IoT 実験用 IO ボード上のタクト・スイッチの押下回数で待ち時間（分）を設定し，設定した待ち時間後に圧電スピーカからピンポン音を鳴らすキッチン・タイマーを製作してみましょう．

サンプル・プログラム example13_chime_timer.py を実行し，LED が白色に点灯してからタクト・スイッチを押すと，押すたびに待ち時間の設定値が 1 分ずつ増えるとともに，LED の色が赤色→緑色→黄色→青色→赤紫色→藍緑色→白色と変化します．LED の色番号 1（赤色）～6（藍緑色）は，待ち時間 1～6 分を示し，色番号 7（白色）は 7 分以上（または未設定状態）を示します．

ボタン操作を止めると（1 秒間，タクト・スイッチを押さなかったとき），設定を完了し，キッチン・タイマーを開始します．タイマー作動中は，1 秒ごとに LED を点滅させ，圧電スピーカから「チッチッ」と秒を刻む音を出力し，点滅色は残り時間（分）を示します．残り時間が無くなると，ピンポン音と LED 色の変化でユーザへタイマーの終了を知らせます．

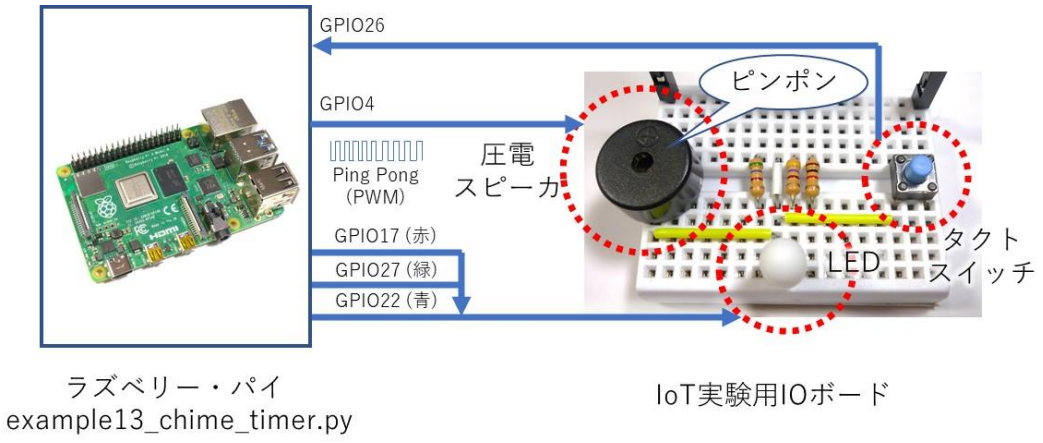

図 1 実験用 IO ボードの全機能を使ってラーメン・タイマーを製作してみよう
タクト・スイッチを押すたびに待ち時間が 1 分ずつ増えるとともに，LED の色が変化する．操作をやめると，タイマーを開始し，残り時間（分）を LED の色で示す

以下に，子機・送信側となる**リスト 1** の example13_chime_timer.py の主な処理内容について説明します．

① カラーLED を 7 色に点灯制御するための関数 led3 を定義します．引数の変数 color は，色番号です．引数の色番号が 0 のときは消灯，1～7 のときは GPIO17，27，22 に接続した LED 素子を色番号に応じて制御します．また，引数値、色番号，色名の表示出力と、処理②の関数 chime によるクリック音のスピーカ出力を行います．引数に 7 よりも大きい値を入力した時は，色番号 7 として LED を白色に点灯制御します．

② 圧電スピーカを制御する関数 chime を定義します．引数の freq は周波数（Hz），変数 t は鳴音時間（秒）です．一例として，処理⑥内の chime(ping_f ,0.5)の場合は，554Hz の音を 0.5 秒間，鳴らします．

③ 以下の処理④～⑥を繰り返し実行します．例外処理 try の expect 部は，LXTerminal からキー割り込み（キー操作：[Ctrl]＋[C]）があったときに GPIO の設定を開放してプログラムを終了するための処理部です．

④ タクト・スイッチによる待ち時間設定のためのユーザ入力処理部です．関数 time で取得した現在時刻に 1.0 秒を加算した値を変数 t へ代入し，時刻が t に達していないときに繰り返し処理を行います．タクト・スイッチが押されたら，変数 color に 1 を加算し，処理①で定義した関数 led3 を実行します．

⑤ キッチン・タイマーの待ち時間処理部です．待ち時間は，変数 color（分）です．変数 color（分）に 60 を乗算して秒数に変換し，変数 t にタイマー終了時間を代入してから，時刻が t に達するまで LED の消灯と点灯を繰り返し処理します．関数 led3 の引数（点灯色）には，残り時間（分）に応じた値を渡します．(t - time()) / 60 で残り時間（分）を計算し，関数 int で小数点以下を切り捨ててから，1 を加算しました．

