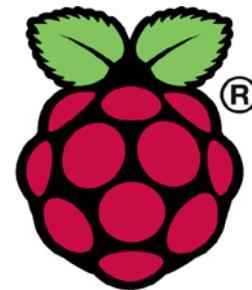


# 抜粋編

## Pythonで組み立てるFMラジオキット

～Pythonプログラミングを完全マスターするための実践キット～

### 組立編



**Raspberry Pi**

スペクトラム・テクノロジー株式会社

<https://spectrum-tech.co.jp>

[sales@spectrum-tech.co.jp](mailto:sales@spectrum-tech.co.jp)

## 組立編 目次

• Raspberry Pi運用マニュアル	ページ
1. RaspberryPiについて	<a href="#">3</a>
2. Linux基本コマンド	<a href="#">4</a>
3. RaspberryPi基本操作	<a href="#">5</a>
4. 日常運用(ウイルススキャン、更新)	<a href="#">6</a>
• FMラジオ組立キット	
1. 概要	<a href="#">8</a>
2. 全体像	<a href="#">9</a>
3. FMラジオ接続	<a href="#">10</a>
4. プログラム作成	
① 画面フレーム作成	<a href="#">13</a>
② 文字表示	<a href="#">15</a>
③ ボタン作成とラジオ局のプリセット	<a href="#">16</a>
④ 時計表示	<a href="#">17</a>
⑤ スクロールバーによる選局	<a href="#">18</a>
⑥ 受信レベル表示	<a href="#">19</a>
⑦ ステレオ／モノの表示	<a href="#">21</a>
⑧ 地域別のFM局自動設定	<a href="#">23</a>
⑨ FM局自動検索、結果出力	<a href="#">25</a>
5. プログラム例	<a href="#">30</a>
6. 今後の検討課題	<a href="#">31</a>

抜粋編になります。目次と内容は一致しません。

# RaspberryPi運用マニュアル

## 1. Raspberry Piについて

既に全世界で1000万台以上販売された手のひらサイズのコンピュータです。  
LinuxベースのRasbianOSで動作しております。

## 2. Linux基本コマンド

### ① システム関係

- 起動: 電源を入れると自動で起動します。

- 再起動: `$ reboot`

又は、`menu>shutdown>reboot`; 左上のメニューから

- 終了: `$ shutdown`

又は、`menu>shutdown>shutdown`; 左上のメニューから

- ログアウト `$ exit`

又は、`menu>shutdown>logout`; 左上のメニューから

- **日本語／英語の入力切替**: キーボードの CTL と `j` を同時に押します (コントロール: 左下と `j`)

# RaspberryPi運用マニュアル

## 2. Linux基本コマンド

### ② ディレクトリ操作、コピー、移動、削除

pi@raspberrypi:~\$ **cd** /home/pi/Documents ディレクトリの切り替え  
 pi@raspberrypi:/home/pi/Documents\$ **ls** ファイルとディレクトリの表示(表示したら操作したいファイルを右クリックでコピーして操作します)

pi@raspberrypi:~\$ **cp** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへコピー  
 pi@raspberrypi:~\$ **mv** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへ移動  
 pi@raspberrypi:~\$ **sudo rm** ファイル名 ファイルの削除  
 便利な機能 **rm -help** コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべてのコマンド共通(マイナスを2個とhelp)

### ③ ユーザ権限、プロセス他

pi@raspberrypi:~\$ **su -** スーパーユーザ(root)に切り替え、パスワードを入力  
 pi@raspberrypi:~\$ **ps a** 現状の動いているプロセスを表示  
 pi@raspberrypi:~\$ **kill** 特定のプロセスを強制終了  
 pi@raspberrypi:~\$ **sudo apt-get install pkg** パッケージのインストールなどに使用  
 pi@raspberrypi:~\$ **date** 日付、時間の設定を行います。  
 pi@raspberrypi:~\$ **sudo leafpad /etc/network/interfaces** インタフェースに記述している内容を変更します。Viよりも使いやすいです。

