楽しいエレクトロニクス工作

第 88 回 CW スピード

JA3FMP 櫻井紀佳

アマチュア無線で CW を運用しているとき、相手方の CW スピードが分かると便利なことがあります。また自分で打っている CW スピードも気になります。電信符号はそれぞれ文字に対する符号の長さが異なり、単位時間当たりの文字数を定義するのは元々難しいと思いますが、参考になる資料があります。

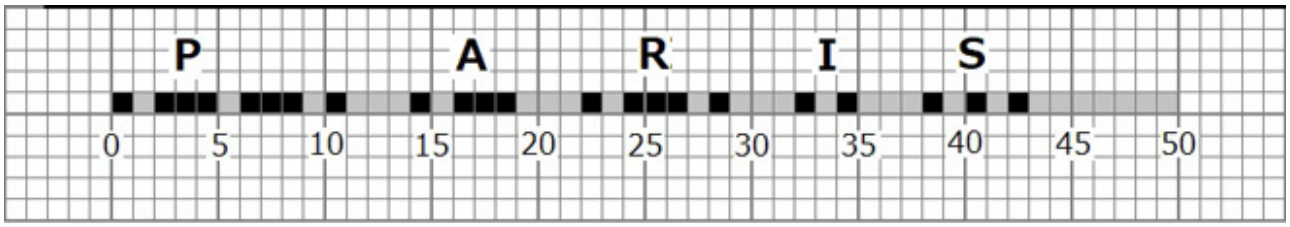
CW の速度は、一般的に WPM(Word Per Minute)で定義され、Word の標準は「PARIS」とされているようです。

短点を 1 単位とした基準で考えると

P = ・— —・	短点 2、長点 2x3、間隔 3、文字間隔 3	小計 14 単位
A = ・—	短点 1、長点 1x3、間隔 1、文字間隔 3	小計 8 単位
R = ・—・	短点 2、長点 1x3、間隔 2、文字間隔 3	小計 10 単位
I = ・・	短点 2、長点 0x3、間隔 1、文字間隔 3	小計 6 単位
S = ・・・	短点 3、長点 0x3、間隔 2、語の間隔 7	小計 12 単位
		合計 50 単位

PARIS は 5 文字なので、文字単位 CPS(Character Per Minute)にすると 1WPM = 5CPM になります。

単純計算　50 単位/5 文字 = 1 文字平均 10 単位



もし 25CPM なら 25 文字/分なので 250 単位/分となり、CW の最速を 125 文字/分とすると、5 倍になって 1,250 単位/分となり、1 単位当たりの時間はそれぞれ 240ms と 48ms となるはずです。

従って現在の 1 単位時間の計測ができれば、**60/1 単位時間 x 10** で 1 分当たりの概略の通信速度が分かります。そこで 1 単位の時間を計測する方法を考えてみます。

