Preface –Getting Started..1
Part1 Modules Introduction..4
1. Crowtail- ESP8266 NodeMCU..4
2. Crowtail- LED..4
3.Crowtail- Button..4
4.Crowtail- Buzzer..4
5. Crowtail- Relay..5
6.Crowtail- Moisture Sensor..5
7. Crowtail- Temperature& Humidity Sensor..6
8. Crowtail- Light Sensor..6
9. Crowtail- Vibration Motor..6
10. Crowtail- Hall Sensor..7
11.Crowtail- Tilt Switch..7
12.Crowtail- OLED..7
13. Crowtail- RGB-LED..7
14. Crowtail- Water Sensor..8
15.Crowtail- Laser Pointer..8
16. One Wrie Waterproof Temperature Sensor..8
17. Crowtail- MOSFET..9
28.Crowtail – Flame Sensor..9
19.Crowtail- Rotary Angle Sensor..9
20.Crowtail- 3-Axis Digital Accelerometer..9
21.Crowtail- PIR Sensor..10
22.Crowtail- 9G Servo..10
23.Crowtail- Magnetic Switch..10
24.Crowtail- Protoboard..11
25.Crowtail- Switch..11
Part2 Crowtail Application..12
Lesson1: LED Control..12
Lesson2: Button Control LED..14
Lesson3: Moisture Sensor..15
Lesson4: Vibration Motor..16
Lesson5: Emergency Light..17
Lesson6: Raining detect..18
Lesson7: Flame detect..19
Lesson8: Motion Detect..19
Lesson9: Hall Switch..20
Lesson10 Breathing Lamp..21
Lesson11 Laser Pointer..22
Lesson12: One Wire Waterproof Temperature Sensor..23
Lesson13: Rainbow Light..27
Lesson14 MOSFET Control Fan..29
Lesson15: Read Data From ADXL345 Digital Accelerometer Sensor..30
Lesson16: Servo Control..32

Lesson17: TWO NodeMCU Communication...33
Lesson18: DHT11 Temperature Display System..35
Lesson19: Wireless Toliet Led Indicator..40

Preface –Getting Started

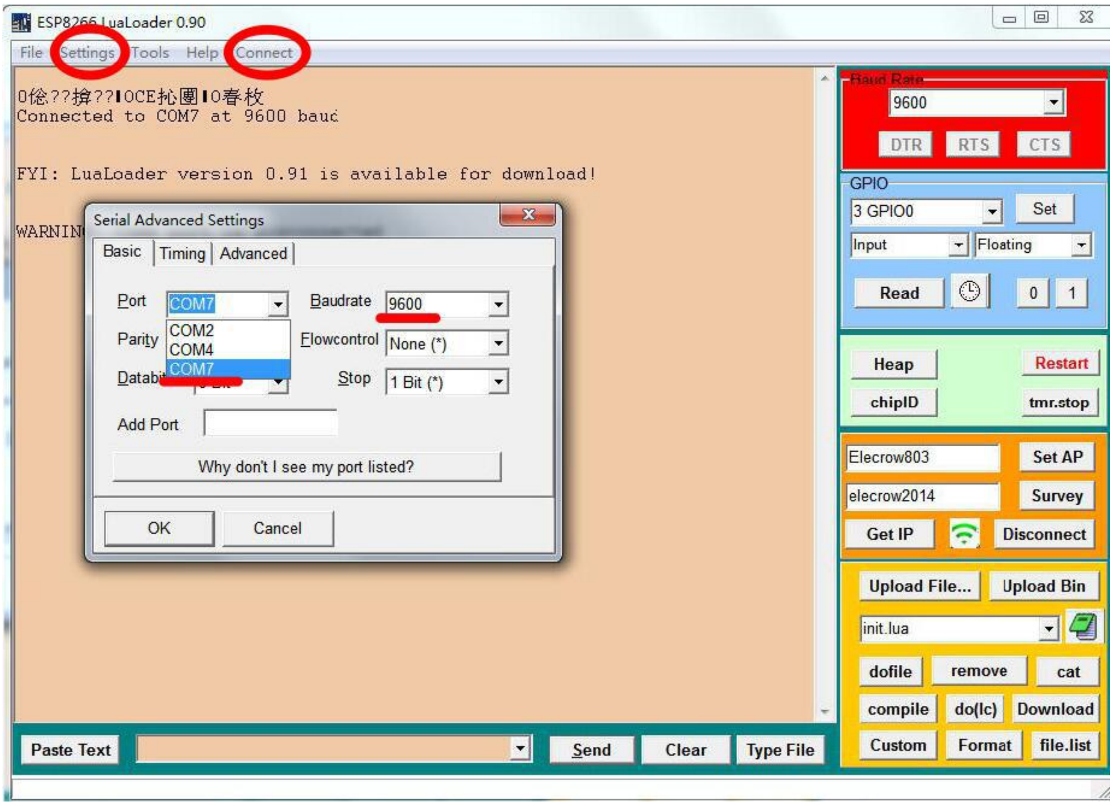
Welcome to the world of ESP8266 NodeMCU. It’s an ESP8266 WiFi module with Crowtail interface. We will use it to make some simple application of fun with various Crowtail modules .

1: How to use ESP8266 NodeMCU

1) Before you use the micro USB cable to connect the ESP8266 NodeMCU with your computer, you need to install the cp2012 USB driver. you can download it from there: https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

2) After succeed connection, we need to choose some development tools such as LuaLoader and luaEditor, we will provide these software. You can open them from Tools folder. The luaEditor is used to programming and debugging, finally it will generate .lua file. The LuaLoader is used to download and compile the .lua file, and it provides some other features. The next will introduce the usage of LuaLoader.

