# Software for Smart Cities, Software and Applications for IoT

## Introduction to Internet of Things

Dimitrios J. Vergados

University of Western Macedonia,
Department of Informatics

10-7-2020

Σκοπός της εργασίας

Ενότητα 1: Εκτέλεση πρώτου προγράμματος σε python

Ενότητα 2: Επικοινωνία με συσκευή IoT

Ενότητα 3: Γραφική απεικόνιση των αποτελεσμάτων

Ενότητα 4: Αποκακρυσμένος έλεγχος συσκευής.

End of presentation

# Σκοπός της εργασίας

Σκοπός της εργασίας

# Σκοπός της εργασίας

Σκοπός της εργασίας είναι η δημιουργία λογισμικού σε python, το οποίο θα μπορεί να διαβάζει μετρήσεις από συσκευή IoT, να στέλνει εντολές σε αυτή, και να απεικονίζει γραφική παράσταση με τα αποτελέσματα

## Σκοπός της εργασίας

# Εργαλεία που θα χρησιμοποιηθούν Ι

1. Γλώσσα python 3, https://wiki.python.org/moin/BeginnersGuide
2. Βιβλιοθήκη paho-mqtt https://pypi.org/project/paho-mqtt για επικοινωνία με το σύστημα επικοινωνία MQTT
3. Βιβλιοθήκη Matplotlib για δημιουργία γραφικών παραστάσεων στη python

# Ενότητα 1: Εκτέλεση πρώτου προγράμματος σε python

# Εγκατάσταση της γλώσσας python στον υπολογιστή

## Linux

Δε χρειάζεται να κάνουμε τίποτα, η γλώσσα είναι ήδη εγκατεστημένη.

## Windows

Ακολουθούμε τις οδηγίες από το σύνδεσμο https://docs.python.org/3/using/windows.html Ιδιαίτερη προσοχή οι παρακάτω επιλογές να είναι επιλεγμένες κατά την εγκατάσταση

1. ``Add python to environment variables'': χρειάζεται για νσ μπορούμε να τρέχουμε προγράμματα από τη γραμμή εντολών
2. ``pip'': είναι απαραίτητο για να μπορούμε να εγκαταστήσουμε τις βιβλιοθήκες που χρειαζόμαστε

## Macintosh

Ακολουθούμε τις οδηγίες: https://docs.python.org/3/using/mac.html

# Έλεγχος ορθής εγκατάστασης της python (Linux, Mac)

Για να επαληθεύσουμε την εγκατάσταση της γλώσσας, ανοίγουμε ένα τερματικό, και γράφουμε

```
python3 --version
pip3 --version
```

Θα πρέπει να δείτε την έκδοση των δύο παραπάνω προγραμμάτων, όπως το παρακάτω:

```
$ python3 --version
Python 3.8.2
$ pip3 --version
pip 20.2.2 from /usr/local/lib/python3.8/dist-packages/pip (python 3.8)
```

# Έλεγχος ορθής εγκατάστασης της python (Windows)

Για να επαληθεύσουμε την εγκατάσταση της γλώσσας, ανοίγουμε ένα τερματικό (Command prompt), και γράφουμε

```
1 python --version
2 pip --version
```

Θα πρέπει να δείτε την έκδοση των δύο παραπάνω προγραμμάτων, όπως το παρακάτω:

```
1 C:\Users\Dimitrios Vergados>python --version
2 Python 3.8.5
3
4 C:\Users\Dimitrios Vergados>pip --version
5 pip 20.1.1 from c:\users\dimitrios vergados\appdata\local\programs\python\
      python38\lib\site-packages\pip (python 3.8)
```

# Δημιουργία project για την εργασία Ι

## Εγκατάσταση προγράμματος για την επεξεργασία των αρχείων python.

Για τη συγκεκριμένη εργασία προτείνεται το πρόγραμμα vscode https://code.visualstudio.com/Download#, αλλά μπορεί να χρησιμοποιηθεί οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου

## Δημιουργία καταλόγου εργασίας για την εργασία

- Σε κάποιο σημείο του συστήματος δημιουργούμε ένα κενό κατάλογο (directory).
- Ανοίγουμε το `vscode` και επιλέγουμε `File` -> `Open Folder`, και επιλέγουμε τον κατάλογο που δημιουργήσαμε προηγουμένως.
- Από το μενού επιλέγουμε `File` -> `New File`. Το αποθηκεύουμε με όνομα που τελειώνει σε `.py`, πχ `iot_example.py`

# Δημιουργία project για την εργασία II

## Εγκατάσταση modules του vscode για διευκόλυνση συγγραφής κώδικα python

- Αφού αποθηκεύσουμε το αρχείο με επέκταση `.py`, το `vscode` θα προτείνει την εγκατάσταση των ακόλουθων 2 modules

  `ms-python.python` και `pylint`

# Εκτέλεση του πρώτου προγράμματος python I

Μέσα από το `vscode`, πληκτρολογούμε το ακόλουθο πρόγραμμα και το αποθηκεύουμε στο αρχείο `.py` που δημιουργήσαμε προηγουμένως.

***ΠΡΟΣΟΧΗ: Σε όλα τα προγράμματα `python` τα κενά στην αρχή της κάθε γραμμής έχουν συντακτική συμμασία, οπότε πρέπει να είμαστε προσεκτικοί να διατηρούνται τα κενά όπως φαίνονται στα παραδείγματα.***

# Εκτέλεση του πρώτου προγράμματος python II

```
1  class IoTExample:
2      def __init__(self):
3          print('Class is created')
4
5      def start(self):
6          print('Starting')
7
8  try:
9      iot_example = IoTExample()
10     iot_example.start()
11 except KeyboardInterrupt:
12     print("Interrupted")
13     try:
14         sys.exit(0)
15     except SystemExit:
16         os._exit(0)
```

# Εκτέλεση του πρώτου προγράμματος python III

Για να εκτελέσουμε το πρόγραμμα, από τη γραμμή εντολών εισερχόμαστε στον κατάλογο που βρίσκεται το πρόγραμμά μας (με την εντολή `cd <directory`), και στη συνέχεια πληκτρολογούμε:

σε linux, mac:

```
1 python3 iot_example.py
```

σε windows:

```
1 python iot_example.py
```

# Εκτέλεση του πρώτου προγράμματος python IV

όπου `iot_example.py` είναι το όνομα του αρχείου. Θα πρέπει να δούμε output όπως το παρακάτω:

```
1 Class is created
2 Starting
```

Εναλλακτικά μπορούμε να ξεκινήσουμε την εκτέλεση μέσα από το `vscode` από το μενού `Run`

# Ενότητα 2: Επικοινωνία με συσκευή IoT

# Γενικά Ι

## Η έξυπνη πρίζα μπορεί

- Να λαμβάνει μετρήσεις ηλεκτρικής κατανάλωσης των συσκευών στην πρίζα
- Να ανοίγει και κλείνει την τροφοδοσία μέσα από την πρίζα
- Να επικοινωνεί ασύρματα μέσω του δικτύου `z-wave` ή άλλων ασύρματων τεχνολογιών

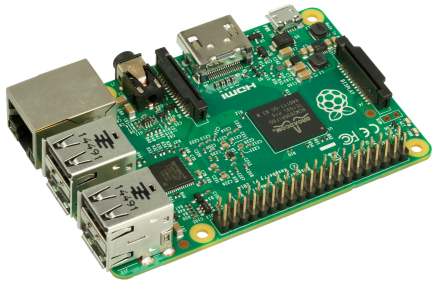
## Smart Plug

# Γενικά II

## Το Raspberry Pie

- Επικοινωνεί ασύρματα με z-wave ή άλλων τεχνολογιών με τις έξυπνες συσκευές
- (Δυνητικά) επεξεργάζεται και αποθηκεύει μετρήσεις, μπορεί να υλοποιεί «έξυπνους κανόνες», κ.α.
- Επικοινωνεί με το διαδίκτυο είτε ασύρματα (Wifi) είτε ενσύρματα (ethernet)
- Επικοινωνεί με τον διακομιστή μηνυμάτων MQTT broker για να του στέλνει τις μετρήσεις και να λαμβάνει εντολές