### ④ モジュール、usb、メモリ、HDDなどの表示

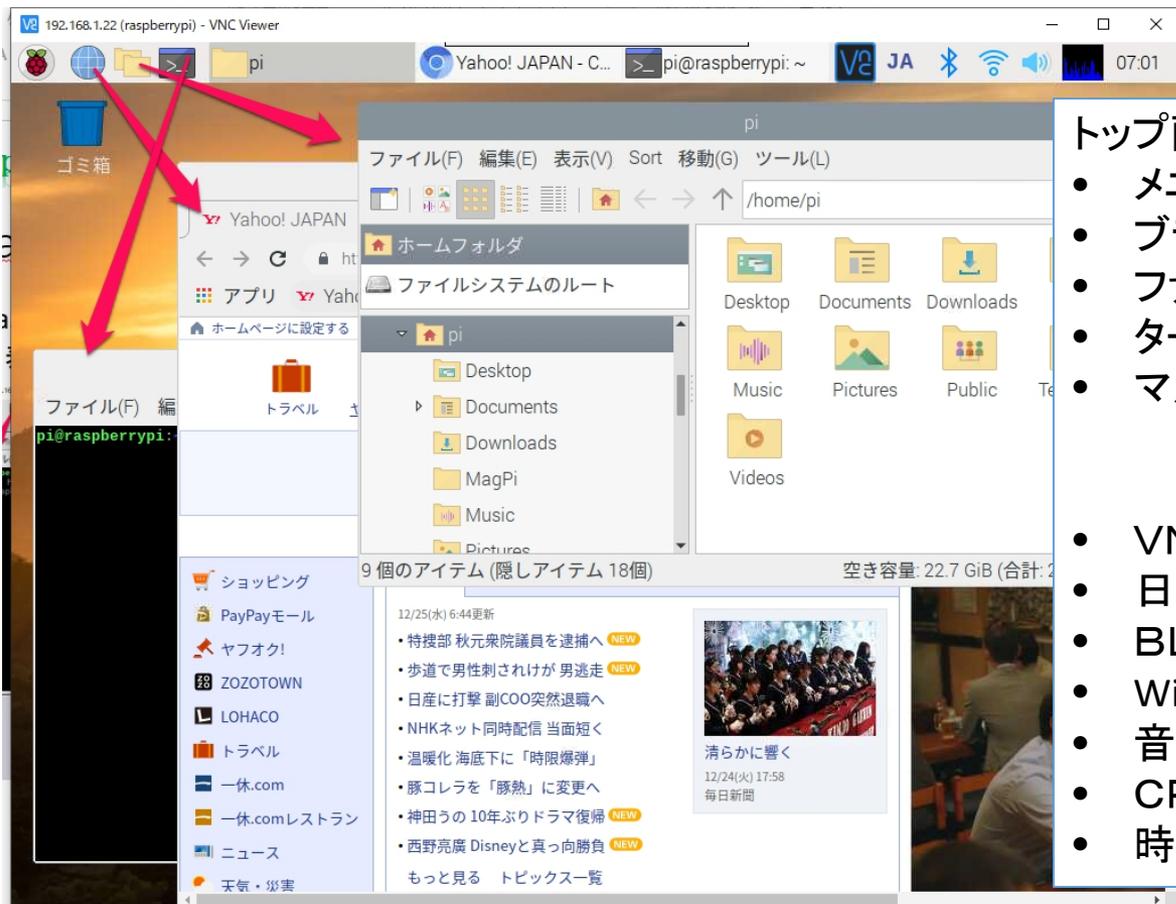
pi@raspberrypi:~\$ **lsmod** linuxのモジュールリスト表示  
 pi@raspberrypi:~\$ **lsusb** usbのデバイス表示  
 pi@raspberrypi:~\$ **free -mt** メモリ使用状態表示  
 pi@raspberrypi:~\$ **df** HDD(マイクロSD)の使用状態表示

## RaspberryPi運用マニュアル

### 3. Raspberry Piの基本操作

#### ① 表示画面と内容

デスクトップ上によく使うコマンド.txtがあります。  
コピーして使ってください



トップ画面(上段のタスクバーで選択)