*Open the LuaLoader, click the menu “Setting”, choose the “Comm port Settings”, and it will popup a “Serial Advance Setting”, set up the right port here then click the “Connect” button on the top of the menu.

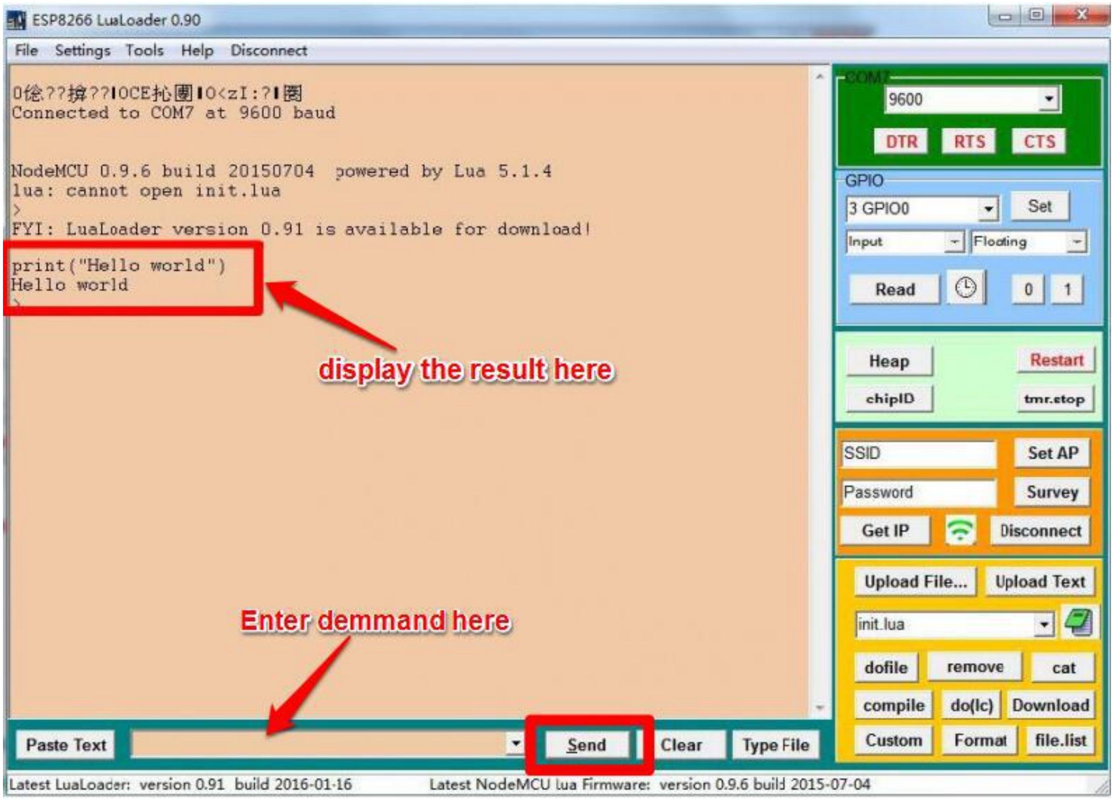


```
0俆??揄??▮OCE扡團▮O春枚
Connected to COM7 at 9600 bauc

FYI: LuaLoader version 0.91 is available for download!

WARNIN
```

Port: COM7 (COM2, COM4, COM7)
Baudrate: 9600
Parity
Flowcontrol: None (*)
Databi
Stop: 1 Bit (*)
Add Port

Baud Rate: 9600
GPIO: 3 GPIO0, Input, Floating
Elecrow803
elecrow2014
init.lua

When the information”Connected to COMX at 9600baud” display in the debug window, it means connecting is successful, and you can input command statement into the bellow edit box. As send command “ print(“Hello world！ ”) ”, then the result will display in the debug window.



