GIS - Task #7984

Understanding The Things Network and embedding it with Jupyter for Well Data

27/03/2019 16:20 - Debojyoti Mallick

Status:	Resolved	**Start date:**	27/03/2019
Priority:	Normal	**Due date:**	
Assignee:	Debojyoti Mallick	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Understanding Things Network and its Role with Python in Visual Studio Code:
https://www.thethingsnetwork.org/docs/applications/python/

Setup and Installing TTN Client: https://www.thethingsnetwork.org/docs/applications/golang/quick-start.html

History

#1 - 28/03/2019 10:57 - Debojyoti Mallick

Debojyoti Mallick wrote:

> Understanding Things Network and its Role with Python in Visual Studio Code: https://www.thethingsnetwork.org/docs/applications/python/
>
> Setup and Installing TTN Client: https://www.thethingsnetwork.org/docs/applications/golang/quick-start.html

First attempt to connect to the Things Network:

```
import time
import ttn

app_id = "foo"
access_key = "ttn-account.eiPq8mEeYRL_PNBZsOpPy-O3ABJXYWulODmQGR5PZzg"

def uplink_callback(msg, client):
print("Received uplink from ", msg.dev_id)
print(msg)

handler = ttn.HandlerClient(app_id, access_key) #app id: "aurodam" ; access_key:
"v2.WIo2jNGQgvpIMXQcC9A-EsAw9UpusVIAmDtdV4_vGTY"__**
```

1. using mqtt client*
   ```
   mqtt_client = handler.data()
   mqtt_client.set_uplink_callback(uplink_callback)
   mqtt_client.connect()
   time.sleep(60)
   mqtt_client.close()
   ```

1. using application manager client
   ```
   app_client =  handler.application()* # ('Error while getting the', ' application: UNKNOWN')_**_***
   ```
   h1. Cannot Find the Application though app_id and access key are mentioned. Maybe have to mention MQTTClient?

```
my_app = app_client.get()
print(my_app)
my_devices = app_client.devices()
print(my_devices)
```

>> From the Things Network API Reference :

This package provides you an easy way to exchange traffic with The Things Network via MQTT and manage your applications.

Possible Solution below : ***

MQTTClient(app_id, access_key, mqtt_address="",
discovery_address="discovery.thethings.network:1900")

*mqtt_address: string this is the address of the handler to which the application was registered, in the {hostname}:{port} format.

discovery_address: string this is the address of the discovery server to use in order to find back the address of the MQTT handler, in the {hostname}:{port} format.

If the mqtt_address is set, the discovery_address doesn't need to be set as it is only used to retrieve the mqtt_address from the discovery server. The constructor returns an MQTTClient object set up with the application informations, ready to be connected to The Things Network.*

Connects and starts the client in the background. This function also re-establishes the client's connection in case it was closed.

client.connect()
close
Disconnects and stops the client from which the method is called.

client.close()

#2 - 28/03/2019 11:00 - Debojyoti Mallick

- % Done changed from 0 to 10

#3 - 29/03/2019 10:35 - Debojyoti Mallick

Debojyoti Mallick wrote:

> Debojyoti Mallick wrote:
>
> > Debojyoti Mallick wrote:
> >
> > > Debojyoti Mallick wrote:
> > >
> > > > Understanding Things Network and its Role with Python in Visual Studio Code:
> > > > https://www.thethingsnetwork.org/docs/applications/python/
> > > >
> > > > Setup and Installing TTN Client: https://www.thethingsnetwork.org/docs/applications/golang/quick-start.html
> > > >
> > > > **First attempt to connect to the Things Network:**
> > > >
> > > > import time
> > > > import ttn
> > > >
> > > > app_id = "foo"
> > > > access_key = "ttn-account.eiPq8mEeYRL_PNBZsOpPy-O3ABJXYWulODmQGR5PZzg"
> > > >
> > > > def uplink_callback(msg, client):
> > > > print("Received uplink from ", msg.dev_id)

print(msg)

handler = ttn.HandlerClient(app_id, access_key) ***#app id: "aurodam" ; access_key: "v2.WIo2jNGQgvpIMXQcC9A-EsAw9UpusVIAmDtdV4_vGTY"__****

