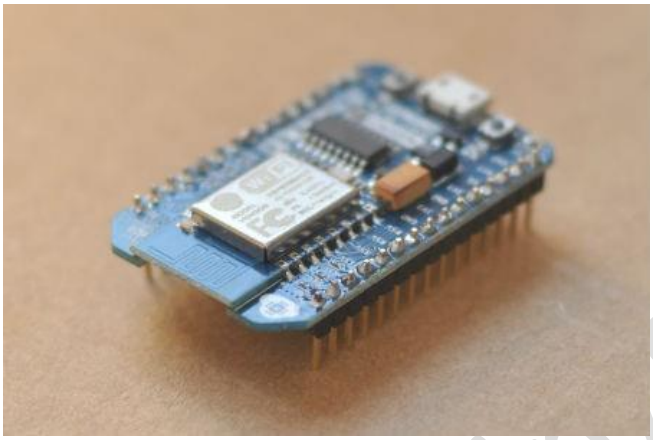
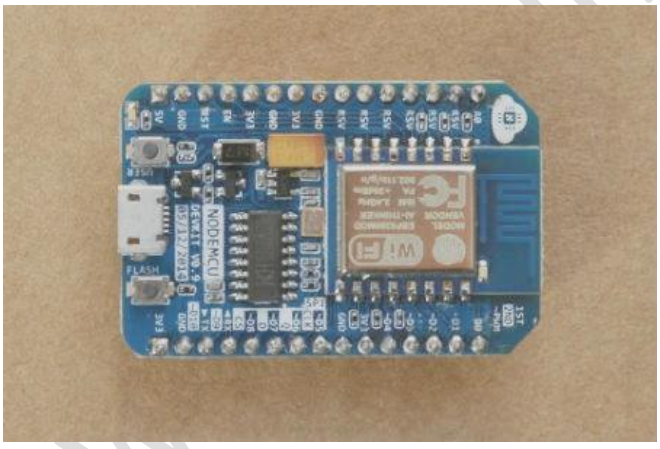
NodeMCUESP8266 Module with Wi-Fi Code: NODEMCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the dev kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson, and spiffs.





Spec:

NodeMCU IoT Module NodeMCU is a IoT Module based on ESP8266 wifi Module. NodeMCU uses Lua Scripting language and is an open source Internet of Things (IoT) platform.Specification of Node-MCU IoT Module

- The Development Kit based on ESP8266, integates GPIO, PWM, IIC, 1-Wire and ADC all in one board.
- Power your developement in the fastest way combinating with NodeMCU Firmware!
- USB-TTL included, plug&play
- 10 GPIO, every GPIO can be PWM, I2C, 1-wire
- PCB antenna

Features of Node-MCU IoT Module

- Open source IoT Platform
- Easily Programmable
- Low cost & Simple to Implement
- WI-FI enabled

Product Details:

Antenna	Internal PCB
GPIO	10 GPIO, every GPIO can be PWM, I2C, 1-wire
Input Voltage	5V DC
Operating Votage	3.3.V DC
Output Power	+19.5dBm in 802.11b mode
Programming Chip	CP2102 Serial / USB Chip
USB	Micro USB port for power, programming and debugging
Weight	Approx. 12g
WiFi type	11 b/g/n Wi-Fi Direct (P2P), soft-AP

Sample code:

Connect to an AP

```
print(wifi.sta.getip())
--nil
wifi.setmode(wifi.STATION)
wifi.sta.config{ssid="SSID",pwd="password"}
-- for older versions of the firmware wifi.sta.config("SSID","password")
```

```
-- wifi.sta.connect() not necessary because wifi.sta.config sets auto-connect
= true
tmr.create():alarm(1000, 1, function(cb_timer)
  if wifi.sta.getip() == nil then
    print("Connecting...")
  else
    cb_timer:unregister()
    print("Connected, IP is "..wifi.sta.getip())
  end
end)
```

Control GPIO

```
ledPin = 1
swPin = 2
gpio.mode(ledPin,gpio.OUTPUT)
gpio.write(ledPin,gpio.HIGH)
gpio.mode(swPin,gpio.INPUT)
print(gpio.read(swPin))
```

HTTP request

```
-- A simple HTTP client
conn = net.createConnection(net.TCP, 0)
conn:on("receive", function(sck, payload) print(payload) end)
conn:on("connection", function(sck)
  sck:send("GET / HTTP/1.1\r\nHost: nodemcu.com\r\n"
           .. "Connection: keep-alive\r\nAccept: */*\r\n\r\n")
end)
conn:connect(80, "nodemcu.com")
```

Doing something similar using the HTTP module:

```
http.get("http://nodemcu.com", nil, function(code, data)
    if (code < 0) then
      print("HTTP request failed")
    else
      print(code, data)
    end
  end)
```

HTTP server

```
-- a simple HTTP server
srv = net.createServer(net.TCP)
srv:listen(80, function(conn)
    conn:on("receive", function(sck, payload)
        print(payload)
        sck:send("HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n<h1>
Hello, NodeMCU.</h1>")
    end)
    conn:on("sent", function(sck) sck:close() end)
end)
```

Connect to MQTT Broker

```
-- init mqtt client with keepalive timer 120sec
```

```
m = mqtt.Client("clientid", 120, "user", "password")

-- setup Last Will and Testament (optional)
-- Broker will publish a message with qos = 0, retain = 0, data = "offline"
-- to topic "/lwt" if client don't send keepalive packet
m:lwt("/lwt", "offline", 0, 0)

m:on("connect", function(con) print ("connected") end)
m:on("offline", function(con) print ("offline") end)

-- on publish message receive event
m:on("message", function(conn, topic, data)
  print(topic .. ":" )
  if data ~= nil then
    print(data)
  end
end)

-- for secure: m:connect("192.168.11.118", 1880, 1)
m:connect("192.168.11.118", 1880, 0, function(conn) print("connected") end)

-- subscribe topic with qos = 0
m:subscribe("/topic",0, function(conn) print("subscribe success") end)
-- or subscribe multiple topic (topic/0, qos = 0; topic/1, qos = 1; topic2 ,
qos = 2)
-- m:subscribe({["topic/0"]=0,["topic/1"]=1,topic2=2}, function(conn)
print("subscribe success") end)
-- publish a message with data = hello, QoS = 0, retain = 0
m:publish("/topic","hello",0,0, function(conn) print("sent") end)

m:close();
-- you can call m:connect again
```

UDP client and server

```
-- a udp server
s=net.createServer(net.UDP)
s:on("receive",function(s,c) print(c) end)
s:listen(5683)

-- a udp client
cu=net.createConnection(net.UDP)
cu:on("receive",function(cu,c) print(c) end)
cu:connect(5683,"192.168.18.101")
cu:send("hello")
```