RaspberryPi を用いた制御技術の習得

○山本遼 A)、後藤伸太郎 A)

A) 工学系技術支援室　装置開発技術系

1　はじめに

装置開発技術支援室システム開発グループ及び精密加工グループでは、実験等で使用される装置や部品の製作を主として行っている。しかしながら装置と呼ばれるものは、機械の部分とは別に制御部やそれに伴う電子回路部がある。今後、機械加工だけでなく制御部の相談や製作の要望にも応えるため、python 言語と RaspberryPi を用いた制御技術を習得することを目的とし、研修を行った。

本稿では、研修で製作した巻線機を軸に、得られた成果や今後の課題について報告する。

2　RaspberryPi とは

RaspberryPi（図 1）とはイギリスのラズベリー財団によって開発された教育用小型コンピューターである。本機を用いた理由は、小型である点、安価に(1000～5000 円)入手することができる点、標準で python が導入されている点、GPIO を持っている点、IOT 化に向いている点、開発に関する情報が豊富な点などである。

図 1. RaspberryPi zero wh

3　python とは

python とは 1991 年に開発されたプログラミング言語の 1 つであり、近年では YouTube や Dropbox、Instagram なども python を用いて開発されており、非常にメジャーなプログラミング言語だと言える。

python を用いた主な理由としては、文法が比較的簡単であること、各分野に特化したライブラリが豊富にあることである。以下に紹介する巻線機の制御は python を用いて行った。

4　RaspberryPi と python を用いた制御の学習

まず RaspberryPi と python を用いた制御を行うにあたり、LED を任意に光らせることから学習を進めた。基本的な if や for などの記述方法についてある程度学習を進めた。

その後、本研修のメインテーマに設定した巻線機について設計し、部品等の仕様を決めた後にプログラムを作成した。プログラムの作成にあたり、巻線機の製作を行っている情報は得られなかったため、類似の開発を行っているものを参考に開発を進めた。

5　巻線機の開発

本研修では複数の制御要素を含む題材として巻線機の設計製作を行った（図 2）。小型（A4 用紙上に収まる大きさ）であること、安価であること、複雑な機構は減らすことを基本目標とし、ワイヤー径に応じた位置制御、主軸の回転制御、ワイヤー折り返しの検知などについて制御を行った。

