Doctors of Intelligence &Technology(DOIT)

LuaNode ESP32 Document

（文档版本：V1.0 软件版本：V 1.0 硬件版本：V 1.0）

2016-11-28

1. Introduction

LuaNode is a SDK for ESP32 dev-kit. The hardware ESP32 dev-kit is provided by DOIT.

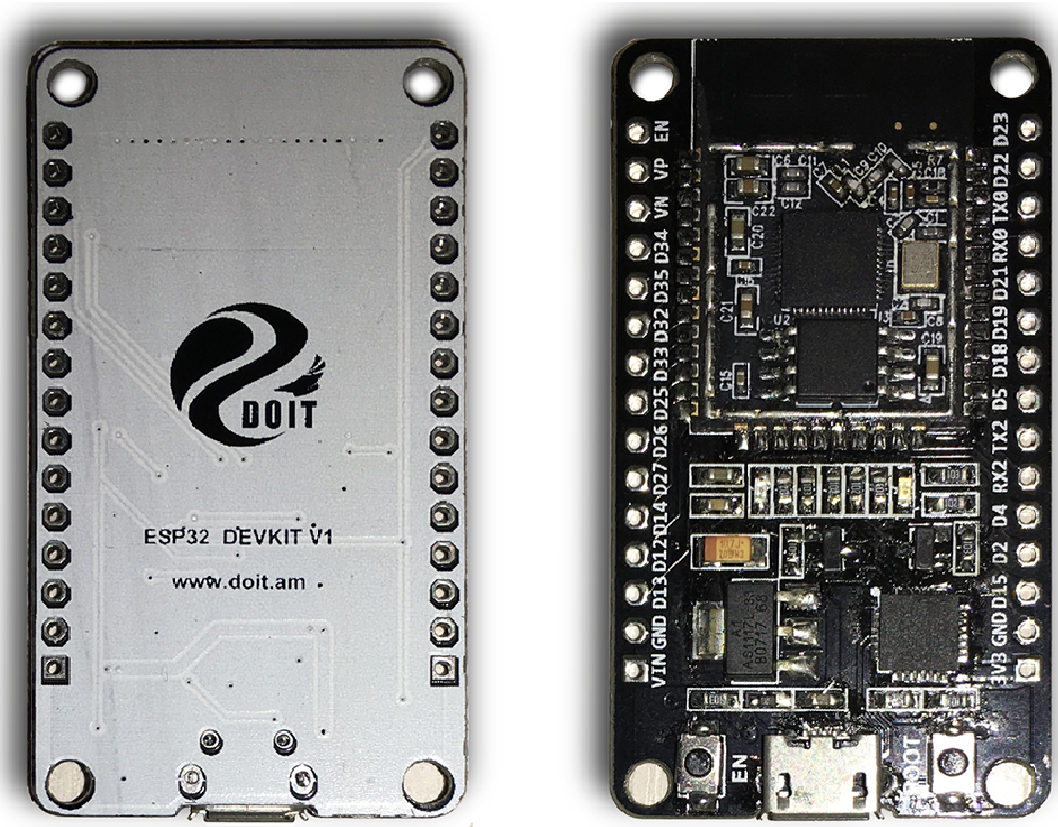

Fig 1.1 ESP32 dev-kit

2. Prepare

2.1 Download *LuaNode* and toolchains

You can download *LuaNode* on Linux by the following command:
git clone https://github.com/Nicholas3388/LuaNode.git

Make sure you have *git* installed, if not, install it by the command: *sudo apt install git*.

Download toolchains by: git clone https://github.com/jmattsson/nodemcu-prebuilt-toolchains.git

2.2 Download other usefull tools

ESPlorer: is a Integrated Development Environment (IDE) for ESP8266 developers. *LuaNode* is compatible for *Esplorer.* Developers can use it to run and test lua program, as well as download lua code to ESP32.

