**IoT : Assignment #2**

*Suvam Basak*

*MTech IT*

20MCMB08

**Python GUI application for the MQTT subscribe operation to the given demo MQTT publish operation using Thingspeak Cloud. with Raspberry Pi + DHT11 Sensor.**

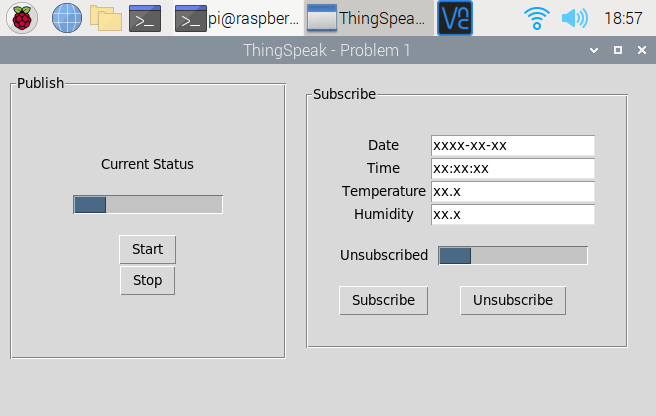


Image 1 : UI

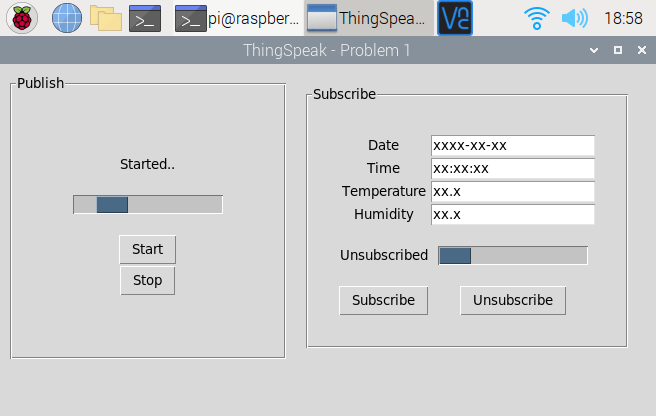


Image 2 : Start Clicked

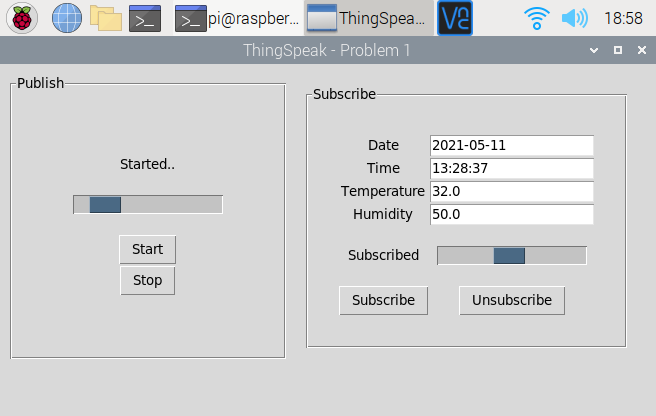


Image 3 : Subscribe Clicked

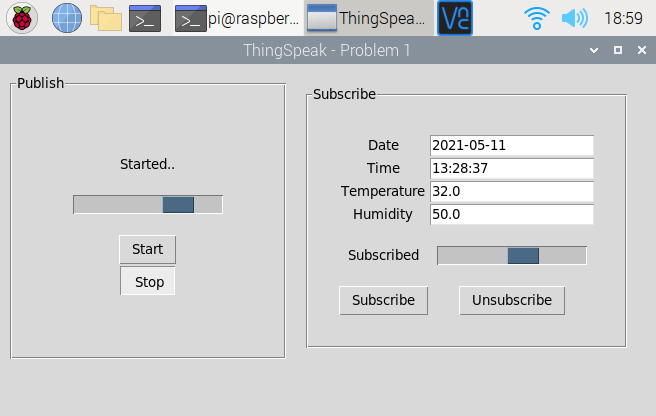


Image 4 : Stop Clicked

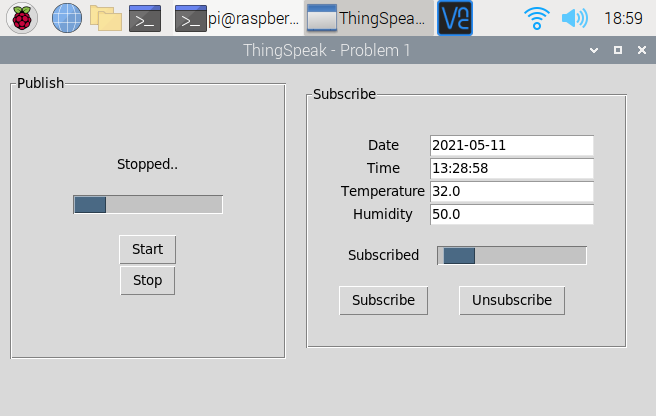


Image 5 : Publication stopped

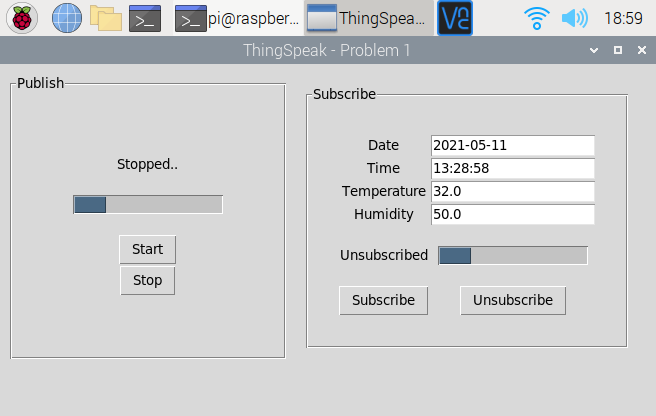


Image 6 : Unsubscribe Clicked

**Source Code**

from \_\_future\_\_ import print\_function

import paho.mqtt.publish as publish

from tkinter import \*

from tkinter import ttk

from urllib.request import urlopen

import json

import time

import threading

import Adafruit\_DHT

class Publish:

'''

ThingSpeak Cloud publish

'''

def \_\_init\_\_(self):

# Thread control

self.control = None

self.publisher\_threat = None

# Set sensor type : Options are DHT11,DHT22 or AM2302

self.sensor = Adafruit\_DHT.DHT11

# Set GPIO sensor is connected to

self.gpio = 4

# The ThingSpeak Channel ID.

# Replace <YOUR-CHANNEL-ID> with your channel ID.

self.CHANNEL\_ID = "1385704"

# The write API key for the channel.

# Replace <YOUR-CHANNEL-WRITEAPIKEY> with your write API key.

self.WRITE\_API\_KEY = "JXMONKXBE6TT13RA"

