```
 1
 2  --***************** init.lua  ****************************
 3  -- Copyright 2016 jankop@volny.cz, MIT license, http://opensource.org/licenses/MIT
 4  print("** init")
 5  if rtcmem.read32(0)==2
 6  then
 7      dofile("sonoffset.lua")
 8  elseif rtcmem.read32(0)==3
 9  then
10      dofile("sonofftel.lua")
11  else
12      dofile("sonoffrun.lua")
13  end
14
15  --***************** sonoffrun.lua ***************************
16  -- Copyright 2016 jankop@volny.cz, MIT license, http://opensource.org/licenses/MIT
17  _,BoRe=node.bootreason()
18  print("** sonoffrun ",BoRe)
19  rtcmem.write32(0,0)
20  GPIO0  = 3 -- button
21  GPIO12 = 6 -- relay (active high)
22  GPIO13 = 7 -- GREEN LED (active low)
23  t1=250000
24  t2=t1
25  ot=0
26  code="sonoff"
27  mode="APO"
28  chan=6
29  on ="color: green; border: 3px #fff outset;"
30  off="color: red; border: 3px #fff outset;"
31  noact="color: black; border: 7px #fff outset;"
32  gpio.mode(GPIO0,gpio.INT)
33  gpio.mode(GPIO12,gpio.OUTPUT)
34  gpio.mode(GPIO13,gpio.OUTPUT)
35  function ron()
36      led=0
37      OnButt=on
38      OffButt=noact
39      rtcmem.write32(1,1)
40      gpio.write(GPIO13,led)
41      gpio.write(GPIO12,rtcmem.read32(1))
42  end
43  function roff()
44      led=1
45      OnButt=noact
46      OffButt=off
47      rtcmem.write32(1,0)
48      gpio.write(GPIO13,led)
49      gpio.write(GPIO12,rtcmem.read32(1))
50  end
51
52  function WaitIP()
53      wifi.sta.connect()
54      tmr.alarm(1,1000,1,
55          function()
56              if wifi.sta.getip()==nil
57              then
58                  print("Wait for Ip...")
59                  led=(led<1 and 1 or led>0 and 0)
60                  gpio.write(GPIO13,led)
61                  tmr.alarm(2,50,0,
62                      function()
63                          led=(led<1 and 1 or led>0 and 0)
64                          gpio.write(GPIO13,led)
65                      end)
66              else
67                  tmr.stop(1)
68                  print("IP is : ",wifi.sta.getip())
69              end
70          end)
71  end
72
73  if rtcmem.read32(1)==1 and BoRe~=0
74  then
75      ron()
76  else
77      roff()
78  end
79  if file.exists("code.inf")
80  then
81      file.open("code.inf","r")
```

```
 82        pcode=file.readline()
 83        pmode=file.readline()
 84        pchan=file.readline()
 85        file.close()
 86        if pcode~=nil
 87        then
 88            code=string.sub(pcode,1,#pcode-1)
 89        end
 90        if pmode~=nil
 91        then
 92            mode=string.sub(pmode,1,#pmode-1)
 93        end
 94        if pchan~=nil
 95        then
 96            chan=(string.sub(pchan,1,#pchan-1))
 97        end
 98    end
 99
100    if mode=="APO"
101    then
102        wifi.setmode(wifi.SOFTAP)
103        cfg={}
104        cfg.channel=chan
105        --cfg.auth=wifi.WPA_WPA2_PSK
106        cfg.auth=wifi.WPA_PSK
107        cfg.ssid="APO"..node.chipid()
108        cfg.pwd= "esp"..node.chipid()
109        wifi.ap.config(cfg)
110        dhcp_config ={}
111        dhcp_config.start = "192.168.4.2"
112        wifi.ap.dhcp.config(dhcp_config)
113        print(wifi.ap.dhcp.start())
114    else
115        wifi.setmode(wifi.STATION)
116        WaitIP()
117    end
118    --  start server
119    srv=net.createServer(net.TCP,60)
120    srv:listen(80,
121        function(conn)
122            at=tmr.now()
123            t1=t2
124            t2=at-ot
125            ot=at
126            if t1<300000 and t2<300000
127            then
128                return
129            end
130            conn:on("receive",
131                function(client,payload)
132                    --send data to client
133                    function Send()
134                        lenght= #buff
135                        buff =
136                            'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n'..
