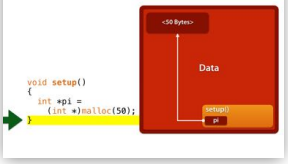
Mobile Networking

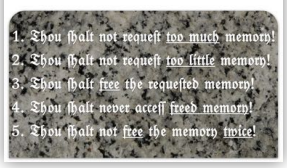
Programming for Engineers
Winter 2015

Andreas Zeller, Saarland University

Dynamic Memory



Dynamic Memory



Structs

- In C we can combine data into a struct (also called record)
- Example: Complex Numbers

Type definition

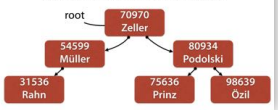
```
struct Complex {
  double real;
  double imag;
};
```

Variable initialisation

```
struct Complex c = {
  3.0,  // real
  4.0   // imag
};
```

Search Trees

- Every node has (up to two) children:
  in the left subtree are all smaller values,
  in the right subtree are all larger values



Today's Topics

- Mobile Networking
- HTTP
- HTML
- Webserver!

Murray Leinster

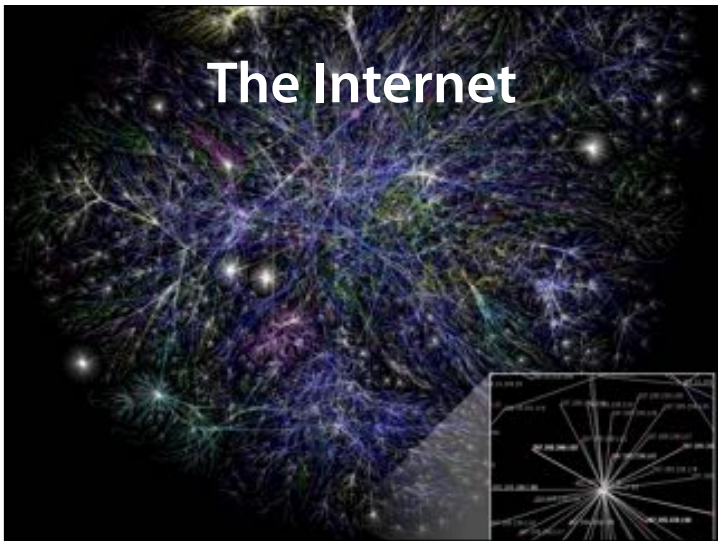
"A Logic Named Joe" (1946)

Murray Leinster, 1896–1975

The computer ... manages the spreading of ninety-four percent of all TV programs, conveys all information about weather, air traffic, special deals… and records every business conversation, every contract… Computers have changed the world. Computers are the civilisation. If we turn them off, we will fall back to a kind of civilisation, of which we have forgotten how it even works.

The aim of a computer network is to have computers communicate with each other

---



Partial map of the Internet based on the January 15, 2005 data found on opte.org. Each line is drawn between two nodes, representing two IP addresses. The length of the lines are indicative of the delay between those two nodes. This graph represents less than 30% of the Class C networks reachable by the data collection program in early 2005.

Wireless Internet

- WLAN = Wireless Local Area Network
- Allows "local" computers to communicate

Wireless Modem

The Arduino ESP8266 shield allows the Arduino to connect to networks, and also to set up its own network

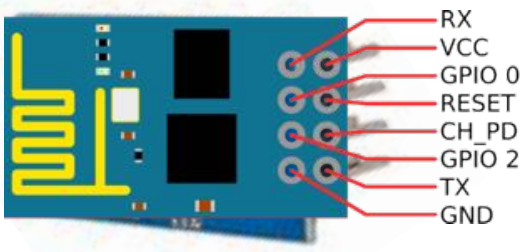
Controlling a Modem

Modems receive

- *data* to be sent
- *commands* to control them

The modem is controlled by so-called AT commands

Connecting the Modem

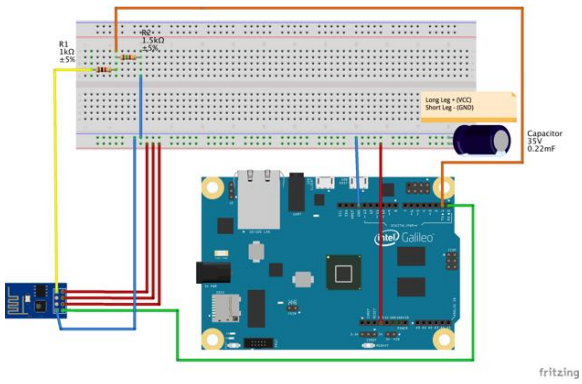


⚠ **Never** connect the wireless module to the **5V** power output of the board. Otherwise the wireless chip will be **damaged permanently.**