- メニュー
  - ブラウザ
  - ファイルマネージャ
  - ターミナル
  - マルチ画面選択
- 
- VNC
  - 日本語入力
  - BLE
  - WiFi
  - 音量
  - CPU使用率
  - 時刻

## RaspberryPi運用マニュアル

### 4. 日常運用

#### ① セキュリティ対策(アンチウイルス更新、スキャン)

- アンチウイルス対策として無料のclamAVをインストールしてます。
- 手動での運用を基本としています。

#### パターンファイル更新

手動スキャン時に更新されます

#### 手動でスキャン

\$ sudo clamscan --infected --remove --recursive  
自動化可能ですが、バックグラウンドで重くなる可能性大。コマンド入力後約5分位かかります。

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
ERROR: /var/log/clamav/freshclam.log is locked by another  
ERROR: Problem with internal logger (UpdateLogFile = /var/  
og).  
root@raspberrypi: ~# leafpad /etc/clamav/freshclam.conf  
root@raspberrypi: ~# freshclam  
ClamAV update process started at Fri J  
main.cvd is up to date (version: 57, sigs: 4216790, f-level: 60, builder: mishh  
ammer)  
daily.cvd is up to date (version: 21862, sigs: 394456, f-level: 63, builder: neo  
)  
bytecode.cvd is up to date (version: 283, sigs: 53, f-level: 65, builder: neo)  
root@raspberrypi: ~# clamscan --infected --remove --recursive  
SCAN SUMMARY  
Known viruses: 4607906  
Engine version: 0.99.2  
Scanned directories: 264  
Scanned files: 2063  
Infected files: 0  
Data scanned: 61.31 MB  
Data read: 49.02 MB (ratio 1.25:1)  
Time: 71.844 sec (1 m 11 s)  
root@raspberrypi: ~#
```

手動でスキャン



# FMラジオ組立キット

難易度を示します  
 A: 上級  
 B: 中級  
 C: 初心者

## 1. 概要

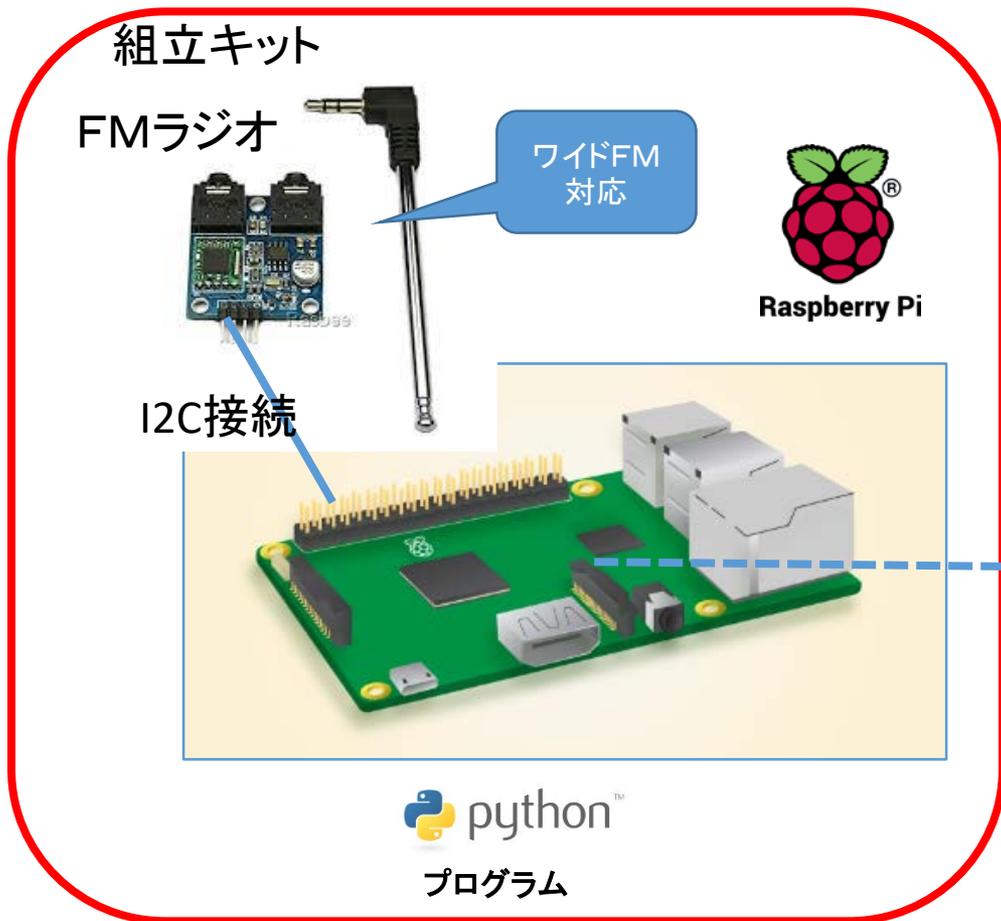
### • 目的

- FMラジオを使って、Pythonのプログラミングを習得することを目的とします。特にFMラジオとのI2C接続による、データ書き込み、データ読み出しと動的な画面作成、コマンド制御を行います。Pythonを使って開発を行う主要な内容を習得できます。

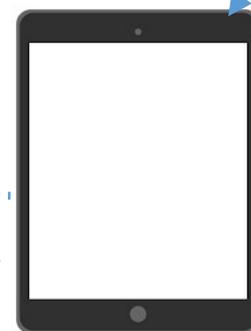