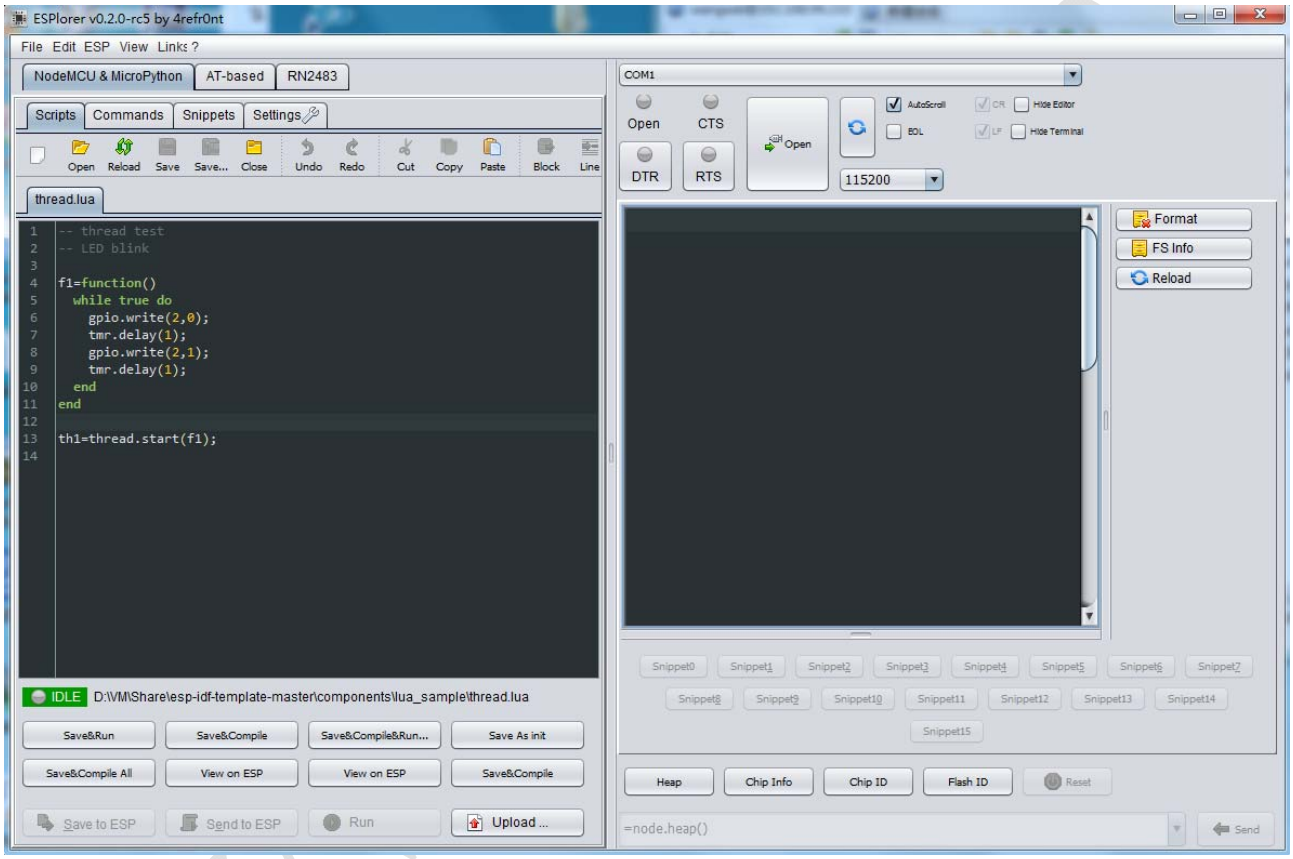

Fig 2.1 ESPlorer

Download ESPlorer from the link: https://github.com/4refr0nt/ESPlorer

esptool.py (Linux): the flash tool to flash firmware in Linux. No need to download, It comes with esp-idf, contains in LuaNode.

