**HCーSR04の検知結果をslackに発信します。**

[こちら](https://drive.google.com/drive/folders/1ArEhGOW7aoltPlpExHF0NC1WF6leyW3I?usp=sharing)からダウンロードしたファイルを以下の通りに配置します。

config\_slack.json　→　.../tmp/PetWatcher/Automate/

distance\_slack.py　→　.../tmp/PetWatcher/Automate/

start\_slackmsg.sh　→　.../tmp/

sensor\_data.py → run it with File called BME280(you can download them [こちら](https://drive.google.com/drive/folders/1ArEhGOW7aoltPlpExHF0NC1WF6leyW3I?usp=sharing))

sensor\_data.py can get sensor data at the same time(distance,humi,temper,pressure),and send them to slack

各自ファイルのコードは以下の通りです。

ーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー

**config\_slack.json:**

{

"APP":"ペット見守り",

"client\_id":"RASPI\_MON",

"work\_dir":"/home/pi/Desktop/tmp/PetWatcher/MONITOR/work",

"interval": 60,

"storage\_url":"http://192.168.11.10:3000/storage/",

"mqtt\_broker":"192.168.11.11:1883",

"pic\_size":[1920, 1080],

"topic\_monitor":"MONITOR/monitor",

"topic\_event":"MONITOR/event",

"cat\_only":false

}

**distance\_slack.py:**

**#!/usr/bin/python3**

**# -\*- coding: utf-8 -\*-**

**import time**

**import sys**

**import os**

**import traceback**

**import json**

**import subprocess**

**import paho.mqtt.client as mqtt**

**import RPi.GPIO as GPIO**

**import MyServer.common as MyCom**

**'''**

**センサーで測距した距離データをMQTTで送信する**

**https://pypi.org/project/paho-mqtt/**

**https://qiita.com/hsgucci/items/6461d8555ea1245ef6c2**

**http://www.steves-internet-guide.com/loop-python-mqtt-client/**

**'''**

**debug\_mode = True**

**#debug\_mode = False**

**# config.jsonを読み、パラメータを取得**

**MyCom.load\_config('./config\_slack.json')**

**my\_id = MyCom.id()**

**interval = MyCom.interval()**

**app\_id = MyCom.app\_id()**

**mqtt\_broker = MyCom.mqtt()**

**topic\_pub = MyCom.topic\_monitor()**

**topic\_sub = MyCom.topic\_event()**

**mqtt\_broker\_addr = mqtt\_broker.split(':')[0]**

**mqtt\_broker\_port = int(mqtt\_broker.split(':')[1])**

**# トピック**

**topic\_pub = os.path.join(app\_id, topic\_pub)**

**topic\_sub = os.path.join(app\_id, topic\_sub)**

**if debug\_mode:**

**print(mqtt\_broker\_addr)**

**print(mqtt\_broker\_port)**

**print(topic\_pub)**

**print(topic\_sub)**

**# アクション実行**

**def do\_action(cmd, opt):**

**if debug\_mode:**

**print("do\_action:" + cmd + " " + opt)**

**ret = subprocess.call([cmd, opt])**

**return ret**

**# MQTT接続時にサブスクライバを登録する関数**

**def regist\_sub(cl, info, flag, rc):**

**if debug\_mode:**

**print("regist\_sub:", rc)**

**if rc != 0:**

**print("Failed to connect MQTT Broker.", + str(rc))**

**os.exit(1)**

**if debug\_mode:**

**print("subscribe:" + topic\_sub)**

**cl.subscribe(topic\_sub)**

**# メッセージ到着時のアクション実行コールバック関数**

**def action(cl, info, msg):**

**if debug\_mode:**

**print("get message:")**

**print(msg.topic + ":" + str(msg.payload))**

**m = json.loads(str(msg.payload))**

**if debug\_mode:**

**print(m)**

**id = m['ID']**

**if id != my\_id:**

**print("Not Mine")**

**return**

**type = m['TYPE']**

**if type != 'ACTION':**

**print('Not supported:' + type)**

**return**

**cmd = act\_list[m['TARGET']]**

**opt = m['OPTION']**

**return do\_action(cmd, opt)**

