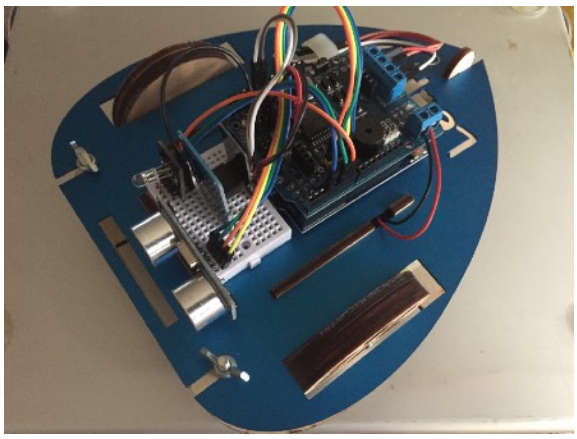
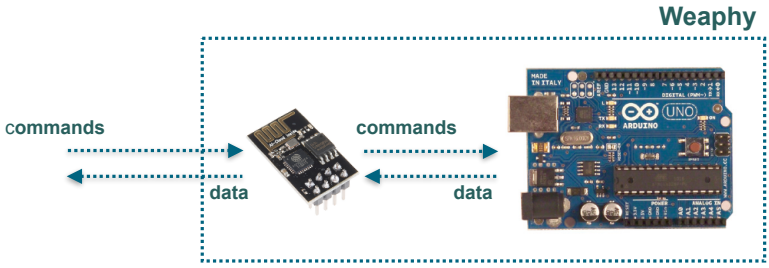
Inleiding (v0.15)

Weaphy is een Leaphy robot die is uitgebreid met een wifi-interface. Een Weaphy ziet er als volgt uit:



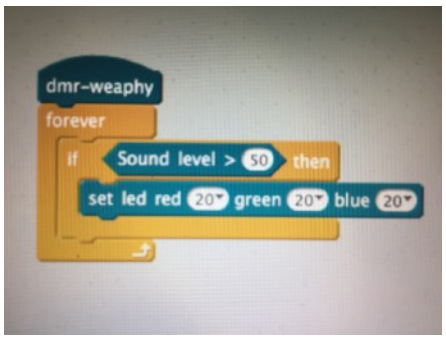
Een Leaphy, en dus ook een Weaphy, is een op Arduino gebaseerde robot. De Weaphy voegt daaraan een op een ESP8266 gebaseerde wifi-adapter toe. De logica in deze wifi-adapter stuurt de ontvangen data door naar de Arduino via een seriële verbinding.



De Arduino in de Weaphy robot kan op twee manieren worden geprogrammeerd om commando's die via wifi worden ontvangen om te zetten in daden (bijvoorbeeld vooruit of achteruit rijden).

De eerste manier is via de Arduino IDE, in C++. Dit programma kan de data ontvangen via een seriele poort -waar de ESP op is aangesloten- dan interpreteren en omzetten in aktie. Een voorbeeld programma hiervoor is opgenomen in Appendix D.

De tweede manier om de Arduino te programmeren is via mBlock, met behulp van een Weaphy-extensie. Via mBlock kunnen programma's worden gegenereerd die op basis van voorgedefinieerde blokken de data op de seriele poort -waar de ESP op is aangesloten- interpreteren en omzetten in aktie. Deze Weaphy-extensie voor mBlock is nog in ontwikkeling.



De ESP8266 kan ook op diverse manieren worden geprogrammeerd: via de Arduino IDE in C++, in de programmeertaal Lua en via een zogenoemde Hayes commandostructuur, zoals dat vroeger bij modems werd gebruikt.

Standaard is een ESP-01 zo geconfigureerd dat deze kan worden bestuurd via zogenoemde Hayes commando's (ook wel bekend als de AT-commando-set). Wil je de ESP-01 met Lua programmeren, zoals dat in deze beschrijving wordt gedaan, dan moet de ESP-01 eerst worden "geflashed"; hoe dat gaat wordt in deze handleiding beschreven.

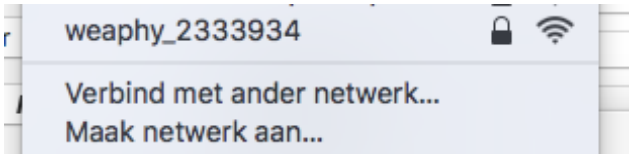
Dit document beschrijft Weaphy in een aantal stappen:
1. Het gebruiken van Weaphy
2. Het bouwen van Weaphy
3. Het voorbereiden van de wifi-adapter
4. Het voorbereiden van de Arduino

Om die stappen uit te kunnen voeren zijn diverse programmaatjes nodig. In Appendix A staat beschreven welke dat zijn en waar deze kunnen worden verkregen.

Het gebruiken van Weaphy

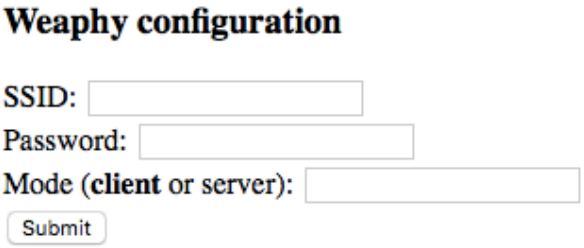
Weaphy de eerste keer aanzetten

Wanneer Weaphy voor de eerste keer wordt aangezet - én alle bestanden uit de Appendix B zijn op de wifi-adapter gezet - dan wordt automatisch een uniek netwerk aangemaakt, bijvoorbeeld weaphy_2333934. Elke wifi-client kan hiermee worden verbonden, bijvoorbeeld een laptop, een tablet of een mobile telefoon. Het standaard wachtwoord is *12345678*.



Wanneer een client is verbonden met dit netwerk, dan kan de Weaphy worden bestuurd.

De Weaphy kan ook worden verbonden aan een ander -reeds bestaand - wifi-netwerk. Dit kan bijvoorbeeld een wifi-netwerk op school zijn of thuis, en Weaphy kan ook worden verbonden met een andere Weaphy, de ene is dan server en de andere(n) client. Om de Weaphy aan een ander wifi-netwerk te koppelen is een configuratie-pagina opgenomen in de Weaphy:



Deze configuratie-pagina is op te roepen vanuit een browser dmv. *http://192.168.4.1/config.html*. Vul de gegevens in en druk op de *submit*-knop. De Weaphy moet daarna worden uitgezet en weer aangezet. De Weaphy onthoudt de instellingen van wifi, zodat bij de volgende keer gebruik hetzelfde netwerk wordt gekozen.