# The hostname of the ThingSpeak MQTT broker.

self.MQTT\_HOST = "mqtt.thingspeak.com"

# You can use any username.

self.MQTT\_USERNAME = "mwa0000022490756"

# Your MQTT API key from Account > My Profile.

self.MQTT\_API\_KEY = "8HQRSH6RX3BHT23Z"

self.T\_TRANSPORT = "websockets"

self.T\_PORT = 80

# Create the topic string.

self.TOPIC = "channels/" + self.CHANNEL\_ID + "/publish/" + self.WRITE\_API\_KEY

def push\_data(self):

# function to publish data in ThingSpeak Cloud

# will be running in thread

while self.control:

try:

# Use read\_retry method. This will retry up to 15 times to

# get a sensor reading (waiting 2 seconds between each retry).

humidity, temperature = Adafruit\_DHT.read\_retry(

self.sensor, self.gpio)

# Validation

if humidity is not None and temperature is not None:

print(

'Temp={0:0.1f}\*C Humidity={1:0.1f}%'.format(temperature, humidity))

else:

print('Failed to get reading. Try again!')

continue

# build the payload string.

payload = "field1=" + str(temperature)+"&field2="+str(humidity)

# attempt to publish this data to the topic.

publish.single(self.TOPIC, payload, hostname=self.MQTT\_HOST, transport=self.T\_TRANSPORT, port=self.T\_PORT, auth={

'username': self.MQTT\_USERNAME, 'password': self.MQTT\_API\_KEY})

time.sleep(5)

except Exception as e:

print('Exception: push\_data ', str(e))

def start(self):

