```
import RPi.GPIO as GPIO # GPIOを利用する
import time             # sleepを利用する

# ポート番号の定義
SWITCH = 18
LED = 21
LED_BLINKING = 1
LED_LIGHTING = 2#モードの定義
led_mode = LED_BLINKING#モードを始めは点滅にしておく

# GPIOの初期化 --- (*1)
GPIO.setmode(GPIO.BCM)
GPIO.setup(SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(LED, GPIO.OUT)


#コールバック関数の定義
def callback_change_switch(ch):
    global led_value
    print("callback", ch)
    if ch != SWITCH: return
    if led_mode == LED_BLINKING:
        led_mode = LED_LIGHTING
    else:
        led_mode = LED_BLINKING


# イベントを設定 --- (*3)
GPIO.add_event_detect(
    SWITCH, # ポート番号
    GPIO.RISING, # イベントの種類
    callback=callback_change_switch, # 関数の指定
    bouncetime=200) # 連続イベントを制限

# LEDを消灯しておく
GPIO.output(LED, GPIO.LOW)


try:
    while True:
        if led_mode ==:#穴埋め
            GPIO.output(LED, GPIO.HIGH)
            time.sleep(0.5)
            GPIO.output(LED, GPIO.LOW)
            time.sleep(0.5)
        else:
            GPIO.output(LED, GPIO.HIGH)
            time.sleep(0.5)

except KeyboardInterrupt:
    GPIO.cleanup()

```

```python
import RPi.GPIO as GPIO # GPIOを利用する
import time             # sleepを利用する

# ポート番号の定義
SWITCH = 18
LED = 21
LED_BLINKING = 1#モードの定義
LED_LIGHTING = 2#モードの定義
led_mode = LED_BLINKING

# GPIOの初期化 --- (*1)
GPIO.setmode(GPIO.BCM)
GPIO.setup(SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(LED, GPIO.OUT)

# LEDを消灯する
GPIO.output(LED, GPIO.LOW)

try:
    while True:
        if GPIO.input(SWITCH):#ポーリング
            if led_mode == LED_BLINKING:
                led_mode = LED_LIGHTING#点灯
            else:
                led_mode = LED_BLINKING#点滅

        if led_mode == LED_BLINKING:
            GPIO.output(LED, GPIO.HIGH)
            time.sleep(0.5)
            GPIO.output(LED, GPIO.LOW)
            time.sleep(0.5)
        else:
            GPIO.output(LED, GPIO.HIGH)
            time.sleep(0.5)

except KeyboardInterrupt:
    GPIO.cleanup()
```

```
import RPi.GPIO as GPIO # GPIOを利用する
import time             # sleepを利用する

# ポート番号の定義
SWITCH = 18
LED = 21
LED_BLINKING = 1#モードの定義
LED_LIGHTING = 2#モードの定義
led_mode = LED_BLINKING
bouncetime = 0.2#連続のボタンの受付時間[sec]
time_switch_previous = 0#前回ボタンをおしてからの経過時間

# GPIOの初期化
GPIO.setmode(GPIO.BCM)
GPIO.setup(SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(LED, GPIO.OUT)

# LEDを消灯する
GPIO.output(LED, GPIO.LOW)

time_mode_started = time.perf_counter()#最初のモードの開始時間

try:
    while True:
        time_now = time.perf_counter()
        #今の時間の取得
        if GPIO.input(SWITCH):
        #ボタンが押されているとき,
            if time_now - time_switch_previous > bouncetime:
            #前にボタンを押してから十分な時間が経っていたら,
                #モードの切り替え
                if led_mode == LED_BLINKING:
                    led_mode = LED_LIGHTING #点灯へ
                else:
                    led_mode = LED_BLINKING #点滅へ
                time_switch_previous = time_now
                #スイッチの状態を変更
                time_mode_started = time_now
                #モードの切り替わり時の更新

        time_mode_spent = time_now-time_mode_started
        #今のモードでの経過時間

        if led_mode == LED_BLINKING:    #点滅モードでの処理

            if  time_mode_spent < 0.5:
            #今のモードで0.5秒が経っていないうちは,
                GPIO.output(LED, GPIO.HIGH) #光る
            elif time_mode_spent < 1.0:
            #今のモードで1.0秒が経っていないうちは,
                GPIO.output(LED, GPIO.LOW)  #消える
            else:
```

```
                #今のモードで1.0秒経ったら時間をリセット
45                  time_mode_started = time_now
46
47          elif led_mode == LED_LIGHTING:  #点灯モードでの処理
48              GPIO.output(LED, GPIO.HIGH)#点灯し続ける
49              """
50              この部分は点滅モード同様に以下のように実装しても良い
51              if time_mode_spend < 0.5:
52                  GPIO.output(LED, GPIO.HIGH) #光る
53              else:                                                    ↵
                #今のモードで0.5秒経ったら時間をリセット
54                  time_mode_started = time_now
55
56              """
57
58          time.sleep(0.005)#周期
59
60
61  except KeyboardInterrupt:
62      GPIO.cleanup()
63
64
```

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

time_start = time.time()#時間の計測開始

try:
    """
    間に時間を計測したい処理を入れる
    例：
    time.sleep(2)
    """

except KeyboardInterrupt:
    pass

finally:
    time_end = time.time()#時間の計測終了
    time_execution = time_end - time_start#かかった時間の計算
    print(time_execution)#結果の表示
    input():
    GPIO.cleanup()

```

```
import RPi.GPIO as GPIO # GPIOを利用する
import time             # sleepを利用する

# ポート番号の定義
SWITCH = 18
LED = 21
led_value = GPIO.LOW
time_now = 0
time_start = 0

# GPIOの初期化
GPIO.setmode(GPIO.BCM)
GPIO.setup(SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(LED, GPIO.OUT)


# スイッチの切り替えイベント関数の定義
def callback_change_switch(ch):
    global led_value #globalは関数内でグローバル変数を操作可能にする
    global time_now = time.time()#ボタンを押した時間を記録
    print("callback", ch)
    if ch != SWITCH:
        print("incorrect switch is pushed!")
        return#呼び出したボタンが違うときに処理をしないためのプログラム
    time_spent = #穴埋め①：かかった時間を計算
    print("time", time_spent)#呼び出しの時間（秒）を表示
    #LEDの状態の切り替え
    if led_value == GPIO.LOW:
        led_value = GPIO.HIGH
    else:
        led_value = GPIO.LOW



GPIO.output(LED, GPIO.LOW)# LEDを消灯しておく
time_start = time.time()#処理の開始時間を取得

# イベント（割り込み）の開始
GPIO.add_event_detect(
    SWITCH, # ポート番号
    GPIO.RISING, # イベントの種類
    callback=callback_change_switch, # 関数の指定
    bouncetime=200) # 連続イベントを制限


try:
    #メインループ
    while True:
        if led_value == GPIO.HIGH:
            #穴埋め②led_valueがHIGHの状態のとき
        else:
            #穴埋め③led_valueがLOWの状態のとき
```

```
53              time.sleep(0.1)
54
55  except KeyboardInterrupt:
56      GPIO.cleanup()
57
58
```

```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
```