Weaphy besturen

Weaphy kan op een aantal manieren worden bestuurd. Als eerste kan Weaphy worden bestuurd via de ingebouwde webpagina die standaard bereikbaar is via *http://192.168.4.1* (Let op: hier moet iets anders worden ingevuld wanneer Weaphy aan een bestaand wifi-netwerk is gekoppeld). Dit levert de volgende pagina op:

Naast het besturen van Weaphy via een webpagina luistert Weaphy ook naar netwerkpakketten (UDP). Deze netwerkpakketten kunnen vanaf zowel een PC als vanaf een mobiel apparaat worden gestuurd. Hiervoor is echter wel een client nodig. In macOS of Linux is deze client standaard aanwezig:

```
echo "forward" | nc -w1 -u 192.168.2.157 54321
```

Met dit commando wordt de tekst “forward" in een netwerkpakket verpakt en gestuurd naar IP adres *192.168.2.157* op poortnummer *54321*. Weaphy luistert standaard naar dit poortnummer *54321.*

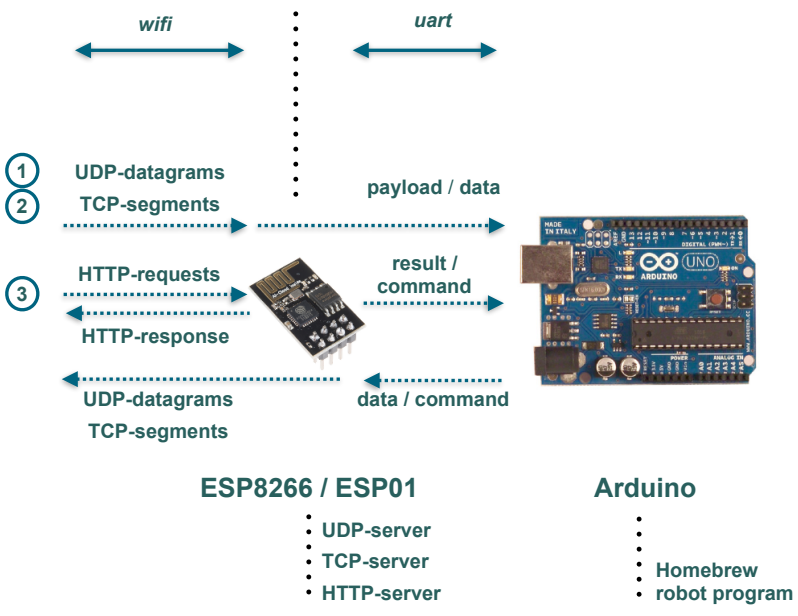
Op bijvoorbeeld een iPhone dient een app te worden geïnstalleerd om hetzelfde te kunnen doen, bijvoorbeeld de TCP-UDP Client van Lorenzo Greco.

Het bouwen van Weaphy

Overzicht

De wifi-adapter van een Weaphy is een zogenoemde ESP-01 / ESP8266 SoC. Hoewel de fysieke grootte van deze ESP-01 mogelijk niet imponeert, hij is vergelijkbaar met een complete computer inclusief processor, geheugen en zelfs ruimte om er bestanden op te zetten. Die ruimte kan bijvoorbeeld worden gebruikt om er programma’s op te bewaren; zo hoeft de ESP-01 niet elke keer opnieuw te worden geprogrammeerd als hij even van de stroom af is geweest.

Voor een Weaphy zijn die programma's bijvoorbeeld een UDP-server, een TCP-



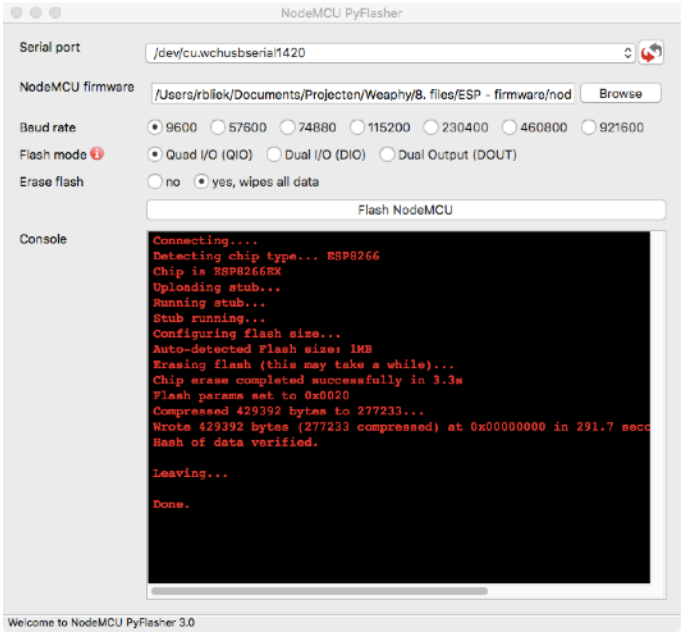
server en een HTTP-server. Deze programma's zijn in de programmeertaal Lua geschreven.
Voordat een ESP-01 wifi-adapter Lua-programma's kan uitvoeren is er een zogenoemde NodeMCU firmware nodig(als vervanging voor de standaard "AT-firmware").

Alle logica voor het opzetten van een wifi-verbinding, het in stand houden van verbindingen en het afhandelen van ontvangen data wordt binnen de ESP-01 wifi-adapter gedaan. Data gestuurd naar deze wifi-adapter, ongeacht of deze via UDP, TCP of HTTP is gestuurd, wordt door de wifi-adapter transparant doorgestuurd naar de Arduino via de seriële poort; de Arduino heeft er zelf geen weet van dat deze via wifi zijn verstuurd. De Arduino ontvangt alleen de commando's via de seriële poort.

Het bouwen van Weaphy bestaat grofweg uit de volgende stappen:
1. De wifi adapter "flashen" met NodeMCU firmware
2. Het programmeren van de wifi adapter
3. Het fysiek aansluiten van de wifi adapter aan de Leaphy/Arduino
4. Het programmeren van de Arduino

De wifi adapter "flashen"
De ESP-01 wifi-adapter wordt standaard geleverd met een firmware waardoor deze zich gedraagt als een modem uit vervlogen tijden (via de zogenoemde Hayes AT-commando set). Voor de Weaphy wordt echter een andere "firmware" gebruikt: de NodeMCU-firmware. Deze "NodeMCU-firmware" moet dus eerst op de ESP-01 wifi-adapter worden gezet. Dit proces heet "flashen" en het hoeft slechts één keer (per wifi adapter).

Om een ESP-01 wifi-adapter te kunnen "flashen", moet er eerst een "NodeMCU-firmware" worden gemaakt. Het maken van een dergelijke "firmware" gaat via een (gratis)



dienst die te bereiken is via een Internet website: <*https://nodemcu-build.com*>. Hier wordt om een email-adres gevraagd; dit adres wordt gebruikt om informatie naartoe te sturen die nodig is om de "firmware" te downloaden wanneer deze klaar is. Naast het email-adres kunnen ook optionele modules worden gekozen die die vervolgens in de "firmware" worden opgenomen. Voor Weaphy zijn de volgende modules relevant: file,

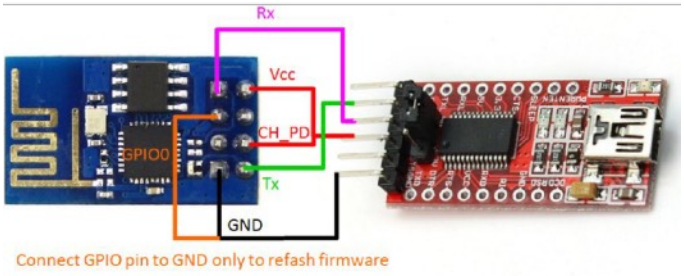
gpio, http, mdns, net, node, tmr, uart, wifi, en optioneel: ds18b20, voor temperatuur meten via DS18B20.

Er worden twee versies gemaakt van de “NodeMCU firmware”: een ‘float' en een ‘integer’ versie. Na enige tijd ontvangt je een email met daarin twee “links”, voor elke versie één. Beide versies kunnen worden gedownload en de resulterende files zijn vervolgens te gebruiken om de ESP-01 wifi-adapter te “flashen".

Het “flashen" kan bijvoorbeeld via de “command-line”, met een programma: esptool. Maar er is ook een wat meer gebruiksvriendelijk programma voor beschikbaar: NodeMCU PyFlasher. Een voorbeeld via een macOS computer met esptool; dit is een Python programma:

```
esptool.py --port /dev/cu.wchusbserial1420 write_flash -fm qio 0x00000 nodemcu-master-10-modules-2017-12-12-16-22-29-float.bin
```

Let op: voor het “flashen" van de ESP-01 moet de ESP-01 anders worden aangesloten. Naast Vcc, GND, RX, TX en CH_PD dient nu ook GPIO0 te worden aangesloten op de GND (en GPIO2 aan “Vcc”, maar “not-connected” werkt ook). Het schema is als volgt:



Dit vergt dus wat soldeerwerk. Het handigste is om een extra USB-serieel adapter te kopen en deze om te solderen met een extra verbinding (oranje verbinding in het schema), zodat deze altijd kan worden gebruikt om te “flashen” (naast een andere USB-serieel adapter die alleen wordt gebruikt om te programmeren).