1. using mqtt client*
   mqtt_client = handler.data()
   mqtt_client.set_uplink_callback(uplink_callback)
   mqtt_client.connect()
   time.sleep(60)
   mqtt_client.close()

1. using application manager client
   app_client = handler.application()* # ('Error while getting the', ' application: UNKNOWN')_**_***
   h1. Cannot Find the Application though app_id and access key are mentioned. Maybe have to mention MQTTClient?

my_app = app_client.get()
print(my_app)
my_devices = app_client.devices()
print(my_devices)

>> From the Things Network API Reference :

This package provides you an easy way to exchange traffic with The Things Network via MQTT and manage your applications.

***Possible Solution below :** ***

MQTTClient(app_id, access_key, mqtt_address="",
discovery_address="discovery.thethings.network:1900")

*mqtt_address: string this is the address of the handler to which the application was registered, in the {hostname}:{port} format.

discovery_address: string this is the address of the discovery server to use in order to find back the address of the MQTT handler, in the {hostname}:{port} format.

If the mqtt_address is set, the discovery_address doesn't need to be set as it is only used to retrieve the mqtt_address from the discovery server. The constructor returns an MQTTClient object set up with the application informations, ready to be connected to The Things Network.*

Connects and starts the client in the background. This function also re-establishes the client's connection in case it was closed.

client.connect()
close
Disconnects and stops the client from which the method is called.

client.close()

h1. First Attempt to Connect to The Things Network and get Readings for Ami_Rana and Baraka

Roles were changed by Aza for Philippe so that he can get administrative privileges to check the status of all the wells and the devices. Previously a collaborator role was assigned which meant all wells (devices) and applications would not be displayed.

Philippe helped me understand how the connection works in Visual Studio and python with uplink_callback(uplink_callback) and Handler ID's (handlers[app_id] = ttn.HandlerClient(app_id, access_key))

*eg. handlers[app_id] = ttn.HandlerClient(app_id, access_key)

1. using mqtt client
   mqtt_clients[app_id] = handlers[app_id].data()
   mqtt_clients[app_id].set_uplink_callback(uplink_callback)
   mqtt_clients[app_id].connect()*

The Concept of time.sleep() was discussed and an interval of 10 mins was set to visualize the data from the device.

eg. time.sleep(600)

Receiving data from multiple devices was discussed which is when a new variable was introduced.

**config = {

'apps': {
'ami_rama': "ttn-account-v2.6UEpQT1kb58BCouvltr5HLOP7CGOwZsLTWYpTo2nK3I", #(app id, access key)
'baraka': "ttn-account-v2.eo-VseYHokva5d5h7J0b9b1fMQqWPU1dzOeBGXdqwrw"   #(app_id, access key)
},
'timeout': 10
} **

Readings were received for both Ami_Rama and Baraka using the following script.

**import time

```
import ttn
import sys

config = {
'apps': {
'ami_rama': "ttn-account-v2.6UEpQT1kb58BCouvltr5HLOP7CGOwZsLTWYpTo2nK3I",
'baraka': "ttn-account-v2.eo-VseYHokva5d5h7J0b9b1fMQqWPU1dzOeBGXdqwrw"
},
'timeout': 10
}

def uplink_callback(msg, client):
print("Received uplink from ", msg.dev_id)
print(msg)

handlers = {}
mqtt_clients = {}

for app_id, access_key in config['apps'].items():
handlers[app_id] = ttn.HandlerClient(app_id, access_key)

    1. using mqtt client
       mqtt_clients[app_id] = handlers[app_id].data()
       mqtt_clients[app_id].set_uplink_callback(uplink_callback)
       mqtt_clients[app_id].connect()

time.sleep(600)

for mqtt_client in mqtt_clients.keys():
mqtt_client.close()**
```

Output received for Baraka:
*Received uplink from catena4450_07_46

MSG(app_id='baraka', dev_id='catena4450_07_46', hardware_serial='0002CC0100000146', port=1, > > > >> counter=418,payload_raw='Fj1DIxQfH2JUkAHiAAA=', payload_fields=MSG(boot=20, error='none', lux=482, p=1006.88, rh=56.25, tDewC=21.383713359605352, tempC=31.12109375, vBat=4.196044921875, wLevel=0, wPressure=0), metadata=MSG(time='2019-03-