### • Pythonプログラミング内容

項番	項目	プログラム内容	難易度
1	フレーム作成	tkinterのフレーム作成	C
2	文字表示	Tkinterのラベル表示	C
3	ボタン作成とラジオ局のプリセット	tkinterのボタン表示、コマンド、I2Cによるラジオ局設定	C
4	時計表示	tkinterの動的ラベル表示	C
5	スクロールバーによる選局	tkinterのスクロール表示、コマンド	B
6	受信レベル表示	tkinterの動的ラベル表示、I2Cによるデータ読込	B
7	ステレオ／モノの表示	同上	B
8	地域別のFM局自動設定	Json形式の読込、ボタン自動作成、ラジオ局自動設定	A
9	FM局自動検索、結果出力	I2Cのデータ読込、スキャン、テキスト出力	A

## FMラジオ組立キット 2.全体像



FMラジオ  
画面例



VNC接続

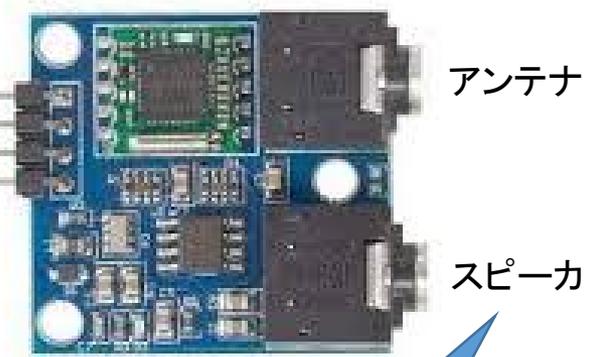
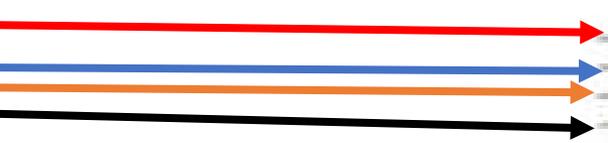
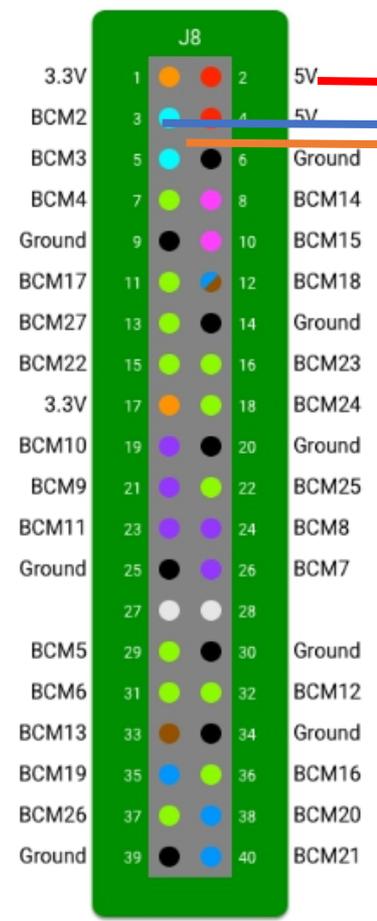
PC、タブレット等

スピーカ(ステレオミニジャック)は別途必要です。

# FMラジオ組立キット

## 3.FMラジオ接続

Raspberry PiとFMラジオの接続



メス-メスのジャンパで接続

Pi	FMラジオ
2	5V
3	SDA
5	SLC
6	GND

スピーカは付属されてません。音量調整付きのスピーカを接続してください

## FMラジオ組立キット 3.FMラジオ接続

### • 単体試験

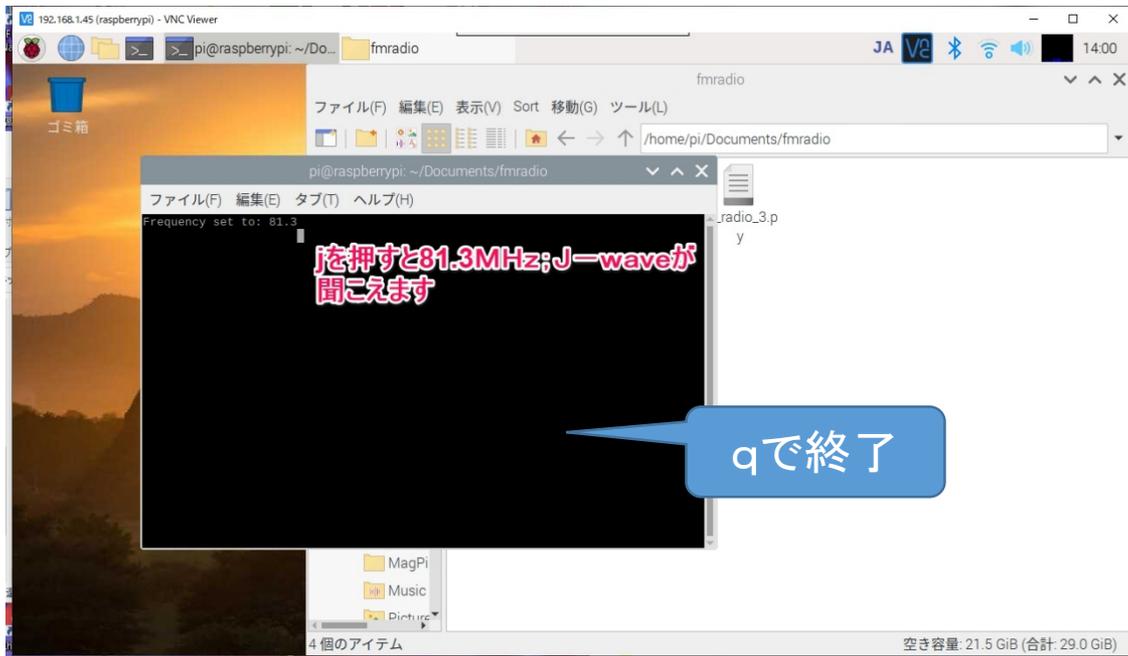
- コマンド画面からi2Cの接続を確認