Er zijn ook handige USB-serieel adapters speciaal voor de ESP-01 (bijvoorbeeld verkrijgbaar via www.benselectronics.nl).

esp8266-01 Programmer CH340

Een tip: je kunt twee van deze adapters kopen, waarbij je er één aanpast (soldeert) voor “flashen”, door aan de onderkant een bruggetje te solderen (zie schema eerder in dit document). Zorg er dan wel voor, tip 2, dat je de aangepaste adapter goed kunt herkennen, bijvoorbeeld door het gele connectortje een andere kleur te geven (met nagellak o.i.d.).

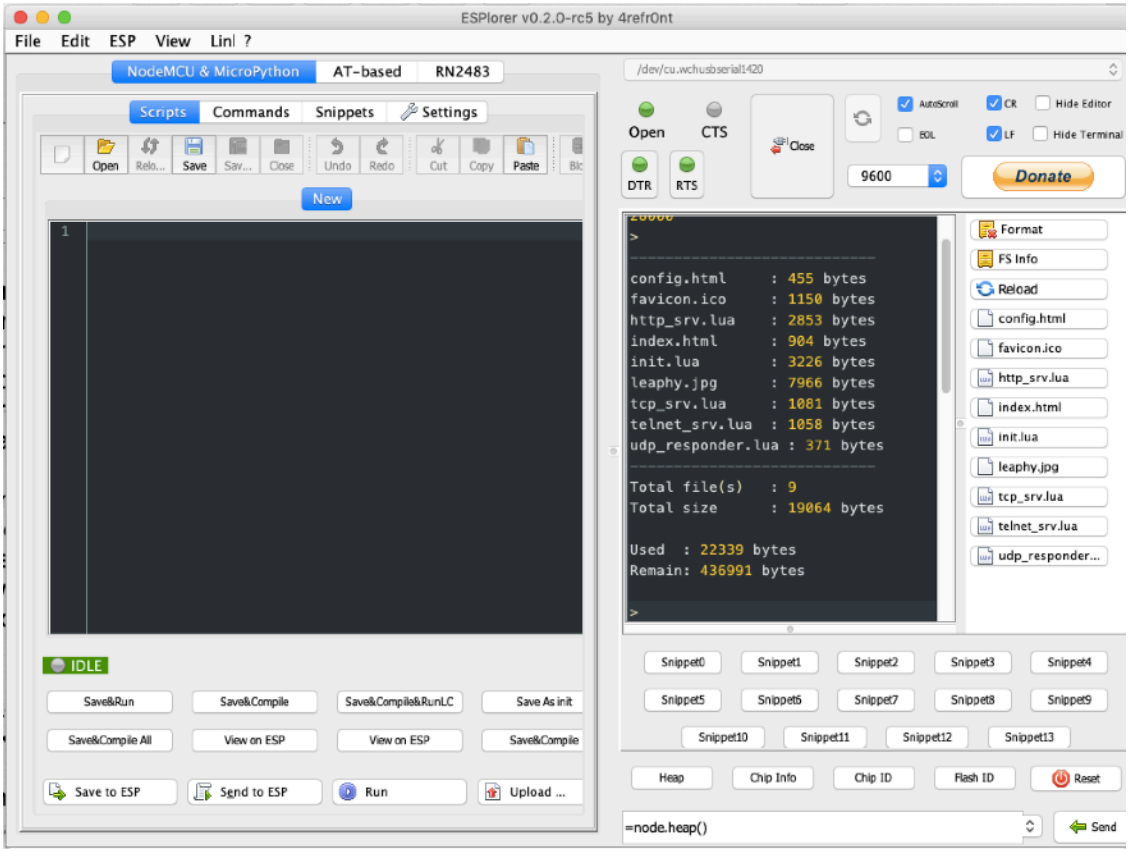
Het programmeren van de wifi adapter

Als het “flashen" van de wifi-adapter is gelukt, kan de adapter geprogrammeerd worden. Zoals eerder beschreven, het “flashen" hoeft maar één keer per adapter; daarna kan de adapter zo vaak geprogrammeerd worden als je maar wilt.

Eigenlijk is programmeren niet het juiste woord. Het programmeren zelf wordt namelijk vooraf op een computer gedaan. Het programmeren gebeurt in de taal Lua. De Lua-bestandjes worden vervolgens naar de wifi-adapter overgezet.

Om Lua-bestandjes op de wifi adapter te zetten is een speciaal programma nodig: ESPlorer is er voor zowel macOS als voor Windos. Met ESPlorer kunnen niet alleen Lua-bestandjes naar de wifi-adapter worden overgezet, maar ook andere bestanden zoals html en config-bestanden.

Met ESPlorer kunnen bestandjes ook weer worden verwijderd of van naam worden veranderd, net als bij een computer met Explorer of Finder.



Om ESPlorer te kunnen gebruiken moet een ESP worden aangesloten via een USB-poort. Let op de bps-rate. De eerste keer dat de ESP wordt aangesloten is dit 115200. Voor Weaphy wordt echter 9600 gebruikt. Dit is gedaan om de communicatie met de Arduino betrouwbaarder te maken. Verwarrend? Jazeker ;-) . Als de ESP niet reageert kies dan een andere bps-rate en probeer het daarna opnieuw.

Om bestanden naar de ESP te sturen kun je de knop “upload" gebruiken. Daarna kun je een bestand kiezen (of meerdere) en bevestigen. Aan de rechterkant van ESPlorer is een knop “reload”. Hiermee kun je de actuele status van de bestanden op de ESP laten zien.

De taal die de wifi-adapter begrijpt is Lua (tenminste, als het “flashen”, zoals eerder beschreven, is gelukt). Van Lua is heel veel informatie beschikbaar via Internet.

Een speciaal bestandje is init.lua. Elke keer wanneer de wifi-adapter aangezet wordt, dan wordt init.lua automatisch gestart. Vanuit init.lua kunnen ook andere programma's worden aangeroepen.

Om ervoor te zorgen dat de wifi-adapter niet vasloopt, bijvoorbeeld door een fout in je programma, kun je init.lua in eerste instantie setup.lua noemen. En pas als alles is getest ende goedbevonden, hernoem je deze in init.lua; dat kan met ESPlorer. Mocht je een wifi-adapter niet meer kunnen benaderen, dan kun je deze het beste opnieuw "flashen", zoals eerder beschreven.

Het aansluiten van de wifi-adapter aan Leaphy

De ESP-01 wifi-adapter werkt op 3.3V en kan daardoor niet rechtstreeks op een Arduino Uno worden aangesloten; de Arduino Uno werkt namelijk op 5V. Hiervoor zijn zogenaamde *level-shifters* nodig. En er zijn kant-en-klaar adaptertjes te koop, bijvoorbeeld: *http://s.aliexpress.com/RJJZFRNb*.

De communicatie tussen Arduino en ESP-01 wifi-adapter is seriëel. Hiervoor zijn zijn twee verbindingen nodig tussen de Arduino en de ESP-01: eentje voor "receive" en eentje voor "transmit". Voor de Arduino worden hiervoor A0/D14 ("receive"), en A1/D15 ("transmit") gebruikt. Op de ESP-adapter heten deze "RX" en "TX", voor resp. "receive" en "transmit". (Let op: deze verbindingen dienen kruislings te worden aangesloten, dus "RX" van de ESP-01 gaat aan A1 ("transmit") en "TX" van de ESP-01 gaat aan A0 ("receive").

