**CODE**

This code folder contains code blocks for the v1.1.0, i.e it has the feature of a WIFI manager and it posts data in every 1 minutes.

**CODE TESTING AND RESULTS**

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | TEST | BEHAVIOUR | RESULTS |
| 1 | Powering up the device for the first time | The device started the captive portal (wifi manager) and a wifi credentials was imputed | PASSED. This behavior is expected. |
| 2 | The device was disconnected from supply and plugged again to test if it will connect to the previously configured router | On power on, the device didn’t start the captive portal but connected to the previously configured router | PASSED. This behavior is expected. |
| 3 | While the device was connected to a router, the router was powered-down but the device was left with power supply | After about a minute while the router was not available, the device started its captive portal (wifi manager) so that the user can input another wifi credentials | PASSED. This behavior is expected. |
| 4 | After inputting another wifi credential,the device was disconnected from power source and connected again. Both configured Routers were made available to check which one the device will connect to. | The device connected to the second router because it was the last configured router | PASSED. This behavior is expected. |
| 5 | The second configured router was then switched off to see if the device will connect to the first configured router | The device tried to connect to the last configured router but when that failed, it started the captive portal and didn’t connect to the first configured router. | FAILED. The end-user module used for the wifi manager can only store a wifi configuration at a time on the device flash. |

**INIT.LUA**

-- otherwise, start up

print('Running main.lc in 5 seconds')

-- dofile('main.lua')

tmr.alarm(0, 5000, tmr.ALARM\_SINGLE, function() dofile("main.lc") end)

**NTP-CLOCK.LUA**

function do\_clock\_sync ()

sntp.sync(ntpserver,

function(sec, usec, server, info)

print('Synced to epoch', sec, 'from server', server)

local tm = rtctime.epoch2cal(rtctime.get())

print(string.format("Now: %04d/%02d/%02d %02d:%02d:%02d", tm["year"], tm["mon"], tm["day"], tm["hour"]+1, tm["min"], tm["sec"]))

end,

function()

print('Clock sync failed, retrying!')

tmr.create():alarm(1000, tmr.ALARM\_SINGLE, function()

do\_clock\_sync()

end)

end

)

end

return { sync = do\_clock\_sync }

**PERIODIC-WORK.LUA**

local dht\_pin = 1

local function readDHT()

local status, temp, humi, temp\_dec, humi\_dec = dht.read(dht\_pin)

if status == dht.OK then

return temp, humi

else

return -1000, -1000

end

end

function periodic\_measurement()

local time = rtctime.get()

local temp, hum = readDHT()

print("Meas: T=" .. temp .. ", RH=" .. hum)

collectgarbage()

local post = require("client\_post")

local json\_t = post.create\_json(temp, "C", "temperature", time, lat, lon)

local json\_h = post.create\_json(hum, "%", "humidity", time, lat, lon)

local send\_table = {{url\_t, json\_t},{url\_h, json\_h}}

post.post\_json(server, send\_table)

end

return { periodic = periodic\_measurement }

**MAIN.LUA**

-- local config has priority over generic

if file.exists("config.lua.local")

then dofile("config.lua.local")

else dofile("config.lua")

end

dofile("wifi\_connect.lc")

-- -- every 1 minutes

cron.schedule("\* \* \* \* \*", function(e)

print("Cron running measurement.")

local run = require("periodic\_work")

run.periodic()

end)

-- every day @ 2:22

cron.schedule("22 2 \* \* \*", function(e)

print("Cron clock sync.")

local clock = require("ntp-clock")

clock.sync()

end)

**CONFIG.LUA**

--- SERVER AND API DEFINITIONS ---

server = "www.terasyshub.io"

port = 443

url\_t = "/api/v1/data"

url\_h = "/api/v1/data"

url\_cred = "/api/v1/keys"

ntpserver = "pool.ntp.org"

--- LOCATION ---

lat = 6.497492

lon = 3.382360

--- DEVICE KEY ---

mykey = "98f4f6409ae547223e4c"

**WIFI\_CONNECT.LUA**

--- Connect to the wifi network ---

print("Connecting to WiFi access point...")

------------------------------------------------

--this is where the captive portal

-------------------------------------------

wifi.sta.disconnect()

wifi.setmode(wifi.STATIONAP)

-----START THE END USER MODULE-----------------------

enduser\_setup.start(

function()

print("Connected" )

end,

function(err, str)

print("enduser\_setup: Err #" .. err .. ": " .. str)

end

);

-------------------------------------------

---the end of the captive portal

-------------------------------------------

wifi.sta.connect()

tmr.create():alarm(2000, tmr.ALARM\_AUTO, function(cb\_timer)

if wifi.sta.getip() == nil then

print("Waiting for IP address...")