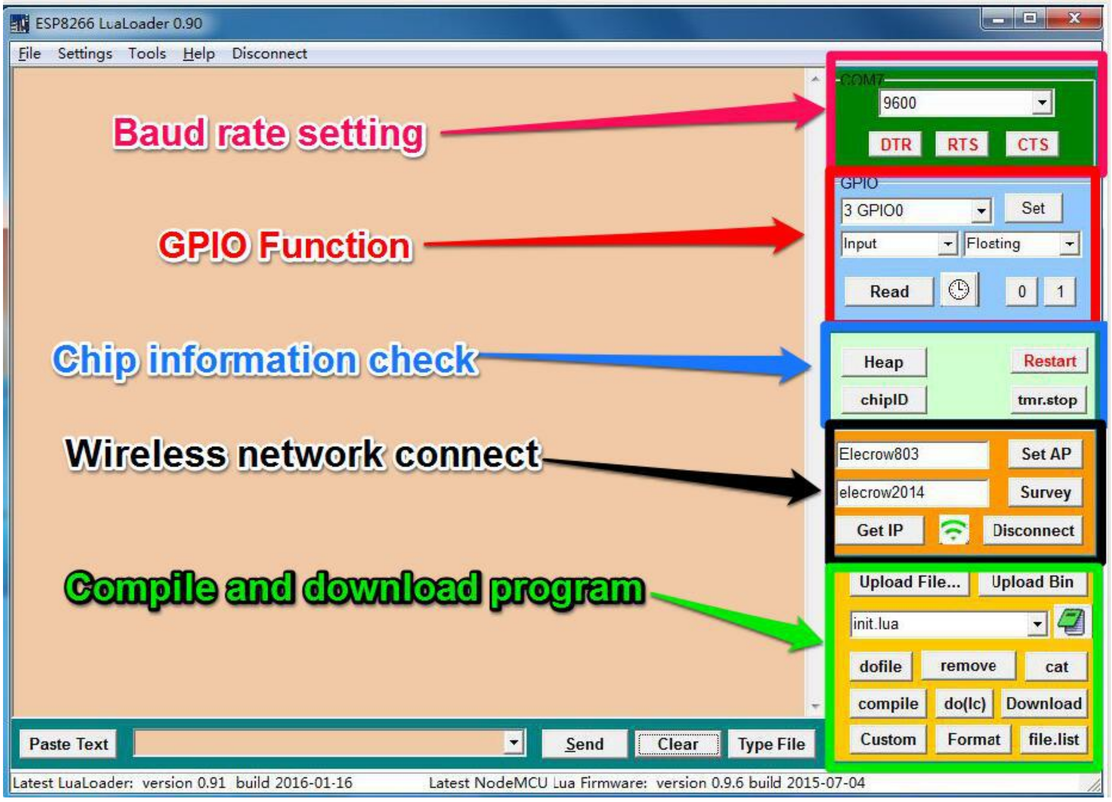
```
0锿??擠??IOCE扡團I0<zI:?I圏
Connected to COM7 at 9600 baud

NodeMCU 0.9.6 build 20150704  powered by Lua 5.1.4
lua: cannot open init.lua
>
FYI: LuaLoader version 0.91 is available for download!
print("Hello world")
Hello world
>
```

display the result here

Enter demmand here

Latest LuaLoader: version 0.91 build 2016-01-16   Latest NodeMCU Lua Firmware: version 0.9.6 build 2015-07-04

Also, you can use some of the function which display on the right side of Lualoader interface.



Baud rate setting

GPIO Function

Chip information check

Wireless network connect

Compile and download program

Elecrow803

elecrow2014

Latest LuaLoader: version 0.91 build 2016-01-16   Latest NodeMCU Lua Firmware: version 0.9.6 build 2015-07-04

Baud rate setting: set the baud rate that you need ,normally set at 9600.

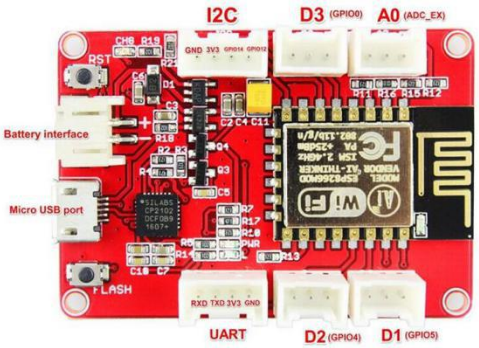
GPIO Function: Choose different GPIO port, set its mode(Input, Output or Interrupt),set GPIO Pull up resistor (Floating, Pull up or Pull down), and “Read” or “Write” operation to the GPIO

port.
Chip information check: check the information about the chip or restart the NodeMCU.
Wireless network connect: Enter the wireless network account and password, you can connect it and get the IP.
Compile and download program: First, click “Upload File…”upload the .lua file that you want to download. Secondly click “compile” to compile the .lua file, through the compile can click “Download” to download the program.

This site: http://www.benlo.com/esp8266/ has its detail introduction.

Part1 Modules Introduction

1. Crowtail- ESP8266 NodeMCU



Description

Crowtail- ESP8266 NodeMCU adds six crowtail interface on the board (one I2C port, one UART port, one analog port and three digital port). Wired up a USB-Serial chip that can upload code. We also provide a connector for 3.7V Lithium polymer batteries and built battery charging.

2. Crowtail- LED

Description

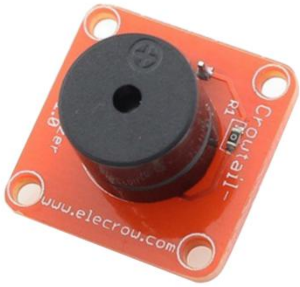
The Crowtail-LED is designed for the beginners of Arduino. It is the best way to step into the Aduino from it's I/O pins. The LED is the best choice to help you learn I/O pins.

3.Crowtail- Button

Description

The Crowtail-Button is a momentary push button which rebounds on its own after it is released. The button outputs a HIGH signal when pressed, and LOW when released.

4.Crowtail- Buzzer



Description

The Crowtail- Buzzer module is a piezo buzzer which can be connected to digital outputs, and will emit a tone when the output is HIGH.

5. Crowtail- Relay

Description

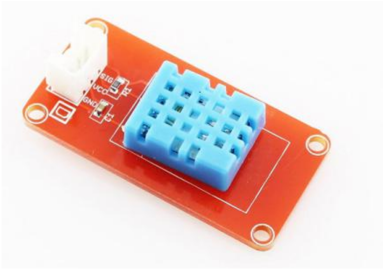
The Crowtail- Relay module is a digital normally-open switch. Through it, you can control circuit of high voltage with low voltage, say 5V on the controller.

6.Crowtail- Moisture Sensor

Description

This Moisture Sensor can be used to detect the moisture of soil or your pet plant's water level, let the plants in your garden reach out for human help. It is connected to Analog I/O port.

7. Crowtail- Temperature& Humidity Sensor



Description

The Crowtail- Temperature& Humidity Sensor is based on DH11 which is a complex sensor with a calibrated digital signal out.It used to detect the temperature & humidity.