Naast "TX" en "RX" dienen de "Vcc" en de "GND" van de ESP-01 te worden aangesloten. De "Vcc" van de ESP-01 op een "5V" pin van de Arduino en de "GND" van de ESP-01 op een "GND" pin van de Arduino.

Het programmeren van de Arduino

Om de werking van de Weaphy te testen is een simpel robot-programma gemaakt en in de Arduino gezet. Dit programma interpreteert de commando's die via een seriële poort worden ontvangen en zet deze om in actie.

De Arduino kan ook worden geprogrammeerd via een extensie in mBlock. Deze extensie is nog in ontwikkeling.

Appendix A - Benodigde software

- **NodeMCU-pyflasher**
  https://github.com/marcelstoer/nodemcu-pyflasher/releases.

- **ESPlorer**
  http://esp8266.ru/esplorer/.

- **esptool.py** (deze is niet nodig als je NodeMCU-pyflasher gebruikt)
  https://github.com/espressif/esptool

Appendix B - Bestanden op de ESP

Overzicht

init.lua / setup.lua
http_srv.lua
udp_srv.lua
tcp_srv.lua
telnet_srv.lua

config.html
index.html

wifi.cfg

Er zijn twee versies van de code: de eerste is het standaard scenario; de tweede betreft het “security-scenario”. Deze appendix beschrijft het standaard scenario.

init.lua / setup.lua

```
-- setup.lua (init.lua)

-- Setup uart, make permanent (last '1' makes permanent))
-- (uart,bps,databits,parity,stopbits,echo,permanent = 1)
uart.setup(0,9600,8,0,1,0,1)

SSID   = "weaphy_"..node.chipid()
PWD    = "12345678"
MODE   = "server"

function launch()
  -- Create index.html if it does not yet exist
  if not(file.exists("index.html"))
  then
    fl = file.open("index.html", "w")
    if fl
    then
      file.write('<!DOCTYPE HTML>\n')
      file.write('<html>\n')
      file.write('<head>\n')
      file.write('<meta content="text/html; charset=utf-8">\n')
      file.write('<title>Weaphy</title>\n')
      file.write('<style>table, th, td {text-align: center;}</style>\n')
      file.write('</head>\n')
      file.write('<body><h3>Hello, I am weaphy_' .. node.chipid() .. '</h3>\n')
      file.write('<table style="width:50%">\n')
      file.write('<tr>\n')
      file.write('<tr><td></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd"
value="forward"></form></td>\n')
      file.write('<td></td>')
      file.write('</tr>\n')
      file.write('<tr>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="left"></
form></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="stop"></
form></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="right"></
form></td>\n')
      file.write('</tr>\n')
      file.write('<tr>\n')
      file.write('<td></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd"
value="backward"></form></td>\n')
      file.write('<td></td>\n')
      file.write('</tr>\n')
      file.write('</table>\n')
      file.write('</body></html>\n')
```

```
      file.close()
      fd = nil
    end
  end

  -- Launch existing servers
  if file.exists("tcp_srv.lua")
  then
    dofile("tcp_srv.lua")
  end

  if file.exists("udp_responder.lua")
  then
    dofile("udp_responder.lua")
  end

  if file.exists("telnet_srv.lua")
  then
    dofile("telnet_srv.lua")
  end

  if file.exists("http_srv.lua")
  then
    dofile("http_srv.lua")
  end
end

function isconnected()
  -- Lets see if we are already connected by getting the IP
  if (MODE == "server")
  then
    ipAddr = wifi.ap.getip()
  else
    ipAddr = wifi.sta.getip()
  end

  resturn( (ipAddr ~= nil) and (ipAddr ~= "0.0.0.0") )
end

-- Let's see if there is a config file for wifi
if file.exists("wifi.cfg")
then
  file.open("wifi.cfg")
  SSID = file.readline()
  PWD  = file.readline()
  MODE = file.readline()
  file.close()

  -- Remove eol
  SSID = string.sub(SSID, 1, string.len(SSID)-1)
  PWD  = string.sub(PWD, 1, string.len(PWD)-1)
  MODE = string.sub(MODE, 1, string.len(MODE)-1)
end

-- Setup wifi, and connect
wifi.setphymode(wifi.PHYMODE_N)

if (MODE == "server")
then
  wifi.setmode(wifi.STATIONAP)
  wifi.ap.config({ssid=SSID, pwd=PWD})
else
  wifi.setmode(wifi.STATION)
  wifi.sta.config({ssid=SSID, pwd=PWD})
  wifi.sta.connect()
end

-- Assume we are connected, so just run the launch code.
launch()
```

http_srv.lua

```
srv=net.createServer(net.TCP)
```

```
srv:listen(80,function(conn)

  conn:on("receive", function(client,payload)

    --print(payload)

    -- first find GET or POST
    if string.find(payload, "GET")
    then
      -- method is GET, so send file (default is index.html)
      tgtfile = string.sub(payload,string.find(payload,"GET /")
              +5,string.find(payload,"HTTP/")-2)

      if tgtfile == ""
      then tgtfile = "index.html"
      end

      -- Check for .html or .ico or .png
      if (string.find(tgtfile, ".html")
          or string.find(tgtfile, ".ico")
          or string.find(tgtfile, ".png") ~= nil)
      then
        local f = file.open(tgtfile,"r")
        if f ~= nil
        then
          client:send(file.read())
          file.close()
        end
      else
        client:send("<html>"..tgtfile.." not found - 404 error.<BR><a href='index.html'>Home</
a><BR>")
      end

    else
      -- no GET so assume POST, find parameters
      -- expected parameters are [cmd] or [ssid] and [password]
      fssid={string.find(payload,"ssid=")}
      fcmd={string.find(payload,"cmd=")}

      if fcmd[2] ~= nil
      then
        foundcmd=string.sub(payload,string.find(payload,"cmd=")
            +4,#payload)
        print(foundcmd)

        -- Assume the request can be OK-ed
        conn:send('HTTP/1.0 200 OK\n')
        conn:send('Server: Weaphy HTTP\n')
        conn:send('\n')
        conn:send('\n')

        local f = file.open("index.html","r")
        if f ~= nil
        then
          client:send(file.read())
          file.close()
        end

      end

      if fssid[2]~=nil
      then
        foundssid=string.sub(payload,string.find(payload,"ssid=")
            +5,string.find(payload,"&password=")-1)
        foundpwd=string.sub(payload,string.find(payload,"password=")
            +9,string.find(payload,"&mode=")-1)
        foundmode=string.sub(payload,string.find(payload,"mode=")
            +5)

        -- Write the result in wifi.cfg
        file.open("wifi.cfg","w+")
        file.write(foundssid .. "\n")
```

```
        file.write(foundpwd .. "\n")
        if foundmode == "server"
        then
          file.write("server\n")
        else
          file.write("client\n")
        end
        file.close()

        -- Assume the request can be OK-ed
        conn:send('HTTP/1.0 200 OK\n')
        conn:send('Server: Weaphy HTTP\n')
        conn:send('\n')
        conn:send('\n')

        f = nil
        tgtfile = nil
        collectgarbage()
      end
    end
  end)

  conn:on("sent",function(conn) conn:close() end)

end)
```

