```
-- for ESP8266 and Lua 2016/09/11 by Jankop Factory
-- Swiss railway clock synchronized from external web NTP server
-- Version 1.1 upgrade memory management
-- For synchronization must be conected to internet
-- http://esp8266.fancon.cz/esp8266-web-swiss-clock-lua
-- tested with:
-- NodeMCU custom build by frightanic.com
-- branch: master
-- commit: 8e48483c825dea9c12b37a4db3d034fccbcba0bf
-- SSL: false
-- modules: dht,file,gpio,net,node,rtctime,sntp,tmr,wifi
-- built on: 2016-09-10 15:55
-- powered by Lua 5.1.4 on SDK 1.5.4.1(39cb9a32)
-- ***************************************************************
ssid = "********"  -- set your router SSID
pass = "********" -- set your router password
TimeZone=2        -- set your time zone
-- ***************************************************************
wifi.setmode(wifi.STATION)
wifi.sta.config(ssid, pass,1)
_,RID=node.bootreason()
myheap=0
buff1=""
srp=60     -- period of sync   [s]
HrsAngle=0 -- angle of pointer
MinAngle=0 -- angle of pointer
SecAngle=0 -- angle of pointer
h=0
n=0
s=0
local floor=math.floor
local DSEC=24*60*60 -- secs in a day
local YSEC=365*DSEC -- secs in a year
local LSEC=YSEC+DSEC    -- secs in a leap year
local FSEC=4*YSEC+DSEC  -- secs in a 4-year interval
local BASE_DOW=4    -- 1970-01-01 was a Thursday
local BASE_YEAR=1970    -- 1970 is the base year
local _days={-1, 30, 58, 89, 119, 150, 180, 211, 242, 272, 303, 333, 364}
local _lpdays={}
for i=1,2  do _lpdays[i]=_days[i]   end
for i=3,13 do _lpdays[i]=_days[i]+1 end
-- decode time from UNIX format
function gmtime(t)
    local y,j,m,d,w
    local mdays=_days
    s=t
    y=floor(s/FSEC)
    s=s-y*FSEC
    y=y*4+BASE_YEAR
    if s>=YSEC then
        y=y+1
        s=s-YSEC
        if s>=YSEC then
            y=y+1
            s=s-YSEC
            if s>=LSEC then
                y=y+1
                s=s-LSEC
            else
                mdays=_lpdays
            end
        end
    end
    j=floor(s/DSEC)
    s=s-j*DSEC
    local m=1
    while mdays[m]<j do m=m+1 end
    m=m-1
    local d=(j-mdays[m])
    w=(floor(t/DSEC)+BASE_DOW)%7
    h=floor(s/3600)
    s=s-h*3600
    n=floor(s/60)
    s=s-n*60
    collectgarbage()
end
-- get time from NTP server
function synct()
    net.dns.resolve("cz.pool.ntp.org",
        function(sk, ip)
            if (ip ~= nil)
```

```
            then
                sntp.sync(ip)
                stat="<span style=\"color: blue;\">SYNC</span>"
            else
                stat="<span style=\"color: red;\">NOSYNC</span>"
            end
        end)
end
-- web page - clock face with included parameters
local function LoadBuff()
    local buff2 = '<!doctype html><head><meta charset="UTF-8"><meta http-equiv="refresh" content="60">\
<title>Swiss Railway Clock</title></head><body style="background:white">\
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="-1100 -1100 2200 2200" width="600" height="600">\
<title>Swiss Railway Clock</title>\
<style type="text/css">\
.bg {stroke: none; fill: white;}\
.fc {stroke: none; fill: black;}\
.h1 {stroke: none; fill: black;}\
.h2 {stroke: none; fill: #aa0000;}\
</style>\
<defs>\
<path id="mark1" d="M -20,-1000 l 40,0 0,100 -40,0 z"/>\
<path id="mark2" d="M -40,-1000 l 80,0 0,240 -80,0 z"/>\
<path id="mark3" d="M -40,-1000 l 80,0 0,300 -80,0 z"/>\
<path id="handh" d="M -50,-600  l 50,-50 50,50 0,800  -100,0 z"/>\
<path id="handm" d="M -40,-850  l 40,-40 40,40 0,1080 -80,0  z"/>\
<g id="hands">\
<path d="M -10,-910 l  10,-10 10,10 2,300 -24,0 z M -13,-390 l  26,0 7,690 -40,0 z"/>\
<path d="M 0,-620 a 120,120 0 0 1 0,240 a 120,120 0 0 1 0,-240 z M 0,-560 a  60,60  0 0 0 0,120 a 60,60 0 0 0 0,-120 z"/>\
</g>\
<g id="face1">\
<use xlink:href="#mark1" transform="rotate(06)"/>\
<use xlink:href="#mark1" transform="rotate(12)"/>\
<use xlink:href="#mark1" transform="rotate(18)"/>\
<use xlink:href="#mark1" transform="rotate(24)"/>\
</g>\
<g id="face2">\
<use xlink:href="#face1"/>\
<use xlink:href="#face1" transform="rotate(30)"/>\
<use xlink:href="#face1" transform="rotate(60)"/>\
<use xlink:href="#mark3"/>\
<use xlink:href="#mark2" transform="rotate(30)"/>\
<use xlink:href="#mark2" transform="rotate(60)"/>\
</g>\
<g id="face">\
<use xlink:href="#face2"/>\
<use xlink:href="#face2" transform="rotate(90)"/>\
<use xlink:href="#face2" transform="rotate(180)"/>\
