Deel 3 overzicht: LES1

* Sockets
  + met RP
  + met ESP32
* Client/server
  + met ESP32
  + met RP
* Hercules tool
  + client en server. H>H H>ESP ESP>H H>RP RP>H ESP>H H>ESP
* Threads, om meerdere clients te connecteren
* HTML basics
* HTTP protocol
* Webserver
* Je eigen website met enkel HTML op ESP32 en op RP
* CSS basics
* Je eigen website met HTML+CSS op ESP32 en op RP
* Javascript basics
* Javascript DOM
* Ajax (en mogelijk indien tijd Fetch, promises, async await)
* Je eigen website met HTML+CSS+JS op ESP32 en op RP
* Websockets!!! ( R5)
* API’s
* MQTT
* Thingsboard
* Node-Red
* Project: Default eindproject voor RP3, sturing en dashboard voor een domotica-beveiliging-toegangscontrole oplossing.

Les 1 : Opdrachten

1. Stuur met Hercules een led op je RP
2. Stuur met Hercules een led op je ESP32
3. Stuur een bericht “button pressed” van je ESP naar Hercules
4. Stuur een bericht “button pressed” van je RP naar Hercules
5. Stuur een led op je RP met een button op je ESP
6. Stuur een led op je ESP met een button op je RP
7. ESP en RP hebben elk een button en een led en sturen elkaars led met hun button RP server, daarna ESP server.
8. Extra voor wie veel tijd heeft => leds sturen via MIT APP INV TCP extention

**Gebruik Socket in RP SERVER programma= Socket basic server RP.py**

**'''**

**RP3-L4 Socket TCP server , test met Hercules!**

**'''**

**import socket**

**import time**

**PORT=7808**

**s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**s.bind(('', PORT))**

**s.listen(5)**

**request=b""**

**conn = None**

**# trukje om IP-adres van je RP te dedecteren**

**def get\_ip\_address():**

**ip\_address = '';**

**s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)**

**s.connect(("8.8.8.8",80))**

**print("s.getsockname()=",s.getsockname())**

**ip\_address = s.getsockname()[0]**

**print("ip\_address=",ip\_address)**

**s.close()**

**return ip\_address**

**try:**

**while True:**

**print("Server is waiting for a connection at",get\_ip\_address(), "and port", PORT)**

**conn, addr = s.accept()**

**print('Server got a connection from' , addr)**

**while True:**

**request = conn.recv(2048)**

**if len(request)> 0:**

**print("len request =",len(request))**

**print("message from client=",request.decode("UTF-8"))**

**else:**

**print(“client disconnected”)**

**break**

**except Exception as e:**

**print("Except>" , e)**

**finally:**

**print("Finished!")**

**Gebruik Socket in RP SERVER programma= Socket RP TCP client sends.py**

**]mport socket**

**SERVER = "192.168.0.172" # vind dit op tab UDP Setup van Hercules**

**PORT = 4443**

**ADDR = (SERVER, PORT)**

**FORMAT = "utf-8"**

**client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**client.connect(ADDR)**

**print("client connected")**

**try:**

**while True:**

**to\_send= input("geef uw tekst die u wil versturen aub!")**

**if to\_send == "QUIT":**

**to\_send = ""**

**message = to\_send.encode(FORMAT)**

**client.sendall(message)**

**break**

**else:**

**message = to\_send.encode(FORMAT)**

**client.sendall(message)**

**print("out of endless loop")**

**except Exception as e:**

**print("exception e=", e)**

**finally:**

**client.close()**

**print("finally")**

**Gebruik Socket ESP32**

[**https://micropython.org/download/esp32/**](https://micropython.org/download/esp32/) **v1.17**

**Thonny v3.13**

https://docs.micropython.org/en/latest/esp8266/tutorial/network\_basics.html

**4. Network basics**

The network module is used to configure the WiFi connection. There are two WiFi interfaces, one for the station (when the ESP connects to a router) and one for the access point (for other devices to connect to the ESP). Create instances of these objects using:

**>>> import** **network**

**>>>** sta\_if = network.WLAN(network.STA\_IF)

**>>>** ap\_if = network.WLAN(network.AP\_IF)

You can check if the interfaces are active by:

**>>>** sta\_if.active()

False

**>>>** ap\_if.active()

True

You can also check the network settings of the interface by:

**>>>** ap\_if.ifconfig()

('192.168.4.1', '255.255.255.0', '192.168.4.1', '8.8.8.8')

The returned values are: IP address, netmask, gateway, DNS.

**4.1. Configuration of the WiFi**

Upon a fresh install the ESP8266 is configured in access point mode, so the AP\_IF interface is active and the STA\_IF interface is inactive. You can configure the module to connect to your own network using the STA\_IF interface.