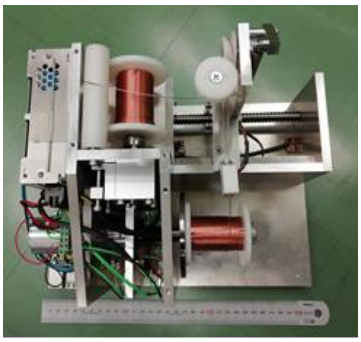

図 2. 製作した巻線機

6　巻線機動作のフローチャートと各部の概要

製作した巻線機は右のフローチャートの手順で動作させた（図 3)。

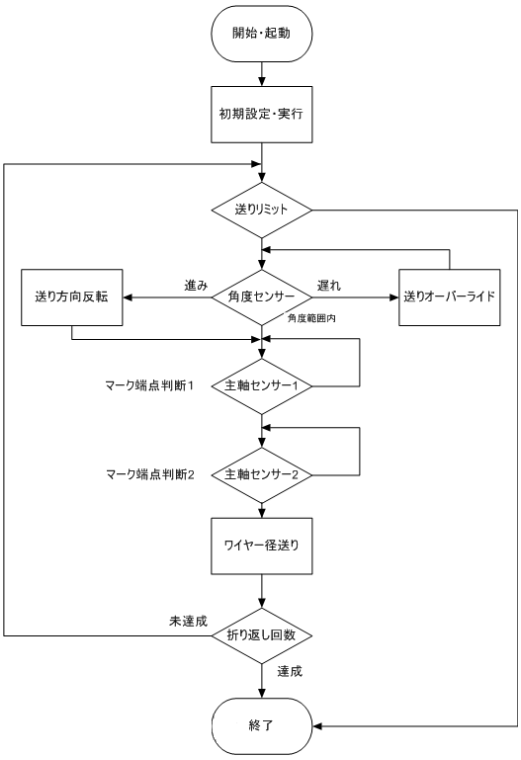

図 3.　巻線機動作フローチャート

1.1　初期値の設定

送りステッピングモーターの初期設定や、ワイヤー径・巻き数等の入力を行う。

1.2　送りリミット

送りテーブルが限界位置に移動した場合、プログラム全体をスキップし終了する。

1.3　角度センサー

巻かれているワイヤーの状態に応じて送りをオーバーライドさせる。ワイヤーが折り返した場合はそれを検知し、送り方向を反転させる。

1.4　主軸センサー1、2

主軸マーク位置を検知しマークの始めから終わりまで回すことで主軸 1 回転とした。

1.5　送りテーブルワイヤー径送り

通常状態では送りテーブルを主軸 1 回転毎にワイヤー径分だけ送る。

1.6　折り返し回数

初期設定で指定した折り返し回数（コイルの層数）に到達したか、否かを判断する。判断の基準は巻き数、層数のどちらでも可能なものとした。

以下からそれぞれの部分についての動作・プログラムについて説明する。

7　主軸 DC モーターの制御

DC モーターの正逆回転・ブレーキについて、モータードライバーTA7291 を用いて制御した。TA7291 は 2 つの入力端子を持っており、その 2 端子のオンオフの状態からモーターを制御することができる。

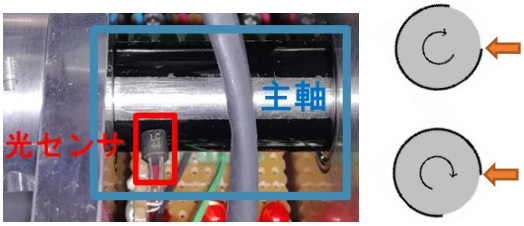

図 4.　主軸と光センサーとその読み取り

また主軸回転位置は、光センサーで主軸マークの読み取りを行った。プログラムではライブラリ「RPI.GPIO」を用いて RaspberryPi の GPIO の入出力を行い、主軸 DC モーターの回転や、回転回数のカウント、主軸原点復帰などを制御した。（図 4）今回の巻線機では光センサーで必要十分であったが、これをロータリーエンコーダー等に変更することで、より正確な回転位置制御を行うこともできると考える。

8　送りステッピングモーターの制御

ステッピングモーターを用いた送りテーブルの位置制御について、モータードライバーL6470 とボールネジを用いて制御した。L6470 では SPI 通信を用いて制御を行うため、ライブラリ「spidev」を用いて GPIO による SPI 通信を行った。L6470 には様々なコマンドや設定を行うことができる（表 1）が、それらについてメインプログラム上で spidev を用いて記述すると、プログラムが長くなってしまうほか、どのようなコマンドかを一目で判断することができない。そこで L6470 のコマンドについて、関数としてまとめた自作のライブラリを作成することで、メインプログラムの記述を簡略化した。また、ステッピングモーターの初期設定について別のモジュールに記述し、読み込む手法を採用することで、こちらもメインプログラムの簡略化とした。