View from above – connectors are at the bottom

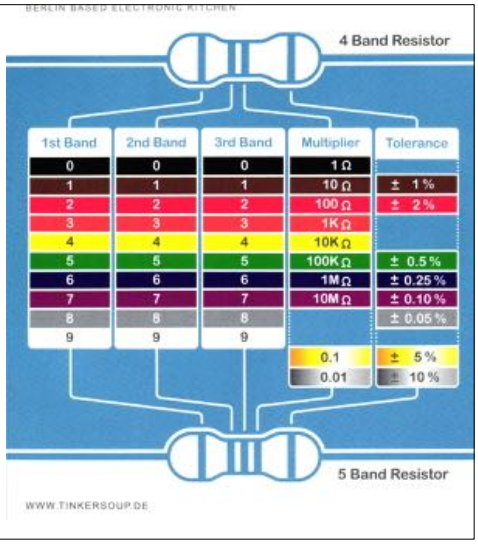
These are connected to the serial output (TX) of the Arduino
Details in assignment sheet

Connecting the Modem



Details in assignment sheet

Resistors



Controlling a Modem

- Modems are controlled by so-called *AT commands* (AT = "Attention")
- AT commands allow to
  - connect to networks,
  - create networks
  - set communication speeds…

The modem is controlled by so-called AT commands

ESP8266 AT Instruction Set

Version 0.30

Espressif Systems IOT Team

Source: https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf
(Will be linked from Webpage)

Table of Contents

---

2. AT+RST – Restart module 9
3. AT+GMR – View version info 9
4. AT+GSLP – Enter deep-sleep mode 10
5. ATE – AT commands echo 10
6. AT+RESTORE – Factory reset 10
7. AT+UART – UART configuration 11
8. AT+UART_CUR – current UART configuration 12
9. AT+UART_DEF – default UART configuration 13
10. AT+SLEEP – sleep mode 14
11. AT+RFPOWER – set maximum value of RF TX Power 14
12. AT+RFVDD – set RF TX Power according to VDD33 15

Creating a Network

- We want to create a *Wifi access point*
- Allows other devices to connect

```
AT+CWMODE=2
```

Wifi mode

Set up an SSID

- Sets the name by which other devices can identify the network
- Also sets password (8–64 characters), authentication mode, and channel (1–12)

```
AT+CWSAP="PFE","12345678",1,4
```

Auth mode

SSID Password Channel

Use your own SSID and a safe password!

Sending Commands

- The modem is connected to the serial port
- It replies with "OK" if everything is fine

```
int issueCommand(char *command) {
  // handle request
  Serial.println(command);
  delay(10);

  // The modem replies with "OK"
  // if everything worked well
  if (!Serial.find("OK"))
    return 0; // Error

  return 1;
}
```

Need to signal this to the user (e.g. through display or blinking LED)

Wifi Setup

```
void setup() {
  // Initialize serial connection
  Serial.begin(115200);
  Serial.setTimeout(5000);

  // Set mode to wifi access point
  if (!issueCommand("AT+CWMODE=2"))
       return;

  // Enable wifi access point with SSID "PFE"
  if (!issueCommand("AT+CWSAP=\"PFE\",\"12345678\",1,4"))
      return;
}
```

Demo

Ports

- We want to run a *service* on the device
- Every computer provides p*orts* for network IP connections
- Ports are numbered from 1 to 65535
- Every service has its own port