⑥ 処理⑤の待ち時間が完了したときにピンポン音を鳴らす処理部です．LED を赤紫色→藍緑色と制御し，圧電スピーカを 554Hz で 0.5 秒間，440Hz で 0.5 秒間，鳴音制御を行います．

リスト 1　キッチン・タイマーのサンプル・プログラム example13_chime_timer.py

```
#!/usr/bin/env python3
# Example 13' チャイム - キッチン・タイマー

port_chime = 4                                  # 圧電スピーカ用 GPIO ポート番号
port_btn = 26                                   # ボタン用 GPIO ポート番号
ping_f = 554                                    # 圧電スピーカ用 周波数 1
pong_f = 440                                    # 圧電スピーカ用 周波数 2
ports = [17, 27, 22, port_chime]                # 赤,緑,青色の LED とスピーカ GPIO
colors= ['消灯','赤色','緑色','黄色','青色','赤紫色','藍緑色','白色']

from RPi import GPIO                            # GPIO モジュールの取得
from time import sleep, time                    # sleep と time モジュールの取得

def led3(color):                                ←①
    if color > 0:                               # 色番号が 0 以上のとき
        print('Timer =', color, 'min. ',end='') # 色番号の指示値を表示
        chime(ping_f ,0.01)                     # クリック音
        if color > 7 or color < 0:              # 色番号の範囲外のとき
            color = 7                           # 白色(7)に設定
        print('Color =', color, colors[color])  # 色番号を表示
    for i in range(3):                          # LED 用ポート番号を変数 i へ
        port = ports[i]                         # ポート番号を ports から取得
        b = (color >> i) & 1                    # 該当 LED への出力値を変数 b へ
        GPIO.output(port, b)                    # ポート番号 port の GPIO を出力に

def chime(freq ,t):                             ←②
    pwm.ChangeFrequency(freq)                   # PWM 周波数の変更
    pwm.start(50)                               # PWM 出力を開始. デューティ 50%
    sleep(t)                                    # t 秒の待ち時間処理
    pwm.stop()                                  # PWM 出力停止

GPIO.setmode(GPIO.BCM)                          # ポート番号の指定方法の設定
for port in ports:                              # 各ポート番号を変数 port へ代入
    GPIO.setup(port, GPIO.OUT)                  # ポート番号 port の GPIO を出力に
GPIO.setup(port_btn, GPIO.IN, pull_up_down=GPIO.PUD_UP) # GPIO 26 を入力に
pwm = GPIO.PWM(port_chime, ping_f)              # PWM 出力用のインスタンスを生成

try:                                            # キー割込(Ctrl+C)の監視を開始
    while True:                                 ←③ # 繰り返し処理
        led3(colors.index('白色'))              # LED を白色に
        color = 0                               # 色番号 0 (消灯) を設定
        t = time() + 1.0                        # タイムアウト変数 t を 1 秒後に設定
        while color == 0 or time() < t:         # 色番号が 0 または時間 t 以内のとき
            b = GPIO.input(port_btn)            # ボタン入力値を変数 b へ代入
④           if b == 0:                          # ボタンが押されていた時
                color += 1                      # 色番号に 1 を追加
                led3(color)                     # LED を色番号で点灯
                while b == 0:                   # ボタンが押されたままのとき
                    b = GPIO.input(port_btn)    # 押されている間は待機する
                sleep(0.1)                      # チャタリング防止
                t = time() + 1.0                # タイムアウト変数 t を 1 秒後に設定
        t = time() + 60 * color                 # タイマー終了時間 t を color 分後に
        while time() < t:                       # タイマー終了するまで繰り返し
⑤           led3(0)                             # LED を消灯に
            sleep(0.5)                          # 0.5 秒の待ち時間処理
            led3(int((t - time()) / 60) + 1)    # タイマー残り時間(分)
            sleep(0.5)                          # 0.5 秒の待ち時間処理
        led3(colors.index('赤紫色'))            # LED を赤紫色に
⑥       chime(ping_f ,0.5)                      # ブザー「Ping」音
        led3(colors.index('藍緑色'))            # LED を藍緑色に
        chime(pong_f, 0.5)                      # ブザー「Pong」音
except KeyboardInterrupt:                       # キー割り込み発生時
    print('¥nKeyboardInterrupt')                # キーボード割り込み表示
    for port in ports:                          # 各ポート番号を変数 port へ代入
        GPIO.cleanup(port)                      # GPIO を未使用状態に戻す
    GPIO.cleanup(port_btn)                      # GPIO を未使用状態に戻す
```