137                            'Content-Length: '.. lenght ..'\r\n'..
138                            'Cache-Control: no-cache\r\n'..
139                            'Pragma: no-cache\r\n'..
140                            'Connection: keep-alive\r\n\r\n'..buff
141                        client:send(string.sub(buff,1, (#buff > 1460) and 1460 or #buff),
142                            function()
143                                if #buff>1460
144                                then
145                                    client:send(string.sub(buff,1461,(#buff > 2920) and
    2920 or #buff),
146                                        function()
147                                            buff=nil
148                                            collectgarbage()
149                                        end)
150                                end
151                            end)
152                    end
153                    -- prepare code page for client
154                    function SendCode()
155                        buff=
156                            '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">\
157    <html><head>\
158    <style type="text/css">\
159    html {font-family: sans-serif;color: brown;background:#ffe4c4}\
160    .submit {width: 12%; height:5vw; font-size: 100%; font-weight: bold; color: black;
       border: 7px #fff outset; border-radius: 4vw;}\
```

```
161    .password {width: 20%; height:3vw; font-size: 100%; font-weight: bold; color: black;}\
162    @media (max-width: 1281px) {\
163    html { font-size: 3vw; font-family: sans-serif;color: brown;background:#ffe4c4}\
164    .submit {width: 40%; height:20vw; font-size: 100%; font-weight: bold; color: black;
       border: 7px #fff outset; border-radius: 15vw;}\
165    .password {width: 60%; height:10vw; font-size: 100%; font-weight: bold; color: black;}}\
166    </style>\
167    <meta content="text/html; charset=utf-8">\
168    <title>SONOFF - Factory</title></head><body>\
169    <center>\
170    <form action="/" method="post">\
171    <h1>Sonoff ID:'..node.chipid()..'<br>\
172    <input type="password" name="CODE" class="password" value=""><br><br>\
173    <input type="submit" name="SUBMIT" class="submit" value="SUBMIT"></h1>\
174    </form></center></body></html>'
175                        Send()
176                    end
177                    -- prepare main page for client
178                    function SendData()
179                        buff=
180                            '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">\
181    <html><head>\
182    <style type="text/css">\
183    html {font-family: sans-serif;color: brown;background:#ffe4c4}\
184    .submit {width: 10%; height:5vw; font-size: 100%; font-weight: bold; border-radius:
       4vw;}\
185    @media (max-width: 1281px) {\
186    html {font-size: 3vw; font-family: sans-serif;color: brown;background:#ffe4c4}\
187    .submit {width: 40%; height:20vw; font-size: 100%; font-weight: bold; border-radius:
       15vw;}}\
188    </style>\
189    <meta content="text/html; charset=utf-8">\
190    <title>SONOFF - Factory</title></head><body>\
191    <center>\
192    <form action="/" method="post">\
193    <h1>Sonoff ID:'..node.chipid()..'<br>\
194    <input type="submit" name="R1ON" class="submit" value="ON" style="'..OnButt..'">\
195    <input type="submit" name="R1OFF" class="submit" value="OFF" style="'..OffButt..'">\
196    </h1>\
197    </form>\
198    node.bootreason: '..BoRe..'<br>\
199    node.heap : '..node.heap()..'</center></body></html>'
200                        Send()
201                    end
202                    -- prepare message "FILE NOT FOUND" for client
203                    function Send404()
204                        buff=
205                            '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">\
206    <html><head>\
207    <meta content="text/html; charset=utf-8">\
208    <title>404</title></head>\
209    <body ><h1>404 - Page not found</h1><br></body></html>'
210                        Send()
211                    end
212                    -- parser of POST parameters
213                    function ParseData()
214                        function cgidecode(str)
215                            return (str:gsub('+', ' '):gsub("%%(%x%x)", function(xx) return
       string.char(tonumber(xx, 16)) end))
216                        end
217                        function parsecgi(str, keys, ignore_invalid)
218                            keyfound = {}
219                            for pair in str:gmatch"[^&]+" do
220                                key, val = pair:match"([^=]*)=(.*)"
221                                if not key then print("1.IQS") end
222                                default = keys[key]
223                                if default == nil then
224                                    if not ignore_invalid then print("2.IQS") end
225                                else
226                                    if type(default) == "table" then default[#default+1] =
       cgidecode(val)
227                                    elseif keyfound[key] then print("3.IQS")
228                                    else
229                                        keyfound[key] = true
230                                        keys[key] = cgidecode(val)
231                                    end
232                                end
233                            end
234                            return keys
235                        end
