EL4012 Lab 4: Transport Layer

Learning Outcomes

1. Set up and configure Thonny
2. Use Thonny to develop software for the Ubuntu Desktop machine
3. Explore TCP Ports 17 and 123 using Python

In this lab we will start to use the VM to connect to the network and exchange data.

## Task 1: Install Thonny

Thonny is a lightweight Python IDE which can be used to develop and download MicroPython code to the Pyboard. It can also be used to write standard Python running on a desktop. It is freely available for both Windows and Linux.

1. Install ‘curl’, a utility for downloading software:  
     
   sudo apt install curl
2. Use curl to download Thonny:  
     
   bash <(curl -s https://thonny.org/installer-for-linux)  
     
   You will then need to press ‘Enter’. It can take a while to download.  *[This failed on me once; if it seems to hang, press ‘Ctrl-C’ (to abandon it) and try again.]*
3. To run Thonny:  
     
   /apps/thonny/bin/Thonny  
   (or go to Activities and search for it)  
     
   You may wish to create a ‘favourite’ icon to make running it easier next time – your choice.
4. When you run Thonny it will open up showing an editor pane at the top, and a ‘shell’ frame at the bottom. The Shell frame will say something like ‘*Python3.7.4 (bundled)*’. This confirms that it is running Python3 correctly.

At the REPL (Read-evaluate-print-loop) prompt ‘>>>’ type:  
  
2\*\*100  
  
This causes Python to calculate 2^100 – the answer is large, and confirms everything is OK

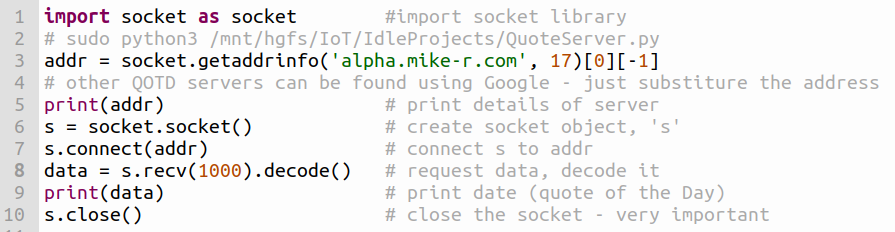
To try out Thonny as an IDE, enter the following code into the editor and save it in your Documents folder. The run it (press green arrow icon below menu bar)

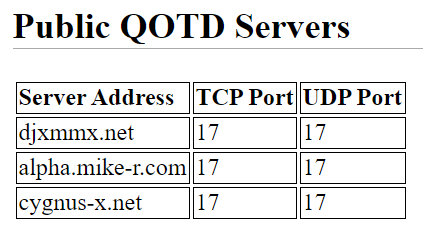
#comment line

for i in range (10):

print(i) # the indent is important!

The result should confirm that the code is running, and printing output to the REPL.

1. Thonny is currently generating and running code which runs on the Ubuntu machine. Later, we will configure it to work with the Microcontroller. The Ubuntu machine already has ‘network access’, so any code you write can use this hardware. The following code will access port 17 of a PC in order to retrieve a ‘Quote of the Day’. Type it in and run it (you may omit the comments, which are there to help your understanding):  
     
     
     
   Each time you run it, you should get a random quote.



1. Port 123 is the ‘well-known port’ for timeservers. The code below retrieves the ‘time’ in the form of a count of seconds, described thus:  
   A Timestamp, Unix time, or POSIX time, is a system for describing points in time, defined as the number of seconds elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds.

It needs to be converted before it can tell us the actual date/time. The conversion can be done by entering the number of seconds at the following website: <https://www.timestampconvert.com/>   
  
from socket import socket, AF\_INET, SOCK\_DGRAM, getaddrinfo

from struct import unpack, calcsize

NTP\_DELTA = 2208988800 # 1970-01-01 00:00:00

NTP\_QUERY = '\x1b' + 47 \* '\0'

NTP\_PACKET\_FORMAT = "!12I"

addr\_info = getaddrinfo("ntp2d.mcc.ac.uk", 123)[0][-1] # Manchester Uni.

sk=socket(AF\_INET, SOCK\_DGRAM)

sk.sendto(NTP\_QUERY.encode('utf-8'), addr\_info)

msg, address = sk.recvfrom(1024)

print("msg = ",msg)

print (address)

unpacked = unpack(NTP\_PACKET\_FORMAT, msg[0:calcsize(NTP\_PACKET\_FORMAT)])

print(unpacked[10] + float(unpacked[11]) /2\*\*32 - NTP\_DELTA)

#go to https://www.timestampconvert.com/ to convert

sk.close() # Don't forget to close the socket

Copy and Paste, run the code, decode the date/time.

Examine the code. Look for familiar commands, words like print, import, socket, etc. The getaddrinfo() function performs a DNS lookup for the time server website in Manchester.

Note the strange binary format of the message coming from the timeserver, and the need to decode it.

Start to look up the syntax of unfamiliar Python commands – there are plenty of websites.