてしまうために，分岐や繰り返しなどのプログラミングの基本的な能力を培う目的の上では不適切となるためである．したがって，本教材では，基本的には画像の入出力部分に絞って OpenCV の機能を用いることとした.

本教材ではカメラの画像から特定の色の部分を抽出して，それを追跡するプログラムを作成するようにするが，その方法には大きく分けて 3 つの方法がある.

(a) 単純に特定の色を抽出する.
(b) 特定の色の領域の輪郭を抽出する.
(c) 再帰を用いて特定の色の領域を抽出する.

(a) の方法は，それぞれのピクセルの色をチェックするだけで済むので，計算量を抑えることができるが，追跡対象の物体と同じ色の物体が他にもカメラに映った時に, 追跡に失敗するという短所がある.

(b) の方法は，追跡対象の物体と同じ色の物体が複数映った場合でも，個別に分離することができるので，最も大きい輪郭を持つものを追跡するようにすることで，追跡を維持することができる．また，処理量も抑えることができるが，プログラミングの難易度が上がってしまい，入門用の課題としては適さないものとなってしまう.

(c) の方法は，再帰プログラミングを理解していれば，簡潔に領域そのものを個別に抽出するプログラムを作成することができる．ただし，再帰プログ

表 2 OpenCV と NumPy を使用した場合に，
特定の色の領域を抽出するプログラム例

```
bgrLower = np.array([0,0,128])
bgrUpper = np.array([100,100,255])
img_mask = cv2.inRange(frame, bgrLower, bgrUpper)
frame[img_mask == 255] = [255,0,0]
mu = cv2.moments(img_mask, False)
x,y = int(mu["m10"]/mu["m00"]), ¥
              int (mu["m01"]/mu["m00"])
```

ラム自体が入門の範疇を大きく超えるうえに，処理量が多く，Raspberry Pi による実時間処理には適さないものとなっている．

以上から，本教材では(a) のプログラムを作成するものを基本課題とし， (b) のプログラム作成を発展課題とすることにした．

いずれにせよ，本教材では分岐や繰り返し文などの基本的なプログラム構文を習得することを目的としているが，プログラミング言語として採用した Python は繰り返し構文である for 文や while 文の実行には非常に時間がかかるという問題がある．これは，Python がプログラムを最適化された実行形式に事前に変換するコンパイラ言語ではなく，プログラムを逐次実行するインタプリタ言語であるうえに，Python がプログラム上で変数の型の指定などを必要とせず，型の推論をおこないながら実行する動的型付け言語であることが理由である．

for 文の実行に時間がかかることは実時間での画像処理にとって極めて致命的である．この問題を回避するには，多次元配列に対する計算を効率的におこなう数値計算ライブラリ NumPy を用いる方法と，JIT（Just In Time：実行時)コンパイラである Numba を用いる方法がある．

NumPy は繰り返し文を高速化するのではなく，配列に対する演算を，MATLAB のように for 文を使わずにベクトル化して記述することで，内部で高速に実行するものである．NumPy は OpenCV と組み合わせることで，画像の中から特定の色を持つ領域を抽出する処理は，重心の座標を求める処理を加えても，for 文や if 文を使わずに 6 行程度で記述することができる（表 2）． しかし，これはすでに述べたように，繰り返し文や分岐文を習得することを目的とした本教材の趣旨に反する．

これに対し，Numba は記述したプログラムを Java のようにコンパイルしながら実行するので，繰り返し文を高速に処理することができる．本教材では，この Numba を利用して繰り返し文と分岐文を利用した画像処理を行うこととした．

表 3 画像処理部分のプログラム例

```
from numba import jit
import cv2

@jit
def test(img):
  height = img.shape[0]
  width = img.shape[1]

  wi = 0
  wj = 0
  num = 0
  for j in range(height):
    for i in range(width):
      if img[j,i,0] < 100 and img[j,i,1] < 100 ¥
                          and img[j,i,2] > 128:
        img[j,i] = 0,0,255
        wi += i
        wj += j
        num += 1

  if num > 0:
    wi /= num
    wj /= num

  return img, wi, wj,num
```

画像処理部分の模範解答としてのプログラムを表 3 に示す．Numba は部分的にコンパイルするため，コンパイルする部分を関数化する必要がある．したがって，作成するプログラムでは，繰り返し，分岐，2 次元配列，関数と，プログラミングの基本要素を全て駆使することになり，演習問題として申し分のない課題となっている．

## ４．モータ制御

使用するモータ 28byj-48 はクローポール式の PM 型ステッピングモータで，クローポール数が合計で 32 本となっている [3]．したがって，1-2 相励磁によ

る駆動の場合, モータ本体は 64 ステップで 1 回転となる．さらに，28byj-48 ギア比 1:64 の歯車も内蔵しており，回転軸自体は 64×64 = 4096 ステップで一回転するようになっている．