8. Crowtail- Light Sensor

Description

The Crowtail- Light Sensor module incorporates a light dependent resistor (LDR), is a commonly used sensor in a wide variety of applications from DIY projects to industrial automation. Typically, the resistance of the LDR or Photo resistor will decrease when the ambient light intensity increases. This means that the output signal from this module will be HIGH in bright light, and LOW in the dark.

9. Crowtail- Vibration Motor

Description

This is a mini vibration motor suitable as a non-audible indicator. When the input is HIGH, the motor will vibrate just like your cell phone on silent mode.

10. Crowtail- Hall Sensor

Description

The Crowtail- Hall Sensor uses the Allegro Hall-effects switches are next generation for the population Allegro 312x and 314x lines of unipolar switches. It switches HIGH(turn off) when the magnetic field disappears.

11.Crowtail- Tilt Switch

Description

The Crowtail- Tilt Switch is the equivalent of a button, and is used as a digital input. Inside the tilt switch is a pair of balls that make contact with the pins when the case is upright. Tilt the case over and the balls don't touch, thus not making a connection.

12.Crowtail- OLED

Description

The Crowtail- OLED is constructed from 128 x 64 dot matrix OLED module. The display offers high brightness, self-emission, high contrast ratio, slim/thin outline, wide viewing angle, wide temperature range and low power consumption.

13. Crowtail- RGB-LED

Description

The Crowtail- RFB-LED module with 4 pcs of WS2812B which is a Chainable & Addressable LED. Users can control all the LED with only one microcontroller pin! Besides, the LED bar can be also chainable, that is, you can connect more than one LED bar together to make your project more dreamful. In this module you can control every LED with different color at the same time.

14. Crowtail- Water Sensor

Description

The Crowtail- water sensor detects water by having a series of exposed traces, The resistor will pull the sensor trace value high until a drop of water shorts the sensor trace to the grounded trace.

15.Crowtail- Laser Pointer

Description

This laser pointer or laser pen is a small handheld device with a laser diode emitting a very narrow coherent low-powered laser beam of visible light, it is frequently used in lecture halls and demonstrations to point at topics of interest on a presentation board. In a school setting, they have become ubiquitous, and they are very useful teaching aids. Please note that DO NOT Point this module to eyes directly, It may cause some blindness, glare and afterimages.

16. One Wrie Waterproof Temperature Sensor

Description

This is a waterproofed version of the DS18B20 Temperature sensor. Handy for when you need to measure something far away, or in wet conditions. While the sensor is good up to 125 degree, the cable is jacketed in PVC so we suggest keeping it under 100 degree.

17. Crowtail- MOSFET

Description

Crowtail- MOSFET enables you to control higher voltage project, say 50VDC, with low voltage, say 5V, on microcontroller. MOSFET is also a kind of switch. There are two screw terminals on the board. One for input power source and the other for device you want to control. Crowtail- MOSFET will pass the power from one end to another when closed.

28.Crowtail – Flame Sensor

Description

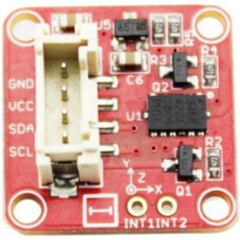
The Crowtail- Flame Sensor can be used to detect fire source or other light sources of the wavelength in the range of 760nm - 1100 nm. It is based on the YG1006 sensor which is a high speed and high sensitive NPN silicon phototransistor.

19.Crowtail- Rotary Angle Sensor

Description

Rotary angle sensor for crowduino.The angular range is 300 degrees with a linear change in value. The resistance value is 10k ohms, perfect for crowduino use. This may also be known as a “potentiometer ”.

20.Crowtail- 3-Axis Digital Accelerometer



Description

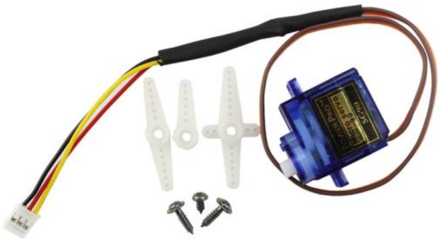
This is a high resolution digital accelerometer providing you at max 3.9mg/LSB resolution and large ±16g measurement range. It's base on an advanced 3-axis IC ADXL345. Have no worry to implement it into your free-fall detection project, cause it's robust enough to survive up to 10,000g shock. Meanwhile, it's agile enough to detect single and double taps. It's ideal for motion detection, Gesture detection as well as robotics.

21.Crowtail- PIR Sensor

Description

The Crowtail-PIR Motion sensor is simply connect to Crowtail base shield and program it. When anyone moves in its detecting range, the sensor outputs HIGH on its SIG PIN.

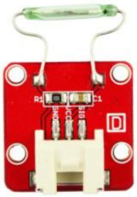
22.Crowtail- 9G Servo



Description

Tower Pro SG90 is a high quality, low-cost servo for all your mechatronic needs. It comes with a 3-pin power and control cable.

23.Crowtail- Magnetic Switch



Description

The Crowtail- Magnetic Sensor is a double-ended type and may be actuated with an electromagnet, a permanent magnet or a combination of both. The magnetic switch is a wonderful tool for designers who would like to turn a circuit on and off based on proximity

24.Crowtail- Protoboard

Description

This Crowtail allows you to add your own circuitry or components to your Crowtail system prototypes.

25.Crowtail- Switch

Description

The Crowtail- Switch is a Latching switch. When the first press the switch, the switch and keep the current adjustment and the button outputs a HIGH signal,namely the self-locking. When the second press the switch, the switch off and switch button to pop up at the same time, outputs a LOW signal.

