

Digital electronics

J.-M Friedt

FEMTO-ST/time & frequency department

`jmfriedt@femto-st.fr`

slides at `jmfriedt.free.fr`

February 19, 2025

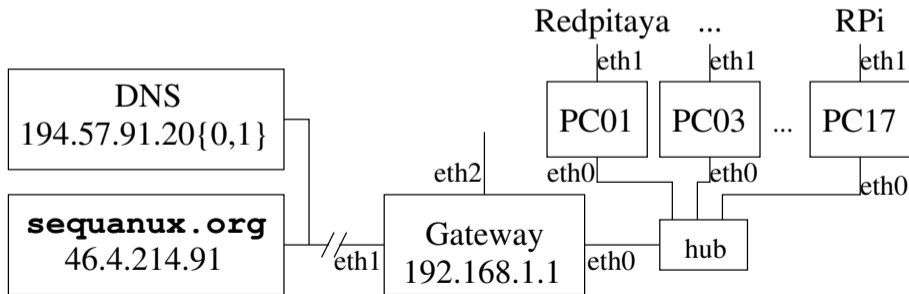
Plan

7 lessons/lab sessions (4-hour long schedules):

1. Executive environments: principles and introduction, getting started with FreeRTOS
2. FreeRTOS, RTEMS, Nuttx ... multitasking and associated methods to make sure shared data and resources are kept in known states (mutex & semaphore)
3. Using the scheduler, mutex and semaphores to solve the “philosopher problem”
4. Embedded systems with GNU/Linux – POSIX compatible operating system
Architecture of an operating system, kernel v.s userspace
Internet connectivity and networking
5. Accessing hardware resources from userspace – memory translation from physical to virtual address space (Memory Management Unit) – `/dev/mem`
6. Accessing hardware resources from a web server – internet connected instrument
7. From userspace to kernel space: character device (*char device*) for communicating between users and the kernel

Network ¹ configuration

- 7 Application layer (DNS, FTP, NTP, HTTP ...)
- 6 Presentation layer (ASCII, EBDIC ...)
- 5 Session layer (socket)
- 4 Transport layer (UDP, TCP, ICMP)
- 3 Network layer (IPv4, IPv6, ARP – arp = link between 2 & 3, route)
- 2 Data link layer – MAC (AppleTalk, 802.3 “Ethernet”: ifconfig)
- 1 Physical layer (RS232, 10BaseT, 802.11 “Wi-Fi”)



¹K. Hafner & M. Lyon, *Where Wizards Stay Up Late*, Simon & Schuster (1998)

Networking commands

- ▶ `ifconfig` for interface configuration: define IP (logical) address or possibly the MAC (hardware) address
- ▶ `route` for routing configuration
- ▶ default routing to the interface subnet
- ▶ private network address range whose packet will not propagate on the internet: **10.x.x.x**, **172.16.x.x** to **172.31.x.x**, **192.168.x.x**
- ▶ all computers have a self address: **127.0.0.1**
- ▶ “hot potato routing”: check if the IP address matches a local subnet, otherwise route to the default gateway
- ▶ UDP (Datagram) = broadcast, connectionless communication, no service quality but low latency/overhead
- ▶ TCP (Transmission Control Protocol) = connected communication making sure each packet has reached the destination in the right order (“TCP stack”) ²
- ▶ ICMP (ping): network management
- ▶ IP-domain name: DNS (set in `/etc/resolv.conf`)

²request retransmission if packet is lost, arranges received packets transmitted through different routes in the right order

Networking commands: from net-tools to ip

net-tools package commands	ip command options
<code>ifconfig -a</code>	<code>ip addr or ip a</code>
<code>ifconfig eth0</code>	<code>ip addr show eth0</code>
<code>ifconfig eth0 IP_ADDR</code>	<code>ip addr add IP_ADDR/24 dev eth0</code>
<code>ifconfig eth0:1 IP_ADDR2</code>	<code>ip addr add IP_ADDR2/24 dev eth0 label eth0:1</code>
<code>ifconfig eth0 IP_ADDR netmask 255.255.255.0</code>	<code>ip addr add IP_ADDR/24 dev eth0</code>
<code>route add default gw IP_GW</code>	<code>ip route add default via IP_GW</code>
<code>ifconfig eth0 up</code>	<code>ip link set eth0 up</code>
<code>ifconfig eth0 down</code>	<code>ip link set eth0 down</code>
	<code>ip addr del IP_ADDR dev eth0</code>
<code>arp -a</code>	<code>ip neigh</code>

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether cc:7e:e7:5f:cf:6e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.55/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet 192.168.2.1/32 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::ce7e:e7ff:fe5f:cf6e/64 scope link
        valid_lft forever preferred_lft forever
```

Netmask (subnet size): /24: first 24 bits of the mask set to 1, equivalent to 255.255.255.0

GNU/Linux network configuration

1. interfaces: ifconfig eth0 192.168.1.1

```
jmfriedt@dhcp-221:~/$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:48:54:55:09:6D
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::248:54ff:fe55:96d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1203876 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1616275 errors:0 dropped:0 overruns:1 carrier:0
          collisions:397422 txqueuelen:1000
          RX bytes:666243681 (635.3 MiB)  TX bytes:1888543246 (1.7 GiB)
          Interrupt:10 Base address:0xe800

eth1      Link encap:Ethernet  HWaddr 00:48:54:39:44:7A
          inet addr:172.16.120.21  Bcast:172.16.120.255  Mask:255.255.255.0
          inet6 addr: fe80::248:54ff:fe39:447a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5818227 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3024232 errors:0 dropped:0 overruns:1 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3049792520 (2.8 GiB)  TX bytes:1779165735 (1.6 GiB)
          Interrupt:11 Base address:0xec00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host

...
```

2. routing table with automatic association of the “appropriate” interface with each subnetwork + *default gateway*,