Espressif flashtool (Windows): to flash the firmware, you'll need a tool. If you are working with Windows, you can use the official flashtool and download it from the following link.

https://www.espressif.com/en/support/download/other-tools?keys=&field_type_tid%5B%5D=13

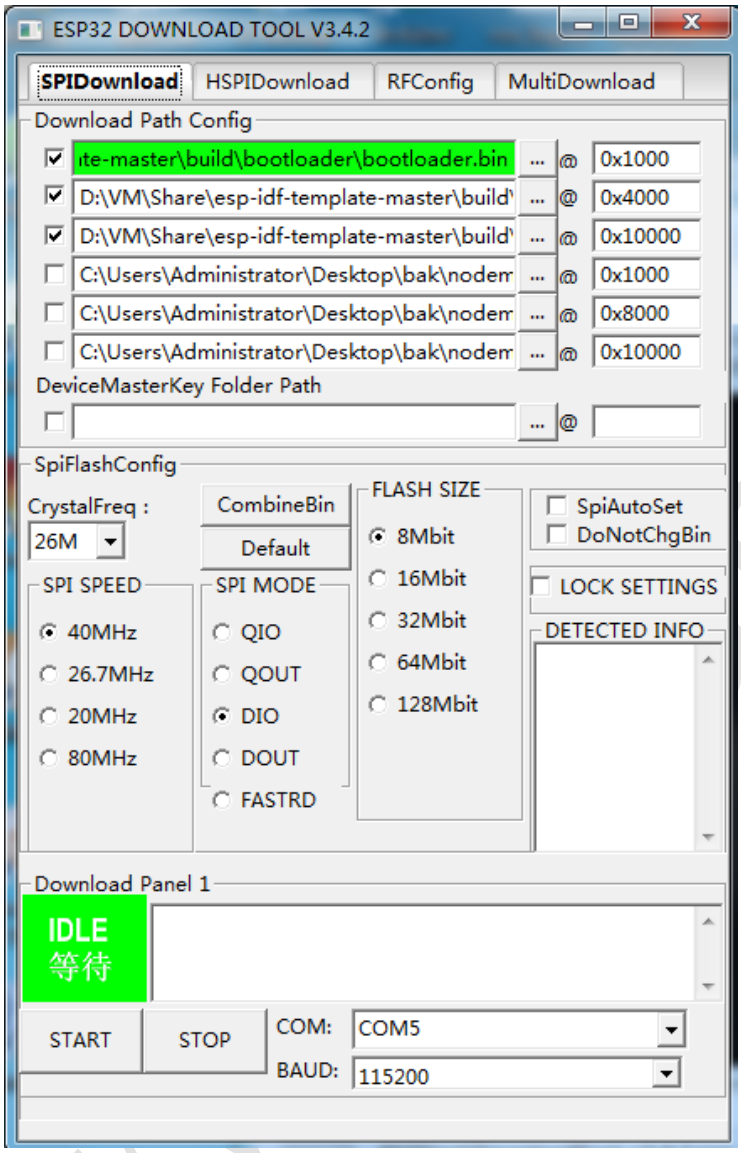

Fig 2.2 ESP32 Flash download tool If you are working with Linux, just use the *esptool.py* to flash the firmware. You can find it in *esp-idf,* contain in LuaNode.

3. Build LuaNode (ESP32)

Change your directory to LuaNode32

* Export ESP32 toolchains by running the following command in Linux terminal, export PATH=/your_path/toolchains/esp32/bin:$PATH

* Export IDF_PATH by running the command, export IDF_PATH=/your_path/LuaNode/LuaNode_Esp32/esp-idf/, where your_path is the path your save LuaNode.

* Change directory to LuaNode32, and then compile the project, run make

The firmware is generated in the *build* directory. The three bins file to be flashed is: *partition-luanode.bin*, *bootload.bin*, *LuaNode.bin*, respectively.

4. Flash firmware

4.1 Linux

The driver cp210x should be installed for Ubuntu (I test on Ubuntu 16.04), so you don't have to install it again. When ESP32 dev-board plug in, the computer can auto detect the device, and you can found the device name */dev/ttyUSB0.*

Make sure you have permission to write the device, you can change the /dev/ttyUSB0 permissions by: *sudo chmod 777 /dev/ttyUSB0*

To flash the firmware, change directory to LuaNode32, where contains a Makefile. Then run *make flash* if there is nothing wrong, the firmware will be start flashing.