else

cb\_timer:unregister()

print("WiFi connection established " )

tmr.create():alarm(1, tmr.ALARM\_SINGLE, function()

local clock = require("ntp-clock")

clock.sync()

end)

end

end)

tmr.create():alarm(10000, tmr.ALARM\_AUTO, function(cb\_timer)

if mykey == nil and wifi.sta.getip() ~= nil and rtctime.get() > 1000000 then

print("Acquiring auth keys.")

local cred = require("client\_credentials")

cred.acquire()

elseif mykey ~= nil then

cb\_timer:unregister()

end

end)

**CLIENT\_POST.LUA**

-- \_G.cjson = sjson

function create\_json (value, unit, type, time, lat, lon)

local data = {}

data.mac = wifi.sta.getmac()

data.location = {lat=lat, lon=lon}

data.timestamp = time

data.metrics = {

{ value = value, unit = unit, type = type }

}

data.key = mykey

return(sjson.encode(data))

end

local function prepare\_post (server, url, json\_s)

local json\_s\_length = string.len(json\_s)

local post\_s = "POST " .. url .. " HTTP/1.1\r\n"

.. "Host: " .. server .. "\r\n"

.. "Accept: \*/\*\r\n"

.. "User-Agent: Mozilla/4.0 (compatible; nodemcu esp8266 Lua;)\r\n"

.. "Connection: keep-alive\r\n"

.. "Content-Length: "..json\_s\_length.."\r\n"

.. "Content-Type: application/json\r\n\r\n"

.. json\_s

return(post\_s)

end

local function parse\_response(response)

local x=response:find("\n")

local code, text = string.match(response:sub(10,x-1), "(%d+) (%a+)")

x=response:find("\r\n\r\n")

local key=response:sub(x+4)

return code, text, key

end

local sk = nil

function post\_json (server, send\_table)

if mykey == nil then

print("No API key, will not send.")

return

end

if sk == nil then sk = tls.createConnection(net.TCP,1) end

sk:on("connection", function(conn)

print("--connected")

local url = send\_table[1][1]

local json= send\_table[1][2]

local post\_s = prepare\_post(server, url, json)

print("--sending: " .. json)

conn:send(post\_s)

end )

sk:on("sent", function(conn, c)

print("--sent")

end)

sk:on("receive", function(conn, c)

local code, text, msg = parse\_response(c)

print("--received: " .. code .. ", " .. text .. ", message: " .. msg)

table.remove(send\_table, 1)

if table.getn(send\_table) > 0 then

local url = send\_table[1][1]

local json= send\_table[1][2]

local post\_s = prepare\_post(server, url, json)

print("--sending: " .. json)

conn:send(post\_s)

else

conn:close()

-- -- Enable deep sleep in μs

-- print("Data logger going to deep sleep now")

-- node.dsleep(300000000)

end

end)

sk:on("disconnection", function(conn) print("--disconnected") end )

sk:connect(port,server)

end

return { create\_json = create\_json, post\_json = post\_json}

**CLIENT\_CREDENTIALS.LUA**

-- \_G.cjson = sjson

local function create\_json\_cred ()

local data = {}

data.mac = wifi.sta.getmac()

return(sjson.encode(data))

end

local function prepare\_post (server, url, json\_s)

local json\_s\_length = string.len(json\_s)

local post\_s = "POST " .. url .. " HTTP/1.1\r\n"

.. "Host: " .. server .. "\r\n"

.. "Accept: \*/\*\r\n"

.. "User-Agent: Mozilla/4.0 (compatible; nodemcu esp8266 Lua;)\r\n"

.. "Connection: close\r\n"

.. "Content-Length: "..json\_s\_length.."\r\n"

.. "Content-Type: application/json\r\n\r\n"

.. json\_s

return(post\_s)

end

local function parse\_response(response)

local x=response:find("\n")

local code, text = string.match(response:sub(10,x-1), "(%d+) (%a+)")

x=response:find("\r\n\r\n")

local key=response:sub(x+4)

return code, text, key

end

local function post\_json\_cred (server, url, json\_s)

local sk = tls.createConnection(net.TCP,1)

sk:on("connection", function(conn)

print("--connected")

local post\_s = prepare\_post(server, url, json\_s)

print("--sending credentials: " .. json\_s)

conn:send(post\_s)

end )

sk:on("sent", function(conn, c)

print("--sent")

end)

sk:on("receive", function(conn, c)

local code, text, key = parse\_response(c)

print("--received: " .. code .. ", " .. text .. ", auth key: " .. key)

mykey=key

conn:close()

end)

sk:on("disconnection", function(conn) print("--disconnected") end )

sk:connect(port,server)

end

function acquire\_credentials()

local json\_cred = create\_json\_cred()

post\_json\_cred(server, url\_cred, json\_cred)

end

return { acquire = acquire\_credentials }

**COMPILE.LUA**

node.compile("wifi\_connect.lua")

node.compile("ntp-clock.lua")

node.compile("client\_post.lua")

node.compile("client\_credentials.lua")

node.compile("main.lua")

node.compile("periodic\_work.lua")