Pycom – multinetwork devices

Part One: MQTT via WiFi
José M. Cámara Nebreda. 2018
IoT devices on offer

• Development modules:
  • WiPy 3.0: WiFi & Bluetooth
  • LoPy: LoRa, WiFi & Bluetooth
  • SiPy: Sigfox, WiFi & Bluetooth
  • PoPy4: LoRa, Sigfox, WiFi & Bluetooth
  • Gpy: LTE-M, WiFi & Bluetooth
  • FiPy: LoRa, Sigfox, LTE-M, WiFi & Bluetooth

• Integrated circuits (OEM):
  • WO1: WiFi & Bluetooth
  • G01: LTE-M, WiFi & Bluetooth
  • LO1: LoRa, WiFi & Bluetooth
  • LO4: LoRa, Sigfox, WiFi & Bluetooth
FiPy

• Features:
  • ESP32 dual core microcontroller
  • USB host port
  • 8 MB flash memory
  • 4 MB RAM memory
  • 3v3 regulator

• Hardware requirements:
  • FiPy module
  • Host PC
  • USB or MicroUSB cable
  • Expansion board (recommended)

• Software requirements:
  • Text editor: ATOM recommended by manufacturer. It will perform as an IDE.
  • Pymakr Plugin for the editor to enable programing the device.
Pymkr Installation

For further information refer to: https://docs.pycom.io/chapter/gettingstarted/installation/pymakr.html
Start – up process

• Connect FiPy to expansion board and then to the host computer via USB cable.
• Remove RTS and CTS jumpers from the expansion board.
• Update the firmware using the pycom firmware update tool: 
  https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html
• Connect to your FiPy board: 
  https://docs.pycom.io/chapter/pymakr/installation/atom.html
• Try: import pycom 
  pycom.heartbeat(False) 
  pycom.rgbled(0x00ff00) 
  this should turn the LED green
• FiPy has a built-in wifi radio module.
• It can be configured for several operation modes:
  • AP (After power-up): this is the default mode. FiPy becomes an Access poit to where othre devices are connected. It can be found as “fipy-wlan-xxxx”. Its password is always www.pycom.io.
  • STA (station): the device is expected to connect to an external router. Three authentication modes are supported:
    • WPA personal: this mode enables FiPy to connect to your home network.
    • WPA Enterprise (EAP – TLS): this enables Fipy to connect to entrepise networks using certificates.
    • WPA Enterprise (EAP – TTLS or PEAP): to connect to Enterprise networks using user/passwork authentication.
# Wifi examples

## AP mode

**CODE:**
```python
code: from network import WLAN
wlan = WLAN() # Default mode is AP
print("Network name: ", wlan.ssid())
print("IP address: ", wlan.ifconfig())
print("Autentication: ", wlan.auth())
```

**OUTPUT:**

Network name: fipy-wlan-0090
IP address: ('0.0.0.0', '0.0.0.0', '0.0.0.0', '0.0.0.0')
Autentication: (3, 'www.pycom.io')

## WPA personal

**CODE:**
```python
code: from network import WLAN
wlan = WLAN(mode=WLAN.STA)
print('Trying')
nets = wlan.scan()
for net in nets:
    if net.ssid == 'my-wifi':
        print('Network found!')
        wlan.connect(net.ssid, auth=(net.sec, 'wifi-key'), timeout=5000)
        print('connection established')
        break
```

**OUTPUT:**

Trying
Network found!
connection established
## WPA Enterprise code

**GENERAL CASE:**
```python
from network import WLAN
wlan = WLAN(mode=WLAN.STA)
wlan.connect(ssid="mywifi", auth=(WLAN.WPA2_ENT, 'user', 'password'), identity='myid', ca_certs='/flash/cert/ca.pem')
```

**OUR CASE:**
```python
import pycom
import machine
Pycom.heartbeat(False)
from network import WLAN
wlan = WLAN(mode=WLAN.STA)
wlan.connect(ssid="wifiubu", auth=(WLAN.WPA2_ENT, 'user', 'password'))
while not wlan.isconnected():
    machine.idle()
pycop.rgbled(0x00ff00) # turn on the LED when connected
```

- Application layer protocol
- Publisher/subscriber model
- Built on top of the protocol stack, provides a messaging mechanism to provide and obtain information in short pieces
- For testing, Pycom recommends the free Adafruit services: [https://io.adafruit.com/](https://io.adafruit.com/)
  - Sign in and you will gain access to you personal account (next slide)
  - You will need to download the mqtt library mqtt.py at: [https://github.com/pycom/pycom-libraries/tree/master/lib/mqtt](https://github.com/pycom/pycom-libraries/tree/master/lib/mqtt)
  - [https://docs.pycom.io/chapter/tutorials/all/mqtt.html](https://docs.pycom.io/chapter/tutorials/all/mqtt.html)
Here you can get your personal API key to include in your programming.
In “Actions” we create two new feeds:
  LED: meant to change the status of the onboard LED
  Temp: meant to display temperature readings from the PySENSE built in thermometer