First activate the station interface:

**>>>** sta\_if.active(**True**)

Then connect to your WiFi network:

**>>>** sta\_if.connect('<your ESSID>', '<your password>')

To check if the connection is established use:

**>>>** sta\_if.isconnected()

Once established you can check the IP address:

**>>>** sta\_if.ifconfig()

('192.168.0.2', '255.255.255.0', '192.168.0.1', '8.8.8.8')

You can then disable the access-point interface if you no longer need it:

**>>>** ap\_if.active(**False**)

Here is a function you can run (or put in your boot.py file) to automatically connect to your WiFi network:

**def** do\_connect():

**import** **network**

sta\_if = network.WLAN(network.STA\_IF)

**if** **not** sta\_if.isconnected():

print('connecting to network...')

sta\_if.active(**True**)

sta\_if.connect('<essid>', '<password>')

**while** **not** sta\_if.isconnected():

**pass**

print('network config:', sta\_if.ifconfig())

**4.2. Sockets**

Once the WiFi is set up the way to access the network is by using sockets. A socket represents an endpoint on a network device, and when two sockets are connected together communication can proceed. Internet protocols are built on top of sockets, such as email (SMTP), the web (HTTP), telnet, ssh, among many others. Each of these protocols is assigned a specific port, which is just an integer. Given an IP address and a port number you can connect to a remote device and start talking with it.

The next part of the tutorial discusses how to use sockets to do some common and useful network tasks.

# 5. Network - TCP sockets

The building block of most of the internet is the TCP socket. These sockets provide a reliable stream of bytes between the connected network devices. This part of the tutorial will show how to use TCP sockets in a few different cases.

The simplest thing to do is to download data from the internet. In this case we will use the Star Wars Asciimation service provided by the blinkenlights.nl website. It uses the telnet protocol on port 23 to stream data to anyone that connects. It’s very simple to use because it doesn’t require you to authenticate (give a username or password), you can just start downloading data straight away.

The first thing to do is make sure we have the socket module available:

**>>> import** **socket**

Then get the IP address of the server:

**>>>** addr\_info = socket.getaddrinfo("towel.blinkenlights.nl", 23)

The getaddrinfo function actually returns a list of addresses, and each address has more information than we need. We want to get just the first valid address, and then just the IP address and port of the server. To do this use:

**>>>** addr = addr\_info[0][-1]

If you type addr\_info and addr at the prompt you will see exactly what information they hold.

Using the IP address we can make a socket and connect to the server:

**>>>** s = socket.socket()

**>>>** s.connect(addr)

Now that we are connected we can download and display the data:

**>>> while** **True**:

**...**  data = s.recv(500)

**...**  print(str(data, 'utf8'), end='')

**...**

**ESP32 voorbeeld van een basic client prog = basic client esp32.py**

''' https://docs.micropython.org/en/latest/esp8266/tutorial/network\_basics.html

4.2. Sockets

Once the WiFi is set up the way to access the network is by using sockets. A socket represents an endpoint on a network device,

and when two sockets are connected together communication can proceed.

Internet protocols are built on top of sockets, such as email (SMTP), the web (HTTP), telnet, ssh, among many others.

Each of these protocols is assigned a specific port, which is just an integer.

Given an IP address and a port number you can connect to a remote device and start talking with it.

The next part of the tutorial discusses how to use sockets to do some common and useful network tasks.

'''

def do\_connect():

import network

sta\_if = network.WLAN(network.STA\_IF)

if not sta\_if.isconnected():

print('connecting to network...')

sta\_if.active(True)

sta\_if.connect('telenet-5432836', pwd)

while not sta\_if.isconnected():

pass

print('network config:', sta\_if.ifconfig()) #network config: ('192.168.0.110', '255.255.255.0', '192.168.0.1', '42.2.24.0')

do\_connect()

# The first thing to do is make sure we have the socket module available:

import socket

#Then get the IP address of the server:

addr\_info = socket.getaddrinfo("192.168.0.172", 5555)

#The getaddrinfo function actually returns a list of addresses, and each address has more information than we need.

#We want to get just the first valid address,

#and then just the IP address and port of the server. To do this use:

addr = addr\_info[0][-1]

print(addr\_info) # [(2, 1, 0, '192.168.0.172', ('192.168.0.172', 8899))]

print(addr) # ('192.168.0.172', 8899)

#Using the IP address we can make a socket and connect to the server:

s = socket.socket()

s.connect(addr)

#Now that we are connected we can download and display the data:

while True:

data = s.recv(500)

print(str(data, 'utf8'), end='')

**ESP32 voorbeeld van een basic server prog = basic server esp32.py**

# ESP32 als simpele server

def do\_connect():

import network

sta\_if = network.WLAN(network.STA\_IF)

if not sta\_if.isconnected():

print('connecting to network...')

sta\_if.active(True) # ESP acts as a station and connects to an acces point

sta\_if.connect('telenet-5432836', 'hwwmbxr7sswM') # use your

while not sta\_if.isconnected():

pass

print('network config:', sta\_if.ifconfig()) #network config: ('192.168.0.110', '255.255.255.0', '192.168.0.1', '42.2.24.0')

do\_connect()

# The first thing to do is make sure we have the socket module available:

import socket

PORT=80

# AF\_INET - use Internet Protocol v4 addresses

# SOCK\_STREAM means that it is a TCP socket.

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.bind(('',PORT)) # specifies that the socket is reachable by any address the machine happens to have

s.listen(5) # max of 5 socket connections

try:

while True:

print("Server is waiting on port", PORT)

conn, addr = s.accept()

print('Server got a connection from' , addr)

while True:

request = conn.recv(2048)

if len(request) > 0:

print("message from client=",request)

print("len request =",len(request))

print("message from client decoded=",request.decode("UTF-8"))

else:

print("client disconnected")

conn.close()

break

except Exception as e:

conn.close()

print('Connection closed error', e)