**# センサーデータの読み取り**

**def get\_distance():**

**if debug\_mode:**

**print("get\_distance")**

**GPIO.output(TRIG, GPIO.LOW)**

**time.sleep(0.3)**

**GPIO.output(TRIG, True)**

**time.sleep(0.00002)**

**#time.sleep(0.00001)**

**GPIO.output(TRIG, False)**

**while GPIO.input(ECHO) == 0:**

**t\_low = time.time()**

**while GPIO.input(ECHO) == 1:**

**t\_high = time.time()**

**t = t\_high - t\_low**

**distance = t/2 \* 340 \* 100**

**# 450cmが測距限界**

**if distance > 450:**

**distance = 450**

**return distance**

**# slackへのメッセージ送信**

**def send\_message(token, channel, distance):**

**if debug\_mode:**

**print("send\_message:" + token + "," + channel)**

**txt = "何かが接近しています。距離 " + str(distance) + " cm"**

**client = slack.WebClient(token=token)**

**client.chat\_postMessage(channel=channel, text=txt)**

**if debug\_mode:**

**print(txt)**

**msg = {"STATUS":"OK", "INFO":""}**

**return msg**

**# センサー読み取り準備**

**GPIO.setwarnings(False)**

**GPIO.setmode(GPIO.BOARD)**

**TRIG = 11**

**ECHO = 13**

**GPIO.setup(TRIG,GPIO.OUT)**

**GPIO.setup(ECHO,GPIO.IN)**

**"""**

**# アクション実行の準備**

**f = open("./config\_slack.json", 'r')**

**act\_conf = json.load(f)**

**f.close()**

**act\_list = {}**

**cmd\_base = act\_conf['CMDDIR']**

**for act in act\_conf['ACTIONS']:**

**print(act)**

**act\_list[act['TARGET']] = os.path.join(cmd\_base, act['COMMAND'])**

**if debug\_mode:**

**print(act\_list)**

**# MQTT Brokerへの接続準備**

**mqtt\_cl = mqtt.Client(protocol=mqtt.MQTTv311)**

**# サブスクライバの準備**

**mqtt\_cl.on\_connect = regist\_sub**

**mqtt\_cl.on\_message = action**

**#mqtt\_broker\_addr="192.168.11.11"**

**#mqtt\_broker\_port=1883**

**#mqtt\_broker\_addr=str(mqtt\_broker\_addr)**

**#mqtt\_broker\_port=int(mqtt\_broker\_port)**

**print("./distance.py159##LIUYI##",mqtt\_broker\_addr,mqtt\_broker\_port)**

**# MQTT Brokerに接続**

**mqtt\_cl.connect(mqtt\_broker\_addr, port=mqtt\_broker\_port)**

**# MQTT処理ループの開始(このループの中で再接続とかもしてくれるとのこと)**

**mqtt\_cl.loop\_start()"""**

**import slack**

**import os**

**slack\_token='xoxb-247578397445-2237502186229-eld4CjB9UBfKCM0t2c5LHNOY'**

**client = slack.WebClient(token=slack\_token)**

**# 測距、データ送信を無限ループ**

**while True:**

**try:**

**# 測距**

**l = get\_distance()**

**# メッセージ組み立て**

**msg = {"ID":my\_id, "TYPE":"DISTANCE", "DISTANCE":l}**

**print(l)**

**msg\_txt="(2021.7.30)distance is! "+str(l)+" cm"**

**client.chat\_postMessage(channel="myserver",text=msg\_txt)**

**# メッセージ送信**

**#mqtt\_cl.publish(topic\_pub, json.dumps(msg))**

**print('sleep: %d' % (interval))**

**except KeyboardInterrupt:**

**break**

**except:**

**print(traceback.format\_exc())**

**pass**

**# 指定時間待ち**

**if debug\_mode:**

**print('sleep: %d' % (interval))**

**time.sleep(interval)**

**"""**

**# MQTT処理ループの終了**

**mqtt\_cl.loop\_stop()**

**# MQTT Brokerからの切断**

**mqtt\_cl.disconnect()"""**

**start\_slackmsg.sh:**

**#!/bin/bash**

**#**

**# slack mes send起動スクリプト**

**#**

**. ./myserver.env**

**. ./cmd**

**. ./goenv.sh**

**. ./nodeenv.sh**

**. ./pythonenv.sh**

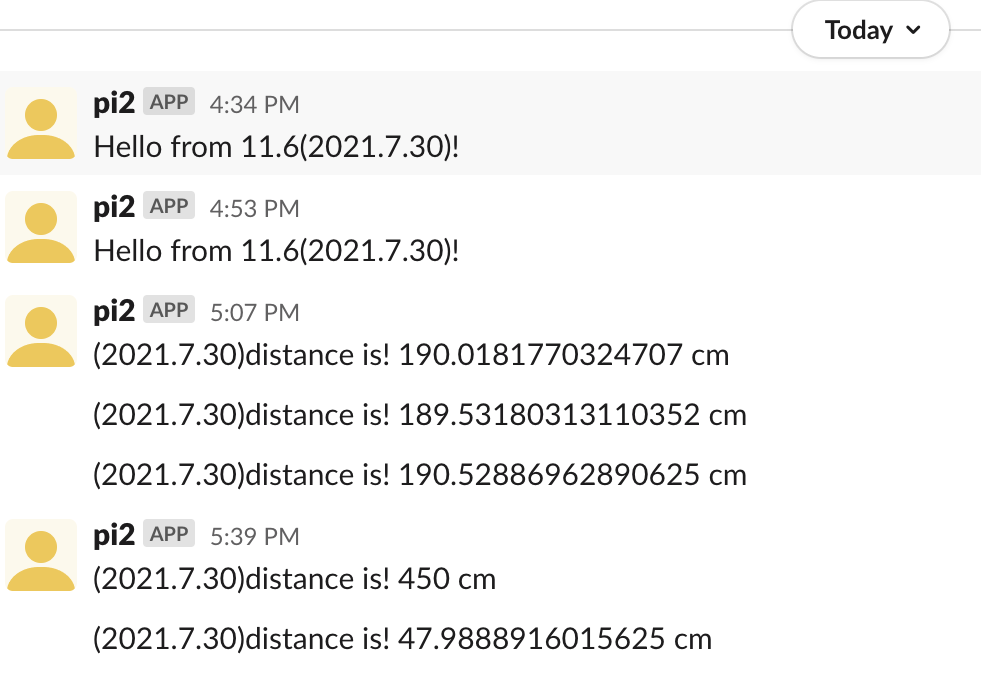
**pushd $MYSERVER\_ROOT/PetWatcher/Automate**