Feeds are input/output:
  When the Fipy publishes a new value, it is updated and displayed online
  When the Fipy subscribes, values changed online are received and read by Fipy
  Fipy can both publish and subscribe to the same feed
MQTT connection example

**Code section**

```python
#mqtt section
print('Trying to connect Adafruit MQTT')
client = MQTTClient('FiPy', 'io.adafruit.com', port=1883, user='checam', password='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx')
client.set_callback(sub_cb)
client.connect()
print('Connection successful')
client.subscribe(topic="checam/feeds/led")
client.subscribe(topic="checam/feeds/temp")
pycom.rgbled(0xff00ff)
```

**Output**

```
Trying to connect Adafruit MQTT
Connection successful
```
Mqtt library functionality

- `connect()`: sets up a socket to communicate with the server
- `disconnect()`: ends the socket
- `publish()`: sends information to a topic on the server
- `subscribe()`: subscribes to a topic on the server
- `set_callback()`: sets up the routine to which incoming subscribed messages are delivered
- `wait_msg()`: wait for subscribed messages to arrive. When arrived the message is processed and sent to a callback routine
- `chk_msg()`: checks whether a message is pending. If it is, proceeds as `wait_msg()`
- `ping()`: checks connection
Pysense board provides several readings:
- Temperature
- Humidity
- Pressure
- Acceleration
- Roll
- Pitch
- Light level
- Altitude

Code section

```python
#pysense section
mp = MPL3115A2(mode=ALTITUDE) # Returns height in meters. Mode may also be set to PRESSURE, returning a value in Pascals
acc = LIS2HH12()
si = SI7006A20()
lit = LTR329ALS01()

while True:
    pitch = acc.pitch()
    roll = acc.roll()
    light = lit.light()
    print("Pitch & Roll: " + '{},{}'.format(pitch, roll))
    print("Light: " + '{:.2f}'.format(light))
    print("Altitude: " + str(mp.altitude()))
    print("Temperature: " + str(si.temperature())+ " deg C and Relative Humidity: " + str(si.humidity()) + " %RH")
    client.publish(topic="checam/feeds/temp", msg=str(si.temperature()))
    client.check_msg()
    time.sleep(5)
```
Pysense example

Output

Pitch & Roll: -3.634626,110.4271
Light: (32, 27)
Altitude: 773.875
Temperature: 34.27464 deg C and Relative Humidity: 38.30771 %RH

Pysense library files

from LIS2HH12 import LIS2HH12
from SI7006A20 import SI7006A20
from LTR329ALS01 import LTR329ALS01
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE
We use the onboard LED to show how commands can be received from the server and therefore from the remote user.

On the Pysense code example we had the line:
```python
client.check_msg()
```

This MQTT method checks whether there is an incoming message:
- In case there is no one, it does nothing.
- If a message has arrived, it is delivered to the routine defined in the MQTT connection code example:
  ```python
  client.set_callback(sub_cb)
  ```

The routine sets what to do with the message:
```python
def sub_cb(topic, msg):
    print(msg)
    print(msg.decode("utf-8"))
    if msg.decode("utf-8") == 'ON':
        pycom.rgbled(0xffffff)
    if msg.decode("utf-8") == 'OFF':
        pycom.rgbled(0x000000)
```

In this case it turns on and off the onboard LED depending on the text of the message.
Micropython mqtt library issues

- Using the mqtt.py module we have encountered the following issue:
  - OSError -1 on method connect().
- For some reason, the getaddrinfo() function must be invoked within the socket.connect() method. To make that possible, we have added two new attributes at __init__(): self.server & self.port.
- We have replaced: self.sock.connect(self.addr) by self.sock.connect(socket.getaddrinfo(self.server,self.port)[0][-1])
- Connection is yet not always successful. A certain delay seems to be necessary in the form:
  ```
  self.sock = socket.socket()
  time.sleep(10)
  self.sock.connect(socket.getaddrinfo(self.server,self.port)[0][-1])
  ```
References

• Although the contents of this presentation are original, most of the information, code and data included are based on the official Pycom website: https://pycom.io/

• A complete guide about the set-up, connection and programming of the development boards including FiPy can be found at: https://docs.pycom.io/