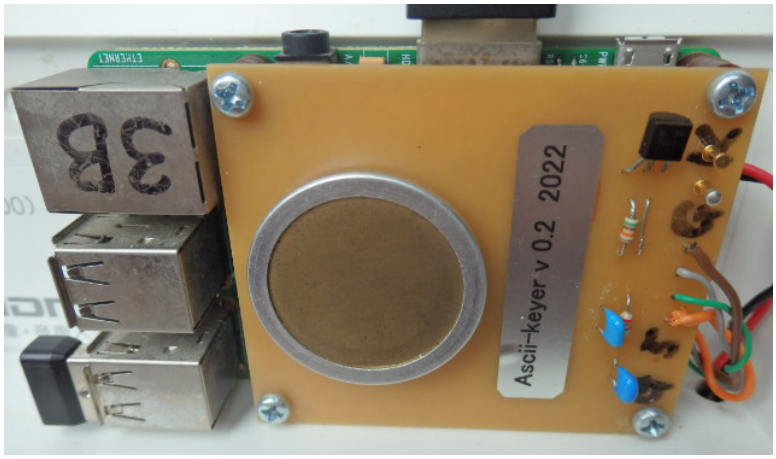
テキスト・ファイルをモールス符号化

RaspberryPi Morse code Python Text file W1AW Qualifying Runs JJC



モールス符号器になった **RaspberryPi**

1 使っていない RaspberryPi の活用

現在の RaspberryPi は、64 ビット CPU を搭載した 4B モデルが主流です。モデル 2B やモデル 3B はそれなりの能力を持っているのですが、引き出しの中で眠っている有様ではないでしょうか。

この稿では、特に使い道がなくて空いている RaspberryPi を再活用して、Python プログラムを演習する方法を示します。そのテーマとして、モールス符号を生成するハードウェアとソフウェアをとりあげます。プロジェクトの名前を Text-keyer とします。

2 Text-keyer の機能

機械送信のモールス発生器には「メモリー機能付きのエレキー」がありますが、それはモールス符号習得者がその能力を使ってモールス符号を生成するので、半手動式の機械モールス符号発生器だといえます。

Text-keyer はそうではなく、操作者にモールス符号化の能力がなくても、PC から送信すべき文字を送ればモールス符号が生成されます。

ARRL の W1AW からは「Qualifying Runs」として、普通文モールス符号によるニュースが放送されています。日本でも、かつては共同通信社から JJC 短波放送が発射されていました。Text-keyer はこれらに似て、用意されたテキスト・ファイルを自動的にモールス符号化する機能をもちます。

長文のモールス符号送信があれば、モールス符号の受信練習に役立ちます。

モールス符号を生成したテキストは、PC 画面に次図のように現れます。

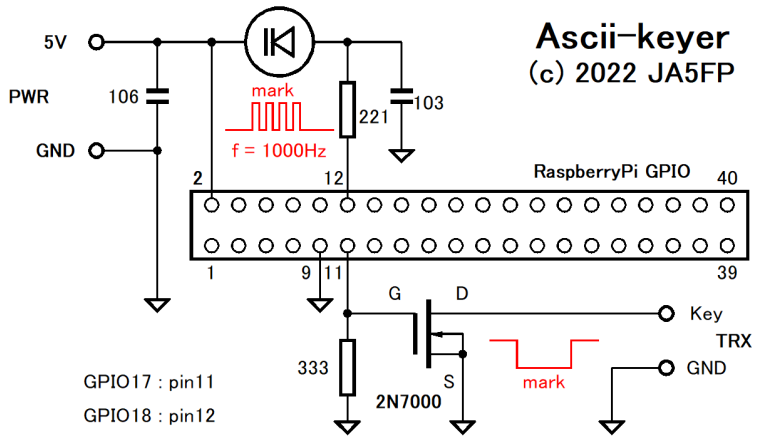
```
Shell
>>> %Run ascii-keyer_v1.2.py
  ワカ゛ハイハネコテ゛アル」
  ナマエハマタ゛ナイ」

  ト゛コテ゛ウマレタカトントケントウカ゛ツカヌ」
  ナンテ゛モウスクライシ゛メシ゛メシタトコロテ゛ニヤーニヤーナイテイタコトタ゛ケハキオクシテイル」
  ワカ゛ハイハココテ゛ハシ゛メテニンケ゛ントイウモノヲミタ」
  シカモアトテ゛キクトソレハシヨセイトイウニンケ゛ンチユウテ゛イチハ゛ント゛ウアクナシユソ゛クタ゛ソウタ゛」
  コノシヨセイトイウノハトキト゛キワレワレヲツカマエテニテクウトイウハナシテ゛アル」
  シカシソノトウシ゛ハナントイウカンカ゛エモナカツタカラヘ゛ツタ゛ンオソロシイトモオモワナカツタ」
  タタ゛カレノテノヒラニノセラレテスートモチアケ゛ラレタトキナンタ゛カフワフワシタカンシ゛カ゛アツタハ゛カリテ゛アル」

Python 3.7.3
```

3　Text-keyer のハードウェアは簡単

RaspberryPi の GPIO に簡単なアダプタを付けるだけで、Text-keyer が作れます。それは、無線機のキーイング回路と符号のモニタ回路だけです。参考回路は次図のとおりです。