4.2 Windows

In Windows, you can install VirtualBox, and then install Linux OS on VirtualBox to build firmware as we said in 4.1 section. Then flash the bin files using official windows download tool. The flash address and other settings are shown as Fig 2.2

Note that after you click *start* button on the download tool, you'll have to click the *EN* button on the ESP32 dev-kit to start flashing! Otherwise, the flashing progress won't start. But in Linux you don't have to press the EN button.

5. Development

5.1 Lua code download

* Run Esplorer as Fig 2.1. Device com port is auto detected, so select the right port to open. The baud rate is 115200.

* Click *open* button on the left to select a Lua file. Also, you can copy Lua code and paste to the

edit area and save it as a new file.

* Finally, click the *Save to ESP* button on the left bottom to download the code to device.

5.2 Lua development

In this section, we introduce how to use lua module to develop application. There are several lua module provided by LuaNode. They are auto loaded when power up.

5.2.1 gpio

It is convenient to set pin as INPUT, OUTPUT mode via gpio module. There is a blue LED connect with pin 2. You can setup pin as OUTPUT mode to turn on/off the LED to test lua gpio module. When the pin 2 output high level, the LED will be turned on, otherwise, turned off.

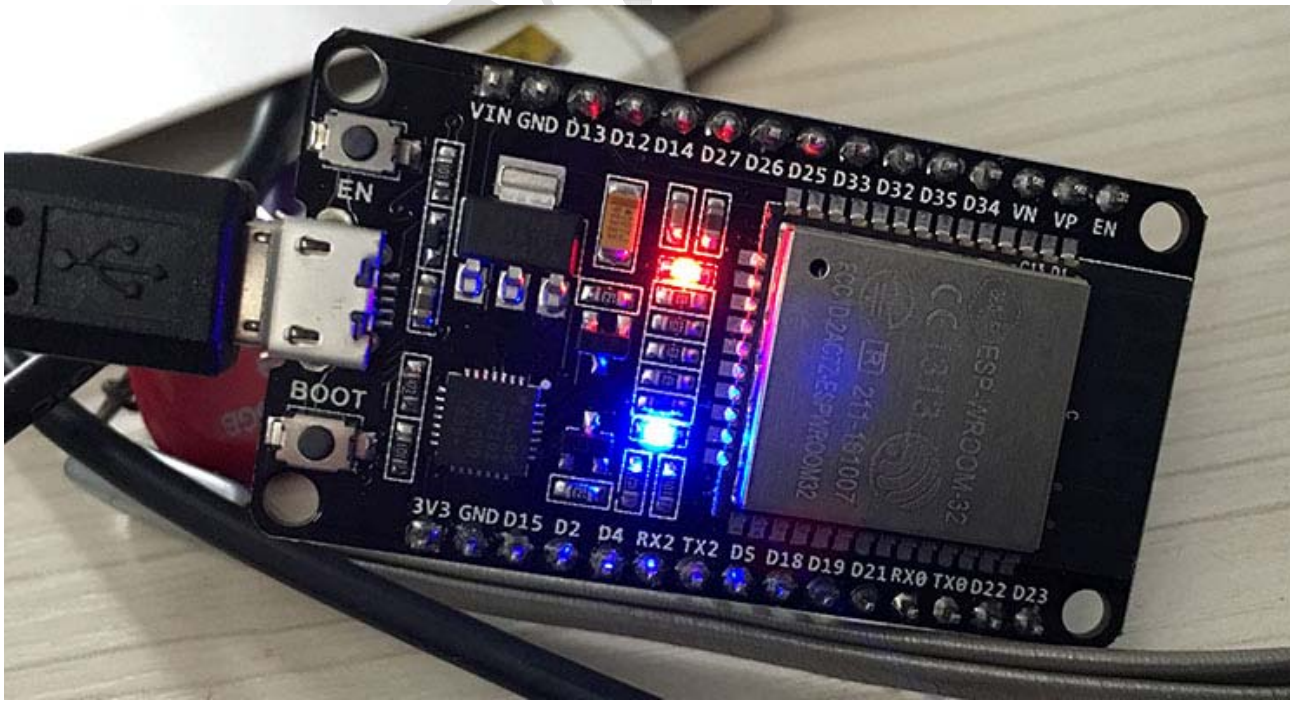

Fig 5.1 LED turn on/off

```
isOff = false

led_blink = function()
   if(isOff) then
      gpio.write(2, 1);      -- turn on
   else
      gpio.write(2, 0);      -- turn off
   end
end

gpio.mode(2,gpio.OUTPUT);   -- set pin mode as output
                                                sample 1.1

period = 1000;
--    repeat function led_blink each second
tmr.register(1, period, tmr.ALARM_AUTO, led_blink);
tmr.start(1);
```

gpio.mode(pin, mode): set pin mode

params: nil

return: heap size

5.2.2 node

node module provide some method to get device information.

Node.heap(): get device heap info

params: nil

return: heap size

node.flashsize(): get device flash size

params: nil

return: flash size

node.chipid(); get chip ID

params: nil

return: chip ID

node.restart(): restart device