ハード的処理では処理する事項が簡単な割には回路が複雑になるように思われ今回はソフト処理を考えることにしました。

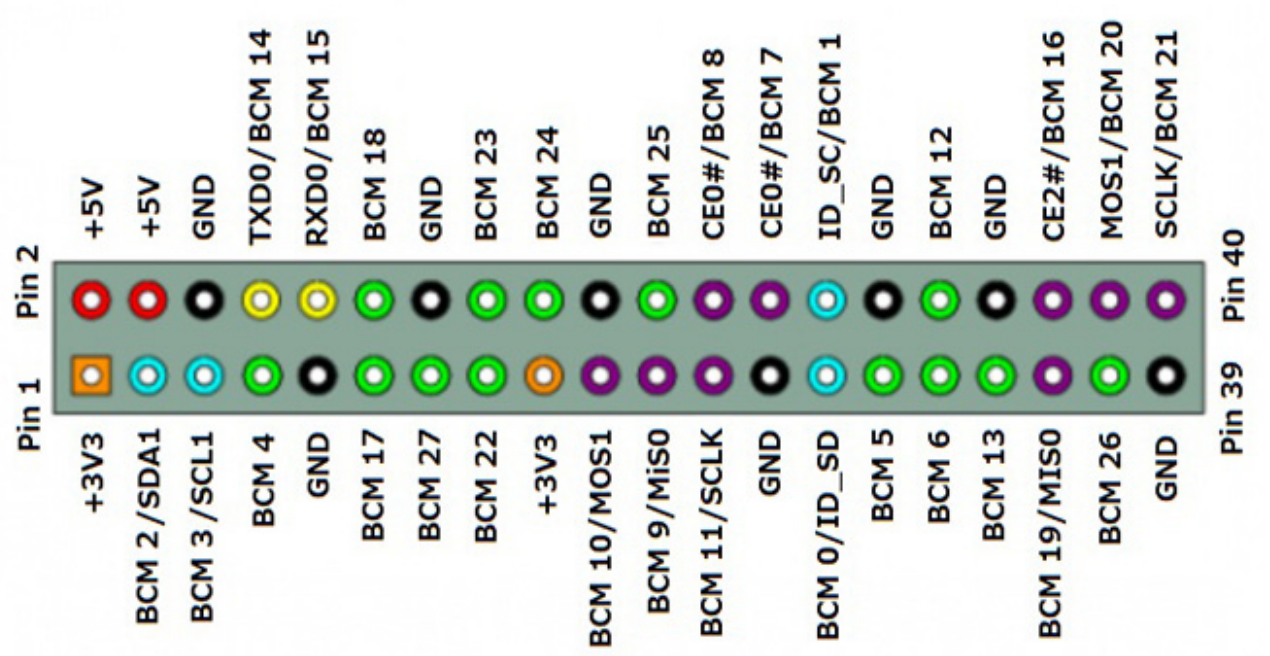
ソフト処理については、以前に買って持っていたシングルボードコンピューター Raspberry Pi 3(以下ラズパイ)を使ってみることにします。このラズパイは多くの入出力端子がついていて、それを制御するソフトも公開されているのでこの用途に使えると思います。

元々ラズパイはボード単位で販売されていますが、液晶ディスプレイとキーボードも一緒に買ってきました。ラズパイ本体の大きさは小さいのでディスプレイの裏側に取り付けていて全体は次のようなものです。

ラズパイ全体(左)

ディスプレイ裏のラズパイ

この写真で分かる通り、このラズパイには多くの入出力端子があって個別に入出力の指定ができるようになっていますが、ある組み合わせで UART のように特別に意味ある組み合わせを指定することもできます。具体的な端子の配列は次のようになっています。



これらの GPIO(General Purpose Input/Output)端子は物理的な番号付け以外に BCM「Broadcom の SOC チャネル」番号が割り振られていてこれで指定することができます。今回は BCM 23 ピンを入力端子として使うことにします。

信号処理のやり方は、CW 符号の内の短点を拾い出して、短点を時間サンプリングすることにします。入力される信号で一番遅い短点は 25 文字/分の 240ms で、一番早い短点が 125 文字/分の 48ms なので、この 48ms に対応したサンプリングとして 10ms としてみました。5 回弱となって少し粗いかもしれませんがこれで試してみます。

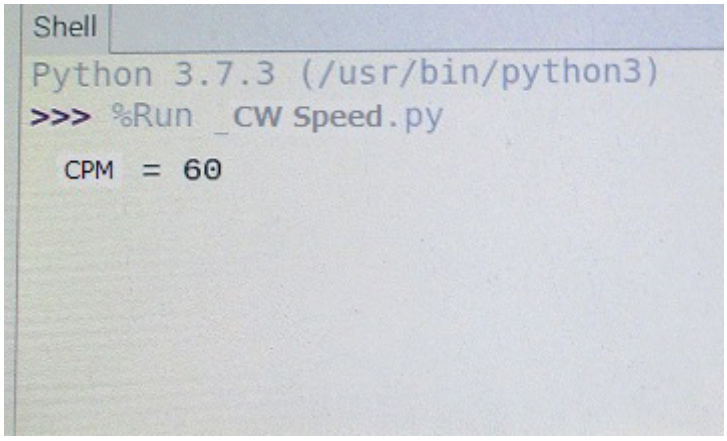
入力端子 BCM 23 はソフトでプルアップしているので、入力された信号が L であればカウントして 10ms の間 sleep し、L の間カウントを加算して信号が H になるまで繰り返します。これによって信号を 10ms でサンプリングしたことになります。短点以外に当然長点も入力されますので、一度入力された信号を測定し、その信号の半分より短い信号が入ればそちらを短点として扱い、その信号の 2 倍より長い信号は長点としてパスするようにしました。

