Si has comprado el sensor de humedad y temperatura DHT22 para la raspberry, todavía no te ha llegado y estás impaciente por empezar a trabajar con él, sigue este hack para simular su lectura y poder importar datos a InfluxDB como si estuvieran sido leídos por el sensor mismo:
Si analizamos el script que funcionaría en la raspberry para leer los datos del sensor podemos observar lo siguiente:

```
#!/usr/bin/python3

import Adafruit_DHT
import json

DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4

while True:
  humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
  if humidity is not None and temperature is not None:
    print(json.dumps({'Temperatura': temperature, 'Humedad': humidity}, indent=2))
  else:
    print("Falló la obtención de datos desde el sensor")
```

Vemos entonces que la temperatura y la humedad son leidas del sensor mediante la función **read_retry** y guardadas en las variables **temperature** y **humidity**, respectivamente. Luego son mostradas por pantalla en formato json por la función print. Entonces, para simular ese comportamiento vamos a crear el siguiente script:

```
#!/usr/bin/python3

from time import time, sleep
import json
from random import uniform

while True:
  sleep(1 - time() % 1)
  print(json.dumps({'Temperatura': uniform(10, 45), 'Humedad': uniform(11,99)},
indent=2))
```

Lo que hacemos es importar la **librería time** para indicar cada cuanto tiempo queremos ejecutar la simulación de lectura e importamos también la **librería random** para generar números aleatorios para esa simulación. Luego le indicamos que el tiempo de lectura sea cada 1 segundo y para guardar cada valor usamos la función uniform de la librería random indicando para la temperatura una lectura de entre 10 y 45 y para la humedad una lectura de entre 11 y 99. Por supuesto usamos una indentación de 2 espacios dado que la indentación superior a dos espacios es para seres inferiores.
Ahora bien, que pasa si queremos guardar la simulación de esas lecturas en una base de datos de registros temporales, como por ejemplo InfluxDB, para poder registrarlas y consultarlas a futuro. Pues analicemos el script original que guarda los datos del sensor en una base de datos InfluxDB:

```
#!/usr/bin/python3

import Adafruit_DHT
import datetime
from influxdb import client as influxdb

# Base de datos
influxHost = xxx
influxPort = xxx
influxDB = xxx
influxUser = xxx
influxPasswd = xxx

# Sensor
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4

# Crear el objeto del sensor usando el bus I2C de la Raspberry
humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)

# Mostrar la fecha en un formato agimable con influxDB 2017-02-26T13:33:49.00279827Z
current_time = datetime.datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%SZ')

influx_metric = [{
  "measurement": "temp_hum",
                "time": current_time,
                "fields": {
                  "Temperatura": temperature,
                  "Humedad": humidity
                 }
}]

# Salvar los datos a la base de datos de InfluxDB
try:
  db = influxdb.InfluxDBClient(influxHost, influxPort, influxUser, InflusPasswd,
InfluxDB)
  db.write_points(influx_metric)
finally:
  db.close()
```

Vemos que para simularlo simplemente tendríamos que cambiar la librería Adafruit_DHT por la random y adaptar la función correspondiente:

```
#!/usr/bin/python3

import json
```

```
from random import uniform
import datetime
from influxdb import client as influxdb

# Variables
influxHost = 'xxx'
influxPort = 'xxx'
influxDB = 'xxx'
influxUser = 'xxx'
influxPasswd = 'xxx'

# Simular el objeto del sensor
humidity, temperature = uniform(10, 45), uniform(10, 45)

# Adaptar el formato de la hora para que influxDB lo entienda
(2017-02-26T13:33:49.00279827Z)
current_time = datetime.datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%SZ')

influx_metric = [{
"measurement": "temp_hume",
               "time": current_time,
               "fields":
                 {
                   "Temperatura": temperature,
                   "Humedad": humidity
                 }
}]

# Salvar mediciones a la base de datos
try:
  db = influxdb.InfluxDBClient(influxHost, influxPort, influxUser, influxPasswd,
influxDB)
  db.write_points(influx_metric)
finally:
  db.close()
# Imprimir también a un archivo de log
#with open('/var/log/dht22.log', 'a') as archivolog:
#  print("Se han guardado los siguientes valores: Humedad:",humidity,
"Temperatura:",temperature, file=archivolog)
```

La diferencia es que ahora no podemos indicar el tiempo de ejecución de esa lectura, por lo que tendremos que crear un servicio en systemd y un temporizador asociado a ese servicio para que dispare el script con la cadencia de tiempo que le indiquemos. Creamos entonces los dos scripts:
Primero el servicio de lectura del sensor simulado (**/etc/systemd/system/DHT22Simulado.service**):

```
[Unit]
Description=Servicio de SystemD para el sensor DHT22
```

```
[Service]
ExecStart=/usr/bin/python3 /root/scripts/SensorDHT22Simulado-LeerYGuardarEnInfluxDB.py
[Install]
WantedBy=default.target
```

Y luego el servicio de temporizador que va a disparar el servicio anterior (**/etc/systemd/system/DHT22Simulado.timer**):

```
[Unit]
Description=Temporizador para el sensor DHT22
[Timer]
OnCalendar=*-*-* *:*:00
[Install]
WantedBy=default.target
```

Logicamente activaremos e iniciaremos ambos servicios:

```
systemctl enable DHT22Simulado.service
systemctl enable DHT22Simulado.timer
systemctl start  DHT22Simulado.service
systemctl start  DHT22Simulado.timer
```

NOTA: Es muy importante que ambos servicios tengan el mismo nombre, aunque distinta extensión.