udp_srv.lua

```
-- udp_srv.lua
-- Transparent UDP <-> serial

udppartnerip = nil
udppartnerport = 54321

udpSocket = net.createUDPSocket()
udpSocket:listen(54321)

udpSocket:on("receive", function(s, data, port, ip)
    print(data)
    udppartnerip = ip
end)

--uart.on("data", "\r",
uart.on("data",
  function(data)
    if udppartnerip ~= nil then
      udpSocket:send(udppartnerport, udppartnerip, data)
    end
end, 0)
```

tcp_srv.lua (optioneel)

```
-- a simple tcp server

-- restart server if needed
if tcp_srv ~= nil then
    tcp_srv:close()
end
tcp_srv = net.createServer(net.TCP, 180)

tcp_srv:listen(54321, function(socket)
    local fifo = {}
    local fifo_drained = true

    local function tcpsend(conn)
        if #fifo > 0 then
            conn:send(table.remove(fifo, 1))
        else
            fifo_drained = true
        end
    end

    local function s_output(str)
```

```
        table.insert(fifo, str)
        if socket ~= nil and fifo_drained then
            fifo_drained = false
            sender(socket)
        end
    end

    node.output(s_output, 0)   -- re-direct output to function s_ouput.

    socket:on("receive", function(c, payload)
        print(payload)
    end)
    socket:on("disconnection", function(c)
        node.output(nil)        -- unregist the redirect output function, output goes to serial
    end)
    socket:on("sent", sender)

    uart.on("data",
    function(data)
      tcpsend(data)
    end, 0)

end)
```

telnet_srv.lua (optioneel)

Deze telnet_srv code heb ik gevonden op Internet. Een telnet-server is handig om de weaphy ook via command-line te kunnen configureren en er bestanden naar toe te sturen.

```
-- a simple telnet server

-- restart server if needed
if telnet_srv ~= nil then
    telnet_srv:close()
end
telnet_srv = net.createServer(net.TCP, 180)

telnet_srv:listen(23, function(socket)
    local fifo = {}
    local fifo_drained = true

    local function sender(c)
        if #fifo > 0 then
            c:send(table.remove(fifo, 1))
        else
            fifo_drained = true
        end
    end

    local function s_output(str)
        table.insert(fifo, str)
        if socket ~= nil and fifo_drained then
            fifo_drained = false
            sender(socket)
        end
    end

    node.output(s_output, 0)   -- re-direct output to function s_ouput.

    socket:on("receive", function(c, l)
        node.input(l)           -- works like pcall(loadstring(l)) but support multiple separate
line
    end)
    socket:on("disconnection", function(c)
        node.output(nil)        -- un-regist the redirect output function, output goes to serial
    end)
    socket:on("sent", sender)

    print("Hello, I am Weaphy_" .. node.chipid())
end)
```

config.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Weaphy Config</title>
</head>
<body>
  <h3>Weaphy configuration</h3>
  <form action="" method="POST">
    SSID: <input type="text" name="ssid"><br>
    Password: <input type="text" name="password"><br>
    Mode (<b>client</b> or server): <input type="text" name="mode"><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

index.html

De <index.html> wordt door Weaphy zelf gecreëerd indien deze voor de eerste keer wordt opgestart. Deze pagina hoeft dus niet op de Weaphy gezet te worden (de reden is dat de naam van de Weaphy uniek gegenereerd wordt in deze pagina).

```
<!DOCTYPE HTML>
<html>
<head>
  <meta  content="text/html; charset=utf-8">
  <title>Weaphy</title>
  <style>table, th, td {text-align: center;}</style>
</head>

<body><h3>Hello, I am weaphy_2333934</h3>
<table style="width:50%">
  <tr>
    <tr>
    <td></td>
    <td>
      <form action="" method="POST">
      <input type="submit" name="cmd" value="forward">
      </form>
    </td>
    <td></td>
  </tr>
  <tr>
    <td>
      <form action="" method="POST">
      <input type="submit" name="cmd" value="left">
      </form>
    </td>
    <td>
      <form action="" method="POST">
      <input type="submit" name="cmd" value="stop">
      </form>
    </td>
    <td>
      <form action="" method="POST">
      <input type="submit" name="cmd" value="right">
      </form>
    </td>
  </tr>
    <tr>
    <td></td>
    <td>
      <form action="" method="POST">
      <input type="submit" name="cmd" value="backward">
      </form>
    </td>
    <td></td>
  </tr>
</table>
</body></html>
```

Appendix C - Bestanden op de ESP (security scenario)

Overzicht

init.lua / setup.lua
http_srv.lua
udp_srv.lua
tcp_srv.lua
telnet_srv.lua

config.html
index.html

wifi.cfg
cl.cfg
secrets.cfg

Er zijn twee versies van de code: de eerste is het standaard scenario; de tweede betreft het “security-scenario”. Deze appendix beschrijft het “security-scenario”. Alleen nieuwe of gewijzigde bestanden worden getoond; de overige bestanden zijn dezelfde als in het standaard scenario (in bovenstaand overzicht zijn deze licht grijs gemaakt; de gewijzigde en nieuwe bestanden cursief).