29T04:55:27.08015751Z', frequency=865.0625, modulation='LORA', data_rate='SF12BW125', airtime=1646592000, coding_rate='4/5', gateways=[MSG(gtw_id='talam-ttn', gtw_trusted=True, timestamp=3091024356, time='2019-03-29T05:05:07Z', channel=0, rssi=-84, snr=7.25, rf_chain=0)]))*

Device for Baraka is in the office for testing and it seems Ami_Rana is not working or transmitting yet.

Which is why the reading for Baraka shows wLevel=0 and wPressure=0 but tempC=31.12109375.

#4 - 29/03/2019 16:04 - Debojyoti Mallick

- Status changed from New to Resolved

- % Done changed from 10 to 100

Debojyoti Mallick wrote:

> Debojyoti Mallick wrote:
>
> > Debojyoti Mallick wrote:
> >
> > > Debojyoti Mallick wrote:
> > >
> > > > Debojyoti Mallick wrote:
> > > >
> > > > > Understanding Things Network and its Role with Python in Visual Studio Code: https://www.thethingsnetwork.org/docs/applications/python/
> > > > >
> > > > > Setup and Installing TTN Client: https://www.thethingsnetwork.org/docs/applications/golang/quick-start.html
> > > > >
> > > > > **First attempt to connect to the Things Network:**
> > > > >
> > > > > ```
> > > > > import time
> > > > > import ttn
> > > > >
> > > > > app_id = "foo"
> > > > > access_key = "ttn-account.eiPq8mEeYRL_PNBZsOpPy-O3ABJXYWulODmQGR5PZzg"
> > > > >
> > > > > def uplink_callback(msg, client):
> > > > > print("Received uplink from ", msg.dev_id)
> > > > > print(msg)
> > > > > ```

handler = ttn.HandlerClient(app_id, access_key) ***#app id: "aurodam" ; access_key: "v2.WIo2jNGQgvpIMXQcC9A-EsAw9UpusVIAmDtdV4_vGTY"__****

1. using mqtt client*
   mqtt_client = handler.data()
   mqtt_client.set_uplink_callback(uplink_callback)
   mqtt_client.connect()
   time.sleep(60)
   mqtt_client.close()

1. using application manager client
   app_client = handler.application()* # ('Error while getting the', ' application: UNKNOWN')_**_***
   h1. Cannot Find the Application though app_id and access key are mentioned. Maybe have to mention MQTTClient?

my_app = app_client.get()
print(my_app)
my_devices = app_client.devices()
print(my_devices)

>> From the Things Network API Reference :

This package provides you an easy way to exchange traffic with The Things Network via MQTT and manage your applications.

***Possible Solution below :** ***

MQTTClient(app_id, access_key, mqtt_address="",
discovery_address="discovery.thethings.network:1900")

*mqtt_address: string this is the address of the handler to which the application was registered, in the {hostname}:{port} format.

discovery_address: string this is the address of the discovery server to use in order to find back the address of the MQTT handler, in the {hostname}:{port} format.

If the mqtt_address is set, the discovery_address doesn't need to be set as it is only used to retrieve the mqtt_address from the discovery server. The constructor returns an MQTTClient object set up with the application informations, ready to be connected to The Things Network.*

Connects and starts the client in the background. This function also re-establishes the client's connection in case it was closed.

client.connect()
close
Disconnects and stops the client from which the method is called.

client.close()

h1. First Attempt to Connect to The Things Network and get Readings for Ami_Rana and Baraka

Roles were changed by Aza for Philippe so that he can get administrative privileges to check the status of all the wells and the devices. Previously a collaborator role was assigned which meant all wells (devices) and applications would not be displayed.