236                        str=string.match(payload,"POST / HTTP/%d%.%d\r\n.+\r\n\r\n(.+=.+)")
```

```
237                         keys={count = 5, start = 1, R1ON="", R1OFF="", CODE="", REFRESH="",
    SUBMIT=""}
238                         parsecgi(str, keys, false)
239                         str=nil
240                         if (passIP==actIP and passPort==actPort)
241                         then
242                             if keys.R1ON=="ON"
243                             then
244                                 ron()
245                             end
246                             if keys.R1OFF=="OFF"
247                             then
248                                 roff()
249                             end
250                         end
251                         if keys.SUBMIT=="SUBMIT"
252                         then
253                             if keys.CODE==code
254                             then
255                                 passIP,passPort=conn:getpeer()
256                             elseif keys.CODE==code.."res"
257                             then
258                                 srv:close()
259                                 rtcmem.write32(0,0)
260                                 dofile("RESTART!")
261                             elseif keys.CODE==code.."set"
262                             then
263                                 srv:close()
264                                 rtcmem.write32(0,2)
265                                 dofile("RESTART!")
266                             elseif keys.CODE==code.."tel"
267                             then
268                                 srv:close()
269                                 rtcmem.write32(0,3)
270                                 dofile("RESTART!")
271                             elseif keys.CODE==code.."off"
272                             then
273                                 passIP,passPort=conn:getpeer()
274                                 roff()
275                             elseif keys.CODE==code.."on"
276                             then
277                                 passIP,passPort=conn:getpeer()
278                                 ron()
279                             end
280                         end
281                         keys=nil
282                     end
283                     -- serve complet POST request with parameters
284                     actIP,actPort=conn:getpeer()
285                     if string.find(payload,"POST / HTTP/%d%.%d\r\n.+\r\n\r\n(.+=.+)")~=nil
286                     then
287                         oldIP=nil
288                         ParseData()
289                         if passIP==actIP and passPort==actPort
290                         then
291                             SendData()
292                         else
293                             SendCode()
294                         end
295                         -- serve POST request without parameters
296                     elseif string.find(payload,"POST /
    HTTP/%d%.%d\r\n.*Content%-Length:%s*%d+.*\r\n\r\n$")~=nil
297                     then
298                         DataLenght=tonumber(string.match(payload,"Content%-Length:%s*(%d+)"))
299                         oldPayload=payload
300                         oldIP=actIP
301                         oldPort=actPort
302                         -- serve separate POST parameters
303                     elseif (#payload==DataLenght) and (oldIP==actIP) and (oldPort==actPort)
304                     then
305                         payload=oldPayload..payload
306                         oldIP=nil
307                         ParseData()
308                         if (passIP==actIP and passPort==actPort)
309                         then
310                             SendData()
311                         else
312                             SendCode()
313                         end
314                         -- serve GET request
315                     elseif string.find(payload,"GET / HTTP/%d%.%d\r\n.+\r\n\r\n$")~=nil
```

```
                    then
                        oldIP=nil
                        if (passIP==actIP and passPort==actPort)
                        then
                            SendData()
                        else
                            SendCode()
                        end
                        -- serve error "FILE NOT FOUND"
                    else
                        Send404()
                    end
                    payload=nil
                end)
        end)
    --button control
    function borz()
        gpio.trig(GPIO0)
        timpres=1
        tmr.alarm(0,200,1, function()
            timpres=timpres+1
            if timpres==15
            then
                pwm.setup(GPIO13, 3, 500)
                pwm.start(GPIO13)
            end
            if timpres==30
            then
                pwm.setup(GPIO13, 10, 500)
                pwm.start(GPIO13)
            end
            if gpio.read(GPIO0)==0
            then
                return
            end
            tmr.stop(0)
            if timpres<15
            then
                gpio.trig(GPIO0, "down",function() borz() end)
                if rtcmem.read32(1)==1
                then
                    roff()
                else
                    ron()
                end
            elseif timpres<30
            then
                roff()
                srv:close()
                rtcmem.write32(0,2)--sonoff SET
                dofile("RESTART!")