## Raspberry Pie



# Γενικά III

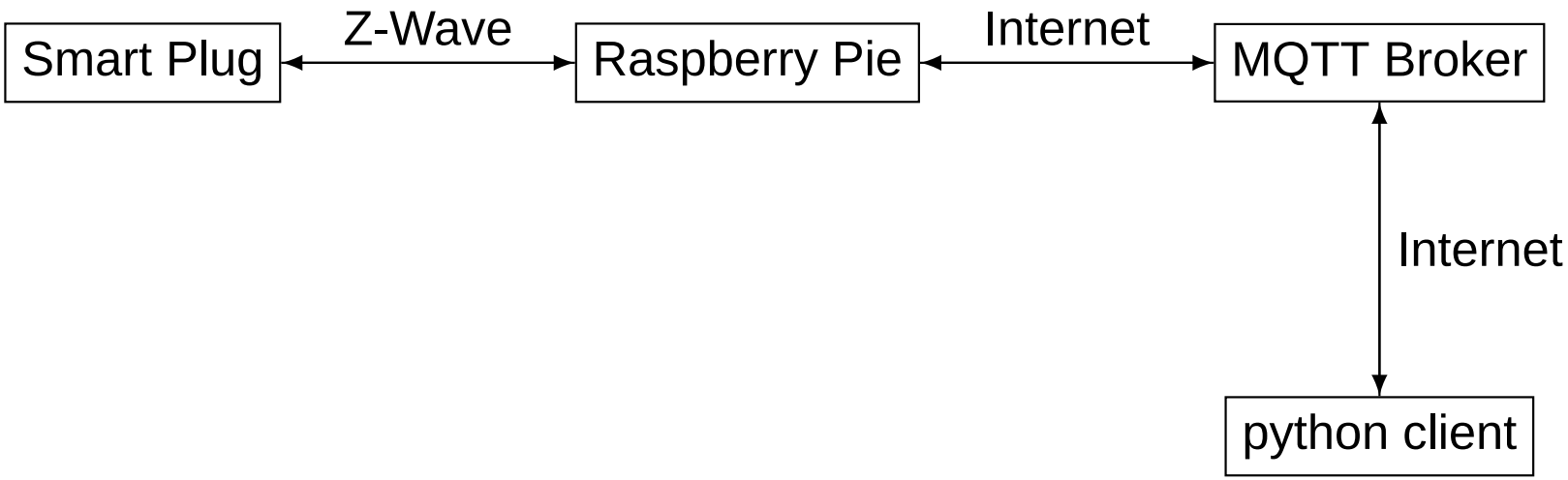
MQTT

Ο διακομιστής MQTT είναι ένα πακέτο λογισμικού το οποίο αναλαμβάνει τη διασύνδεση παραγωγών μηνυμάτων με τους αποδέκτες αυτών των μηνυμάτων:

- Δέχεται τα δεδομένα από τις έξυπνες συσκευές
- Επιτρέπει σε εφαρμογές να συνδεθούν (subscribe) σε κάποιο/α topic. Το topic είναι μια συμβολοσειρά (string) που καθορίζει που από τα μηνύματα «ενδιαφέρεται» να λαμβάνει η εφαρμογή
- Να δρομολογεί τα εισερχόμενα μυνήματα στις αντίστοιχες εφαρμογές, ανάλογα με τα topics που παρακολουθεί η κάθε μία

# Γενικά IV

## Αρχιτεκτονικό διάγραμμα



# MQTT topic για τον έλεγχο του smart plug

Τα MQTT topics που θα χρησιμοποιήσουμε για να λάβουμε δεδομένα σε αυτή την άσκηση είναι:

- `hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts/state`: Μας δείχνει τη στιγμιαία κατανάλωση του φορτίου σε Watt
- `hscnl/hscnl02/command/ZWaveNode005_Switch/command`: Μας δείχνει τις εντολές που έχουν σταλεί προς το smart plug
- `hscnl/hscnl02/state/ZWaveNode005_Switch/state`: Μας δείχνει την κατάσταση της πρίζας (`ON` η `OFF`)

Επίσης θα μπορούμε να στείλουμε εντολές στο topic: - `hscnl/hscnl02/sendcommand/ZWaveNode005_Switch`: Οι επιτρεπόμενες τιμές είναι `ON` για να ενεργοποιήσουμε το φορτίο ή `OFF` για να απενεργοποιήσουμε το φορτίο.

# Βιβλιοθήκη διασύνδεσης με το διακομιστή MQTT από τη γλώσσα python

Για τη διασύνδεση θα χρησιμοποιήσουμε τη βιβλιοθήκη `paho-mqtt`. Η εγκατάσταση γίνεται με τη βοήθεια του προγράμματος `pip` ως εξής:

- linux: `pip3 install paho-mqtt`
- windows: `pip install paho-mqtt`

Επίσης για να μπορούμε να χρησιμοποιήσουμε κλάσεις της βιβλιοθήκης στο αρχείο κώδικά μας, πρέπει να εισάγουμε την βιβλιοθήκη. Στο πάνω μέρος του προγράμματος προσθέτουμε

```
1 import paho.mqtt.client as mqtt
```

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε Ι

Η βασική κλάση που χρησιμοποιούμε είναι είναι η κλάση `mqtt.Client`. Για να χρησιμοποιήσουμε τη βιβλιοθήκη θα πρέπει αρχικά να δημιουργήσουμε ένα τέτοιο αντικείμενο με την εντολή

```
1 self.client = mqtt.Client()
```

(χρησιμοποιούμε `self.client = ... client =` ώστε η μεταβλητή να είναι διαθέσιμη και στις υπόλοιπες μεθόδους της κλάσης μας)

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε II

Οι μέθοδοι και παράμετροι της κλάσης `mqtt.Client` που θα χρησιμοποιήσουμε είναι οι ακόλουθες:

- `self.client.on_connect = self._on_connect`: Με αυτή την εντολή ορίζουμε ότι μόλις ολοκληρωθεί η εγκατάσταση της σύνδεσης, θα κληθεί η μέθοδος `_on_connect(self, client, userdata, flags, rc)` της τρέχουσας κλάσης. Θα πρέπει επίσης να παρέχουμε την υλοποίηση της μεθόδου.

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε III

- `self.client.on_log = self._on_log`: Με αυτή την εντολή ορίζουμε ότι η συνάρτηση `_on_log(self, client, userdata, level, buf)` θα χρησιμοποιείται για την παραγωγή διαγνωστικών μηνυμάτων σχετικά με τη σύνδεση με το MQTT. Θα πρέπει επίσης να παρέχουμε την υλοποίηση της μεθόδου.
- `self.client.on_message = self.on_message`: Με αυτή την εντολή ορίζουμε ότι κάθε φορά που θα λαμβάνεται ένα μήνυμα θα καλείται η συνάρτηση `_on_message(self, client, userdata, msg)`. Θα πρέπει επίσης να παρέχουμε την υλοποίηση της μεθόδου.