# function to start the push data thread

self.control = True

self.publisher\_threat = threading.Thread(target=self.push\_data)

self.publisher\_threat.start()

def stop(self):

# funtion to stop push data thread

self.control = False

self.publisher\_threat.join()

class Subscribe:

'''

ThingSpeak Cloud subscribe

'''

def \_\_init\_\_(self):

# Chennel API (result=1 :: take most current data / last entry)

self.URL = 'https://api.thingspeak.com/channels/1385704/feeds.json?results=1'

# Test

# self.URL = 'https://api.thingspeak.com/channels/1385093/feeds.json?results=1'

def fetch\_update(self):

# function to fetch date from Chennel API

with urlopen(self.URL) as url:

# parse JSON

data = json.loads(url.read().decode())

# return data in format -> (Date,Time,Temperature,Humidity)

return (

data['feeds'][-1]['created\_at'].split('T')[0],

data['feeds'][-1]['created\_at'].split('T')[1][:-1],

data['feeds'][-1]['field1'],

data['feeds'][-1]['field2']

)

# print(data['feeds'][-1])

# print('Temp: ', data['feeds'][-1]['field1'])

# print('Hume: ', data['feeds'][-1]['field2'])

# print('Date: ', data['feeds'][-1]['created\_at'].split('T')[0])

# print('Time: ', data['feeds'][-1]['created\_at'].split('T')[1])

class GUI:

'''

Graphical User Interface for ThingSpeak Cloud publish and subscribe

'''

def \_\_init\_\_(self):

# object of Publish class and flag to track thread status (not running True | running False).

self.publisher = Publish()

self.pub\_flag = True

# object of Subscribe class | thread | thread control | and thread status flag.

self.subscriber = Subscribe()

self.sub\_flag = True

self.subscriber\_thread = None

self.control = None

# gui

self.root = Tk()

self.root.title('ThingSpeak - Problem 1')

# frame for start publishing

self.pub\_frame = LabelFrame(

self.root, text='Publish', padx=61, pady=61)

self.pub\_frame.grid(row=0, column=0, padx=10, pady=10)

# frame for subscribing

self.sub\_frame = LabelFrame(

self.root, text='Subscribe', padx=30, pady=30)

self.sub\_frame.grid(row=0, column=1, padx=10, pady=10)

# Status View publishing

self.status\_text = StringVar()

self.status\_text.set('Current Status')

self.status\_view = Label(self.pub\_frame, textvariable=self.status\_text)

self.status\_view.grid(row=0, column=0)

# Status View subscrbe

self.date\_view = Label(self.sub\_frame, text='Date')

self.time\_view = Label(self.sub\_frame, text='Time')

self.temperature\_view = Label(self.sub\_frame, text='Temperature')

self.humidity\_view = Label(self.sub\_frame, text='Humidity')

self.date\_view.grid(row=0, column=0)

self.time\_view.grid(row=1, column=0)

self.temperature\_view.grid(row=2, column=0)

self.humidity\_view.grid(row=3, column=0)

# Status view for subsscribing

self.subscription\_status\_text = StringVar()

self.subscription\_status\_text.set('Unsubscribed')

self.subscription\_status = Label(

self.sub\_frame, textvariable=self.subscription\_status\_text, anchor=W)

self.subscription\_status.grid(row=4, column=0)

# Data view

self.date = Entry(self.sub\_frame)

self.time = Entry(self.sub\_frame)

self.temperature = Entry(self.sub\_frame)

self.humidity = Entry(self.sub\_frame)

self.date.grid(row=0, column=1)

self.time.grid(row=1, column=1)

self.temperature.grid(row=2, column=1)

self.humidity.grid(row=3, column=1)

self.date.insert(0, 'xxxx-xx-xx')

self.time.insert(0, 'xx:xx:xx')

self.temperature.insert(0, 'xx.x')

self.humidity.insert(0, 'xx.x')

# Progress bar publishing

self.pub\_prog = ttk.Progressbar(self.pub\_frame, orient=HORIZONTAL,

length=150, mode='indeterminate')

self.pub\_prog.grid(row=1, column=0, pady=20)