```
jmfriedt@dhcp-221:~/$ /sbin/route
Kernel IP routing table

```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	*	255.255.255.0	U	0	0	0	eth0
172.16.120.0	*	255.255.255.0	U	0	0	0	eth1
default	172.16.120.254	0.0.0.0	UG	0	0	0	eth1

GNU/Linux Wi-Fi configuration

```
root@satellite:/home/jmfriedt# iwlist scan | grep -B5 FreeW
```

```
Cell 07 - Address: EE:40:00:73:33:02
Channel:11
Frequency:2.462 GHz (Channel 11)  --
Quality=55/70 Signal level=-55 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 08 - Address: 2A:C1:B0:8A:83:C1
Channel:4
Frequency:2.427 GHz (Channel 4)  --
Quality=26/70 Signal level=-84 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 11 - Address: EE:40:00:73:33:03
Channel:11
Frequency:2.462 GHz (Channel 11)  --
Quality=55/70 Signal level=-55 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 22 - Address: 22:2A:C4:65:D5:D2
Channel:11
Frequency:2.462 GHz (Channel 11)  --
Quality=17/70 Signal level=-93 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Frequency:2.462 GHz (Channel 11)
Quality=16/70 Signal level=-94 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 28 - Address: 68:A3:78:00:F5:C5
Channel:4
Frequency:2.427 GHz (Channel 4)
Quality=19/70 Signal level=-91 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 29 - Address: 68:A3:78:00:F5:C6
Channel:4
Frequency:2.427 GHz (Channel 4)
Quality=16/70 Signal level=-94 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 33 - Address: 14:0C:76:B4:06:99
Channel:7
Frequency:2.442 GHz (Channel 7)
Quality=25/70 Signal level=-85 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 35 - Address: 14:0C:76:B4:06:98
Channel:7
Frequency:2.442 GHz (Channel 7)
```

GNU/Linux Wi-Fi configuration

```
root@satellite:/home/jmfriedt# iwlist scan | grep -B5 FreeW
```

```
Cell 07 - Address: EE:40:00:73:33:02
Channel:11
Frequency:2.462 GHz (Channel 11)
Quality=55/70 Signal level=-55 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 08 - Address: 2A:C1:B0:8A:83:C1
Channel:4
Frequency:2.427 GHz (Channel 4)
Quality=26/70 Signal level=-84 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 11 - Address: EE:40:00:73:33:03
Channel:11
Frequency:2.462 GHz (Channel 11)
Quality=55/70 Signal level=-55 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 22 - Address: 22:2A:C4:65:D5:D2
Channel:11
Frequency:2.462 GHz (Channel 11)
Quality=17/70 Signal level=-93 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 25 - Address: 92:84:62:B4:1C:56
Channel:11
```

```
Frequency:2.462 GHz (Channel 11)
Quality=16/70 Signal level=-94 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 28 - Address: 68:A3:78:00:F5:C5
Channel:4
Frequency:2.427 GHz (Channel 4)
Quality=19/70 Signal level=-91 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
Cell 29 - Address: 68:A3:78:00:F5:C6
Channel:4
Frequency:2.427 GHz (Channel 4)
Quality=16/70 Signal level=-94 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 33 - Address: 14:0C:76:B4:06:99
Channel:7
Frequency:2.442 GHz (Channel 7)
Quality=25/70 Signal level=-85 dBm
Encryption key:on
ESSID:"FreeWifi_secure"
```

```
Cell 35 - Address: 14:0C:76:B4:06:98
Channel:7
Frequency:2.442 GHz (Channel 7)
Quality=25/70 Signal level=-85 dBm
Encryption key:off
ESSID:"FreeWifi"
```

```
ifdown wlan0          # ifconfig wlan0 down
iwconfig wlan0 essid "FreeWifi" ap EE:40:00:73:33:02
ifup wlan0            # dhclient wlan0
```


Automating network configuration

- ▶ Network interface default settings in `/etc/network/interfaces`

- ▶ Example

```
auto lo
iface lo inet loopback
```

```
iface usb0 inet dhcp
```

```
auto eth0
iface eth0 inet static
    address 192.168.1.55
    netmask 255.255.255.0
    gateway 192.168.1.1
```

- ▶ manually activate or deactivate an interface: `ifup` and `ifdown`³ (in `/sbin`)
- ▶ prevent Network Manager from messing with `ifup` settings: edit configuration in `/etc/NetworkManager/conf.d/` and add

```
[keyfile]
unmanaged-devices=interface-name:eth*
```

```
or nmcli dev set eth0 managed no or
```

```
[ifupdown]
managed=False
```

³`ifupdown` package for Debian GNU/Linux

TCP server (C)

- ▶ a server *waits* for connections, a client *connects* to a server to initiate communication
- ▶ telnet: the “universal” interactive TCP client
- ▶ netcat: UDP/TCP command line swiss army knife

```
#include <sys/socket.h>
#include <resolv.h>
#include <unistd.h>
#include <strings.h>
#include <arpa/inet.h>

#define MY_PORT      9999
#define MAXBUF      1024

int main()
{int sockfd;
 struct sockaddr_in self;
 char buffer[MAXBUF];
 // socket type (AF = IPv4, STREAM=TCP)
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 bzero(&self, sizeof(self));
 self.sin_family = AF_INET;
 self.sin_port = htons(MY_PORT);
 self.sin_addr.s_addr = INADDR_ANY;
 bind(sockfd, (struct sockaddr*)&self, sizeof(self));
 listen(sockfd, 20); // wait for incoming connection
 while (1)
 {struct sockaddr_in client_addr;
 int mysize, clientfd;
 unsigned int addrlen=sizeof(client_addr);
 clientfd = accept(sockfd, (struct sockaddr*)&client_addr, &addrlen);
 printf("%s:%d connected\n", inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
 mysize=recv(clientfd, buffer, MAXBUF, 0);
 send(clientfd, buffer, taille, 0);
 close(clientfd);
 }
 close(sockfd); return(0); // Clean up (should never get here)
}
```

1. socket (protocol)
2. bind (port)
3. listen (blocking wait)
4. accept
5. send/recv
6. close

Beware of *endianness*

TCP server (Python)

- ▶ a server *waits* for connections
- ▶ a client *connects* to a server to initiate communication
- ▶ telnet: the “universal” interactive TCP client
- ▶ netcat: UDP/TCP command line swiss army knife