Ports

```
#   http://www.iana.org/assignments/port-numbers
#
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# $FreeBSD: src/etc/services,v 1.89 2002/12/17 23:59:10 eric Exp $
#   From: @(#)services   5.8 (Berkeley) 5/9/91
#
# WELL KNOWN PORT NUMBERS
#
rtmp              1/ddp    #Routing Table Maintenance Protocol
tcpmux            1/udp     # TCP Port Service Multiplexer
tcpmux            1/tcp     # TCP Port Service Multiplexer
#                           Mark Lottor <MKL@nisc.sri.com>
nbp               2/ddp    #Name Binding Protocol
compressnet       2/udp     # Management Utility
compressnet       2/tcp     # Management Utility
compressnet       3/udp     # Compression Process
compressnet       3/tcp     # Compression Process
#                           Bernie Volz <VOLZ@PROCESS.COM>
echo              4/ddp    #AppleTalk Echo Protocol
#                 4/tcp    Unassigned
#                 4/udp    Unassigned
rje               5/udp     # Remote Job Entry
rje               5/tcp     # Remote Job Entry
#                           Jon Postel <postel@isi.edu>
zip               6/ddp    #Zone Information Protocol
#                 6/tcp    Unassigned
#                 6/udp    Unassigned
echo              7/udp     # Echo
```

```
mit-ml-dev       85/udp     # MIT ML Device
mit-ml-dev       85/tcp     # MIT ML Device

... 12000 lines more ...



#                           David Reed <--none--->
#               47809-47999 Unassigned
nimcontroller   48000/udp    # Nimbus Controller
nimcontroller   48000/tcp    # Nimbus Controller
nimspooler      48001/udp    # Nimbus Spooler
nimspooler      48001/tcp    # Nimbus Spooler
nimhub          48002/udp    # Nimbus Hub
nimhub          48002/tcp    # Nimbus Hub
nimgtw          48003/udp    # Nimbus Gateway
nimgtw          48003/tcp    # Nimbus Gateway
#                           Carstein Seeberg <case@nimsoft.no>
#               48004-48555 Unassigned
isnetserv       48128/tcp   # Image Systems Network Services
isnetserv       48128/udp   # Image Systems Network Services
blp5            48129/tcp   # Bloomberg locator
blp5            48129/udp   # Bloomberg locator
#               48130-48555 Unassigned
com-bardac-dw   48556/udp    # com-bardac-dw
com-bardac-dw   48556/tcp    # com-bardac-dw
#                           Nicholas J Howes <nick@ghostwood.org>
#               48557-49150 Unassigned
#               49151       IANA Reserved
```

```
discard          9/tcp      # Discard
#                           Jon Postel <postel@isi.edu>
#               10/tcp      Unassigned
#               10/udp      Unassigned
systat          11/udp      # Active Users
systat          11/tcp      # Active Users
#                           Jon Postel <postel@isi.edu>
#               12/tcp      Unassigned
#               12/udp      Unassigned
daytime         13/udp      # Daytime (RFC 867)
daytime         13/tcp      # Daytime (RFC 867)
#                           Jon Postel <postel@isi.edu>
#               14/tcp      Unassigned
#               14/udp      Unassigned
#               15/tcp      Unassigned [was netstat]
#               15/udp      Unassigned
#               16/tcp      Unassigned
#               16/udp      Unassigned
qotd            17/udp      # Quote of the Day
qotd            17/tcp      # Quote of the Day
#                           Jon Postel <postel@isi.edu>
msp             18/udp      # Message Send Protocol
msp             18/tcp      # Message Send Protocol
#                           Rina Nethaniel <---none--->
chargen         19/udp      # Character Generator
chargen         19/tcp      # Character Generator
ftp-data        20/udp      # File Transfer [Default Data]
ftp-data        20/tcp      # File Transfer [Default Data]
ftp             21/udp      # File Transfer [Control]
ftp             21/tcp      # File Transfer [Control]
#                           Jon Postel <postel@isi.edu>
```