Part2 Crowtail Application

Lesson1: LED Control

LED control is basic on NodeMCU. In this lesson we will learn how to control the LED, you can make the led blink and control the last time it shines or the brightness.

Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- LED x 1

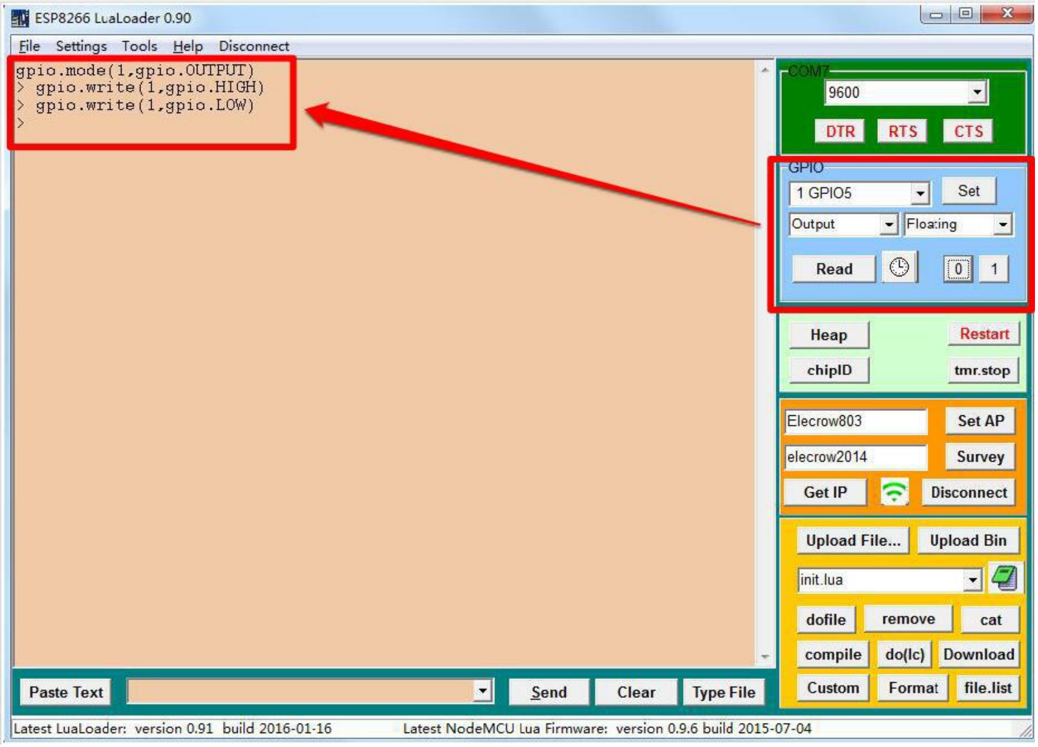
Crowtail- Cable x 1

USB Cable x 1

Hardware Connection

Plug the Crowtail-LED to the D1 port on NodeMCU by a 3pin Crowtail-Cable as following.

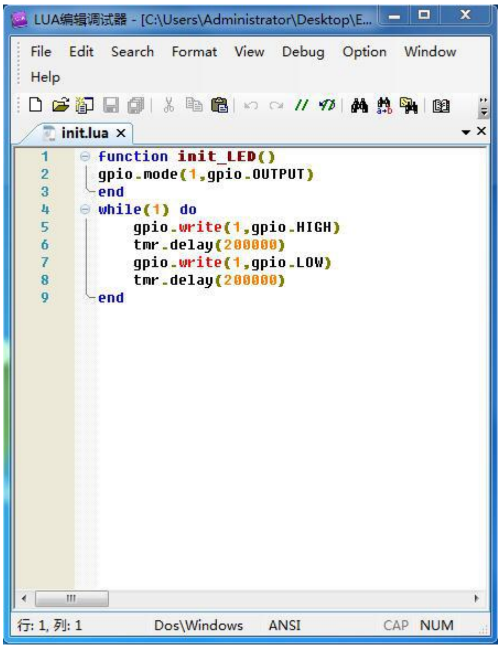
Open the LuaLoader. Click the button in the GPIO function area, choose GPIO5 port, set it as Output mode, click ”1” the LED will light up, click “0” the LED will light off.



```
gpio.mode(1,gpio.OUTPUT)
> gpio.write(1,gpio.HIGH)
> gpio.write(1,gpio.LOW)
>
```