```
import socket
import string
while True:
    sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('127.0.0.1', 4242))
    print("Waiting for connection")
    sock.listen(1)
    conn, addr = sock.accept()
    with conn:
        print('connected from ',addr)
        while True:
            data=conn.recv(1)
            if data:
                data=data.decode()
                print(data)
                if 'q' in data:
                    sock.shutdown(socket.SHUT_RDWR)
                    sock.close()
                    break
```

Easiest way to test a TCP server: telnet IP port

UDP server/client⁴

A server waits for a connection

```
#include <sys/socket.h>
#include <resolv.h>
#include <arpa/inet.h>

#define BUFSIZE          1024

void alltoupper(char* s)
{while ( *s != 0 ) *s++ = toupper(*s);}

int main()
{ char buffer[BUFSIZE];
  struct sockaddr_in addr;
  int sd, addr_size, bytes_read;

  sd = socket(PF_INET, SOCK_DGRAM, 0);
  addr.sin_family = AF_INET;
  addr.sin_port = htons(9999);
  addr.sin_addr.s_addr = INADDR_ANY;
  bind(sd, (struct sockaddr*)&addr, sizeof(addr));
  do {bzero(buffer, BUFSIZE);addr_size = BUFSIZE;
    bytes_read=recvfrom(sd,buffer,BUFSIZE,0, \
      (struct sockaddr*)&addr,&addr_size);
    printf("Connect: %s:%d %s\n",inet_ntoa(addr.sin_addr),\
      ntohs(addr.sin_port), buffer);
    alltoupper(buffer);
    sendto(sd,buffer,bytes_read,0,(struct sockaddr*)&addr, \
      addr_size);
  } while ( bytes_read > 0 );
  close(sd);return 0;
}
```

A client connects to a server to start a transaction

```
#include <sys/socket.h>
#include <resolv.h>
#include <arpa/inet.h>

#define BUFSIZE          1024

int main(int argc, char **argv)
{
  char buffer[BUFSIZE];
  struct sockaddr_in addr;
  int sd, addr_size;

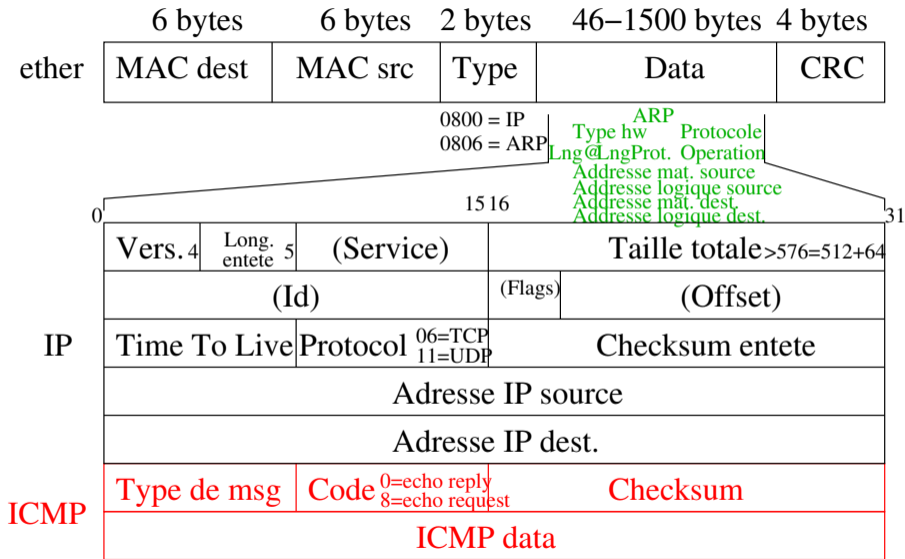
  if ( argc != 2 )
    {printf("usage: %s <msg>\n", argv[0]);exit(0);}
  sd = socket(PF_INET, SOCK_DGRAM, 0);
  addr.sin_family = AF_INET;
  addr.sin_port = htons(9999);
  inet_aton("127.0.0.1", &addr.sin_addr);
  sendto(sd, argv[1], strlen(argv[1])+1, 0, \
    (struct sockaddr*)&addr, sizeof(addr));
  bzero(buffer, BUFSIZE);
  addr_size = sizeof(addr);
  recvfrom(sd,buffer,BUFSIZE,0,(struct sockaddr*)&addr,\
    &addr_size);
  printf("Reply: %s:%d %s\n",inet_ntoa(addr.sin_addr), \
    ntohs(addr.sin_port), buffer);
  close(sd);
  return 0;
}
```

Cross-compile for the targeted embedded board – portability by embedding OS

or echo "toto" | nc -u IP 9999

⁴<http://www.cs.utah.edu/~swalton/listings/sockets/programs/>

Raw IP and ICMP



Question: what is the ICMP identifier in the IP packet header? `sudo tcpdump -i lo -x -e during ping lo`

Raw IP and ICMP

```
$ arp -a
? (192.168.0.1) at 00:13:d3:8d:d3:97 [ether] on eth0
$ arp -d 192.168.0.1
$ ping 192.168.0.11