FTP = File Transfer Protocol → zum Übertragen von Dateien

```
systat          11/tcp      # Active Users
#                           Jon Postel <postel@isi.edu>
#               12/tcp      Unassigned
#               12/udp      Unassigned
daytime         13/udp      # Daytime (RFC 867)
daytime         13/tcp      # Daytime (RFC 867)
#                           Jon Postel <postel@isi.edu>
#               14/tcp      Unassigned
#               14/udp      Unassigned
#               15/tcp      Unassigned [was netstat]
#               15/udp      Unassigned
#               16/tcp      Unassigned
#               16/udp      Unassigned
qotd            17/udp      # Quote of the Day
qotd            17/tcp      # Quote of the Day
#                           Jon Postel <postel@isi.edu>
msp             18/udp      # Message Send Protocol
msp             18/tcp      # Message Send Protocol
#                           Rina Nethaniel <---none--->
chargen         19/udp      # Character Generator
chargen         19/tcp      # Character Generator
ftp-data        20/udp      # File Transfer [Default Data]
ftp-data        20/tcp      # File Transfer [Default Data]
ftp             21/udp      # File Transfer [Control]
ftp             21/tcp      # File Transfer [Control]
#                           Jon Postel <postel@isi.edu>
ssh             22/udp      # SSH Remote Login Protocol
ssh             22/tcp      # SSH Remote Login Protocol
#                           Tatu Ylonen <ylo@cs.hut.fi>
telnet          23/udp      # Telnet
telnet          23/tcp      # Telnet
```

SSH = Secure Shell → zum Einwählen in andere Rechner

```
qotd            17/udp      # Quote of the Day
qotd            17/tcp      # Quote of the Day
#                           Jon Postel <postel@isi.edu>
msp             18/udp      # Message Send Protocol
msp             18/tcp      # Message Send Protocol
#                           Rina Nethaniel <---none--->
chargen         19/udp      # Character Generator
chargen         19/tcp      # Character Generator
ftp-data        20/udp      # File Transfer [Default Data]
ftp-data        20/tcp      # File Transfer [Default Data]
ftp             21/udp      # File Transfer [Control]
ftp             21/tcp      # File Transfer [Control]
#                           Jon Postel <postel@isi.edu>
ssh             22/udp      # SSH Remote Login Protocol
ssh             22/tcp      # SSH Remote Login Protocol
#                           Tatu Ylonen <ylo@cs.hut.fi>
telnet          23/udp      # Telnet
telnet          23/tcp      # Telnet
#                           Jon Postel <postel@isi.edu>
                24/udp      # any private mail system
                24/tcp      # any private mail system
#                           Rick Adams <rick@UUNET.UU.NET>
smtp            25/udp      # Simple Mail Transfer
smtp            25/tcp      # Simple Mail Transfer
#                           Jon Postel <postel@isi.edu>
#               26/tcp      Unassigned
#               26/udp      Unassigned
nsw-fe          27/udp      # NSW User System FE
nsw-fe          27/tcp      # NSW User System FE
#                           Robert Thomas <BThomas@F.BBN.COM>
```

SMTP = Simple Mail Transfer Protocol → liefert e-mail aus

```
netrjs-4        74/udp     # Remote Job Service
netrjs-4        74/tcp     # Remote Job Service
#                          Bob Braden <Braden@ISI.EDU>
                75/udp     # any private dial out service
                75/tcp     # any private dial out service
#                          Jon Postel <postel@isi.edu>
deos            76/udp     # Distributed External Object Store
deos            76/tcp     # Distributed External Object Store
#                          Robert Ullmann <ariel@world.std.com>
                77/udp     # any private RJE service
                77/tcp     # any private RJE service
#                          Jon Postel <postel@isi.edu>
vettcp          78/udp     # vettcp
vettcp          78/tcp     # vettcp
#                          Christopher Leong <leong@kolmod.mlo.d
finger          79/udp     # Finger
finger          79/tcp     # Finger
#                          David Zimmerman <dpz@RUTGERS.EDU>
http            80/udp      www www-http # World Wide Web HTTP
http            80/tcp      www www-http # World Wide Web HTTP
#                          Tim Berners-Lee <timbl@W3.org>
hosts2-ns       81/udp     # HOSTS2 Name Server
hosts2-ns       81/tcp     # HOSTS2 Name Server
#                          Earl Killian <EAK@MORDOR.S1.GOV>
xfer            82/udp     # XFER Utility
xfer            82/tcp     # XFER Utility
#                          Thomas M. Smith <Thomas.M.Smith@lmco.com>
mit-ml-dev      83/udp     # MIT ML Device
mit-ml-dev      83/tcp     # MIT ML Device
#                          David Reed <--none--->
ctf             84/udp     # Common Trace Facility
```

HTTP = HyperText Transfer Protocol → Liefert websiten aus

Create a Server

- Allow multiple connections

```
AT+CIPMUX=1
```

- Enable a server on a given port

```
AT+CIPSERVER=1,80
```

Port

Use your own SSID and a safe password!