            else
                roff()
                srv:close()
                rtcmem.write32(0,3)--sonoff TEL
                dofile("RESTART!")
            end
        end)
    end

    gpio.trig(GPIO0, "down",
        function() borz() end)

    tmr.alarm(3,10000,1,
        function()
            if wifi.sta.getip()==nil and mode~="APO"
            then
                WaitIP()
            end
            if node.heap()<10000
            then
                srv:close()
                rtcmem.write32(0,0)
                dofile("RESTART!")
            end
        end)

    --*************** sonoffset.lua ***************************************

    -- Copyright 2016 jankop@volny.cz, MIT license, http://opensource.org/licenses/MIT
    _,BoRe=node.bootreason()
```

```
397   print("** sonoffset ",BoRe)
398   rtcmem.write32(0,0)
399   rtcmem.write32(1,0)
400   GPIO0  = 3 -- button
401   GPIO12 = 6 -- relay (active high)
402   GPIO13 = 7 -- GREEN LED (active low)
403   code="sonoff"
404   mode="APO"
405   chan=6
406   pwm.setup(GPIO13, 3, 500)
407   pwm.start(GPIO13)
408   gpio.mode(GPIO0,gpio.INT)
409   gpio.mode(GPIO12, gpio.OUTPUT)
410   gpio.write(GPIO12,0)
411   if file.exists("code.inf")
412   then
413       file.open("code.inf","r")
414       pcode=file.readline()
415       pmode=file.readline()
416       pchan=file.readline()
417       file.close()
418       if pcode~=nil
419       then
420           code=string.sub(pcode,1,#pcode-1)
421       end
422       if pmode~=nil
423       then
424           mode=string.sub(pmode,1,#pmode-1)
425       end
426       if pchan~=nil
427       then
428           chan=(string.sub(pchan,1,#pchan-1))
429       end
430   end
431   ssid, passw = wifi.sta.getconfig()
432   wifi.setmode(wifi.STATIONAP)
433   cfg={}
434   cfg.channel=chan
435   --cfg.auth=wifi.WPA_WPA2_PSK
436   cfg.auth=wifi.WPA_PSK
437   cfg.ssid="SET"..node.chipid()
438   cfg.pwd= "esp"..node.chipid()
439   wifi.ap.config(cfg)
440   dhcp_config ={}
441   dhcp_config.start = "192.168.4.2"
442   wifi.ap.dhcp.config(dhcp_config)
443   wifi.ap.dhcp.start()
444   --start server
445   srv=net.createServer(net.TCP,60)
446   srv:listen(80,
447       function(conn)
448           conn:on("receive",
449               function(client,payload)
450                   actIP,actPort=conn:getpeer()
451                   if string.match(wifi.ap.getip(),"%d+%.%d+%.%d+%.")~=string.match(actIP,
      "%d+%.%d+%.%d+%.")
452                   then
453                       return
454                   end
455                   function Send()
456                       lenght= #buff
457                       buff =
458                           'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n'..
459                           'Content-Length: '.. lenght ..'\r\n'..
460                           'Cache-Control: no-cache\r\n'..
461                           'Pragma: no-cache\r\n'..