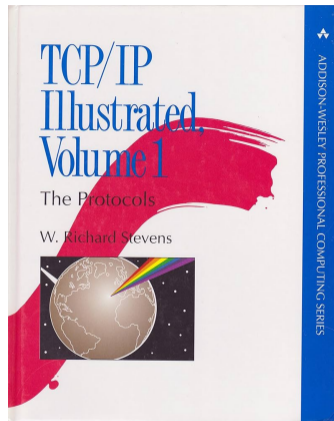
# tcpdump -i eth0 -XX # XX = show ethernet header
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:06:33.462038 ARP, Request who-has 192.168.0.1 tell 192.168.0.11, length 28
    0x0000:  ffff ffff ffff cc7e e75f cf6e 0806 0001  .....~_.n....
    0x0010:  0800 0604 0001 cc7e e75f cf6e c0a8 000b  .....~_.n....
    0x0020:  0000 0000 0000 c0a8 0001  .....
16:06:33.462590 ARP, Reply 192.168.0.1 is-at 00:13:d3:8d:d3:97 (oui Unknown), length 46
    0x0000:  cc7e e75f cf6e 0013 d38d d397 0806 0001  .~_.n.....
    0x0010:  0800 0604 0002 0013 d38d d397 c0a8 0001  .....
    0x0020:  cc7e e75f cf6e c0a8 000b 0000 0000 0000  .~_.n.....
    0x0030:  0000 0000 0000 0000 0000 0000  .....
16:06:33.462607 IP 192.168.0.11 > 192.168.1.1: ICMP echo request, id 1906, seq 1, length 64
    0x0000:  0013 d38d d397 cc7e e75f cf6e 0800 4500  .....~_.n..E.
    0x0010:  0054 1191 4000 4001 a6bb c0a8 000b c0a8  .T..@.@.....
    0x0020:  0101 0800 ff64 0772 0001 e9c2 4c55 c90c  ....d.r....LU..
    0x0030:  0700 0809 0a0b 0c0d 0e0f 1011 1213 1415  .....
    0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  .....!"#$%
    0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
    0x0060:  3637  .....67
```

Raw IP & ICMP

A good understanding of network protocols is needed for applications related to

- ▶ security: firewall or eddicated router (iptables), packet filtering, quality of service and minimizing packet handling duration,
- ▶ remote instrument control. Even without consider data security, avoid DoS attacks.
- ▶ all OSI layers are often not needed for embedded applications: understand the whole communication chain to only keep the mandatory parts (e.g. raw-IP).

In our examples, the argument to `socket` defines the transport protocol (`SOCK_DGRAM` (UDP), `SOCK_STREAM` (TCP)).

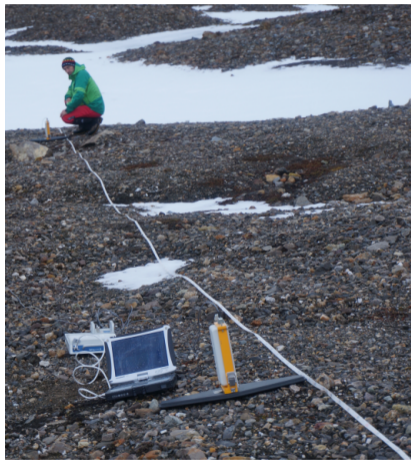
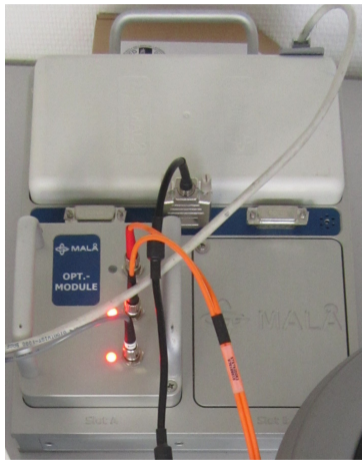


`SOCK_RAW` \Rightarrow 20 byte header including source and destination IPs followed by payload (routing)

```
45 00 00 34 19 a9 00 00 3c ff 66 20 7f 00 00 01      <- source = 127.0.0.1
7f 00 00 01 30 30 30 30 30 30 30 30 30 30 30 30     <- dest = 127.0.0.1
```

Raw Ethernet

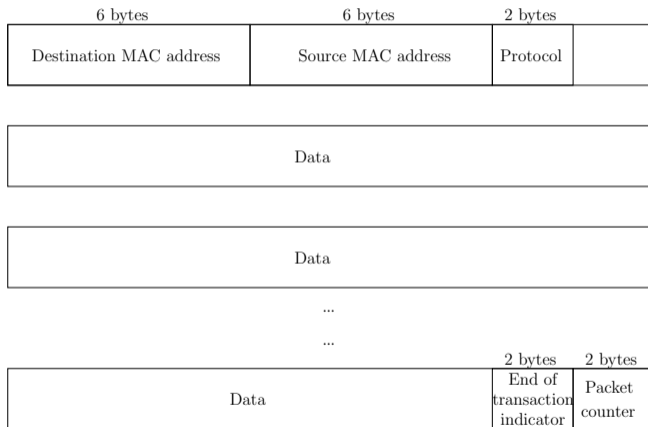
For light, low power embedded systems: raw-Ethernet (Malå ProEx ⁵), no routing (point to point link between PC and RADAR control unit)



⁵<http://sourceforge.net/projects/proexgprcontrol/>,

Raw Ethernet

For light, low power embedded systems: raw-Ethernet (Malå ProEx ⁵), no routing (point to point link between PC and RADAR control unit)

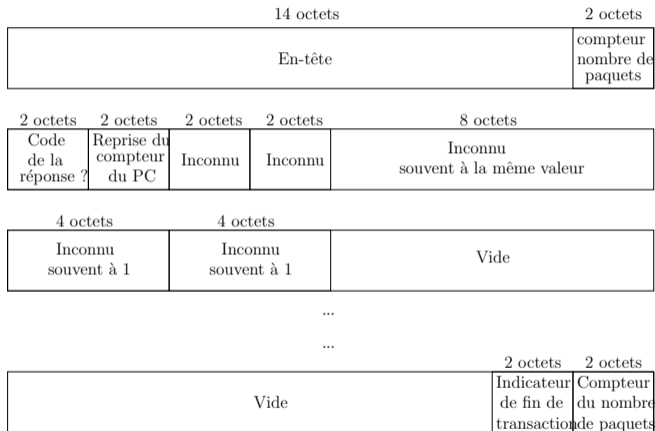


⁵<http://sourceforge.net/projects/proexgprcontrol/>,

A. Hugeot, J.-M Friedt, *A low cost approach to acoustic filters acting as GPR cooperative targets for passive sensing*, IWAGPR 2015

Raw Ethernet

For light, low power embedded systems: raw-Ethernet (Malå ProEx ⁵), no routing (point to point link between PC and RADAR control unit)



⁵<http://sourceforge.net/projects/proexgprcontrol/>,

A. Hugeot, J.-M Friedt, *A low cost approach to acoustic filters acting as GPR cooperative targets for passive sensing*, IWAGPR 2015

Listening to packets ... tcpdump

Example: tcpdump port 80 -i lo -X

The image shows a terminal window titled 'jmfriedt@rugged: ~' with the following content:

```
root@rugged:/home/jmfriedt# tcpdump -i lo -X
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EMULOMB (Ethernet), capture size 262144 bytes
11:30:54.890446 IP6 localhost.38026 > localhost.http: Flags [S], seq 3573235918, win 43690, options [mss 65476,sackOK,TS val 1688894 ecr 0,nop,uscale 7], length 0
0x0000: 5000 0000 0028 0540 0000 0000 0000 0000  ....(.@.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 348a 0050 34fb 40ce  .....P..@.
0x0030: 0000 0000 a002 aaaa 0030 0000 0204 ffc4  .....0.....
0x0040: 0402 080a 0019 c53e 0000 0000 0103 0307  .....>.....
11:30:54.890470 IP6 localhost.http > localhost.38026: Flags [S.], seq 1199692453, ack 3573235919, win 43690, options [mss 65476,sackOK,TS val 1688894 ecr 1688894,nop,uscale 7], length 0
0x0000: 5000 0000 0028 0540 0000 0000 0000 0000  ....(.@.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 0050 948a 4781 daa5  .....P..G...
0x0030: d4fb 40cf a012 aaaa 0030 0000 0204 ffc4  ..@.....0.....
0x0040: 0402 080a 0019 c53e 0019 c53e 0103 0307  .....>.....
11:30:54.890494 IP6 localhost.38026 > localhost.http: Flags [.], ack 1, win 342, options [nop,nop,TS val 1688894 ecr 1688894], length 0
0x0000: 5000 0000 0020 0540 0000 0000 0000 0000  .....@.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 348a 0050 34fb 40cf  .....P..@.
0x0030: 4781 daa5 8010 0156 0028 0000 0101 080a  G.....V..(.....
0x0040: 0019 c53e 0019 c53e  .....>.....
11:30:54.810584 IP6 localhost.38026 > localhost.http: Flags [P.], seq 1:414, ack 1, win 342, options [nop,nop,TS val 1688899 ecr 1688894], length 413
0x0000: 5000 0000 015d 0540 0000 0000 0000 0000  ....e.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 348a 0050 34fb 40cf  .....P..@.
0x0030: 4781 daa6 8018 0156 01c5 0000 0101 080a  G.....V.....
0x0040: 0019 c543 0019 c53e 4745 5420 2f20 4854  ...C...>GET../HT
0x0050: 5490 2f31 2e31 0d0a 486f 7374 3a20 6c6f  IP/1.1.1..Host:lo
0x0060: 6361 5c68 6f73 740d 0a55 7365 722d 4167  calhost.,User-Ag
0x0070: 656e 743a 204d 6f7a 696c 6c61 2f35 2e30  ent:Mozilla/5.0
0x0080: 2028 5831 313b 204c 696e 7578 2069 3638  ,(X11:Linux,i68
0x0090: 363b 2072 763a 3331 2e30 2920 4765 636b  6;rv:31.0)Geck
0x00a0: 6f2f 3230 3130 3031 3031 2046 6972 6566  o/20100101,Firef
0x00b0: 6f78 2f33 312e 3020 4963 6577 6561 7365  ox/31.0,Iceweas
0x00c0: 6c2f 3331 2e35 2e33 0d0a 4163 6365 7074  /31.5.3..Accept
0x00d0: 3a20 7465 7874 2f68 746d 6c2c 6170 706c  :.text/html,appl
0x00e0: 6963 6174 696f 6e2f 7868 746d 6c2b 786d  ication/xhtml+xml
0x00f0: 6c2c 6170 706c 6963 6174 696f 6e2f 786d  l,application/xml
0x0100: 6c3b 713d 302e 392c 2a2f 2a3b 713d 302e  l;q=0.9,*/*;q=0.
0x0110: 380d 0a41 6363 6570 742d 4c61 6e67 7561  8..Accept-Langua
0x0120: 6765 3a20 656e 2d95 532c 696e 3b71 3d30  ge;en-US,en;q=0
0x0130: 2e35 0d0a 4163 6365 7074 2d45 6e63 6f64  .S..Accept-Encod
0x0140: 696e 673a 2067 7a69 702c 2064 6566 6c61  ing:gzip,defla
0x0150: 7465 0d0a 444e 543a 2031 0d0a 436f 6e6e  te..I;I:1..Conn
```

Below the terminal is a browser window showing the output of a curl command:

```
jmfriedt@rugged:~$ curl http://localhost:38026/
Hello World
```

The browser address bar shows 'localhost' and the page content is 'Hello World Ceci est un test.'

Listening to packets ... tcpdump

Example: tcpdump port 80 -i lo -X

```
0x0120: 6765 3a20 656e 2d55 532c 656e 3b71 3d30  ge:,en-US,en;q=0
0x0130: 2e35 0d0a 4163 6365 7074 2d45 6e63 6f64  .S.,Accept-Encod
0x0140: 696e 673a 2067 7a69 702c 2064 6566 6c61  ing:,gzip,.defla
0x0150: 7465 0d0a 444e 543a 2031 0d0a 436f 6e6e  te.,INT:1.,Conn
0x0160: 6563 7469 6f6e 3a20 6b65 6570 2d61 6c69  ection:keep-ali
0x0170: 7665 0d0a 0d0a
ve....
11:29:13.002767 IP6 localhost.http > localhost.38024; Flags [.], ack 303, win 350, options [nop,nop,TS val 1663422 ecr 1663422], length 0
0x0000: 5000 0000 0020 0640 0000 0000 0000 0000  .....@.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 0050 9488 84cc 0c3d  .....P.....=
0x0030: 4271 c33d 8010 015e 0028 0000 0101 080a  Bq=...^(.....
0x0040: 0019 61be 0019 61be
..a..a.
11:29:13.003061 IP6 localhost.http > localhost.38024; Flags [P.], seq 1:315, ack 303, win 350, options [nop,nop,TS val 1663422 ecr 1663422], length 314
0x0000: 5000 0000 015a 0640 0000 0000 0000 0000  ....z@.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 0050 9488 84cc 0c3d  .....P.....=
0x0030: 4271 c33d 8018 015e 0162 0000 0101 080a  Bq=...^b.....
0x0040: 0019 61be 0019 61be 4854 5450 2f31 2e31  ..a..a,HTTP/1.1
0x0050: 2032 3030 204f 4b0d 0a44 6174 653a 2057  .200_OK.,Date:W
0x0060: 6964 2c20 3036 204d 6179 2032 3031 3520  ed.,06_May,2015.
0x0070: 3039 3a32 393a 3133 2047 4d54 0d0a 5365  09:29:13,GMT..Se
0x0080: 7276 6572 3a20 4170 6163 6865 2f32 2e34  rver:Apache/2.4
0x0090: 2e31 3020 2844 6962 6361 6e29 0d0a 4c61  .10,(Debian).La
0x00a0: 7374 2d4d 6f64 6966 6365 643a 2057 6964  st-Modified:Wed
0x00b0: 2c20 3036 204d 6179 2032 3031 3520 3039  .,06_May,2015,09
0x00c0: 3a32 303a 3534 2047 4d54 0d0a 4954 6167  :20:54 GMT.,ETag
0x00d0: 3a20 2231 662d 3631 3636 3634 6665 3131  :,"1f-519664fe11
0x00e0: 6463 3322 0d0a 4163 6365 7074 2d52 616e  d3".Accept-Ran
0x00f0: 5765 733a 2062 7974 6573 0d0a 436f 6e74  ges:,bytes.,Cont
0x0100: 656e 742d 4c65 6e67 7468 3a20 3331 0d0a  ent-Length:31..
0x0110: 4b65 6570 2d41 6c69 7655 3a20 7469 6d65  Keep-Alive:time
0x0120: 6f75 743d 352c 206d 6178 3d31 3030 0d0a  out=5.,max=100..
0x0130: 436f 6e6e 6563 7469 6f6e 3a20 4b65 6570  Connection:Keep
0x0140: 2d41 6c69 7655 0d0a 436f 6e74 656e 742d  -Alive.,Content-
0x0150: 5479 7065 3a20 7465 7874 2f68 746d 6c0d  Type:text/html.
0x0160: 0a0d 0a48 656c 6c6f 2057 6f72 6c64 0a0a  ..Hello,World..
0x0170: 4365 6369 2065 7374 2075 6e20 7465 7374  Ceci.est,un.test
0x0180: 2e0a
..
11:29:13.003070 IP6 localhost.38024 > localhost.http; Flags [.], ack 315, win 350, options [nop,nop,TS
0x0000: 5000 0000 0020 0640 0000 0000 0000 0000  .....@.....
0x0010: 0000 0000 0000 0001 0000 0000 0000 0000  .....
0x0020: 0000 0000 0000 0001 3488 0050 4271 c33d  .....PBq=
0x0030: 84cc 0d77 8010 015e 0028 0000 0101 080a  ..w...^(.....
0x0040: 0019 61be 0019 61be
..a..a.
11:29:17.906610 IP6 localhost.http > localhost.38024; Flags [F.], seq 315, ack 303, win 350, options
0x0000: 5000 0000 0020 0640 0000 0000 0000 0000  .....@.....
```

```
jmfriedt@rugged: ~
jmfriedt@rugged:~$ cat /var/www/html/index.html
Hello World