init.lua / setup.lua

```
-- setup.lua (init.lua)

-- Setup uart, make permanent (last '1' makes permanent))
-- (uart,bps,databits,parity,stopbits,echo,permanent = 1)
uart.setup(0,9600,8,0,1,0,1)

SSID   = "weaphy_"..node.chipid()
PWD    = "12345678"
MODE   = "server"

function createpages()
  -- Create index.html if it does not yet exist
  if not(file.exists("index.html"))
  then
    fl = file.open("index.html", "w")
    if fl
    then
      file.write('<!DOCTYPE HTML>\n')
      file.write('<html>\n')
      file.write('<head>\n')
      file.write('<meta content="text/html; charset=utf-8">\n')
      file.write('<title>Weaphy</title>\n')
      file.write('<style>table, th, td {text-align: center;}</style>\n')
      file.write('</head>\n')
      file.write('<body><h1>Hello, I am weaphy_' .. node.chipid() .. '</h3>\n')
      file.write('<table style="width:50%">\n')
      file.write('<tr>\n')
      file.write('<tr><td></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd"
value="forward"style=" font-size:30px"></form></td>\n')
      file.write('<td></td>')
      file.write('</tr>\n')
      file.write('<tr>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="left"
style="font-size:30px"></form></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="stop"
style="font-size:30px"></form></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="right"
style="font-size:30px"></form></td>\n')
      file.write('</tr>\n')
```

```
      file.write('<tr>\n')
      file.write('<td></td>\n')
      file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="backward"
style="font-size:30px"></form></td>\n')
      file.write('<td></td>\n')
      file.write('</tr>\n')
      file.write('</table>\n')
      file.write('</body></html>\n')
      file.close()
      fl = nil
    end -- if fl
  end -- if not(file.exists("index.html"))

  -- Read security clearance level from scl.cfg
  -- If applicable find security level clearance questions and answers
  if file.exists("scl.cfg")
  then
    -- Read security clearance level
    file.open("scl.cfg")
    line = file.readline()
    scl = tonumber(string.sub(line, 1, string.len(line)-1))
    sclindex = scl
    file.close()

    while (sclindex > 0)
    do
      -- first determine security question for sclindex security clearance level
      if file.exists("secrets.cfg")
      then
        file.open("secrets.cfg")
        sclquestion = ""
        repeat
          line = file.readline()
          if line ~= nil
          then
            foundsclline = (sclindex ==
                              (tonumber(string.sub(line,
                                                   1,
                                                   string.find(line,":")-1
                                                  )
                                        )
                              )
                           )
            if foundsclline
            then
              sclquestion = string.sub(line,
                                       string.find(line,":")+1,
                                       string.find(line,"=")-1
                                      )
            end -- foundsclline
          else eol = true
          end -- line ~= nil
        until (eol or foundsclline)
        file.close()
      end -- if file.exists("secrets.cfg")

      -- Create page hat is returned at security clearance level = sclindex
      -- sclquestion now contains the question for this level
      pagename = (tostring(sclindex) .. "index.html")
      fl = file.open(pagename, "w+")
      if fl
      then
        file.write('<!DOCTYPE HTML>\n')
        file.write('<html>\n')
        file.write('<head>\n')
        file.write('<meta content="text/html; charset=utf-8">\n')
        file.write('<title>Weaphy</title>\n')
        file.write('<style>table, th, td {text-align: center;}</style>\n')
        file.write('</head>\n')
        file.write('<body><h1>Hello, I am weaphy_' .. node.chipid() .. '</h3>\n')

        file.write('<table style="width:50%">\n')
```

```
        file.write('<tr>\n')
        file.write('<tr><td></td>\n')

        if (sclindex <= 3)
        then
          file.write('<td><form action="" method="POST"><input type="submit" name="cmd"
value="forward"style=" font-size:30px"></form></td>\n')
        end

        file.write('<td></td>')
        file.write('</tr>\n')
        file.write('<tr>\n')

        if (sclindex <= 2)
        then
          file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="left"
style="font-size:30px"></form></td>\n')
        end

        file.write('<td><form action="" method="POST"><input type="submit" name="cmd" value="stop"
style="font-size:30px"></form></td>\n')

        if (sclindex <= 2)
        then
          file.write('<td><form action="" method="POST"><input type="submit" name="cmd"
value="right" style="font-size:30px"></form></td>\n')
        end

        file.write('</tr>\n')
        file.write('<tr>\n')
        file.write('<td></td>\n')

        if (sclindex <= 3)
        then
          file.write('<td><form action="" method="POST"><input type="submit" name="cmd"
value="backward" style="font-size:30px"></form></td>\n')
        end

        file.write('<td></td>\n')
        file.write('</tr>\n')
        file.write('</table>\n')

        -- write active question and an input field
        file.write('\n')
        file.write(sclquestion .. '\n')
        file.write('<form action="" method="POST">')
        file.write('Answer: <input type="text" name="secret"><br>')
        file.write('<input type="submit" value="Submit">')
        file.write('</form>')

        file.write('</body></html>\n')
        file.close()
        fl = nil
      end --  if fl
      sclindex = sclindex - 1
    end -- while (sclindex > 0)
  end -- if file.exists("scl.cfg")
end -- function createpages()

function launch()
  -- Launch existing servers
  if file.exists("tcp_srv.lua")
  then
    dofile("tcp_srv.lua")
  end

  if file.exists("udp_responder.lua")
  then
    dofile("udp_responder.lua")
  end

  if file.exists("telnet_srv.lua")
```

```
  then
    dofile("telnet_srv.lua")
  end

  if file.exists("http_srv.lua")
  then
    dofile("http_srv.lua")
  end
end -- function launch()

function isconnected()
  -- Lets see if we are already connected by getting the IP
  if (MODE == "server")
  then
    ipAddr = wifi.ap.getip()
  else
    ipAddr = wifi.sta.getip()
  end

  return( (ipAddr ~= nil) and (ipAddr ~= "0.0.0.0"))
end -- function isconnected()

-- start with creating all nescessary html files
createpages()

-- Let's see if there is a config file for wifi
if file.exists("wifi.cfg")
then
  file.open("wifi.cfg")
  SSID = file.readline()
  PWD  = file.readline()
  MODE = file.readline()
  file.close()

  -- Remove eol
  SSID = string.sub(SSID, 1, string.len(SSID)-1)
  PWD  = string.sub(PWD, 1, string.len(PWD)-1)
  MODE = string.sub(MODE, 1, string.len(MODE)-1)
end -- if file.exists("wifi.cfg")

-- Setup wifi, and connect
wifi.setphymode(wifi.PHYMODE_N)

if (MODE == "server")
then
  wifi.setmode(wifi.STATIONAP)
  wifi.ap.config({ssid=SSID, pwd=PWD})
else
  wifi.setmode(wifi.STATION)
  wifi.sta.config({ssid=SSID, pwd=PWD})
  wifi.sta.connect()
end -- if (MODE == "server")

-- Assume we are connected, so just run the launch code.
launch()
```

http_serv.lua

```
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)

  conn:on("receive", function(client,payload)
    -- first find GET or POST
    if string.find(payload, "GET")
    then
      -- method is GET, extract uri, and be sure to check syntax
      uri = ""
      if string.find(payload, "GET /")
      then
        if string.find(payload, "HTTP/")
        then
          uri = string.sub(payload,string.find(payload,"GET /")
                  +5,string.find(payload,"HTTP/")-2)
```

```
      end
    end

    if uri == ""
    then
      -- nothing found, assume default = index.html
      tgtfile = "index.html"
    else
      tgtfile = uri
    end

    -- Check scl
    scl=0
    if file.exists("scl.cfg")
    then
      file.open("scl.cfg")
      line = file.readline()
      scl = tonumber(string.sub(line, 1, string.len(line)-1))
      file.close()
    end

    -- If scl ~= 0 then add scl to uri
    if scl ~= 0
    then
      tgtfile = (tostring(scl) .. tgtfile)
    end

    -- Check for .html or .ico or .png
    if (string.find(tgtfile, ".html")
        or string.find(tgtfile, ".ico")
        or string.find(tgtfile, ".png") ~= nil)
    then
      local f = file.open(tgtfile,"r")
      if f ~= nil
      then
        client:send(file.read())
        file.close()
      end
    else
      client:send("<html>"..tgtfile.." not found - 404 error.<BR><a href='index.html'>Home</
a><BR>")
    end

  else
    -- no GET so assume POST, find parameters
    -- expected parameters are
    -- 1. [cmd]
    -- 2. [ssid] and [password]
    -- 3. [secret]

    fssid={string.find(payload,"ssid=")}
    fcmd={string.find(payload,"cmd=")}
    fsecret={string.find(payload,"secret=")}

    -- 1. [cmd]
    if fcmd[2]~=nil
    then
      foundcmd=string.sub(payload,string.find(payload,"cmd=")
          +4,#payload)
      print(foundcmd)

      -- Assume the request can be OK-ed
      conn:send('HTTP/1.0 200 OK\n')
      conn:send('Server: Weaphy HTTP\n')
      conn:send('\n')
      conn:send('\n')

      local f = file.open("index.html","r")
      if f ~= nil
      then
        client:send(file.read())
        file.close()
```

```
  end
end

-- 2. [ssid] and [password]
if fssid[2]~=nil
then
  foundssid=string.sub(payload,string.find(payload,"ssid=")
      +5,string.find(payload,"&password=")-1)
  foundpwd=string.sub(payload,string.find(payload,"password=")
      +9,string.find(payload,"&mode=")-1)
  foundmode=string.sub(payload,string.find(payload,"mode=")
      +5)

  -- Write the result in wifi.cfg
  file.open("wifi.cfg","w+")
  file.write(foundssid .. "\n")
  file.write(foundpwd .. "\n")
  if foundmode == "server"
  then
    file.write("server\n")
  else
    file.write("client\n")
  end
  file.close()

  -- Assume the request can be OK-ed
  conn:send('HTTP/1.0 200 OK\n')
  conn:send('Server: Weaphy HTTP\n')
  conn:send('\n')
  conn:send('\n')

  f = nil
  tgtfile = nil
  collectgarbage()
end

-- 3. [secret]
if fsecret[2]~=nil
then
  foundsecret=string.sub(payload,string.find(payload,"secret=")
      +7)

  -- Read security clearance level from scl.cfg
  if file.exists("scl.cfg")
  then
    -- Read security clearance level
    file.open("scl.cfg")
    line = file.readline()
    scl = tonumber(string.sub(line, 1, string.len(line)-1))
    file.close()
  else
    -- assume security clearance level is 0
    scl = 0
  end

  -- Compare answer to that in secret.cfg (depending on security clearance level)
  if (scl ~= 0 and file.exists("secrets.cfg"))
  then
    file.open("secrets.cfg")
    correctanswer = false
    repeat
      line = file.readline()
      if line ~= nil
      then
        sclindex = tonumber(string.sub(line, 1, string.find(line,":")-1))

        if (scl == sclindex)
        then
          sclanswer = string.sub(line, string.find(line,"=")+1)
          -- remove eol before compare
          sclanswer = string.sub(sclanswer, 1, string.len(sclanswer)-1)
          correctanswer = (sclanswer == foundsecret)
```

```
            end
          else eol = true
          end
        until (eol or correctanswer)
        file.close()

        print(sclindex, foundsecret, string.len(foundsecret), sclanswer, string.len(sclanswer),
correctanswer)

        -- If answer is correct, write the new security clearance level in scl.cfg
        if (correctanswer)
        then
          scl = scl - 1
          file.open("scl.cfg","w+")
          file.write(tostring(scl) .. "\n")
          file.close()
        end
      end

      -- Assume the request can be OK-ed
      conn:send('HTTP/1.0 200 OK\n')
      conn:send('Server: Weaphy HTTP\n')
      conn:send('\n')
      conn:send('\n')

    end
  end
end)

conn:on("sent",function(conn) conn:close() end)

end)
```

