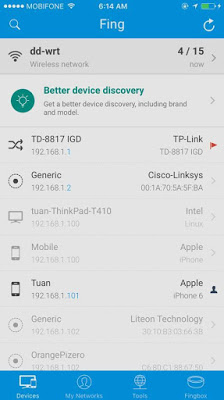
# **[Demo 18:](http://www.iotsharing.com/2017/05/how-to-use-arduino-esp32-to-display-spi-led-matrix.html) [How to testing/debugging IoT application (TCP, UDP, HTTP, MQTT)](http://www.iotsharing.com/2017/06/how-to-make-iot-testing-application-using-python.html)**

**1. Introduction**  
- When working with IoT application, we often work with protocols such as TCP/UDP, HTTP, MQTT, ... So we need to know some tools/utilities for debugging, testing. Beside the most famous tool for Internet monitor is [Wireshark](https://www.wireshark.org/download.html" \t "http://www.iotsharing.com/2017/06/_blank), I often write some simple application for testing, debugging using Python. I choose Python because it is easy to setup, learn, use. Here are some simple Python programs that are useful for our purpose.  
**1.1 Scan network**  
Before going to next steps we must ensure that IoT nodes exist on the network.  
We can use **"ping ip\_or\_mdns\_name\_of\_node"**  
or an application called Fing-Network Scanner that is available on [Apple store](https://itunes.apple.com/us/app/fing-network-scanner/id430921107?mt=8" \t "http://www.iotsharing.com/2017/06/_blank) and [Google store](https://play.google.com/store/apps/details?id=com.overlook.android.fing" \t "http://www.iotsharing.com/2017/06/_blank)

[](https://4.bp.blogspot.com/-cbY7Pl4zmFg/WUOhdbYBTrI/AAAAAAAAEEY/iaSt4ceSY1gld-34pHtyJ2ujp-dSmv_mQCLcBGAs/s1600/fing.jpg)

**Figure: Fing network scanner, alive node has black color**

**2. Testing TCP**  
- Create a TCP server and TCP client. Client will send the data to server, server convert to upper case and respond to client.

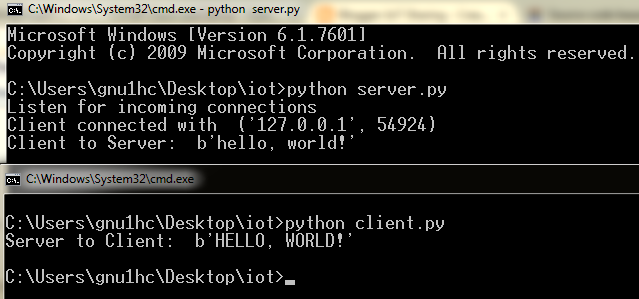
**2.1 TCP Server**

|  |
| --- |
| import socket  # bind all IP address  HOST = '  0.0.0.0  '  # Listen on Port  PORT = 44444  #Size of receive buffer  BUFFER\_SIZE = 1024  # Create a TCP/IP socket  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  # Bind the socket to the host and port  s.bind((HOST, PORT))  print('Listen for incoming connections')  s.listen(1)  while True:  # Accept the incomming connection  conn, addr = s.accept()  # Print the info of client  print ('Client connected with ' , addr)  # Receive BUFFER\_SIZE bytes data  data = conn.recv(BUFFER\_SIZE)  if data:  #print received data  print('Client to Server: ' , data)  # Convert to upper case and send back to Client  conn.send(data.upper())  # Close connection  conn.close() |

**2.2 TCP Client**

|  |
| --- |
| import socket  # bind all IP address  HOST = '  0.0.0.0  '  # Listen on Port  PORT = 44444  #Size of receive buffer  BUFFER\_SIZE = 1024  # Create a TCP/IP socket  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  # Bind the socket to the host and port  s.bind((HOST, PORT))  print('Listen for incoming connections')  s.listen(1)  while True:  # Accept the incomming connection  conn, addr = s.accept()  # Print the info of client  print ('Client connected with ' , addr)  # Receive BUFFER\_SIZE bytes data  data = conn.recv(BUFFER\_SIZE)  if data:  #print received data  print('Client to Server: ' , data)  # Convert to upper case and send back to Client  conn.send(data.upper())  # Close connection  conn.close() |

**2.3 Result**

[](https://2.bp.blogspot.com/-_mHDmGYRCtg/WTZjVc4TpsI/AAAAAAAAEBc/VszbfjtFfCE53rzGoxDUtDacReTEUOO7wCPcB/s1600/tcp.png)

**Figure: Testing TCP**

**3. Testing UDP**  
- Create a UDP server and UDP client. Client will send the data to server, server convert to upper case and respond to client.