Ceci est un test.
jmfriedt@rugged:~$
```

BBC News - Home Portail numérique FE... Accédez à vos e-ma... TorrentFlux

Hello World Ceci est un test.

Listening to packets ... wireshark

Former-ethereal, wireshark for a graphical user interface to analyzer packet contents

The screenshot displays the Wireshark 1.12.1 interface. The main window shows a list of captured packets on the loopback interface 'lo'. The selected packet (No. 6) is an HTTP 200 OK response. Below the packet list, the raw bytes and their hexadecimal representation are visible, showing the text 'Hello World' and 'Ceci est un test.'.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	:::1	:::1	TCP	94	38022->80 [SYN] Seq=0 win=43690 Len=0 MSS=65476 SA
2	0.000019000	:::1	:::1	TCP	94	80->38022 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 M
3	0.000034000	:::1	:::1	TCP	86	38022->80 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=
4	0.010705000	:::1	:::1	HTTP	388	GET / HTTP/1.1
5	0.010739000	:::1	:::1	TCP	86	80->38022 [ACK] Seq=1 Ack=303 Win=44800 Len=0 TSva
6	0.011205000	:::1	:::1	HTTP	400	HTTP/1.1 200 OK (text/html)
7	0.011214000	:::1	:::1	TCP	86	38022->80 [ACK] Seq=303 Ack=315 Win=44800 Len=0 TS
8	1.083335000	:::1	:::1	HTTP	369	GET /favicon.ico HTTP/1.1
9	1.083533000	:::1	:::1	HTTP	586	HTTP/1.1 404 Not Found (text/html)
10	1.083545000	:::1	:::1	TCP	86	38022->80 [ACK] Seq=586 Ack=815 Win=45952 Len=0 TS
11	1.087472000	:::1	:::1	HTTP	399	GET /favicon.ico HTTP/1.1
12	1.087650000	:::1	:::1	HTTP	586	HTTP/1.1 404 Not Found (text/html)

Raw packet data for packet 6:

```
0140  6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41  nnection : Keep-A
0150  6c 69 76 65 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79  live..Content-Ty
0160  70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a 0d  pe: text/html...
0170  0a 48 65 6c 6c 6f 20 57 6f 72 6c 64 0a 0a 43 65  .Hello World..Ce
0180  63 69 20 65 73 74 20 75 6e 20 74 65 73 74 2e 0a  ci est un test..
```

Listening to packets ... wireshark

Former-ethereal, wireshark for a graphical user interface to analyzer packet contents

The screenshot shows the Wireshark 1.12.1 interface. The main window displays a list of captured packets on the interface 'Loopback: lo'. Packet 6 is selected, showing details for Ethernet II, Internet Protocol Version 6, and Hypertext Transfer Protocol. The packet details show a 400-byte GET request for '/favicon.ico' from 192.168.1.1 to 192.168.1.1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	:::1	:::1	TCP	94	38022->80 [SYN] Seq=0 Win=43690 Len=0 MSS=65476 S...
2	0.000019000	:::1	:::1	TCP	94	80->38022 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 M...
3	0.000034000	:::1	:::1	TCP	86	38022->80 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=...
4	0.010705000	:::1	:::1	HTTP	388	GET / HTTP/1.1
5	0.010739000	:::1	:::1	TCP	86	80->38022 [ACK] Seq=1 Ack=303 Win=44800 Len=0 TSva...
6	0.011205000	:::1	:::1	HTTP	400	HTTP/1.1 200 OK (text/html)
7	0.011214000	:::1	:::1	TCP	86	38022->80 [ACK] Seq=303 Ack=315 Win=44800 Len=0 TS...
8	1.083335000	:::1	:::1	HTTP	369	GET /favicon.ico HTTP/1.1
9	1.083533000	:::1	:::1	HTTP	586	HTTP/1.1 404 Not Found (text/html)
10	1.083545000	:::1	:::1	TCP	86	38022->80 [ACK] Seq=586 Ack=815 Win=45952 Len=0 TS...
11	1.087472000	:::1	:::1	HTTP	399	GET /favicon.ico HTTP/1.1
12	1.087650000	:::1	:::1	HTTP	586	HTTP/1.1 404 Not Found (text/html)

Frame 6: 400 bytes on wire (3200 bits), 400 bytes captured (3200 bits) on interface 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 6, Src: ::1 (:::1), Dst: ::1 (:::1)
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 38022 (38022), Seq: 1, Ack: 303, Len: 314
Hypertext Transfer Protocol

```
0050 d1 c1 00 17 d1 c0 48 54 54 50 2f 31 2e 31 20 32 .....HT TP/1.1 2
0060 30 30 20 4f 4b 0d 0a 44 61 74 65 3a 20 57 65 64 00 OK..D ate: Wed
0070 2c 20 30 36 20 4d 61 79 20 32 30 31 35 20 30 39 09
0080 3a 32 32 3a 32 33 20 47 4d 54 0d 0a 53 65 72 76 , 06 May 2015 09
0090 65 72 3a 20 41 70 61 63 68 65 2f 32 2e 34 2e 31 :22:23 G MT..Serv
00a0 30 20 28 44 65 62 69 61 6e 29 0d 0a 4c 61 73 74 er: Apac he/2.4.1
00b0 2d 4d 6f 64 69 66 69 65 64 3a 20 57 65 64 2c 20 0 (Debian)..Last
00c0 30 36 20 4d 61 79 20 32 30 31 35 20 30 39 3a 32 -Modify d: Wed,
00d0 30 3a 35 34 20 47 4d 54 0d 0a 45 54 61 67 3a 20 06 May 2 015 09:2
00e0 22 31 66 2d 35 31 35 36 36 34 66 65 31 31 64 63 "1f-5156 64felli
00f0 33 22 0d 0a 41 63 63 65 70 74 2d 52 61 6e 67 65 3*..Acce pt-Range
0100 73 2a 2c 62 79 74 65 73 0d 0a 43 6f 6e 74 65 66 s: bytes ..Conten
0110 74 2d 4c 65 6e 67 74 68 3a 20 33 31 0d 0a 4b 65 t-Length : 31..Ke
0120 65 70 2d 41 6c 69 76 65 3a 20 74 69 6d 65 6f 75 ep-Alive : timeou
0130 74 7d 2e 3a 20 6d 6f 70 3d 21 79 20 0d 0a 47 64 ArF_max =100..Ca
```

TCP/IP tools

- traceroute