462                           'Connection: keep-alive\r\n\r\n'..buff
463                       client:send(string.sub(buff,1, (#buff > 1460) and 1460 or #buff),
464                           function()
465                               if #buff>1460
466                               then
467                                   client:send(string.sub(buff,1461,(#buff > 2920) and
      2920 or #buff),
468                                       function()
469                                           buff=nil
470                                           collectgarbage()
471                                       end)
472                               end
473                           end)
474                   end
475                   -- send data to client
```

```
476                     function SendData()
477                         if mode~="APO"
478                         then
479                             stasta=wifi.sta.status()
480                             staip=wifi.sta.getip()
481                             if staip==nil then staip="No IP address" end
482                             if stasta==0 then pohl="STATION IDLE"end
483                             if stasta==1 then pohl="STATION CONNECTING"end
484                             if stasta==2 then pohl="STATION WRONG PASSWORD"end
485                             if stasta==3 then pohl="STATION NO AP FOUND"end
486                             if stasta==4 then pohl="STATION CONNECT FAIL"end
487                             if stasta==5 then pohl="STATION GOT IP"end
488                             buff=
489                                 '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN">\
490     <html><head>\
491     <style type="text/css">\
492     @media (max-width: 1281px) {\
493     html{font-size:5vw;font-family:sans-serif;}\
494     .su{width:30%;height:10vw;font-size:80%;font-weight:bold;color:black;border:4px #fff
    outset;border-radius:15vw;}\
495     .ps{width:60%;height:10vw;font-size:100%;font-weight:bold;color:black;}}\
496     </style>\
497     <meta content="text/html;charset=utf-8"><title>SONOFF</title></head>\
498     <body style="color:brown;background:#ffe4c4">\
499     <h1>Sonoff ID:'..node.chipid()..'</h1>\
500     <span style="color: blue">'..pohl..'<br>\
501     IP address : '..staip..'</span>\
502     <form action="/"method="post">\
503     Set your WiFi Router parameters:<br>\
504     <input type="text" class="ps" name="SSID"value="'..ssid..'"> SSID<br>\
505     <input type="text" class="ps" name="PASSW"value="'..passw..'"> PASSWORD<br><br>\
506     Set password for relay control:<br>\
507     <input type="text"class="ps"name="CODE"value="'..code..'"> PASSCODE<br><br>\
508     Set single Access Point mode only:\
509     <input type="checkbox"name="SQUARE"value="CHECKED"><br><br>\
510     AP WiFi channel<br>\
511     <input type="text" class="ps" name="CHANNEL"value="'..chan..'"><br><br>\
512     <input type="submit"class="su"name="BUTTON"value="SUBMIT">\
513     <input type="submit"class="su"name="BUTTON"value="REFRESH IP">\
514     <input type="submit"class="su"name="BUTTON"value="EXIT"></form>\
515     node.heap : '..node.heap()..'\
516     </body></html>'
517                         else
518                             buff=
519                                 '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN">\
520     <html><head>\
521     <style type="text/css">\
522     @media (max-width: 1281px) {\
523     html{font-size:5vw;font-family:sans-serif;}\
524     .su{width:30%;height:10vw;font-size:80%;font-weight:bold;border:4px #fff
    outset;border-radius:15vw;}\
525     .ps{width:60%;height:10vw;font-size:100%;font-weight:bold;}}\
526     </style>\
527     <meta content="text/html;charset=utf-8"><title>SONOFF</title></head>\
528     <body style="color:brown;background:#ffe4c4">\
529     <h1>Sonoff ID:'..node.chipid()..'</h1>\
530     <span style="color: blue">Access point mode only<br>\
531     IP address : 192.168.4.1</span>\
532     <form action="/"method="post">\
533     Set your WiFi Router parameters:<br>\
534     <input type="text" class="ps" name="SSID" style="color: LightGrey" value="'..ssid..'">
    SSID<br>\
535     <input type="text" class="ps" name="PASSW" style="color: LightGrey" value="'..passw..