表 1. L6470 コマンド抜粋

コマンドの記述	コマンドのバイナリー・コード					動　　作
	765	4	3	21	0	
NOP	000	0	0	00	0	何も実行されません。
SetParam（ADR, Value）	000	[ADR]				アドレス（ADR）のレジスターに、Valueの値を書き込みます。
GetParam（ADR）	001	[ADR]				アドレス（ADR）のレジスターに格納された値を返します。（読み出し）
Run（DIR, Spd）	010	1	0	00	D	目標速度（Spd）とモーターの方向（DIR）を指定して、定速回転をさせます。
StepClock（DIR）	010	1	1	00	D	デバイスをステップ・クロック・モードにして、方向（DIR）を指定します。（外部クロックで駆動）
Move（DIR, N_Step）	010	0	0	00	D	DIR方向にN_Stepのマイクロ・ステップを行います。（モーターが動いているときには実行されません）
GoTo（ABS_POS）	011	0	0	00	0	最短経路で、ABS_POSの位置へモーターを動かします。
GoTo_DIR（DIR, ABS_POS）	011	0	1	00	D	DIR方向を指定して、ABS_POSの位置へモーターを動かします。
GoUntil（ACT, DIR, Spd）	100	0	A	01	D	SWがオンになるまで、Spdの速度でDIR方向に動作を実行し、SWがオンになると、その時のACTビットの動作が実行されてSoftStopが起きます。
ReleaseSW（ACT, DIR）	100	1	A	01	D	SWがオフになるまで、最低速度でDIR方向に動作を実行し、SWがオフになると、その時のACTビットの動作が実行されてHardStopが起きます。
GoHome	011	1	0	00	0	HOME位置にモーターを持ってきます。
GoMark	011	1	1	00	0	MARK位置にモーターを持ってきます。
ResetPos	110	1	1	00	0	ABS_POSレジスタをリセットします。（HOME位置を決めます）
ResetDevice	110	0	0	00	0	デバイスを、電源投入時の状態にリセットします。
SoftStop	101	1	0	00	0	減速期間の後に、モーターを停止します。（保持トルクあり）
HardStop	101	1	1	00	0	直ちにモーターを停止します。（保持トルクあり）
SoftHiZ	101	0	0	00	0	減速期間の後に、ブリッジをハイ・インピーダンス状態にします。（保持トルクなし）
HardHiZ	101	0	1	00	0	直ちにブリッジをハイ・インピーダンス状態にします。（保持トルクなし）
GetStatus	110	1	0	00	0	STATUSレジスターの値を返します。

9　送りテーブルのリミットセンサー

送りテーブルのリミットについて、光センサーを用いて制御した。送りテーブルがリミット位置にくると光センサーが反応し、それを GPIO で読み取ることで、リミット位置を判断し、try: except KeyboardInterrupt:を用いることでプログラム全体から脱出する。

10　ワイヤー角度センサー

巻かれるワイヤーは常に一定の太さで曲がり等がなく、巻くボビンの長さも寸法が正確であれば、計算どおりに巻くことができる。しかし実際にはワイヤー径の誤差などから計算通りとはいかず（図 5 左）、正常に巻くためには巻かれているワイヤーの状態を把握する必要がある。

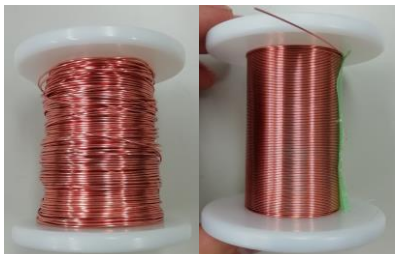

図 5.　コイルの状態の比較

巻かれているワイヤーの状態を、角度センサーを用いて計測し（図 6）、Arduino を用いて特定の条件の時に信号を RaspberryPi に発信することでワイヤーの状態を確認し、制御を行った。Arduino を用いた理由は、RaspberryPi のアナログ入出力端子が少ない点、GPIO を節約したかった点、メインプログラムを簡略化できる点である。送りに遅れが生じた場合、角度センサーが一定の値になるまで送りをオーバーライドさせ、ワイヤーがボビン端で折り返した場合、角度センサーは逆方向に動かされ、それを基準として、送りも逆方向へ動くよう制御した。これにより、ワイヤー径やボビンの誤差の影響があっても綺麗に巻くことが可能となった（図 5 右）。