ラズパイには Thonny Python IDE(Integrated Development Environment)という統合開発環境がついていて、この上でソフトの開発ができます。使い方や特徴など多くの情報がインターネットに載っているのでそちらを見て頂きたいのですが、コンパイラーではないので昔の BASIC に似ていてコードを書いて即実行できます。

そのソフトは次のようなものです。

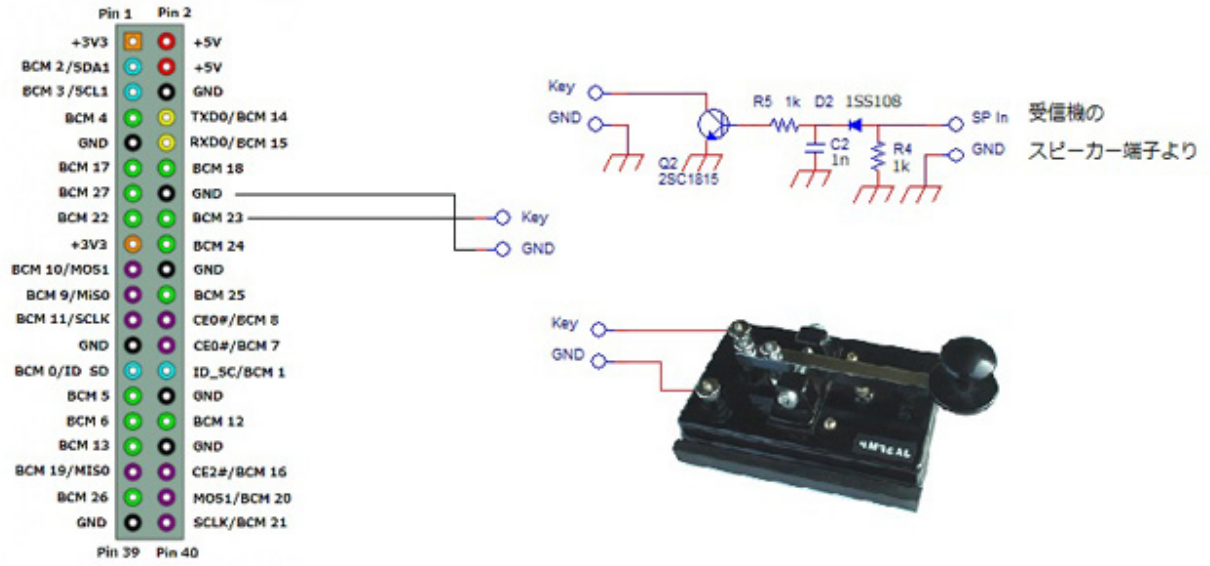
```
import RPi.GPIO as GPIO
import time

# st=Sampling Time, dt=Temporary Dot Time, dr=Befor dt

st=0.01

while True:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(23,GPIO.IN,pull_up_down=GPIO.PUD_UP)
    dt=0
    dr=48

    while 0==GPIO.input(23):
        time.sleep(st)
        dt=dt+st

    if dt!=0:
        if dt>(2*dr):
            dt=dr
        dr=dt

        CPM=60/(dt*10)
        print("CPM =",int(CPM))

    GPIO.cleanup()
```

このソフトを動かした結果は次のようになりました。



この結果の表示を大きな文字にしたいのですがソフトが得意ではなくすぐに対応できていません。

今回の実験では GPIO の端子直接キーイングしてみましたが、受信信号に接続しなければ意味がありません。実はスピーカーからでる電信音でキーイングするのは結構難しくて、過去の実験記事「第 31 回続 CW 復調改善 その 2」を参考にインターフェースを考えています。



GPIO とのインターフェース

今回入力端子を直接キーイングしているので結構バラつきが大きく、平均値が出るようにソフトを変更した方が良いかも知れません。また受信した信号とのインターフェースはもう少し検討してみます。