Web Setup

```
void setup() {
  // Initialize serial connection
  Serial.begin(115200);
  Serial.setTimeout(5000);

  // Set mode to wifi access point
  if (!issueCommand("AT+CWMODE=2"))
       return;

  // Enable wifi access point with SSID "PFE"
  if (!issueCommand("AT+CWSAP=\"PFE\",\"12345678\",1,4"))
      return;

  // Enable multiple TCP/UDP connections
  if (!issueCommand("AT+CIPMUX=1"))
      return;

  // Enable TCP server on port 80
  if (!issueCommand("AT+CIPSERVER=1,80"))
      return;
}
```

Receiving Data

```
char *read_data(int *id) {
  // If a client connects, the modem sends a string
  // +IPD,<ID>,<len>[,<remote IP>,<remote port>]:<data>

  // Wait for connection from a client
  if (!Serial.findUntil("+IPD,", "\r"))
    return NULL;

  // read ID
  *id = Serial.parseInt();
  if (!Serial.findUntil(",", "\r"))
   return NULL;

  // read length
  int len = Serial.parseInt();

  // ignore until colon
  if (!Serial.findUntil(":", "\r"))
   return NULL;
```

```
  // read ID
  *id = Serial.parseInt();
  if (!Serial.findUntil(",", "\r"))
   return NULL;

  // read length
  int len = Serial.parseInt();

  // ignore until colon
  if (!Serial.findUntil(":", "\r"))
   return NULL;

  // allocate data
  char *data = (char *)malloc(len + 1);
  if (data == NULL)
    return NULL;

  // Fill it
  Serial.readBytes(data, len);

  // And we're done
  data[len] = '\0';

  return data;
}
```

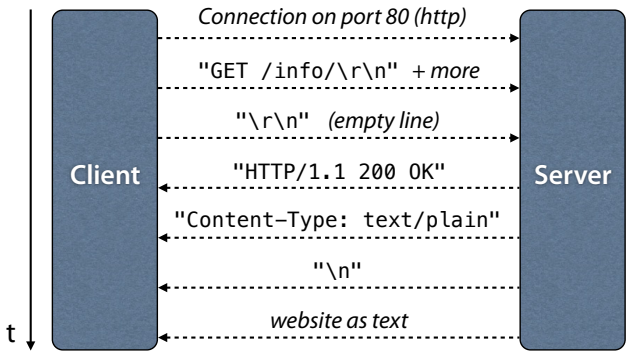
Sending Data

```
void send_data(char *data, int id) {
  // To send data, use "AT+CIPSEND=<id>,<len>\r\n",
  // followed by data
  int len = strlen(data);

  Serial.print("AT+CIPSEND=");
  Serial.print(id);
  Serial.print(",");
  Serial.println(len);
  delay(20);

  Serial.write(data, len);
  delay(100);
}
```

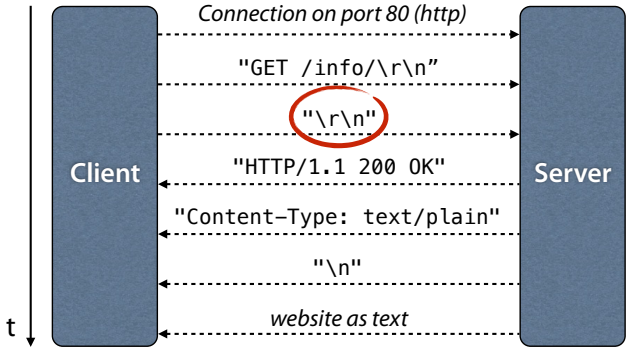
Web

- A *Web server* (= a computer) waits on port 80 for a *Web client* (= another computer) to initiate a connection.
- The client sends a request for a specific website
- The server then delivers this website

http://192.168.4.1/info/



http://192.168.4.1/info/



Waiting for Empty Line

- Aside from the GET command the browser also sends information about itself
- We read until we see an empty line
- An empty line consists of two consecutive '\n' (newline symbol)
- There can also be '\r' (carriage return) characters in-between

Processing HTML

```
void process_data(char *data, int id) {
    // We ignore all requests except for GET
    if (strncmp(data, "GET", strlen("GET")))
      return;

    // This is where extra processing of data
    // may take place

    send_html("<h1>Hello, world</h1>", id);
}
```