    '"> PASSWORD<br><br>\
536     Set password for relay control:<br>\
537     <input type="text"class="ps"name="CODE"value="'..code..'"> PASSCODE<br><br>\
538     Set single Access Point mode only:\
539     <input type="checkbox"name="SQUARE"value="CHECKED" checked="checked"><br><br>\
540     AP WiFi channel<br>\
541     <input type="text" class="ps" name="CHANNEL"value="'..chan..'"><br><br>\
542     <input type="submit"class="su"name="BUTTON"value="SUBMIT">\
543     <input type="submit"class="su"name="BUTTON"value="REFRESH IP">\
544     <input type="submit"class="su"name="BUTTON"value="EXIT"></form>\
545     node.heap : '..node.heap()..'\
546     </body></html>'
547                         end
548                         Send()
549                     end
550                     -- send error "FILE NOT FOUND" to client
```

```
551                     function Send404()
552                         buff=
553                             '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">\
554     <html><head>\
555     <meta content="text/html; charset=utf-8">\
556     <title>404</title></head>\
557     <body >\
558     <h1>404 - Page not found</h1><br>\
559     </body></html>'
560                         Send()
561                     end
562                     function ParseData()
563                         -- parser of POST parameters
564                         function cgidecode(str)
565                             return (str:gsub('+', ' '):gsub("%%(%x%x)", function(xx) return
        string.char(tonumber(xx, 16)) end))
566                         end
567                         function parsecgi(str, keys, ignore_invalid)
568                             keyfound = {}
569                             for pair in str:gmatch"[^&]+" do
570                                 key, val = pair:match"([^=]*)=(.*)"
571                                 if not key then print("1.IQS") end
572                                 default = keys[key]
573                                 if default == nil then
574                                     if not ignore_invalid then print("2.IQS") end
575                                 else
576                                     if type(default) == "table" then default[#default+1] =
        cgidecode(val)
577                                     elseif keyfound[key] then print("3.IQS")
578                                     else
579                                         keyfound[key] = true
580                                         keys[key] = cgidecode(val)
581                                     end
582                                 end
583                             end
584                             return keys
585                         end
586                         str=string.match(payload,"POST / HTTP/%d%.%d\r\n.+\r\n\r\n(.+=.+)")
587                         keys={count = 6, start = 1, SSID="", PASSW="", BUTTON="", CODE="",
        SQUARE="", CHANNEL=""}
588                         parsecgi(str, keys, false)
589
590                         if keys.BUTTON=="SUBMIT" and keys.SQUARE=="CHECKED"
591                         then
592                             mode="APO"
593                         end
594                         if keys.BUTTON=="SUBMIT" and keys.SQUARE~="CHECKED"
595                         then
596                             mode="STA"
597                         end
598
599                         if keys.BUTTON=="SUBMIT"
600                         then
601                             ssid=keys.SSID
602                             code=keys.CODE
603                             temp=string.match(keys.CHANNEL,"(%d+)")
604                             if temp~=nil
605                             then
606                                 temp=tonumber(temp)
607                                 if temp>0 and temp<14
608                                 then
609                                     chan=temp
610                                 end
611                             end
612                             if #keys.PASSW > 7
613                             then
614                                 passw=keys.PASSW
615                             end
616                             wifi.sta.config(ssid,passw,1)
617                         end
618
619                         if keys.BUTTON=="EXIT"
620                         then
621                             srv:close()
622                             file.open("code.inf", "w")
623                             file.writeline(code)
624                             file.writeline(mode)
625                             file.writeline(chan)
626                             file.close()
627                             dofile("RESTART!")
628                         end
```

```
629                     keys=nil
630                 end
631                 -- serve POST request with parameters
632                 if string.find(payload,"POST / HTTP/%d%.%d\r\n.+\r\n\r\n(.+=.+)")~=nil
633                 then
634                     oldIP=nil
635                     ParseData()
636                     SendData()
637                     -- serve POST request without parameters
638                 elseif string.find(payload,"POST /
    HTTP/%d%.%d\r\n.*Content%-Length:%s*%d+.*\r\n\r\n$")~=nil
639                 then
640                     DataLenght=tonumber(string.match(payload,"Content%-Length:%s*(%d+)"))
641                     oldIP=actIP
642                     oldPort=actPort
643                     oldPayload=payload
644                     -- serve GET request
645                 elseif string.find(payload,"GET / HTTP/%d%.%d\r\n.+\r\n\r\n$")~=nil
646                 then
647                     oldIP=nil
648                     SendData()
649                     -- serve separate POST parameters
650                 elseif (#payload==DataLenght) and (oldIP==actIP) and (oldPort==actPort)
651                 then
652                     payload=oldPayload..payload
653                     oldIP=nil
654                     ParseData()
655                     SendData()
656                     -- serve error "FILE NOT FOUND"
657                 else
658                     Send404()
659                 end
660                 payload=nil
661                 collectgarbage()
662             end)
663     end)
664 --button control
665 function borz()
666     gpio.trig(GPIO0)
667     timpres=1
668     tmr.alarm(0,100,1, function()
669         timpres=timpres+1
670         if gpio.read(GPIO0)==0
671         then
672             return
673         end
674         tmr.stop(0)
675         if timpres > 1
676         then
677             srv:close()
678             dofile("RESTART!")