<use xlink:href="#face2" transform="rotate(270)"/>\
</g>\
</defs>\
<circle class="bg" r="1024"/>\
<circle  r="1050" fill-opacity="0" stroke="grey" stroke-width="50"/>\
<use xlink:href="#face" class="fc"/>\
<g>\
<use xlink:href="#handh" class="h1" transform="rotate('..HrsAngle..')"/>\
<animateTransform attributeName="transform" type="rotate" begin="0s" from="0,0,0" to="360,0,0" dur="12h" repeatCount="indefinite" />\
</g>\
<g>\
<use xlink:href="#handm" class="h1" transform="rotate('..MinAngle..')"/>\
<animateTransform attributeName="transform" type="rotate" begin="0s" from="0,0,0" to="360,0,0" dur="60min" repeatCount="indefinite" />\
</g>\
<g>\
<use xlink:href="#hands" class="h2" transform="rotate('..SecAngle..')"/>\
<animateTransform attributeName="transform" type="rotate" begin="0s" by="6" calcMode="discrete" dur="1s" accumulate="sum" repeatCount="indefinite"/>\
</g>\
</svg><br>\
Reset ID : '..RID..'<br> Heap : '..myheap..'<br> Time Zone : '..TimeZone..'<br> Time State : '..stat..'</body></html>'
    local lenght= #buff2
    buff1 = 'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n'..
        'Content-Length: '.. lenght ..'\r\n'..
        'Cache-Control: max-age=120\r\n'..
        'Connection: Keep-Alive\r\n\r\n'..buff2
    buff2=nil
```

```
156         collectgarbage()
157     end
158     -- web server
159     function StartServer()
160         srv = net.createServer(net.TCP, 120)
161         srv:listen(80, function (conn)
162             conn:on("receive",
163                 function (client, request)
164                     if string.find(request, "GET / HTTP/1.1") ~= nil
165                     then
166                         gmtime(rtctime.get())
167                         HrsAngle=360/12*(h+TimeZone+n/60)
168                         MinAngle=360/60*(n+s/60)
169                         SecAngle=360/60*s
170                         myheap=node.heap()
171                         LoadBuff()
172                         collectgarbage()
173                         client:send(string.sub(buff1,1, (#buff1 > 1460) and 1460 or #buff1),
174                             function()
175                                 if #buff1>1460
176                                 then
177                                     client:send(string.sub(buff1,1461,(#buff1 > 2920) and
    2920 or #buff1),
178                                         function()
179                                             if #buff1 > 2920
180                                             then
181                                                 client:send(string.sub(buff1,2921,#buff1),
182                                                     function()
183                                                     buff1=nil
184                                                     collectgarbage()
185                                                     end)
186                                             end
187                                         end)
188                                 end
189                             end)
190                         collectgarbage()
191                     else
192                         client:send('HTTP/1.1 404 Not found\r\n'..
193                             'Content-Type: text/html\r\n'..
194                             'Content-Length: 25\r\n'..
195                             'Connection: close\r\n\r\n'..
196                             '<h1>Page not found !</h1>')
197                         client:close()
198                     end
199                 end)
200         end)
201     end
202     -- main loop
203     tmr.alarm(0,1000,1,
204         function()
205             if wifi.sta.getip()==nil
206             then
207                 print("Wait for IP")
208             else
209                 ipa,_,_=wifi.sta.getip()
210                 print("IP is ",ipa)
211                 tmr.stop(0)
212                 synct()
213                 print("Server start")
214                 StartServer()
215                 -- periodic time sync
216                 tmr.alarm(1,srp*1000,1,
217                     function()
218                         synct()
219                         print("My heap :",myheap)
220                         collectgarbage()
221                     end)
222             end
223         end)
224
225
226
227
228
```