```
jmfriedt@vm1:~$ traceroute www.whitehouse.gov
traceroute to www.whitehouse.gov (23.214.186.191), 30 hops max, 60 byte packets
 1  10.10.0.254 (10.10.0.254)  0.377 ms  0.370 ms  0.362 ms
 2  vss-5b-6k.fr.eu (46.105.123.252)  0.954 ms  0.952 ms  0.947 ms
 3  rbx-g1-a9.fr.eu (178.33.100.29)  2.509 ms  2.509 ms  2.505 ms
 4  * * *
 5  ldn-5-6k.uk.eu (213.251.128.18)  48.180 ms * *
 6  * * *
 7  ae10.mpr2.lhr2.uk.zip.zayo.com (64.125.31.194)  6.492 ms  8.722 ms  7.263 ms
 8  ae5.mpr1.lhr15.uk.zip.zayo.com (64.125.21.10)  4.519 ms  4.515 ms  4.496 ms
 9  94.31.61.250.IPYX-074083-001-ZY0.above.net (94.31.61.250)  4.504 ms  4.500 ms  4.495 ms
10  a23-214-186-191.deploy.static.akamaitechnologies.com (23.214.186.191)  4.493 ms  4.488 ms  4.793 ms
```

- nslookup

```
jmfriedt@rugged:~$ nslookup www.femto-st.fr
```

```
Server:          130.67.15.198
Address:         130.67.15.198#53
```

```
Non-authoritative answer:
```

```
Name:   www.femto-st.fr
Address: 195.83.19.10
```

HTTP tools

- The swiss army knife: nc (netcat)

```
echo -e "GET http://jmfriedt.free.fr HTTP/1.0\n\n" | nc jmfriedt.free.fr 80
```

- wget – GET ethod (answer: 0:47 1:57 2:25)

```
dest=Vesoul
```

```
wget -q -O- "http://reiseauskunft.bahn.de/bin/query.exe/fn?revia=yes&\
```

```
existOptimizePrice=1&country=FRA&\
```

```
dbkanal_007=L01_S01_D001_KIN0001_qf-bahn_LZ003&\
```

```
ignoreTypeCheck=yes&S=Besancon+Franche+Comte+TGV&REQ0JourneyStopsS0A=7&Z=${dest}&\
```

```
REQ0JourneyStopsZ0A=7&trip-type=single&date=Me%2C+07.01.15&time=08%3A37&\
```

```
timesel=depart&returnTimesel=depart&optimize=0&infant-number=0&tariffTravellerType.1=E&\
```

```
tariffTravellerReductionClass.1=0&tariffTravellerAge.1=&qf-trav-bday-1=&tariffClass=2&\
```

```
start=1&qf.bahn.button.suchen=" | grep -A1 uratio | grep ^[0-9] | tr '\n' ' ' ,
```

- curl – POST method ⁶

```
month=12
```

```
year=2014
```

```
curl -s --cookie-jar test --data \
```

```
"anzahlprofile=1&einheit=meter&vonkurz=LFPG&nachkurz=LYR+&landetime=&zwpunkte=&\
```

```
adresse=&profil=profilangeben&day=15&month=$month&year=$annee&flightnumber=XX001&\
```

```
von=PARIS%2C+FRANCE+++++++&uptime=00%3A30&obentime=04%3A00&flughoehe0=10000&\
```

```
downtime=00%3A30&nach=LONGYEARBYEN%2C+NORWAY+++++++&send=Send+request" \
```

```
http://www.helmholtz-muenchen.de/epcard/eng_flugoutput.php | grep Sv
```

⁶J.-M Friedt, *Cartographier le bout du monde*, GNU/Linux Magazine France 185 (Sept. 2015)

Probing the network

▶ scan network: netdiscover

Currently scanning: 192.168.69.0/16 | Screen View: Unique Hosts

5 Captured ARP Req/Rep packets, from 4 hosts. Total size: 300

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.1.1	e4:54:e8:ad:cb:0a	1	60	Dell Inc.
192.168.1.3	e4:54:e8:ad:ca:ae	2	120	Dell Inc.
192.168.2.1	e4:54:e8:ad:ca:ae	1	60	Dell Inc.
192.168.2.1	e4:54:e8:ad:cb:0a	1	60	Dell Inc.

...

▶ scan network: sudo nmap -sn 192.168.1.1/24

Starting Nmap 7.93 (<https://nmap.org>) at 2025-02-18 09:17 CET

Nmap scan report for 192.168.1.1

Host is up (0.0016s latency).

Nmap scan report for 192.168.1.3

Host is up (0.0015s latency).

Nmap scan report for 192.168.1.42

Host is up (0.00023s latency).

Nmap done: 256 IP addresses (3 hosts up) scanned in 2.51 seconds

▶ list open ports: sudo nmap 192.168.1.1

▶ OS identification: sudo nmap -O 192.168.1.1

Internet documentation

Request for Comments (RFC) at www.ietf.org/rfc

Most common protocols

- ▶ HTTP = RFC 2068 (Jan. 1997) and 2616 (Jun. 1999),
- ▶ SMTP = RFC 772 ⁷ (Sep. 1980) and 821 (Aug. 1982)
- ▶ FTP = RFC114 et successeurs, RFC542 (Aug. 1973), RFC959 (Oct. 1985)

Available services, used protocol and connection port are standardized and documented:
`/etc/services`

Too low computational power makes vulnerable to **DoS attacks**.

⁷tools.ietf.org/html/rfc772

Practical demonstration

1. configure the network so the embedded board can communicate through a TCP/IP (IP = Internet Protocol – `ifconfig`, TCP = connected protocol = `rlogin/ssh`) link with the host computer
2. launch a web server on the embedded board (`lighttpd`) and display the output of an `index.html` static web page
3. check that you can control the LED state through the standard `/sys/class/leds` interface
4. create a dynamic web interface (CGI: Common Gateway Interface) in which a script parses the argument of the URL and acts on the LED accordingly. Be careful with permissions (who owns the web server? who owns the LED?)