params: nil

return: nil

5.2.3 tmr

tmr is a timer module, which provides timer function. To delay some time, use methods such as,

tmr.delay(s): to delay seconds

params: seconds to delay

return: nil

tmr.delay_ms(ms): delay milli-second

params: msec to delay

return: nil

tmr.delay_us(us): delay micro second

params: micro seconds to delay

return: nil

tmr.register(id, ms, mode, callback): register a callback to a timer

params:

id: timer id

ms: msec to delay

mode: timer mode, can be tmr.ALARM_SINGLE/tmr.ALARM_AUTO

callback: callback when times up

return: nil

tmr.unregister(id): unregister callback

params: timer id

return: nil

A sample to use timer module is show as sample 1.1

5.2.4 file

File is a module provided to access SPIFFS via Lua.

file.open(name, mode): open a file

params:

name: file name

mode: open mode, can be 'r'(read only), 'w'(write only),

'w+'(write append),'r+'(write read)

return: true if open successfully, otherwise, false

file.close(): close an open file, should open a file first

params: nil

return: nil

file.write(content): write content to an open file

params: content

return: return true if write successfully, otherwise, return nil

file.read([len]): read len (read all content if no params) characters from an open file

params: len to read, read all content if no params, the parameter is optional

return: nil

file.writeline(content): write a line of content to an open file

params: content

return: return true if write successfully, otherwise, return nil

file.readline(): read a line from an open file

params: nil

return: read content

file.fsinfo(): print FS information

params: nil

return:

remain: remaining capacity

used: used capacity

total: total FS capacity

example:

```
r,u,t = file.fsinfo();
```

file.remove(name): remove a file from FS

params: name of file to be removed

return: nil

file.list(): list content stored in FS

params: nil

return: content stored in FS

example:

```
l = file.list();
for k,v in pairs(l) do
```

```
        print(k, v);
    end
```

file.format(): format file system, remove all content stored in FS

params: nil

return: nil

5.2.5 wifi

wifi is a module provided to access wifi function.