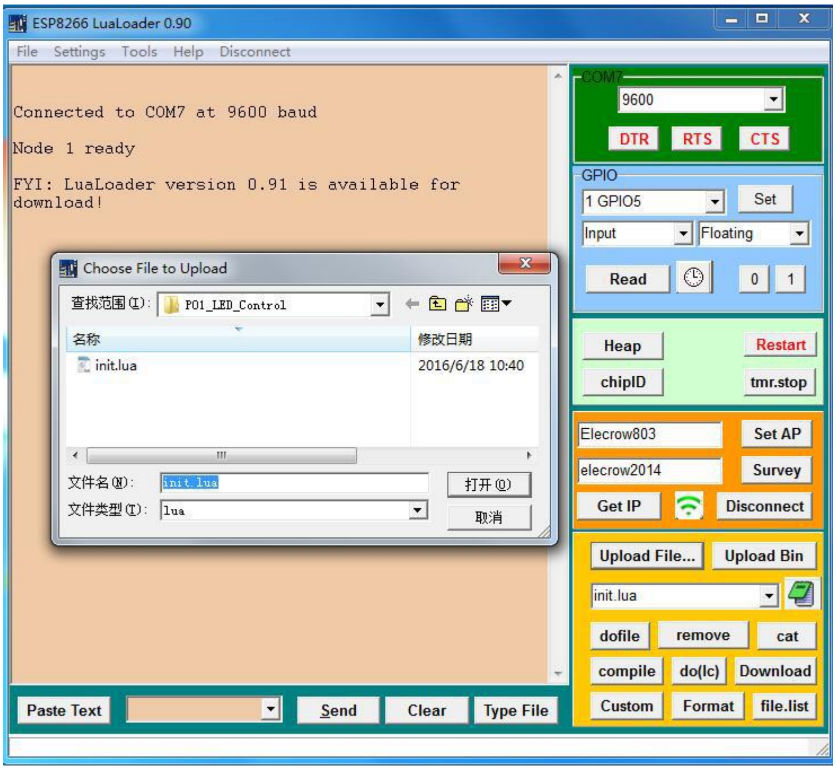
Latest LuaLoader: version 0.91 build 2016-01-16 Latest NodeMCU Lua Firmware: version 0.9.6 build 2015-07-04

Also, we can download the code to the NodeMCU. Copy the bellow code and paste it to LuaEditor, then save the file and named it “init.lua”


LUA编辑调试器 - [C:\Users\Administrator\Desktop\E...
File Edit Search Format View Debug Option Window Help
init.lua
行: 1, 列: 1 Dos\Windows ANSI CAP NUM

```
function init_LED()
gpio.mode(1,gpio.OUTPUT)
end
while(1) do
     gpio.write(1,gpio.HIGH)
     tmr.delay(200000)
     gpio.write(1,gpio.LOW)
     tmr.delay(200000)
end
```

Open the LuaLoader and follow the below steps before download program. First, click "Upload File…" to choose ESP8266 NodeMCU kit demo code -> P01_LED _Control -> init.lua

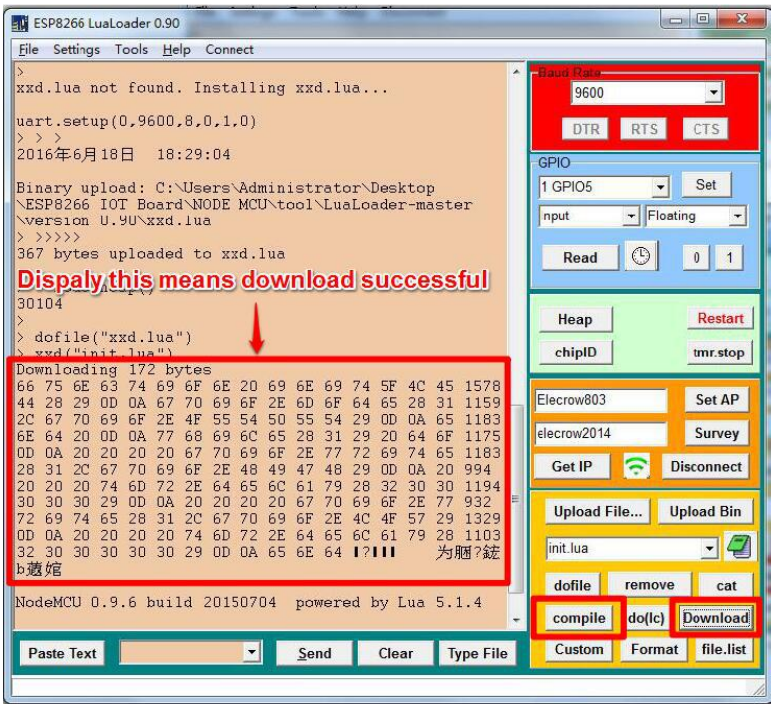


```
Connected to COM7 at 9600 baud

Node 1 ready

FYI: LuaLoader version 0.91 is available for
download!
```

Baud: 9600
GPIO: 1 GPIO5; Input; Floating
Elecrow803
elecrow2014
init.lua

查找范围(I): P01_LED_Control
init.lua 2016/6/18 10:40
文件名(N): init.lua
文件类型(T): lua

Secondly, click "compile" to compile the .lua file and then click "Download" to download the .lua file



ESP8266 LuaLoader 0.90

```
>
xxd.lua not found. Installing xxd.lua...

uart.setup(0,9600,8,0,1,0)
> > >
2016年6月18日   18:29:04

Binary upload: C:\Users\Administrator\Desktop
\ESP8266 IOT Board\NODE MCU\tool\LuaLoader-master
\version 0.90\xxd.lua
> >>>>>
367 bytes uploaded to xxd.lua
30104
>
> dofile("xxd.lua")
> xxd("init.lua")
Downloading 172 bytes
66 75 6E 63 74 69 6F 6E 20 69 6E 69 74 5F 4C 45 1578
44 28 29 0D 0A 67 70 69 6F 2E 6D 6F 64 65 28 31 1159
2C 67 70 69 6F 2E 4F 55 54 50 55 54 29 0D 0A 65 1183
6E 64 20 0D 0A 77 68 69 6C 65 28 31 29 20 64 6F 1175
0D 0A 20 20 20 20 67 70 69 6F 2E 77 72 69 74 65 1183
28 31 2C 67 70 69 6F 2E 48 49 47 48 29 0D 0A 20 994
20 20 20 74 6D 72 2E 64 65 6C 61 79 28 32 30 30 1194
30 30 30 29 0D 0A 20 20 20 20 67 70 69 6F 2E 77 932
72 69 74 65 28 31 2C 67 70 69 6F 2E 4C 4F 57 29 1329
0D 0A 20 20 20 20 74 6D 72 2E 64 65 6C 61 79 28 1103
32 30 30 30 30 30 29 0D 0A 65 6E 64 ▌?▌▌▌     为腼?鋐
b薳婠

NodeMCU 0.9.6 build 20150704  powered by Lua 5.1.4
```

Dispaly this means download successful

Baud Rate: 9600

GPIO: 1 GPIO5, nput, Floating

Elecrow803

elecrow2014

init.lua

The Crowtail- LED will be blink every 0.2 second.