# Progress bar subscribing

self.sub\_prog = ttk.Progressbar(self.sub\_frame, orient=HORIZONTAL,

length=150, mode='indeterminate')

self.sub\_prog.grid(row=4, column=1, pady=20, columnspan=2)

# adding button in publishing frame.

self.start = Button(self.pub\_frame, text='Start',

command=self.start\_pub)

self.stop = Button(self.pub\_frame, text='Stop', command=self.stop\_pub)

self.start.grid(row=2, column=0)

self.stop.grid(row=3, column=0)

# adding button in subscribe frame.

self.sub = Button(self.sub\_frame, text='Subscribe',

command=self.start\_sub)

self.cancel = Button(self.sub\_frame, text='Unsubscribe',

command=self.stop\_sub)

self.sub.grid(row=5, column=0)

self.cancel.grid(row=5, column=1)

self.root.mainloop()

def loader(self):

# function runs in a different thread and update the data of text view

while self.control:

# get current data.

current\_state = self.subscriber.fetch\_update()

print(current\_state)

# update all text view

self.date.delete(0, END)

self.date.insert(0, current\_state[0])

self.time.delete(0, END)

self.time.insert(0, current\_state[1])

self.temperature.delete(0, END)

self.temperature.insert(0, current\_state[2])

self.humidity.delete(0, END)

self.humidity.insert(0, current\_state[3])

time.sleep(3)

def start\_sub(self):

# on subscribe click event

# starts leader thead

if self.sub\_flag:

self.sub\_flag = False

print('Start sub')

# start thread

self.control = True

self.subscriber\_thread = threading.Thread(target=self.loader)

self.subscriber\_thread.start()

# start progress bar and chnage subscription status

self.sub\_prog.start(10)

self.subscription\_status\_text.set('Subscribed')

def stop\_sub(self):

# on unsubscribe click event

# stops the leader thead

if not self.sub\_flag:

self.sub\_flag = True

print('Stop sub')

# stop infinite loop

self.control = False

# stop progress bar and chnage subscription status

self.sub\_prog.stop()

self.subscription\_status\_text.set('Unsubscribed')

def start\_pub(self):

# on click event of publish

# starts publisher thread

if self.pub\_flag:

self.pub\_flag = False

print('Start pub')

self.publisher.start()

self.pub\_prog.start(20)

self.status\_text.set('Started..')

def stop\_pub(self):

# on click event of publish

# starts publisher thread

if not self.pub\_flag:

self.pub\_flag = True

print('Stop pub')

self.publisher.stop()

self.pub\_prog.stop()

self.status\_text.set('Stopped..')

# start GUI

GUI()

**Python GUI application to collect and store sensor data locally in the MySQL DB (LAMP stack installed in the Raspberry Pi). Connect to Raspberry Pi + DHT11 Sensor.**