Sending HTML

```
void send_html(char *data, int id) {
  // We always send the same page
  char output[2048];

  sprintf(output,
"HTTP/1.1 200 OK\r\n\
Content-Type: text/html\r\n\
Content-Length: %d\r\n\r\n%s",
  strlen(data), data);

  send_data(output, id);
}
```

Demo

Im Browser Adresse http://192.168.0.42/info/ eingeben – auf serieller Ausgabe sehen, was ankommt

---

Inputs

- Idea: control LED via the website
- turn on with http://192.168.4.1/on/
- turn off with http://192.168.4.1/off/

Inputs

```
void process_data(char *data, int id) {
    // We ignore all requests except for GET
    if (strncmp(data, "GET", strlen("GET")))
      return;

    if (strncmp(data, "GET /on", strlen("GET /on")) == 0)
    {
      turn_led_on();
      send_html("<h1>LED is on</h1>", id);
    }
    else if (strncmp(data, "GET /off", strlen("GET /off")) == 0)
    {
      turn_led_off();
      send_html("<h1>LED is off</h1>", id);
    }
    else {
      send_html("<h1>Hello, world</h1>", id);
    }
}
```

Demo

Links

- In HTML by using
  `<a href="URL">text</a>`
  one can link to other websites
- URLs without a host name (www.foo.com) link to the same host

Outputting Links

```
client.println("<p>");
client.println("LED <a href=\"/on\">turn on</a>");
client.println(" | ");
client.println("<a href=\"/off\">turn off</a>");
client.println("</p>");
```

produces

```
<p>
LED <a href="/on">turn on</a>
 |
<a href="/off">turn off</a>
</p>
```

\" = Anführungszeichen innerhalb einer Zeichenkette

Inputs with Links

```
void process_data(char *data, int id) {
    // We ignore all requests except for GET
    if (strncmp(data, "GET", strlen("GET")))
      return;

    if (strncmp(data, "GET /on", strlen("GET /on")) == 0)
    {
      turn_led_on();
      send_html("<h1>LED is on</h1><a href=\"/off\">turn off</a>", id);
    }
    else if (strncmp(data, "GET /off", strlen("GET /off")) == 0)
    {
      turn_led_off();
      send_html("<h1>LED is off</h1><a href=\"/on\">turn on</a>", id);
    }
    else {
      send_html("<h1>Hello, world</h1><a href=\"/on\">turn on</a>", id);
    }
}
```

Demo

Access Control

- Behind a router or computer your Arduino is invisible to the internet
- On the internet anyone can access your device and record "secret" URLs
- Before you put your program on the Internet, please contact your friendly computer scientist

HTTP

HTML

IP

PORT

SHIELD

SHARING

ROUTER

INTERNET

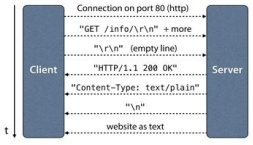
Creating a Network

- We want to create a *Wifi access point*
- Allows other devices to connect

```
AT+CWMODE=2
```

Wifi mode

http://192.168.0.42/info/



Web Setup

```
void setup() {
  // Initialize serial connection
  Serial.begin(115200);
  Serial.setTimeout(5000);

  // Set mode to wifi access point
  if (!issueCommand("AT+CWMODE=2"))
      return;

  // Enable wifi access point with SSID "PFE"
  if (!issueCommand("AT+CWSAP=\"PFE\",\"12345678\",1,4"))
      return;

  // Enable multiple TCP/UDP connections
  if (!issueCommand("AT+CIPMUX=1"))
      return;

  // Enable TCP server on port 80
  if (!issueCommand("AT+CIPSERVER=1,80"))
      return;
}
```

Receiving Data

```
char *read_data(int *id) {
  // If a client connects, the modem sends a string
  // +IPD,<ID>,<len>[,<remote IP>,<remote port>]:<data>

  // Wait for connection from a client
  if (!Serial.findUntil("+IPD,", "\r"))
    return NULL;

  // read ID
  *id = Serial.parseInt();
  if (!Serial.findUntil(",", "\r"))
   return NULL;

  // read length
  int len = Serial.parseInt();

  // ignore until colon
  if (!Serial.findUntil(":", "\r"))
   return NULL;
```