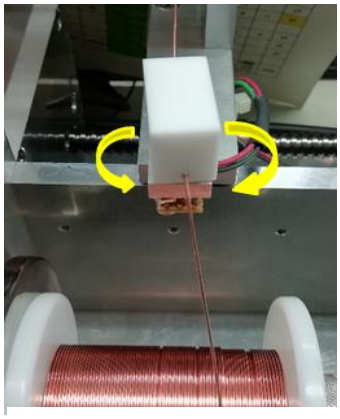

図 6.　角度センサー

Arduino 基盤は大きくスペースを使ってしまうため、マイコン（ATMEGA328P-PU）に角度センサーのプログラムを書き込むことで基盤を必要最小限の大きさになるようにした。

11　GUI アプリケーションの作成

巻線機の制御について上記の通り行ったが、このプログラムを動作させるには、ターミナルと呼ばれる windows でいうコマンドプロンプトのようなものを使用しなければならない。これを開発者以外の人にも簡単的に使用できるようにするため、ライブラリ「Tkinter」を用いて GUI アプリケーションを作成した。（図 7）

現在は RaspberryPi にモニタ類を接続するか、遠隔操作することで操作しているが、タッチスクリーンと GUI を用いることで、装置単体でより簡単に操作を行うことができると考える。

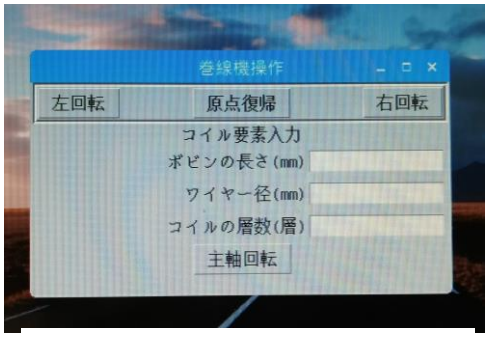

図 7.　作成した操作画面

12　得られた成果

以下に得られた成果についてまとめる。

- RaspberryPi の基本的な技術の習得
- Python でのプログラミング概要や、開発のためのライブラリについての基礎知識習得
- 他 PC やスマートフォンからの RaspberryPi を遠隔操作
- 角度センサー等を用いるための電子回路の技術習得
- Arduino の基本的な技術の習得
- RaspberryPi とマイコンの使い分けの理解
- 装置製作に伴う部品の選定などの理解
- 巻線機を小型・安価にするための加工や機構の工夫

装置全体を作ることで、どのような技術や知識が必要か、段取りはどのように行えばよいかについて学ぶことができた。また、当初は RaspberryPi のみで全ての制御を行えると考えていたが、製作を進めていく中で、マイコンやドライバーに役割を分担することで、メインプログラムの簡略化ができると共に、段階的な開発ができるということが分かった。

13　巻線機における今後の課題

- ワイヤーテンションの制御

現在は、板バネとすべり軸受けを組み合わせた機構によりワイヤーにテンションをかけているが、ワイヤー径が太くなるとテンションが足りず、角度センサーの誤動作につながってしまう。今後、ワイヤーテンションの状態を計測・制御し、常にテンションを一定に保つ機構が必要であると考える。

- インターネットを利用した通知や運転状態の確認

RaspberryPi を用いることで IOT 化を容易に達成することができる。今後、装置の動作異常や、コイルの巻き終わりを通知するといった機能を追加したい。

- 主軸モーターの高速化
- 制御パネルの製作

14 おわりに

RaspberryPi は色々な分野で使用されているため、多くの情報を得ることができるが、それゆえどれが必要で使える情報なのかを判断することが非常に困難であった。しかしながらプログラミングや電子回路について初心者であった私でも、ある程度形になるものを製作することができた点から、RaspberryPi は理解しやすく、また応用のきく便利なツールであるといえる。

今後、業務における装置開発に活用する他にも、業務依頼書の管理や集計といったファイル管理アプリケーションの開発などへの応用や、プログラミング体験実習など多方面での応用も可能であると考える。

最後に本研修を行う機会を与えてくださった装置開発技術支援室の皆様に深く感謝の意を示します。