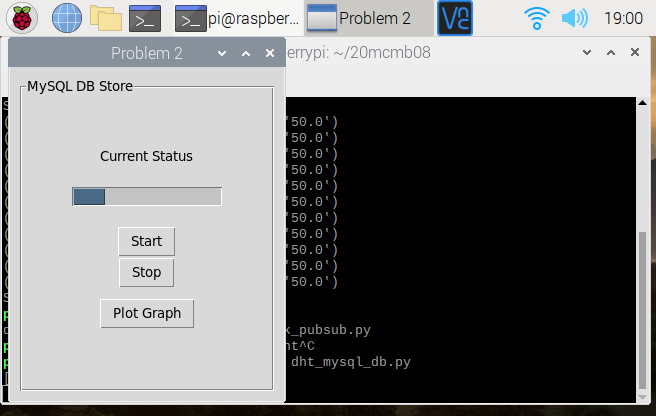


Image 7 : UI

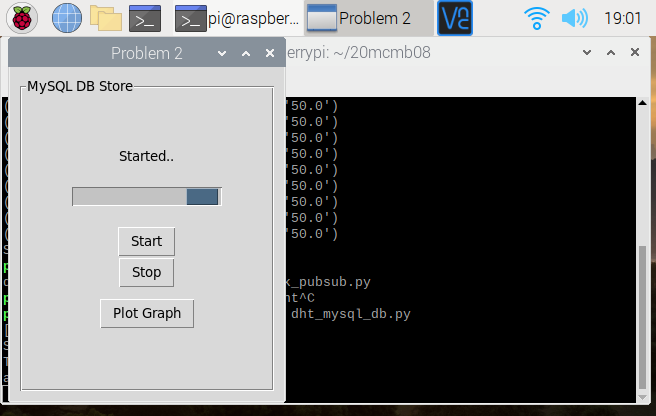


Image 8 : Start Clicked

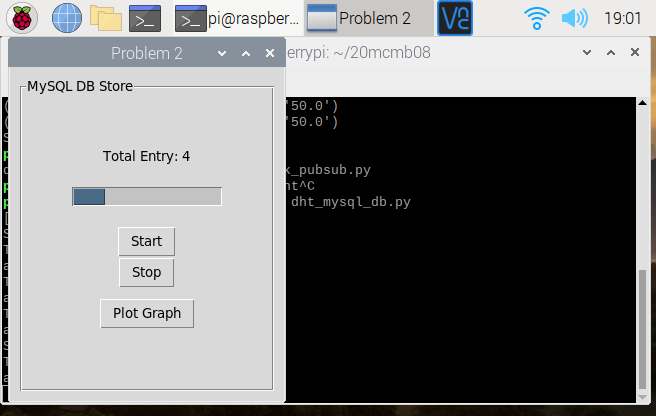


Image 9 : Stop Clicked

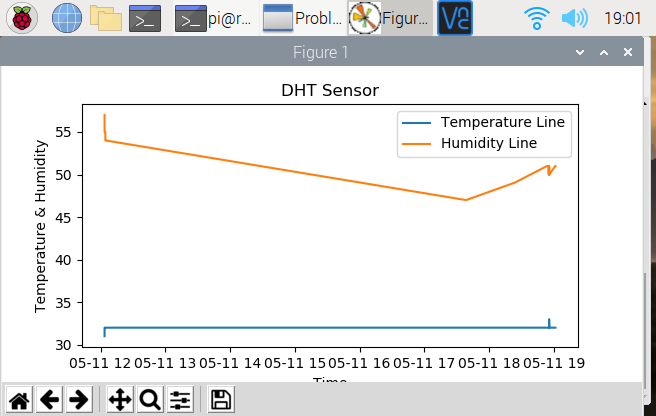


Image 10 : Plot Graph Clicked

**Source Code**

from tkinter import \*

from tkinter import ttk

from matplotlib import pyplot as plt

import pymysql

import threading

import time

import Adafruit\_DHT

# Test

# import random

class Database:

'''

MySQL Database

'''

def \_\_init\_\_(self):

# Total entry count.

self.entry\_count = 0

# Credentials

self.host = 'localhost'

# self.user = 'admin'

# self.password = 'admin'

# self.dbname = '20mcmb08'

self.user = 'phpmyadmin'

self.password = 'scisnks99'

self.dbname = 'phpmyadmin'

# Connect

try:

self.db = pymysql.connect(

self.host, self.user, self.password, self.dbname)

print("[\*] Database Connected.")

except Exception as e:

print("\n\n[\*\*] Exception :: \_\_init\_\_ :: " + str(e))

print('\n\n')

# Auto commit and cursor.

self.db.autocommit(True)

self.cursor = self.db.cursor()

def fetch\_all(self):

# Function to fetch all data from the table dht.

SQL = "SELECT \* FROM `dht`"

# Execute and fetch result.

try:

self.cursor.execute(SQL)

result = self.cursor.fetchall()

# print('fetch\_all: ', result)

return result

except Exception as e:

print('\n[\*\*] Database :: fetch\_all :: ' + str(e))

return None

def add\_new(self, temperature, humidity):

# Function to insert new data.

SQL = "INSERT INTO `dht` (`id`, `time`, `temperature`, `humidity`) VALUES (NULL, NULL, '%s', '%s')" % (

temperature, humidity)

# Execute

try:

self.cursor.execute(SQL)

self.entry\_count += 1

print('add\_new:', temperature, humidity)

except Exception as e:

print('\n[\*\*] Exception :: add\_new :: ' + str(e))

def get\_entry\_count(self):

# Function to get total data entry.

return self.entry\_count

class Sensor:

'''

DHT11 Sensor

'''

def \_\_init\_\_(self):

# Thread control

self.control = None

self.sensor\_threat = None

# Set sensor type : Options are DHT11,DHT22 or AM2302

self.sensor = Adafruit\_DHT.DHT11

# Set GPIO sensor is connected to

self.gpio = 4

# Database object

self.db = Database()

def sense(self):

# Thread to sense and store data in database.

while self.control:

try:

time.sleep(3)

# Use read\_retry method. This will retry up to 15 times to

# get a sensor reading (waiting 2 seconds between each retry).

humidity, temperature = Adafruit\_DHT.read\_retry(

self.sensor, self.gpio)

# TEST

# humidity = random.randint(0, 50)

# temperature = random.randint(0, 50)

if humidity is not None and temperature is not None:

print(

'Temp={0:0.1f}\*C Humidity={1:0.1f}%'.format(temperature, humidity))

# Insert data

self.db.add\_new(temperature, humidity)

else:

print('Failed to get reading. Try again!')

except Exception as e:

print('Sense:', str(e))

def start(self):

# Function to start the thread

self.control = True

self.sensor\_threat = threading.Thread(target=self.sense)

self.sensor\_threat.start()

def stop(self):

# Function to stop the thread

self.control = False

self.sensor\_threat.join()

# return total data entry

return self.db.get\_entry\_count()

class GUI:

'''

Graphical User Interface

'''

def \_\_init\_\_(self):

# Sensor object

self.flag = True

self.sensor = Sensor()

self.root = Tk()

self.root.title('Problem 2')

# create frame for start publishing

self.pub\_frame = LabelFrame(

self.root, text='MySQL DB Store', padx=50, pady=50)

self.pub\_frame.grid(row=0, column=0, padx=10, pady=10)

# Status View

self.status\_text = StringVar()

self.status\_text.set('Current Status')

self.status\_view = Label(self.pub\_frame, textvariable=self.status\_text)

self.status\_view.grid(row=0, column=0)

# Progress bar

self.pub\_prog = ttk.Progressbar(self.pub\_frame, orient=HORIZONTAL,

length=150, mode='indeterminate')

self.pub\_prog.grid(row=1, column=0, pady=20)

# adding button in publishing frame.

self.start = Button(self.pub\_frame, text='Start',

command=self.start\_pub)

self.start.grid(row=2, column=0)

self.stop = Button(self.pub\_frame, text='Stop', command=self.stop\_pub)

self.stop.grid(row=3, column=0)

self.plot = Button(

self.pub\_frame, text='Plot Graph', command=self.graph)

self.plot.grid(row=4, column=0, pady=10)

self.root.mainloop()

def graph(self):

# Fetch all data from database and show the graph

result = Database().fetch\_all()

if None == result:

print('Error: GUI graph..')

return

# X-axis values

time = []

# Y-axis values

temperature = []

humidity = []

# make list of time, temperature, humidity

for row in result:

time.append(row[1])

temperature.append(row[2])

humidity.append(row[3])

# plot

plt.plot(time, temperature, label='Temperature Line')

plt.plot(time, humidity, label='Humidity Line')

plt.xlabel('Time')

plt.ylabel('Temperature & Humidity')

plt.title('DHT Sensor')

plt.legend()

# function to show the plot

plt.show()

def start\_pub(self):

# Start button click event handle

# Starting the thread on click

if self.flag:

self.flag = False

print('Start pub')

self.sensor.start()

# Start progress bar and change status view

self.pub\_prog.start(20)

self.status\_text.set('Started..')

def stop\_pub(self):

# Stop button clickevent handle

# Stoping the thread on click

if not self.flag:

self.flag = True

print('Stop pub')

entry = self.sensor.stop()

self.pub\_prog.stop()

self.status\_text.set('Stopped..')

time.sleep(1.5)

# Show total entry

self.status\_text.set('Total Entry: '+str(entry))

# Start GUI

GUI()

**Database**

CREATE TABLE `dht` (

`id` int(11) NOT NULL,

`time` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE current\_timestamp(),

`temperature` float NOT NULL,

`humidity` float NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `dht`

ADD PRIMARY KEY (`id`);

ALTER TABLE `dht`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;