scl.cfg

```
4
```

secrets.cfg

Het secrets.cfg bevat de vragen en antwoorden bij elk security clearance level (het aantal levels staat in scl.cfg; en deze wordt automatisch aangepast als de gebruiker vragen goed beantwoordt). Als je andere vragen -en antwoorden- wilt gebruiken, dan hoef je alleen het secrets.cfg bestand aan te passen en scl.cfg.

Een voorbeeld van secrets.cfg is:

```
1:Wat is het antwoord op deze laatste vraag?=42
2:Hoeveel delen gaan er in een trilogie?=5
3:Welke Adams heeft een liftershandboek geschreven?=douglas
4:Wat is de naam van een manisch depressieve robot?=marvin
```

De syntax per lijn is:
<level> : <vraag> = <antwoord>

Alles in kleine letters.

Appendix D - Arduino Code

Het volgende programma is een simpel robot-programma die zorgt dat alle commando's die via de seriële poort worden gelezen en geïnterpreteerd worden en vervolgens omgezet in acties (bijvoorbeeld: de motoren worden aangezet zo dat de robot vooruit gaat). Dit programma is slechts een voorbeeld en het is nagenoeg compleet (al zijn nog niet alle commando's geïmplementeerd).

```
#include <SoftwareSerial.h>

#define DEBUG 1
```

```
// Hardware Serial
// D00 RX
// D01 TX

// D02

// Leaphy L298P Buzzer
#define BUZZER       4

// Leayphy LED
#define LED_BLUE     3
#define LED_GREEN    5
#define LED_RED      6

// Leaphy Ultrasone
#define US_ECHO      8
#define US_TRIG      7

// D09

// Leaphy L298P Motor A/B
#define L298P_MA_SPEED 10 // D10 L298P Motor A PWM Speed control
#define L298P_MA_DIR   12//  D12 L298P Motor A Direction
#define L298P_MB_SPEED 11 // D11 L298P Motor B PWM Speed control
#define L298P_MB_DIR   13 // D13 L298P Motor B Direction

// Leaphy mySerial for ESP
#define SER_RX  14
#define SER_TX  15
#define SER_BPS 9600

#define CMD_FORWARD  "forward"
#define CMD_BACK     "backward"
#define CMD_LEFT     "left"
#define CMD_RIGHT    "right"
#define CMD_LED      "led"
  #define PRM_LED_RED    "red"
  #define PRM_LED_GREEN  "green"
  #define PRM_LED_BLUE   "blue"
#define CMD_STOP     "stop"
#define CMD_START    "start"

#define CMD_GET      "get"
#define CMD_SET      "set"

#define CMD_SEND     "send"

#define CMD_NETWORK  "network"

#define CMD_LENGTH 20

#define SPD_MIN 70   // minimum pwm speed (motor requires minimum pwm for movement)
#define SPD_MAX 255  // max = 255
#define SPD_INC 10   // increments

SoftwareSerial mySerial(SER_RX, SER_TX);

uint8_t led_red, led_blue, led_green; // LED
bool    spd_forward;    // speed-direction
uint8_t spd_left, spd_right; // pwm speed left/right 0 - 255

String getValue(String data, int nr)
{
  int found;
  int prmBase;
  int prmEnd;
  int curChr;
  int strEnd;
  char chrread;
  boolean eol,rdy;
```

```
  found = 0;
  prmBase = 0;
  prmEnd = 0;
  curChr= 0;
  strEnd = data.length()-1;

  while ((curChr < strEnd) && (found < nr))
  { // There are still characters in data to examine

    chrread = data.charAt(curChr);
    eol = ((curChr >= strEnd) || (chrread == '\n') || (chrread == '\r'));

    while ((chrread ==' ') && (!eol))
    { // Discard spaces
      curChr++;
      chrread = data.charAt(curChr);
      eol = ((curChr >= strEnd) || (chrread == '\n') || (chrread == '\r'));
    }

    // prmBase points to the first non-space and/or nothing more to examine

    if (!eol)
    { // There is a parameter found, try to find the end of it
      found++;
      prmBase = curChr;
      prmEnd = prmBase + 1;

      chrread = data.charAt(prmEnd);
      eol = ((prmEnd >= strEnd) || (chrread == '\n') || (chrread == '\r'));

      while ((chrread != ' ') && (!eol))
      { // try to find the end of the paramter, one chr at a a time
        prmEnd++;
        chrread = data.charAt(prmEnd);
        eol = ((prmEnd >= strEnd) || (chrread == '\n') || (chrread == '\r'));
      }
    }

    // prmBase points to first character, and
    // prmEnd points to the next space OR EOL

    curChr=prmEnd+1;
  }

  if (found == nr)
  {
    return data.substring(prmBase,prmEnd);
  }
  else
  {
    return "";
  }
}

String readline()
{
  bool eol;
  char rdchar;
  String data = "";

  eol = false;

  while (!mySerial.available())
  {
    // Wait for something to appear on serial
  }

  do
  {
    if (mySerial.available())
    {
      rdchar = mySerial.read();
```

```
      // QaD: first check if it is (part of) Orion protocol [FF 55 00 04 07 data]
      if (rdchar == 0xff)
      { // Suspect orion-packet found
        if (mySerial.available())
        {
          rdchar = mySerial.read();
          if (rdchar == 0x55)
          { // Assume orion-packet found, discard preamble (3 octets), and read next
            for (int i = 0; i <= 3; i++)
            {
              if (mySerial.available())
              {
                rdchar = mySerial.read();
              }
            }
          }
        }
      }
      // End of QaD

      data.concat(rdchar);
      delay(3);
    }
  }
  while ((rdchar != '\r') && (rdchar != '\n'));

  return data;
}

void setspeed()
{ // Set direction and speed of motors
  if (spd_forward)
  { // going forward
    digitalWrite(L298P_MA_DIR, HIGH);
    digitalWrite(L298P_MB_DIR, HIGH);
  }
  else
  { // going backward
    digitalWrite(L298P_MA_DIR, LOW);
    digitalWrite(L298P_MB_DIR, LOW);
  }

  // Set speed motors
  analogWrite(L298P_MA_SPEED, spd_left);
  analogWrite(L298P_MB_SPEED, spd_right);
}

void backward()
{ // Go backward at minimum speed
  spd_forward = false;
  spd_left  = SPD_MIN;
  spd_right = SPD_MIN;
  setspeed();
}

void forward()
{ // Go forward at minimum speed
  spd_forward = true;
  spd_left  = SPD_MIN;
  spd_right = SPD_MIN;
  setspeed();
}

void left()
{ // Turn left, by decrasing speed left motor OR increasing right motor
  if (spd_left> (SPD_MIN + SPD_INC))
  { // Decrease speed left motor
    spd_left = (spd_left - SPD_INC);
  }
  else
  { // Increase speed right motor (if possible)
```

```
    spd_right = ((spd_right + SPD_INC) % SPD_MAX);
  }
  setspeed();
}

void right()
{ // Turn right by decreasing speed right motor OR increasing left motor
  if (spd_right > (SPD_MIN + SPD_INC))
  { // Decrease speed right motor
    spd_right = (spd_right - SPD_INC);
  }
  else