実際の制御は，28byj-48 のクローポールを励磁する 4 本の信号線を，モータドライバーを介して Raspberry Pi の GPIO に接続し，その GPIO の出力を Python によるプログラミングで制御することによっておこなう．

プログラム上では，信号の出力パターンを順送りするか逆送りにするかで回転方向を，信号を送る回数で回転角度を，信号を送る周期で回転速度を制御する．回転速度については，28byj-48 は 1000[Hz]（1000 ステップ／秒）程度までの信号に対応できるので，最速で約 4 秒で 1 回転となる．

ただし，本教材は入門用としているので，モータの制御は回転方向だけとし，画像処理と合わせて次のようなプログラムを作成するものとしている．

(Step1) カメラからの画像をキャプチャし, 特定の色を持つピクセルを抽出する．
(Step2) 抽出されたピクセルの分布の重心座標を求める．
(Step3) 画像の中心に対して重心座標が左右のどちら側にあるか判断して, モータの回転方向を定める．

以上のステップをプログラム化するによって，特定の色を持った物体をカメラが追跡するアプリケーションを作成できる．このプログラムはおおむね 150 行程度で記述できる. GPIO の出力を制御する部分から丁寧にプログラムを作り上げていけば，入門用の課題としては十分に歯応えのある課題となる．

## ５．発展課題

理解の早い学生であれば，本教材程度の内容では物足りない場合や，あるいは，さらに高度なプログラム課題を欲することが考えられる．その場合，画像処理の部分では，輪郭抽出で特定の色の領域を抽出することや，テンプレートマッチングによる物体の追跡を実装することが考えられる．また，モータの制御の部分については，簡単な PID 制御を取り入れるということも考えられる. ただし, この場合は，画像の取得と処理の実行速度に注意しなければならない．

単純に画像を取得して処理をしてから，モータの制御をおこなう場合，画像のキャプチャ速度に信号出力の周期が影響を受ける．すなわち，画像を 1 フレーム取得してから，1 ステップ分だけモータが回転することになる．実際に画像のキャプチャ速度を秒間 30 フレームとした場合, モータが 1 回転するまでにおよそ 138 秒を要した. 1 回転に 4096 ステップ要するので，秒間 30 ステップであり，秒間 30 フレームにほぼ等しい値となっている．また，キャプチャ速度を秒間 60 フレームとした場合は，約半分の 70 秒を要した．したがって，より高速にモータを動かすには, 画像取得の周期の影響を受けないように, モータの制御を別々に並行しておこなう必要がある. そのためには並列処理プログラミングが必要となるので，これを発展課題とすることもできる．

ちなみに，モータの制御と画像処理を並列化した場合は，約 6 秒で 1 回転した．ただし，モータへ送信する信号の周期を 1000[Hz]以下にするために，回転の 1 ステップごとに 1 / 1000 秒のウェイトを挟んでいる．このウェイトだけであれば，約 4 秒で 1 回転（4096 ステップ）するはずだが，プログラムの起動直後，カメラからの画像を取得するまでに，1 秒程度の時間を要したので，約 6 秒での 1 回転となった．

また，このウェイト時間を調整することで，モータの回転速度を制御できるので，PID 制御を取り入れることができる．この場合は，画像処理よりもシステム制御に重きを置いた課題となる．さらには，並列処理を通信を利用する形に発展させ，画像処理を別の計算機に任せることで，より高度な処理を実現させる課題へと発展もできる．すなわち，本教材は画像処理を入り口として，より IoT との関わりの深い部分へと学習を進めていくことができるようになっている．

## ６．おわりに

近年注目されている IoT に関する入門教材として, 画像処理とモータ制御によって物体を追跡するプログラムの作成を題材とする教材を作成した．Raspberry Pi を用いることで, マイコンの処理能力では対応できない画像処理にも対応し，ある程度複雑なソフトウェアでハードウェアを制御することを学べる教材となっている．実際にはインターネットを介した通信プログラムを作成するところまでは教材化していないが，並列処理を発展させる形で通信部分を組み込むことも可能である．

また，本教材を簡略化したものを，中学生を対象としたサレジオ高専の体験入学で使用する予定である．この体験入学での中学生の反応や，高専 1，2 年生を対象とした教材の試用によって，本教材をブラッシュアップしたうえで，実際に授業で展開することを予定している．

**文献**

1) 金丸隆志, 「実例で学ぶ RaspberryPi 電子工作」, 講談社, pp.181-268, (2015)
2) 28BYJ-48 Datasheet – Digikey, https://www.digikey.jp/ja/datasheets/mikroelektronika/mikroelektronika-step-motor-5v-28byj48-datasheet
3) ステッピングモーター (28BYJ-48) を分解して仕組みを調べてみた, https://stupiddog.jp/note/archives/1209