```
$ cd /home/pi/Documents/fmradio
```

```
$ python3 radio_jp.py
```

- コマンド画面でjを押すと81.3MHzが聞こえます
- 周波数の変更は、64行目変更してください

このプログラムを  
基本に作成します。



全てpython3を  
使用します。

コマンド

```
$ cd /home/pi/Documents/fmradio
```

```
$ python3 radio_jp.py
```

```
デスクトップ#murakami_keno@ビジネス#技術関係#fmラジオ#radio_jp.py - sakura 2.2.0.1
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
40 freqL = freq14bit & 0xFF
41 data = [0 for i in range(4)]
42 init = 0x80
43 data[0] = freqL
44 data[1] = 0xB0
45 data[2] = 0x10
46 data[3] = 0x00
47 try:
48     i2c.write_i2c_block_data(address, init, data)
49     print("Radio Muted")
50 except IOError:
51     subprocess.call(['i2cdetect', '-y', '1'])
52
53
54 if __name__ == '__main__':
55     init_radio(i2c_address)
56     frequency = 81.3 # sample starting frequency
57     # terminal user input infinite loop
58     stdscr = curses.initscr()
59     curses.noecho()
60     try:
61         while True:
62             c = stdscr.getch()
63             if c == ord('j'): # set to 81.3
64                 frequency = 81.3
65                 set_freq(i2c_address, frequency)
66                 time.sleep(1)
67             elif c == ord('f'): # set to 80.0
68                 frequency = 80.0
69                 set_freq(i2c_address, frequency)
70                 time.sleep(1)
71             elif c == ord('w'): # increment by 1
72                 frequency += 1
73                 set_freq(i2c_address, frequency)
74                 sleep(1)
75             elif c == ord('s'): # decrement by 1
76                 frequency -= 1
```

プログラムのエディタには  
さくらエディタ等をお使いく  
ださい。

## FMラジオ組立キット 4.プログラム作成

### ① 画面フレーム作成

- Tkinter (pythonのGUIツールで一番メジャ) を使って画面を作成します。
- <https://docs.python.org/ja/3.7/library/tkinter.html>
- Wedgetが使える、gui1.pyをサンプルに作成してください。

```
$ cd /home/pi/Documents/fmradio/tkinter
```

```
$ python3 gui1.py
```

#### • フレームの表題を作成

- Radio\_jp.pyをコピーしてフレームを作成します  
メインルーチンの上の定義します。

```
class Application(tk.Frame):
```

```
    def __init__(self, master=None):
        super().__init__(master)
        self.pack()
        self.create_widgets()
        self.root = tk.Tk()
        self.root.mainloop()
    def create_widgets(self):
```

#### コマンド

```
$ cd
/home/pi/Documents/fmradio/tkinter
$ python3 gui1.py
```

fmradio/dist/に  
st\_radio\_3.pyで動作  
確認できます。

ヒントの  
み記載

```
pi@raspberrypi: ~/Documents/fmradio/tkinter
) タブ(T) ヘルプ(H)
5 6 7 8 9 a b c d e f
40: -----
50: -----
60: 60 -----
70: -----
pi@raspberrypi:~ $ cd /home/pi/Documents/fmradio
pi@raspberrypi:~/Documents/fmradio $ ls
ST_banner3.png fmstation.txt radio_jp.py st_radio_3.py
pi@raspberrypi:~/Documents/fmradio $ python3 radio_jp.py
pi@raspberrypi:~/Documents/fmradio $ python3
Python 3.7.3 (default, Apr 3 2019, 05:39:12)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
pi@raspberrypi:~/Documents/fmradio $ cd /home/pi/Documents/fmradio/tkinter
pi@raspberrypi:~/Documents/fmradio/tkinter $ ls
gui1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 gui1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 gui1.py
```

The screenshot shows a terminal window where the user navigates to the project directory and runs the program. Below the terminal, a window titled 'tkinter trial' is visible, containing three buttons: 'Hello', 'Clear', and 'QUIT'.

# FMラジオ組立キット

## 4.プログラム作成

### ④ 時計表示

- Tkinterのlabelを使って画面に時計を作成
- time1.pyをサンプルに作成してください。

```
$ python3 time1.py
```

- def create\_widgets(self)内等に作成してゆきます。

```
def create_widgets(self):
```

```
    self.w = tk.Label(self)
```

```
    self.w.pack(side="top")
```

```
def clock(self):
```

```
    now = datetime.now()
```

```
    timelabeled = ("%s年%s月%s日 %s時%s分%s秒" % (now.year, now.month,
now.day, now.hour, now.minute, now.second))
```