  { // Increase speed left motor (if possible)
    spd_left = ((spd_left + SPD_INC) % SPD_MAX);
  }
  setspeed();
}

void increase()
{ // Increase speed in 5 percentage points
  // Note: if one of both motors is (almost) at 100% (=255), direction will change
  //       more intelligent solution may be required in future
  spd_right = ((spd_right + SPD_INC) % SPD_MAX);
  spd_left = ((spd_left + SPD_INC) % SPD_MAX);
  if (spd_right <= SPD_MIN)
  {
    spd_right = SPD_MIN;
  }
  if (spd_left <= SPD_MIN)
  {
    spd_left = SPD_MIN;
  }
  setspeed();
}

void decrease()
{ // Decrease speed in 5 percentage points
  // Note: if one of both motors is (almost) at 0%, direction will change
  //       more intelligent solution may be required in future
  if (spd_right >= (SPD_MIN + SPD_INC))
  {
    spd_right = (spd_right - SPD_INC);
  }
  else
  {
    spd_right = SPD_MIN;
  }

  if (spd_left >= (SPD_MIN + SPD_INC))
  {
    spd_left = (spd_left - SPD_INC);
  }
  else
  {
    spd_left = SPD_MIN;
  }
  setspeed();
}

void setled(uint8_t red, uint8_t green, uint8_t blue)
{
  analogWrite(LED_RED, red);
  analogWrite(LED_GREEN, green);
  analogWrite(LED_BLUE, blue);

  #ifdef DEBUG
    Serial.println("LED set");
  #endif
}

void setup()
{
```

```
  char data;

  // Initialize LED
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  led_red = 0;
  led_blue = 0;
  led_green = 0;
  setled(led_red, led_green, led_blue);

  // Initialize L298P Motor Shield, both motors: Speed = 0;
  pinMode(L298P_MA_SPEED, OUTPUT);
  pinMode(L298P_MA_DIR, OUTPUT);
  analogWrite(L298P_MA_SPEED, 0);
  digitalWrite(L298P_MA_DIR, HIGH);
  pinMode(L298P_MB_SPEED, OUTPUT);
  pinMode(L298P_MB_DIR, OUTPUT);
  analogWrite(L298P_MB_SPEED, 0);
  digitalWrite(L298P_MB_DIR, HIGH);
  // Set standaard direction to forward and speed to 80, both motors
  spd_forward = true;
  spd_left = 80;
  spd_right = 80;

  // Initialize Serial 1 (hw-serial) and 2 (sw-serial)
  Serial.begin(115200);
  mySerial.begin(SER_BPS);

  // Read and discard garbage on serial 2
  while (mySerial.available())
  {
    data = mySerial.read();
  }
}

void loop()
{
  char data;
  uint8_t i, steps;
  bool isEndOfeyword;
  String line, command, separator, parm1, parm2;


  // read one line from mySerial
  line = readline();

  command = getValue(line,1);

  if (command == CMD_STOP)
  {
    spd_left = 0;
    spd_right = 0;
    setspeed();
  }

  if (command == CMD_FORWARD)
  {
    if (spd_forward)
    { // increase forward speed
      increase();
    }
    else
    { // go forward
      forward();
    }
  }

  if (command == CMD_BACK)
  {
    if (!spd_forward)
    { // increase backward speed
```

```
      increase();
    }
    else
    { // go backward
      backward();
    }
  }

  if (command == CMD_LEFT)
  {
    left();
  }

  if (command == CMD_RIGHT)
  {
    right();
  }

  if (command == CMD_STOP)
  {
    spd_left = 0;
    spd_right = 0;
    setspeed();
  }

  if (command == CMD_LED)
  {
    parm1   = getValue(line,2);

    if (parm1 == "on")
    { // SWITCH LED ON
      setled(255, 255, 255);
    }

    if (parm1 == "off")
    { // SWITCH LED OFF
      setled(0, 0, 0);
    }

    if (parm1 == PRM_LED_RED)
    { // SWITCH LED to red
      led_red=255;
      led_green=0;
      led_blue=0;
      setled(led_red, led_green, led_blue);
    }

    if (parm1 == PRM_LED_BLUE)
    { // SWITCH LED to blue
      led_red=0;
      led_green=0;
      led_blue=255;
      setled(led_red, led_green, led_blue);
    }

    if (parm1 == PRM_LED_GREEN)
    { // SWITCH LED to green
      led_red=0;
      led_green=255;
      led_blue=0;
      setled(led_red, led_green, led_blue);
    }
  }

  if (command == CMD_NETWORK)
  { // network
  }

  if (command == CMD_SET)
  { // SET something
  }
}
```

Appendix E - BNF

Voorbeeld van van Weaphy commando-structuur in BNF-notatie

```
<commandline> ::= <command> <eol>

<command>  ::= {
                 <get> | <set>
                 | <stop> | <start> | <move> | <led>
                 | <message>
                 | <network>
                 | <tell>
               }

# Generic commands, sending
<move>     ::= { forward | backward } <id> [<steps>]
<turn>     ::= { left | right } <id> [<degrees>]
<led>      ::= led <id> { on | off | <octet> }
<stop>     ::= stop <id>
<start>    ::= start <id>

# Generic commands, receiving
<move>     ::= { forward | backward } [<steps>]
<turn>     ::= { left | right } [<degrees>]
<led>      ::= led { on | off | <octet> }
<stop>     ::= stop
<start>    ::= start

# Commands triggered (only) by Leaphy
<get>      ::= get myid
               | get myname
               | get version
               | get network

<set>      ::= set myname <identifier>

<netid>    ::= <identifier>
<password> ::= <identifier>

<tell>     ::= tell <id> <string>

<network>  ::= network { join <netid> <password>
                         | create <netid> <password> |
                         | leave [<netid>]
                       }

# Responses received by Leaphy
<responseline> ::= { <response> | <command> } <eol>
<response>     ::= response {   myid <id>
                              | myname <identifier> [<id>]
                              | version <octet>
                              | message <quote> <string> <quote>
                              | network { created
                                          | connected
                                          | notconnected
                                          | netid <netid>
                                        }
                            }

# identifiers
<identifier> ::= <letter> [ <nospace> ]*
<id>         ::= <octet> | "all"
<octet>      ::= <digit> [ <digit> [ <digit> ]]

# Special characters
<delimeter>  ::= " " [<delimeter>]
<quote>      ::= """
<space>      ::= " "
<eol>        ::= CR [LF] | LF [CR]

# Characters

<string>     ::= { <letter> | <digit> | <space> | <char> } [ <string> ]
```

```
<letter>     ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<letter>     ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
<digit>      ::= 0|1|2|3|4|5|6|7|8|9
<special>    ::= _|!
<nospace>    ::= <letter> | <digit> | <special>
```