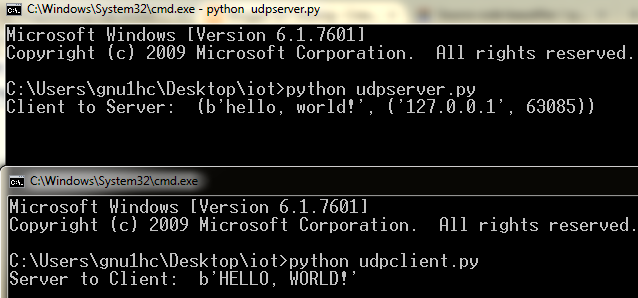
**3.1 UDP Server**

|  |
| --- |
| import socket  # bind all IP address  HOST = '0.0.0.0'  # Listen on Port  PORT = 44444  #Size of receive buffer  BUFFER\_SIZE = 1024  # Create a TCP/IP socket  s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  # Bind the socket to the host and port  s.bind((HOST, PORT))  while True:  # Receive BUFFER\_SIZE bytes data  # data is a list with 2 elements  # first is data  #second is client address  data = s.recvfrom(BUFFER\_SIZE)  if data:  #print received data  print('Client to Server: ' , data)  # Convert to upper case and send back to Client  s.sendto(data[0].upper(), data[1])  # Close connection  s.close() |

**3.2 UDP Client**

|  |
| --- |
| import socket  # Ip of local host  HOST = '127.0.0.1'  # Connect to Port  PORT = 44444  #Size of send buffer  BUFFER\_SIZE = 1024  # data to sent to server  data = 'hello, world!'  # Create a TCP/IP socket  s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  # send data to server  s.sendto(bytearray(data, 'utf-8'), (HOST, PORT))  # Receive response from server  # data is a list with 2 elements  # first is data  #second is client address  data = s.recvfrom(BUFFER\_SIZE)  # Close connection  s.close()  print ('Server to Client: ' , data[0]) |

**3.3 Result**

[](https://4.bp.blogspot.com/-WCnLXk0TXj4/WTZq7Pl--wI/AAAAAAAAEBc/GG32cEIYOdYpd-JiqT2oSc1GZMoVGDdKgCPcB/s1600/udp.png)

**Figure: Testing UDP**

**4. Testing HTTP**

- In order to test HTTP we create HTTP GET or POST request and monitor the response (data and status code) from server.We will use the Requests library in Python.

- Install it by using command line: "**sudo pip install requests"**and to use it just **"import requests"**

**4.1 HTTP GET request**

|  |
| --- |
| import requests  # we use HTTP GET to get content of web page  r = requests.get('http://iotsharing.com')  # print status code response  print (r.status\_code)  # print header response  print(r.headers)  # print content response  print (r.text) |

**4.2 HTTP POST request**  
- Using **"r = requests.post(http://iotsharing.com/post)  "**  
**4.3 Result**

[IMG_256](https://1.bp.blogspot.com/-b4HryQe4SRA/WTbF82UfwdI/AAAAAAAAEBs/z0W9c9VXFBQY_GPg_by6lw0TB-giTB5xgCLcB/s1600/httptest.png)

**Figure: Response for HTTP request testing**

**5. Testing MQTT Client**

- In order to test MQTT Client we will use paho MQTT. You can refer [Demo 15: How to build a system to update Price Tag automatically using Arduino ESP32](http://www.iotsharing.com/2017/05/how-to-build-system-to-update-price-tag.html" \t "http://www.iotsharing.com/2017/06/_blank)

**5.1 Testing MQTT publish**

|  |
| --- |
| import sys  try:  import paho.mqtt.publish as publish  except ImportError:  import os  import inspect  cmd\_subfolder = os.path.realpath(os.path.abspath(os.path.join(os.path.split(inspect.getfile( inspect.currentframe() ))[0],"../src")))  if cmd\_subfolder not in sys.path:  sys.path.insert(0, cmd\_subfolder)  import paho.mqtt.publish as publish  mosquitto\_ip = "192.168.1.104"  publish.single("test/message", hostname=mosquitto\_ip) |

**5.2 Testing MQTT subscribe**

|  |
| --- |
| import sys  try:  import paho.mqtt.subscribe as subscribe  except ImportError:  # This part is only required to run the example from within the examples  # directory when the module itself is not installed.  #  # If you have the module installed, just use "import paho.mqtt.subscribe"  import os  import inspect  cmd\_subfolder = os.path.realpath(os.path.abspath(os.path.join(os.path.split(inspect.getfile( inspect.currentframe() ))[0],"../src")))  if cmd\_subfolder not in sys.path:  sys.path.insert(0, cmd\_subfolder)  import paho.mqtt.subscribe as subscribe  import paho.mqtt.client  mosquitto\_ip = "192.168.1.104"  #callback function will be invoked when message is available  def print\_msg(client, userdata, message):  print("%s : %s" % (message.topic, message.payload))  #subscribe all topics  subscribe.callback(print\_msg, "#", hostname=mosquitto\_ip) |