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε IV

- `self.client.tls_set_context(ssl.SSLContext(ssl.PROTOCOL_TLSv1_2))`: Με αυτή τη μέθοδο ορίζουμε ότι θα χρησιμοποιήσουμε κρυπτογράφυση για την μετάδοση/λήψη των μυνημάτων
- `self.client.username_pw_set('iotlesson', 'YGK0tx5pbtkK2WkCBvJlJWCg')`: Με αυτή τη μέθοδο ορίζουμε το όνομα χρήστη με το password
- `self.client.connect('phoenix.medialab.ntua.gr', 8883)`: Με αυτή τη μέθοδο ξεκινάμε τη σύνδεση με τον MQTT broker, και ορίζουμε το DNS όνομα του διακομιστή, και την πόρτα που θα γίνει η σύνδεση

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε V

- `client.subscribe('hscnl/hscnl02/state/ZWaveNode005_Switch/state')`: Αυτή η μέθοδος καλείται αφού έχει πραγματοποιηθεί η σύνδεση, ώστε να αρχίσει παρακολουθεί το topic που είναι στην παράμετρο
- `self.client.loop_forever()`: Με αυτή την μέθοδο μπλοκάρουμε την εκτέλεση του προγράμματος ώστε να μην τερματίσει και να αναμένει για μηνύματα

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε VI

- `self.client.loop_start()`: Αυτή η μέθοδος χρησιμοποιείται όταν θέλουμε ο MQTT Client να ***μήν*** μπλοκάρει, και να επιτρέψει τη συνέχιση του προγράμματος. Χρησιμοποιείται όταν θέλουμε να διατηρήσουμε τον έλεγχο της εκτέλεσης του προγράμματος, για να δείχνουμε τα γραφικά
- `self.client.publish('hscnl/hscnl02/sendcommand/ZWaveNode005_Switch', 'ON')`: Αυτή η μέθοδος χρησιμοποιείται για την αποστολή μηνύματος προς τον MQTT broker. Η πρώτη παράμετρος είναι το topic, ενώ η δεύτερη παράμετρος είναι το περιεχόμενο του μηνύματος.

# Κλάσεις και μέθοδοι της βιβλιοθήκης `paho-mqtt` που θα χρησιμοποιήσουμε VII

- `self.client.disconnect()`: Αυτή η μέθοδος χρησιμοποιείται για για την αποσύνδεση από τον MQTT broker.

# Οργάνωση του κώδικα Ι

Σύμφωνα με τα παραπάνω, ο σκελετός της κλάσης μας πρέπει να διαμορφωθεί ως εξής:

```
1  import paho.mqtt.client as mqtt
2  import ssl
3  import os
4  import sys
5
6
7  class IoTExample:
8      def __init__(self):
9          self._establish_mqtt_connection()
10
11     # call this method to start the client loop
12     def start(self):
13         self.client.loop_forever()
14
15     # call this method to disconnect from the broker
16     def disconnect(self, args=None):
```

# Οργάνωση του κώδικα II

```
17          self.client.disconnect()  # disconnect
18
19      # You need to fill this in, here we will place the commands
20      # For connecting to the broker
21      def _establish_mqtt_connection(self):
22          print("I need to be completed")
23
24      # This is the callback that will be called after the connection is
25      # established
26      def _on_connect(self, client, userdata, flags, rc):
27          print("I need to be completed")
28
29      # This is the callback that will be called whenever a new message
30      # is received
31      def _on_message(self, client, userdata, msg):
32          print(msg.topic+' '+str(msg.payload))
33
34      # This is the callback that will be called whenever a new log
35      # event is created
36      def _on_log(self, client, userdata, level, buf):
37          print('log: ', buf)
```

# Οργάνωση του κώδικα III

```
38
39
40 try:
41     iot_example = IoTExample()
42     iot_example.start()
43 except KeyboardInterrupt:
44     print('Interrupted')
45     try:
46         iot_example.disconnect()
47         sys.exit(0)
48     except SystemExit:
49         os._exit(0)
```