Why the led blink every 0.2 second? How to control it keep lighting more longer? Now let me tell you. In this demo code, first define the gpio1 into output port, then let the gpio1 constantly switch between high and low level, set the time interval as 0.2 second and make it cycle. You can change the tmr.delay() parameter in the program to change the interval.

Demo code:

```
function init_LED()
gpio.mode(1,gpio.OUTPUT)     --Set the port mode
end
while(1) do
       gpio.write(1,gpio.HIGH)     -- gpio5 is the port 1
       tmr.delay(200000)           --delay 0.2s
       gpio.write(1,gpio.LOW)
       tmr.delay(200000)
end
```

In this lesson we learn the following function.

gpio.mode()

gpio.write()

tmr.delay()

Lesson2: Button Control LED

In the lesson1, we have learned how to control a led turn on for one second, and turn off for one second. Do you want to control it yourself by using the button? In this lesson, we will tell you how to realize it. When pressed the button, the led was lighten and turn off when released.

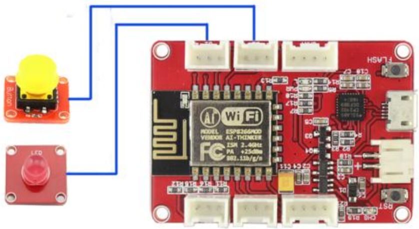
Material:

Crowtail- LED x 1

Crowtail- Button x 1

Crowtail- ESP8266 NodeMCU x 1

Hardware Connection



Open the LuaLoader, click the "Upload File…" and choose ESP8266 NodeMCU kit demo code -> P02_Button_Control_Led -> init.lua. After the upload completed then click "Download " to download the init.lua file.

Demo code:

```
gpio.mode(1,gpio.OUTPUT)    --Setup the port mode
gpio.mode(2,gpio.INPUT)
while(1)    do
gpio.read(2)                    -- read port 2(gpio4)
 tmr.delay(200000)       --Delay 0.2 s
if (gpio.read(2)==1) then
gpio.write(1,gpio.HIGH)    -- set the port 1 high level
end
if (gpio.read(2)==0) then
gpio.write(1,gpio.LOW)       --set the port 1 low level
end
end
```

When you press the button, the led lighting , the led out when loosen the button.

Lesson3: Moisture Sensor

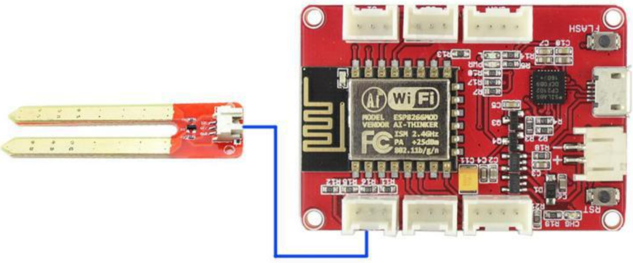
Do you want to know if your pet plant need some water in garden? This Moisture sensor can help you. This Moisture Sensor can be used to detect the moisture of soil or judge if there is water around the sensor, let the plants in your garden reach out for human help. They can be very easy to use, just insert it into the soil and then it will read . With the help of this sensor, it will be realizable to make the plant remind you : hey, i am thirsty now, please give me some water.

Material:

Crowtail- Moisture sensor x 1

Crowtail- ESP8266 NodeMCU x 1

Hardware Connection



Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P03_ Moisture_sensor -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Demo code:

```
while(1)            do
b=adc.read(0)     -- analog read A0
print (b)
tmr.delay(2000000)     --delay 2s
end
```

you can see the sensor value in the LuaLoader debug window.

Lesson4: Vibration Motor

Tilt Switch is the equivalent of a button, it can output different signal when it upright and tilt. In this lesson, we will learn how to control the vibration motor by the tilt switch. When the tilt switch is slant, the vibration motor will be vibrating.

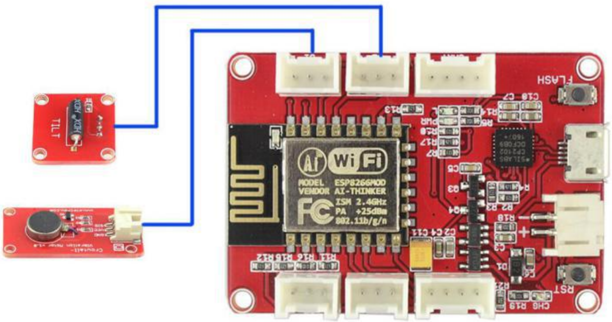
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- Vibration Motor x 1

Crowtail- Tilt Switchx 1

Hardware Connection



Open the LuaLoader, click the “Upload File…” to choose ESP8266 NodeMCU kit demo code -> P04_Vibration_Motor -> init.lua. After the upload completed then click "Download " to download the init.lua file.