Philippe helped me understand how the connection works in Visual Studio and python with uplink_callback(uplink_callback) and Handler ID's (handlers[app_id] = ttn.HandlerClient(app_id, access_key))

*eg. handlers[app_id] = ttn.HandlerClient(app_id, access_key)

1. using mqtt client
   mqtt_clients[app_id] = handlers[app_id].data()
   mqtt_clients[app_id].set_uplink_callback(uplink_callback)
   mqtt_clients[app_id].connect()*

The Concept of time.sleep() was discussed and an interval of 10 mins was set to visualize the data from the device.

eg. time.sleep(600)

Receiving data from multiple devices was discussed which is when a new variable was introduced.

**config = {

'apps': {
'ami_rama': "ttn-account-v2.6UEpQT1kb58BCouvltr5HLOP7CGOwZsLTWYpTo2nK3I", #(app id, access key)
'baraka': "ttn-account-v2.eo-VseYHokva5d5h7J0b9b1fMQqWPU1dzOeBGXdqwrw"   #(app_id, access key)
},
'timeout': 10
} **

Readings were received for both Ami_Rama and Baraka using the following script.

**import time

import ttn
import sys

config = {
'apps': {
'ami_rama': "ttn-account-v2.6UEpQT1kb58BCouvltr5HLOP7CGOwZsLTWYpTo2nK3I",
'baraka': "ttn-account-v2.eo-VseYHokva5d5h7J0b9b1fMQqWPU1dzOeBGXdqwrw"
},
'timeout': 10
}

def uplink_callback(msg, client):
print("Received uplink from ", msg.dev_id)
print(msg)

handlers = {}
mqtt_clients = {}

for app_id, access_key in config['apps'].items():
handlers[app_id] = ttn.HandlerClient(app_id, access_key)

1. using mqtt client
   mqtt_clients[app_id] = handlers[app_id].data()
   mqtt_clients[app_id].set_uplink_callback(uplink_callback)
   mqtt_clients[app_id].connect()

time.sleep(600)

for mqtt_client in mqtt_clients.keys():
mqtt_client.close()**

Output received for Baraka:
*Received uplink from catena4450_07_46

MSG(app_id='baraka', dev_id='catena4450_07_46', hardware_serial='0002CC0100000146', port=1, > > > >> counter=418,payload_raw='Fj1DIxQfH2JUkAHiAAA=', payload_fields=MSG(boot=20, error='none', lux=482, p=1006.88, rh=56.25, tDewC=21.383713359605352, tempC=31.12109375, vBat=4.196044921875, wLevel=0, wPressure=0), metadata=MSG(time='2019-03-29T04:55:27.08015751Z', frequency=865.0625, modulation='LORA', data_rate='SF12BW125', airtime=1646592000, coding_rate='4/5',

gateways=[MSG(gtw_id='talam-ttn', gtw_trusted=True, timestamp=3091024356, time='2019-03-29T05:05:07Z', channel=0, rssi=-84, snr=7.25, rf_chain=0)]))*

Device for Baraka is in the office for testing and it seems Ami_Rana is not working or transmitting yet.

Which is why the reading for Baraka shows wLevel=0 and wPressure=0 but tempC=31.12109375.

Filtering Data is made possible : "You can get what you want"

eg. def uplink_callback(msg, client):
print("Received uplink from ", msg.dev_id)
print("Values Received", msg.app_id, "wLevel", msg.payload_fields.wLevel, "wPressure", msg.payload_fields.wPressure, "tempC", msg.payload_fields.tempC)

In the above case I would only like the values/readings for water level(wLevel), water pressure (wPressure) and temperature (tempC).

The output will show up as below:
Received uplink from catena4450_07_46
Values Received baraka wLevel 0 wPressure 0 tempC 32.43359375

h1.In addition to working on Visual Studio, you can get all of these results in Jupyter by using the same script as they are using the same scripting language.

CONCLUSIONS

- Installed the TTN client at our end [import ttn works]
- We have established a connection with the TTN network for our devices using the app_id and the access_key.
- We have now found success with receiving the values at our end and can also play with them. [Even values of multiple wells]
- We can set a desired time interval for these values. [time.sleep()]
- This script can be run on Jupyter and you can receive the results.

Over to Philippe to eventually use it as a service to be used with GISAF and Thank you so much for all the help.