```
    self.w.config(text = timelabeled, )
```

```
    self.root.after(1000,self.clock)
```

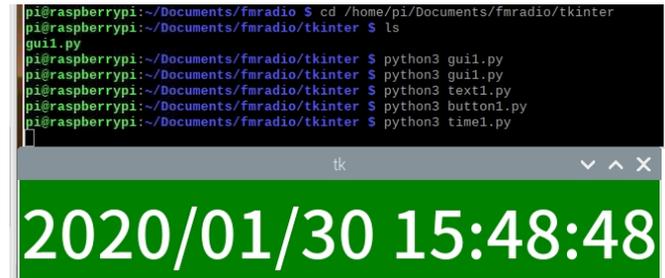
コマンド

```
$ cd
```

```
/home/pi/Documents/fmradio/tkinter
```

```
$ python3 time1.py
```

```
pi@raspberrypi:~/Documents/fmradio $ cd /home/pi/Documents/fmradio/tkinter
pi@raspberrypi:~/Documents/fmradio/tkinter $ ls
gui1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 gui1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 gui1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 text1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 button1.py
pi@raspberrypi:~/Documents/fmradio/tkinter $ python3 time1.py
```



動的表示  
個所

# FMラジオ組立キット

## 4.プログラム作成

コマンド

```
$ cd /home/pi/Documents/fmradio/
```

### ⑦ ステレオ／モノ表示

- FMラジオのI2C読み出しデータからステレオ／モノを抽出
- Tea5767spec.pdfの17ページ>table21 : 3byte目の7bit目になります。
- Stereo=0, それ以外がmonoになります。
- I2Cからのビット抽出方法

```
def stereo(self, address):
    """stereo,mono"""
    self.ste_text.set("")
    time.sleep(0.1)
    try:
        results=i2c.read_i2c_block_data(address,0)
        stmon = (results[2] & 0x80)>>7
        if stmon==0:
            ste="stereo"
        else:
            ste="mono"
        self.ste_text.set(ste)
    except IOError:
        subprocess.call(['i2cdetect', '-y', '1'])
```

Table 21. Format of 3rd data byte

7 (MSB)	6	5	4	3	2	1	0 (LSB)
STEREO	IF6	IF5	IF4	IF3	IF2	IF1	IF0

Table 22. Description of 3rd data byte bits

Bit	Symbol	Description
7	STEREO	Stereo indication: if STEREO = 1 then stereo reception; if STEREO = 0 then mono reception
6 to 0	PLL[13:8]	IF counter result

ビット抽出

Results[2]:2byte目のデータ(0byteから始まるので3番目) & 0x80:7bit目を抽出する16進(10000000)の係数 >>7:7ビット右にシフトして、7ビット目のみを抽出  
またstmonの値は、10進で表示されます。

変換サイト<https://hogegege.tk/tool/number.html>

# FMラジオ組立キット

## 4.プログラム作成

コマンド

```
$ cd /home/pi/Documents/fmradio/
```

### ⑧ 地域別のFM局自動設定

- FMラジオの地域別データをJson形式 (pythonの辞書形式) でデータを読み込み、FM局のプリセットボタンを自動設定します。
- サンプルデータ

```
station={"kanto":{"J-wave":81.3,"FM-Tokyo":80.0,"NHK-FM":82.5,"TBS":90.5,"文化放送":91.6,"日本放送":93.0}}
```

```
area="kanto"
```

- プリセットボタンの自動設定

```
def create_widgets(self):  
    buttons=[]  
    for i in station[area]:  
        freq=station[area][i]  
        buttons.append(tk.Button(self,text=i,command=self.select(freq)))  
    buttons[-1].pack(side="top",pady=10)
```

area:今後地域が追加された場合を想定して  
i:FM局名  
buttons:リスト形式で、周波数追加してゆきます

# FMラジオ組立キット

## 4.プログラム作成

コマンド

```
$ cd /home/pi/Documents/fmradio/
```

### ⑨ FM局自動検索、結果出力

- FM局を自動検索、検索結果をtxtで出力すると同時に、FM局を聴取します。
- 現在設定しているFM局の周波数をi2Cで読み出す関数を定義
- Tea5767spec.pdfの16、17ページ>table 17: 1byte目の0-5bit目、table 19: 2byte目の0-7bit目になります。