4　Python のソースコード

```
''' "ascii-keyer.py v 1.2" (c) 2022 Yukihisa Aida, JA5FP
    A python script, reads stored text file, converts it's ascii characters to adaptive morse-cod
    then keys on/off CW transmitter by these symboles stream.
    Hard ware requires a RaspberryPi computer and a tiny circuit to drive keyer of transmitter.
    Usage is simple, import "info.txt" file or type-in your own message on the script.
    If you need to change keying speed or message, edit "wait" value or "message" text.
    Word spacing is set to 5 dots length (not fit regular code standard as 7 dots length).
    Befor using, type "sudo pigpiod" to use library.
'''

# coding: utf-8
#!/usr/bin/env python
import os
```

```
import time
import pigpio

#KEING SPEED in second of dot
dot=0.1

#MESSAGE to send
with open("neko.txt") as source:                    # message from file
    message = source.read()
#message = 'CQ CQ CQ de JA5FP JA5FP JA5FP k'        # own message
#message = ' モールスムセンツウシンニハ、ベツヒヨウダイ一ゴウニカカゲルモールスフゴウヲ
モチイナケレバナラナイ」'

pi = pigpio.pi()
pi.set_mode(17, pigpio.OUTPUT)                      # keyed TTL at pin11
pi.set_mode(18, pigpio.OUTPUT)                      # keyed tone at pin12

MORSE_CODE = {'A':'-+', 'B':'+---', 'C':'+-+-', 'D':'+--', 'E':'-',
              'F':'--+-', 'G':'++-', 'H':'----', 'I':'--', 'J':'-+++',
              'K':'+-+', 'L':'-+--', 'M':'++', 'N':'+-', 'O':'+++',
              'P':'-++-', 'Q':'++-+', 'R':'-+-','S':'---', 'T':'+',
              'U':'--+', 'V':'---+', 'W':'-++', 'X':'+--+', 'Y':'+-++', 'Z':'++--',
              '1':'-++++', '2':'--+++', '3':'---++', '4':'----+', '5':'-----',
              '6':'+----', '7':'++---', '8':'+++--', '9':'++++-', '0':'+++++',
              chr(10):'L', chr(32):'S', chr(33):'+-+-++', chr(34):'-+--+-',
              chr(36):'---+--+', chr(38):'-+---', chr(39):'-++++-', chr(40):'+-++-',
              chr(41):'+-++-+', chr(42):'+-++-+', chr(43):'-+-+-', chr(44):'++--++',
              chr(45):'+----+', chr(46):'-+-+-+', chr(47):'+--+-', chr(58):'+++---',
              chr(59):'+-+-+-', chr(61):'+---+', chr(63):'--++--', chr(64):'-++-+-',
              chr(91):'+-++-', chr(93):'+-++-+', chr(95):'--++-+',
              'ア':'++-++', 'イ':'-+', 'ウ':'--+',  'エ':'+-+++',  'オ':'-+---',
              'カ':'-+--', 'キ':'+-+--', 'ク':'---+', 'ケ':'+-++', 'コ':'++++',
              'サ':'+-+-+', 'シ':'++-+-', 'ス':'+++-+', 'セ':'-+++-', 'ソ':'+++-',
              'タ':'+-', 'チ':'--+-', 'ツ':'-++-', 'テ':'-+-++', 'ト':'--+--',
              'ナ':'-+-', 'ニ':'+-+-', 'ヌ':'----', 'ネ':'++-+', 'ノ':'--++',
              'ハ':'+---', 'ヒ':'++--+', 'フ':'++--', 'ヘ':'-', 'ホ':'+--',
              'マ':'+--+', 'ミ':'--+-+', 'ム':'+', 'メ':'+---+', 'モ':'+--+-',
              'ヤ':'-++', 'ユ':'+--++', 'ヨ':'++',
              'ラ':'---', 'リ':'++-', 'ル':'+-++-', 'レ':'+++', 'ロ':'-+-+',
              'ワ':'+-+', 'ヰ':'-+--+', 'ヱ':'-++--', 'ヲ':'-+++', 'ン':'-+-+-',
              '一':'-++++', '二':'--+++', '三':'---++', '四':'----+', '五':'-----',
              '六':'+----', '七':'++---', '八':'+++--', '九':'++++-', '〇':'+++++',
              '^^e3^^82^^99':'--', '^^e3^^82^^9a':'--++-', 'ー':'-++-+', '」':'-+-+--',
```

```
              ' (':'+-++-+', ') ':'-+--+-', '、':'-+-+-+'
             }

def main():
    while True:
        print(message + chr(10))
        length = len(message)
        i = 0
        while length:
            code = MORSE_CODE[message[i].upper()]
            print(message[i].upper(), end='')
            for cipher in code:
                if(cipher == '+'):                # dash
                    pi.write(17, True)
                    pi.hardware_PWM(18, 880, 500000)
                    time.sleep(dot*3)             # dash length 3:standard 2:2dots

                    pi.write(17, False)
                    pi.hardware_PWM(18, 0, 0)
                    time.sleep(dot)               # tail length
                if(cipher == '-'):                # dot
                    pi.write(17, True)
                    pi.hardware_PWM(18, 880, 500000)
                    time.sleep(dot)               # dot length wait:standard
                    pi.write(17, False)
                    pi.hardware_PWM(18, 0, 0)
                    time.sleep(dot)               # tail length
                if(cipher == 'S'):
                    time.sleep(dot*0)             # word separation 0:5dots 1:6dots 2:7dots
            time.sleep(dot*2)                     # character separation wait*2:standard
            i += 1
            length -= 1
        time.sleep(10)                            # seconds for next transmit.
        print(chr(10) )
        print("*** 繰り返します。 ***")

if __name__ == "__main__":
    main()
```