Wifi.setmode(mode): set wifi mode

params: mode, can be wifi.STATION, wifi.SOFTAP (default), wifi.STATIONAP, wifi.NULLMODE

return: nil

wifi.start(): wifi start working

params: nil

return: nil

example:

```
wifi.setmode(wifi.SOFTAP);
```

```
wifi.start();
```

wifi.stop(): stop wifi working

params: nil

return: nil

wifi.sta.config({ssid,pwd,[auto]}): set ssid for wifi station

params:

ssid: ssid to connect

pwd: password

auto: auto connect, optional parameter

return: nil if auto=false, otherwise, print connection information

wifi.sta.connect(): connection to ssid

params: nil

return: nil

example:

```
wifi.setmode(wifi.STATION);
wifi.start();
wifi.sta.config({ssid="ssid", pwd="passwd",auto=false});
wifi.sta.connect();
```

wifi.sta.disconnect(): stop connecting to ssid

params: nil

return: nil

wifi.sta.getconfig(): get station config information

params: nil

return: return the config information

5.2.6 net

net is a module provided for access network

net.createConnection(type, secure): create a net connection

params:

type: connection type, can be net.TCP, net.UDP

secure: 1 for encrypted, 0 for plain

return: a net.socket module

net.socket:on(event, callback): register callback function for specific events. The events including: connection, sent, receive, disconnection.

Params:

event: event to monitor, The events including: connection, sent, receive, disconnection

callback: function when the specific event occur

return: nil

example:

```
sv = net.createConnection(net.TCP, 0);
sv:on("receive", function(sock, c) print(c) end);
sv:connect(80, "192.168.1.100");
```

net.socket:send(str): send content via socket

params: string to be sent

return: nil

net.createServer(type, timeout): create a net server

params:

type: net.TCP or net.UDP

timeout: TCP server timeout

return: net.server module

net.server:listen(port, [ip], function(net.socket)): listen on port from IP

params:

port: listen port

ip: optional, ip address, if omit, listen localhost

function(net.socket): callback when someone connect to server, the callback function parameter is a net.socket.

Net.server:on(): UDP server only, register callback for specific events

net.server:send(): UDP server only, send data

5.2.7 thread

thread is a module provided for create thread in lua

thread.start(func): start running a thread

params: thread body to run

return: thread id

example:

```
func = function()
    print("thread start");
end
```

```
th1 = thread.start(func);
```

thread.stop(th): stop running a thread

params: the only one parameter is the thread id to be stopped

return: nil

example:

```
func = function()
    while true do
        tmr.delay(1);
        print("thread start")
    end
end
th1 = thread.start(func);
thread.stop(th1);
```

thread.status(th): get status of a thread

params: the thread to be checked

return: status of the thread

example:

```
func = function()
    while true do
        tmr.delay(1);
```

```
        print("thread start")
    end
end
th1 = thread.start(func);
stat = thread.status(th1);
print(stat);
```

thread.suspend(th): suspend a running thread (this method still have bug)

params: the thread to be suspended

return: nil

5.2.8 lpeg

LPeg is a new pattern-matching library for Lua, based on Parsing Expression Grammars (PEGs). For more details of Lpeg, visit the homepage: http://www.inf.puc-rio.br/~roberto/lpeg/

The *lpeg* module provided method to access lua Lpeg.

Lpeg.match(pattern, str): match a patten against a string

params: the pattern and the string

return: nil if nothing match, else return the index of match

example:

```
lpeg.match(lpeg.P'a', 'aaa');
```

lpeg.P: match a string literally

lpeg.S: match anything in a set

lpeg.R: match anything in a range

5.2.9 utils

utils is a module provide some useful methods, such as base64 encode. More useful methods will be added to this module in the future.

Utils.base64_encode(str): encode a string with base64

params: the string to be encoded

return: encoded string

example:

```
enc = utils.base64_encode("hello");
print(enc);
```

utils.base64_decode(str): decode a string encoded by base64

params: the string to be decoded

return: decoded string

example:

```
enc = utils.base64_encode("hello");
print(enc);
dec = utils.base64_decode(enc);
print(dec);
```

6. Acknowledgement and References

NodeMCU: https://github.com/nodemcu/nodemcu-firmware

esp-idf: https://github.com/espressif/esp-idf

Lua-RTOS-ESP32: https://github.com/whitecatboard/Lua-RTOS-ESP32

esp32-mqtt: https://github.com/tuanpmt/esp32-mqtt

Lpeg: http://www.inf.puc-rio.br/~roberto/lpeg/

DOIT: http://www.doit.am