679         else
680             gpio.trig(GPIO0,"down",function() borz() end)
681         end
682     end)
683 end
684 gpio.trig(GPIO0,"down",function() borz() end)
685
686
687 --****************** sonofftel.lua **********************
688     --- Copyright 2016 jankop@volny.cz, MIT license, http://opensource.org/licenses/MIT
689 _,BoRe=node.bootreason()
690 print("** sonofftel ",BoRe)
691 rtcmem.write32(0,0)
692 rtcmem.write32(1,0)
693 GPIO0  = 3 -- button
694 GPIO12 = 6 -- relay (active high)
695 GPIO13 = 7 -- GREEN LED (active low)
696 chan=6
697 pwm.setup(GPIO13, 10, 500)
698 pwm.start(GPIO13)
699 gpio.mode(GPIO0,gpio.INT)
700 gpio.mode(GPIO12, gpio.OUTPUT)
701 gpio.write(GPIO12,0)
702 if file.exists("code.inf")
703 then
704     file.open("code.inf","r")
705     temp=file.readline()
706     temp=file.readline()
707     pchan=file.readline()
708     file.close()
```

```
709         if pchan~=nil
710         then
711             chan=(string.sub(pchan,1,#pchan-1))
712         end
713     end
714     wifi.setmode(wifi.SOFTAP)
715     cfg={}
716     cfg.channel=chan
717     --cfg.auth=wifi.WPA_WPA2_PSK
718     cfg.auth=wifi.WPA_PSK
719     cfg.ssid="TEL"..node.chipid()
720     cfg.pwd= "esp"..node.chipid()
721     wifi.ap.config(cfg)
722     dhcp_config ={}
723     dhcp_config.start = "192.168.4.2"
724     wifi.ap.dhcp.config(dhcp_config)
725     wifi.ap.dhcp.start()
726     telnet_srv = net.createServer(net.TCP, 180)
727     telnet_srv:listen(2323, function(socket)
728         local fifo = {}
729         local fifo_drained = true
730         local function sender(c)
731             if #fifo > 0 then
732                 c:send(table.remove(fifo, 1))
733             else
734                 fifo_drained = true
735             end
736         end
737         local function s_output(str)
738             table.insert(fifo, str)
739             if socket ~= nil and fifo_drained then
740                 fifo_drained = false
741                 sender(socket)
742             end
743         end
744         node.output(s_output, 0)
745         socket:on("receive", function(c, l)
746             node.input(l)
747         end)
748         socket:on("disconnection", function(c)
749             node.output(nil)
750         end)
751         socket:on("sent", sender)
752         print("Welcome to Espresso.")
753         print("node.chipid() : ",node.chipid())
754     end)
755     --button control
756     function borz()
757         gpio.trig(GPIO0)
758         timpres=1
759         tmr.alarm(0,100,1, function()
760             timpres=timpres+1
761             if gpio.read(GPIO0)==0
762             then
763                 return
764             end
765             tmr.stop(0)
766             if timpres > 1
767             then
768                 telnet_srv:close()
769                 dofile("RESTART!")
770             else
771                 gpio.trig(GPIO0,"down",function() borz() end)
772             end
773         end)
774     end
775     gpio.trig(GPIO0,"down",function() borz() end)
776
```