# Παράδειγμα επιτυχημένης εκτέλεσης Ι

```
$ python3 iot_example.py
Trying to connect to MQTT server...
log:  Sending CONNECT (u1, p1, wr0, wq0, wf0, c1, k60) client_id=b''
log:  Received CONNACK (0, 0)
Connected with result code 0
log:  Sending SUBSCRIBE (d0, m1) [(b'hscnl/hscnl02/state/
     ZWaveNode005_ElectricMeterWatts/state', 0)]
log:  Sending SUBSCRIBE (d0, m2) [(b'hscnl/hscnl02/command/
     ZWaveNode005_Switch/command', 0)]
log:  Sending SUBSCRIBE (d0, m3) [(b'hscnl/hscnl02/state/ZWaveNode005_Switch
     /state', 0)]
log:  Received SUBACK
log:  Received SUBACK
log:  Received SUBACK
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/command/
     ZWaveNode005_Switch/command', ...  (2 bytes)
hscnl/hscnl02/command/ZWaveNode005_Switch/command b'ON'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_Switch/state', ...  (2 bytes)
hscnl/hscnl02/state/ZWaveNode005_Switch/state b'ON'
```

# Παράδειγμα επιτυχημένης εκτέλεσης II

```
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_Switch/state', ...  (2 bytes)
hscnl/hscnl02/state/ZWaveNode005_Switch/state b'ON'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_Switch/state', ...  (2 bytes)
hscnl/hscnl02/state/ZWaveNode005_Switch/state b'ON'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_ElectricMeterWatts/state', ...  (3 bytes)
hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts/state b'7.8'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_ElectricMeterWatts/state', ...  (4 bytes)
hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts/state b'44.4'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/command/
     ZWaveNode005_Switch/command', ...  (3 bytes)
hscnl/hscnl02/command/ZWaveNode005_Switch/command b'OFF'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_Switch/state', ...  (3 bytes)
hscnl/hscnl02/state/ZWaveNode005_Switch/state b'OFF'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_Switch/state', ...  (3 bytes)
hscnl/hscnl02/state/ZWaveNode005_Switch/state b'OFF'
```

# Παράδειγμα επιτυχημένης εκτέλεσης III

```
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_ElectricMeterWatts/state', ...  (1 bytes)
hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts/state b'0'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_Switch/state', ...  (3 bytes)
hscnl/hscnl02/state/ZWaveNode005_Switch/state b'OFF'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_ElectricMeterWatts/state', ...  (1 bytes)
hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts/state b'0'
log:  Received PUBLISH (d0, q0, r0, m0), 'hscnl/hscnl02/state/
     ZWaveNode005_ElectricMeterWatts/state', ...  (1 bytes)
hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts/state b'0'
^CInterrupted
log:  Sending DISCONNECT
```

# Ενότητα 3: Γραφική απεικόνιση των αποτελεσμάτων

# Γενικά Ι

Σε αυτή την ενότητα θα προσθέσουμε ένα διάγραμμα για να απεικονίζουμε τις μετρούμενες τιμές, και 2 κουμπιά για να ανοίγουμε και να κλείνουμε τη συσκευή.

Για τη διασύνδεση θα χρησιμοποιήσουμε τη βιβλιοθήκη `matplotlib`. Η εγκατάσταση γίνεται με τη βοήθεια του προγράμματος `pip` ως έξής:

- linux: `pip3 install matplotlib`
- windows: `pip install matplotlib`

Επίσης για να μπορούμε να χρησιμοποιήσουμε κλάσεις της βιβλιοθήκης στο αρχείο κώδικά μας, πρέπει να εισάγουμε την βιβλιοθήκη. Στο πάνω μέρος του προγράμματος προσθέτουμε

# Γενικά II

```
1 import matplotlib.pyplot as plt
2 from matplotlib.widgets import Button
```

Επίσης θα χρειαστούμε τα ακόλουθα includes:

```
1 from threading import Timer
2 from datetime import datetime
```

# Γραφική απεικόνιση μετρήσεων Ι

Αρχικά θα προσθέσουμε τις ακόλουθες μαθόδους:

Για την αρχικοποίηση του γραφήματος

```
def _prepare_graph_window(self):
                # Plot variables
        plt.rcParams['toolbar'] = 'None'
        self.ax = plt.subplot(111)
        self.dataX = []
        self.dataY = []
        self.first_ts = datetime.now()
        self.lineplot, = self.ax.plot(
            self.dataX, self.dataY, linestyle='--', marker='o', color='b')
        self.ax.figure.canvas.mpl_connect('close_event', self.disconnect)
        self.finishing = False
        self._my_timer()
```

# Γραφική απεικόνιση μετρήσεων II

Για την ανανέωση του γραφήματος όταν λαμβάνονται νέα αποτελέσματα ή αλλάζει η απεικόνιση του άξονα x

```
    def _refresh_plot(self):
        if len(self.dataX) > 0:
            self.ax.set_xlim(min(self.first_ts, min(self.dataX)),
                             max(max(self.dataX), datetime.now()))
            self.ax.set_ylim(min(self.dataY) * 0.8, max(self.dataY) * 1.2)
            self.ax.relim()
        else:
            self.ax.set_xlim(self.first_ts, datetime.now())
            self.ax.relim()
        plt.draw()
```

# Γραφική απεικόνιση μετρήσεων III

Για να προσθέτουμε μια νέα τιμή στο γράφημα:

```
1     def _add_value_to_plot(self, value):
2         self.dataX.append(datetime.now())
3         self.dataY.append(value)
4         self.lineplot.set_data(self.dataX, self.dataY)
5         self._refresh_plot()
```

# Γραφική απεικόνιση μετρήσεων IV

Για την αυτόματη μετακίνιση του γραφήματος δεξιά κάθε δευτερόλεπτο

```
1     def _my_timer(self):
2         self._refresh_plot()
3         if not self.finishing:
4             Timer(1.0, self._my_timer).start()
```

# Γραφική απεικόνιση μετρήσεων V

Επίσης θα πρέπει να τροποποιήσουμε το χειριστή `_on_message` ώς εξής:

```
1     def _on_message(self, client, userdata, msg):
2         if msg.topic == 'hscnl/hscnl02/state/ZWaveNode005_ElectricMeterWatts
              /state':
3                 self._add_value_to_plot(float(msg.payload))
4         print(msg.topic+' '+str(msg.payload))
```

# Γραφική απεικόνιση μετρήσεων VI

Τέλος θα αλλάξουμε τις μεθόδους `__init__()` και `start` ώς εξής:

```
1     def __init__(self):
2         self.ax = None
3         self._establish_mqtt_connection()
4         self._prepare_graph_window()
5
6     def start(self):
7         if self.ax:
8             self.client.loop_start()
9             plt.show()
10        else:
11            self.client.loop_forever()
```

# Ενότητα 4: Αποκακρυσμένος έλεγχος συσκευής.

# Αποκακρυσμένος έλεγχος συσκευής I

Θα φτιάξουμε 2 χειριστές για τα κουμπιά:

```
1     def _button_on_clicked(self, event):
2         self.client.publish(
3             'hscnl/hscnl02/sendcommand/ZWaveNode005_Switch', 'ON')
4
5     def _button_off_clicked(self, event):
6         self.client.publish(
7             'hscnl/hscnl02/sendcommand/ZWaveNode005_Switch', 'OFF')
```

# Αποκακρυσμένος έλεγχος συσκευής II

Πρέπει να προσθέσουμε 2 κουμπιά και ένα πεδίο κειμένου στο γράφημα:

```
        axcut = plt.axes([0.0, 0.0, 0.1, 0.06])
        self.bcut = Button(axcut, 'ON')
        axcut2 = plt.axes([0.1, 0.0, 0.1, 0.06])
        self.bcut2 = Button(axcut2, 'OFF')
        self.state_field = plt.text(1.5, 0.3, 'STATE: -')
        self.bcut.on_clicked(self._button_on_clicked)
        self.bcut2.on_clicked(self._button_off_clicked)
```

# End of presentation

End of presentation

## End of presentation

# Questions?

Questions?

dvergados@uowm.gr