```python
import RPi.GPIO as GPIO
import time
import os

# GPIOピンの設定
BUTTON1_PIN = 18  # ボタン1（開始・停止）
BUTTON2_PIN = 23  # ボタン2（リセット・ラップ）

# 状態の定義
INITIAL = 0       # 初期状態
RUNNING = 1       # 計測中
PAUSED = 2        # 一時停止中

# グローバル変数の初期化
start_time = 0       #開始した時刻
elapsed_time = 0     #計測した時間
state = INITIAL      #ストップウォッチの状態
lap_times = []       #ラップの保存


# ボタン1（開始・停止・再開）のコールバック関数
def start_stop_callback(channel):
    global state, start_time, elapsed_time#グローバル変数を関数内で使う

    if state == INITIAL:
        # 初期状態なら計測開始
        start_time = time.time()#計測開始時刻を記録
        state = RUNNING
        #print("¥n計測開始")

    elif state == RUNNING:
        # 計測中なら一時停止
        elapsed_time += time.time() - start_time
        #これまでの計測時間を計算
        state = PAUSED
        #print(f"¥n一時停止: 経過時間 = {elapsed_time:.3f} 秒")

    elif state == PAUSED:
        # 一時停止中なら計測再開
        start_time = time.time()#再計測開始時刻を記録
        state = RUNNING
        #print("¥n計測再開")

# ボタン2（リセット・ラップ）のコールバック関数
def reset_lap_callback(channel):
    global state, elapsed_time, lap_times, start_time

    if state == RUNNING:
        # 計測中ならラップタイムを記録
        current_time = time.time() - start_time + elapsed_time
        lap_times.append(current_time)#配列に要素を追加する関数(append)
        #print(f"¥nラップ {len(lap_times)}: {current_time:.2f} 秒")
```

```
52
53      elif state == PAUSED or state == INITIAL:
54          # 一時停止中または初期状態ならリセットして初期状態に戻す
55          elapsed_time = 0
56          lap_times = []
57          state = INITIAL
58          #print("¥nリセット: 経過時間とラップタイムをクリア")
59
60
61  # GPIOを初期化
62  GPIO.setmode(GPIO.BCM)
63  GPIO.setup(BUTTON1_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
64  GPIO.setup(BUTTON2_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
65
66
67  # 割り込みイベントを設定（開始）
68  GPIO.add_event_detect(BUTTON1_PIN, GPIO.FALLING, callback=
    start_stop_callback, bouncetime=300)
69  GPIO.add_event_detect(BUTTON2_PIN, GPIO.FALLING, callback=
    reset_lap_callback, bouncetime=300)
70
71  print("ストップウォッチ準備完了")
72
73
74  try:
75      # メインループ
76      while True:
77          #
            画面クリアする関数（画面をクリアすることで表示を更新し続ける）
78          os.system('clear')
79
80          # 現在の経過時間を計算
81          if state == RUNNING:
82              current_time = time.time() - start_time + elapsed_time
83          else:
84              current_time = elapsed_time
85
86          # 現在の経過時間を表示
87          print(f"経過時間: {current_time:.2f} 秒")
88          # ラップタイムの表示
89          for i, lap in enumerate(lap_times, start=1):
90              print(f"ラップ {i}: {lap:.2f} 秒")
91
92
93          time.sleep(0.01) #約0.01秒ごとに表示を更新
94
95  except KeyboardInterrupt:
96      pass
97  finally:
98      GPIO.cleanup()
99      print("¥nプログラム終了")
100
```