**echo "#################################"**

**echo "Start distance\_slack.py"**

**./distance\_slack.py &**

**ーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー**

./start\_slackmsg.sh を実行すれば距離データ自動的にslackに発信します。

**get\_sensor\_data.py**

**get temp,hum,pressure,distance at the same time,and send them to slack**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import time

import sys

import traceback

import os

import json

import subprocess

import paho.mqtt.client as mqtt

import RPi.GPIO as GPIO

import slack

import os

#debug\_mode = True

debug\_mode = False

import os,sys

retval = os.getcwd()

print(retval)

# \u30a2\u30af\u30b7\u30e7\u30f3\u5b9f\u884c

def do\_action(cmd, opt):

if debug\_mode:

print("do\_action:" + cmd + " " + opt)

ret = subprocess.call([cmd, opt])

return ret

# \u30e1\u30c3\u30bb\u30fc\u30b8\u5230\u7740\u6642\u306e\u30a2\u30af\u30b7\u30e7\u30f3\u5b9f\u884c\u30b3\u30fc\u30eb\u30d0\u30c3\u30af\u95a2\u6570

def action(cl, info, msg):

if debug\_mode:

print("get message:")

print(msg.topic + ":" + str(msg.payload))

m = json.loads(str(msg.payload))

if debug\_mode:

print(m)

id = m['ID']

if id != my\_id:

print("Not Mine")

return

type = m['TYPE']

if type != 'ACTION':

print('Not supported:' + type)

return

cmd = act\_list[m['TARGET']]

opt = m['OPTION']

return do\_action(cmd, opt)

# \u30bb\u30f3\u30b5\u30fc\u30c7\u30fc\u30bf\u306e\u8aad\u307f\u53d6\u308a

def get\_distance():

if debug\_mode:

print("get\_distance")

GPIO.output(TRIG, GPIO.LOW)

time.sleep(0.3)

GPIO.output(TRIG, True)

time.sleep(0.00002)

#time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(ECHO) == 0:

t\_low = time.time()

while GPIO.input(ECHO) == 1:

t\_high = time.time()

t = t\_high - t\_low

distance = t/2 \* 340 \* 100

# 450cm\u304c\u6e2c\u8ddd\u9650\u754c

if distance > 450:

distance = 450

return distance

# \u30bb\u30f3\u30b5\u30fc\u8aad\u307f\u53d6\u308a\u6e96\u5099

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

TRIG = 11

ECHO = 13

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

"""

# \u30a2\u30af\u30b7\u30e7\u30f3\u5b9f\u884c\u306e\u6e96\u5099

f = open("./distance\_config.json", 'r')

act\_conf = json.load(f)

f.close()

act\_list = {}

cmd\_base = act\_conf['CMDDIR']

for act in act\_conf['ACTIONS']:

print(act)

act\_list[act['TARGET']] = os.path.join(cmd\_base, act['COMMAND'])

"""

# \u52d5\u4f5c\u8a2d\u5b9a

#debug\_mode = True

#debug\_mode = False

use\_BME280 = True

use\_DHT11 = False

if use\_DHT11:

import DHT11.dht11\_func as MySensor

elif use\_BME280:

import BME280.bme280\_func as MySensor

# \u30c7\u30fc\u30bf\u53d6\u5f97\u3001\u9001\u4fe1\u3092\u7121\u9650\u30eb\u30fc\u30d7

slack\_token='xoxb-247578397445-2237502186229-eld4CjB9UBfKCM0t2c5LHNOY'

client = slack.WebClient(token=slack\_token)

#os.chdir()

my\_id="MIZUKI"

while True:

try:

# \u30c7\u30fc\u30bf\u3092\u8aad\u307f\u8fbc\u3080

# \u6e2c\u8ddd

l = get\_distance()

t,p,h = MySensor.get\_data()

print(t,p,h)

print(l)

# \u30e1\u30c3\u30bb\u30fc\u30b8\u7d44\u307f\u7acb\u3066

msg = {"ID":my\_id,"TYPE":"SENSORDATA", "DISTANCE":l,"TEM":t,"PRES":p,"HUMI":h}

#print(l)

msg\_txt="(2021.8.18)sensor data! "+str(msg)

client.chat\_postMessage(channel="myserver",text=msg\_txt)

#input("test")

time.sleep(10)

except KeyboardInterrupt:

break

except:

print(traceback.format\_exc())

pass