```
def getFreq(self,address):
```

```
    frequency = 0.0
```

```
    results=i2c.read_i2c_block_data(address,0)
```

```
    frequency = ((results[0]&0x3F) << 8) + results[1]
```

```
    # Determine the current frequency using the same high side formula as above
```

```
    frequency = round(frequency * cof / 4 - 225000) / 1000000;
```

```
    return frequency
```

Table 17. Format of 1st data byte

7 (MSB)	6	5	4	3	2	1	0 (LSB)
RF	BLF	PLL13	PLL12	PLL11	PLL10	PLL9	PLL8

Table 18. Description of 1st data byte bits

Bit	Symbol	Description
7	RF	Ready Flag: if RF = 1 then a station has been found or the band limit has been reached; if RF = 0 then no station has been found
6	BLF	Band Limit Flag: if BLF = 1 then the band limit has been reached; if BLF = 0 then the band limit has not been reached
5 to 0	PLL[13:8]	setting of synthesizer programmable counter after search or preset

Table 19. Format of 2nd data byte

7 (MSB)	6	5	4	3	2	1	0 (LSB)
PLL7	PLL6	PLL5	PLL4	PLL3	PLL2	PLL1	PLL0

Results[0]:1byte目のデータ(0byteから始まるので1番目)  
 & 0x3F:0-5bit目を抽出する16進(111111)の係数  
 <<8:8ビット左にシフトして、次の2byte目と接続する  
 Cof:このFMラジオのクリスタルオシレータの周波数、周波数の計算式は、このFMラジオ固有のものになります

## FMラジオ組立キット 4.プログラム作成

コマンド

```
$ cd /home/pi/Documents/fmradio/
```

### ⑨ FM局自動検索、結果出力

- FM局を自動検索、検索結果をtxtで出力すると同時に、FM局を聴取します。
- 検索結果出力例

Fmstation.txt 例

```
2020-01-28 17:36:28.436764 81.3MHz, signal:13  
2020-01-28 17:36:43.752500 90.5MHz, signal:15  
2020-01-28 17:46:09.716881 81.3MHz, signal:14  
2020-01-28 17:46:17.784808 82.5MHz, signal:13  
2020-01-28 17:48:02.905317 81.3MHz, signal:13
```

現在、開始は、スクロールバーの位置からとなりますが、スクロールバーと連携していないため、自動検出後、次の局を検索する場合にバーを動かす必要があります。今後連携を検討

## FMラジオ組立キット 5.プログラム例

### •プログラム例

```
$ cd /home/pi/Documents/fmradio/dist  
$ python3 st_radio_3.py 動作確認できます。  
• プログラムは難読化しています。
```

### •プログラムの送付

- Pythonプログラムの弊社作成例を商品到着後、1-2週間後に送付しますので、連絡ください。その際には、購入日とお名前を連絡してください。

コマンド

```
$ cd  
/home/pi/Documents/fmradio/dist  
$ python3 st_radio_3.py
```

連絡先

スペクトラム・テクノロジー株式会社

村上 正彦

TEL: 04-2990-8881 電話は平日の10-17時のみ

E-mail: [sales@spectrum-tech.co.jp](mailto:sales@spectrum-tech.co.jp)

HP: <https://spectrum-tech.co.jp/>

line@ : @htr2462r

# FMラジオ組立キット

## 6. 今後の検討課題

### ① 検索時のスクロールバー連携

- 現在、開始は、スクロールバーの位置からとなりますが、スクロールバーと連携していないため、自動検出後、次の局を検索する場合にバーを動かす必要があります。今後連携を検討

### ② 地域を選択機能

- 現状はプログラム上で、地域を設定しているが、リストボックス又はmysetなどの設定ファイルで作り込み
- 地域毎のFM局データはDBなどで作成し、全国対応

### ③ 録音機能

- タイマ設定による録音機能

### ④ グラフ化

- 受信レベルをグラフ化