Demo code:

```
gpio.mode(1,gpio.OUTPUT)     -- set port1    output mode
gpio.mode(2,gpio.INPUT)        -- set port2    input mode
while(1)    do
gpio.read(2)
 tmr.delay(200000)              --delay 2s
if (gpio.read(2)==1) then
gpio.write(1,gpio.HIGH)
end
if (gpio.read(2)==0) then
gpio.write(1,gpio.LOW)
end
end
```

When the tilt switch slope, the vibration motor vibrating. Otherwise, the vibration motor was not vibrating.

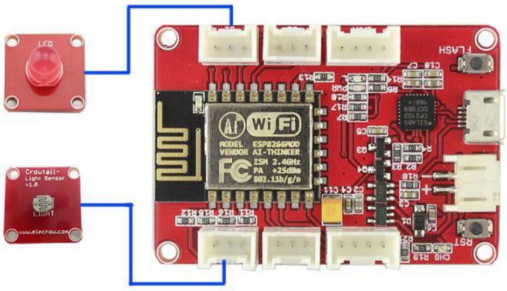
Lesson5: Emergency Light

Do you want to design a emergency light, which can automatic working when the light dimmed, and it will stop when the light brighter. So it can be used for emergency light system. In this lesson, we will help you about this application.

Material:

Crowtail- ESP8266 NodeMCU x 1
Crowtail- Light Sensor x 1
Crowtail- LED x 1

Hardware Connection



Open the LuaLoader, click the "Upload File..." to choose ESP8266 NodeMCU kit demo code -> P05_Emergency_Light -> init.lua. After the upload completed then click "Download " to download the init.lua file.

Demo code:

```
while (1) do
b=adc.read(0)          --Analog read A0
if (b<100) then
gpio.write(1,gpio.HIGH)
else
gpio.write(1,gpio.LOW)
end
end
```

Under the environment of light, blocking light sensor , led will turn on. No block ,led turn off.

Lesson6: Raining detect

In our daily life, we play computer game or watch TV all-possessed indoor,sometime raining, our clothes usually be wetted in the outdoor, but if there is a sensor that it can remind us that when it begin to rain outside, tragedy will not happen to us. In this lesson, we will tell you how to build it. When raining outside, the LED inside will be lighted.

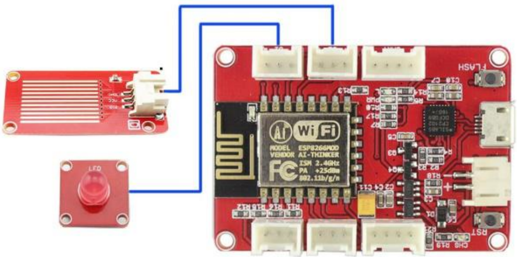
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- LED x 1

Crowtail- Water sensor x 1

Hardware Connection



Open the LuaLoader, click the "Upload File…"choose ESP8266 NodeMCU kit demo code -> P06_Rainning_detect -> init.lua. After the upload completed then click "Download " to download the init.lua file.

Demo code:

```
a=gpio.read(2)     --Digital read port 2
if (a==1) then
    gpio.write(1,gpio.HIGH)
if   (a==0) then
    gpio.write(1,gpio.LOW)
end
end
```

When detect raining outside, the LED lights up.

Lesson7: Flame detect

Have you seen the fire-fighting robots? Why it can find the source of fire?In this lesson, we will tell you the principle about it and you can make a robot like that.

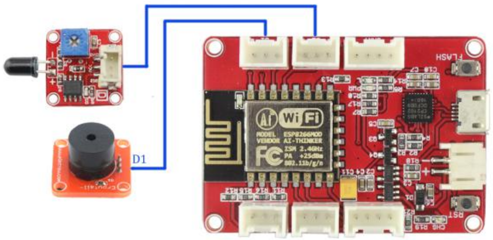
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- Flame Sensor x 1

Crowtail- buzzer x 1

Hardware Connection



Open the LuaLoader, click the “Upload File...”choose ESP8266 NodeMCU kit demo code -> P07_Flame_detect -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Demo code:

```
a=gpio.read(2)
if (a==1) then
      gpio.write(1,gpio.LOW)
if    (a==0) then
      gpio.write(1,gpio.HIGH)
end
end
```

When detect the flame, buzzer alarming. The adjustable resistance may be adjusted to vary sensitivity of detection.

Lesson8: Motion Detect

Do you want to make something to detect movement , there is a PIR Motion Sensor that can help you, when there is an object in its detection range and has a movement, this action will be captured and respond as alarm or take photo.

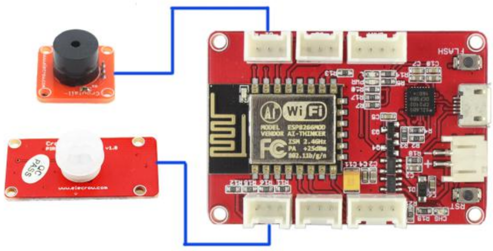
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- PIR Motion Sensor x1

Crowtail- Buzzer x1

Hardware Connection



Open the LuaLoader, click the “Upload File...”choose ESP8266 NodeMCU kit demo code -> P08_Motion_detect -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Demo code:

```
while(1)            do
b=gpio.read(2)
print (b)
tmr.delay(2000000)
if(b==1) then
      gpio.write(1,gpio.HIGH)
     else
      gpio.write(1,gpio.LOW)
     end

end
```

After successfully download the code, you will observed that: when there is a motive object in the detection range of the sensor , the buzzer will alarm.

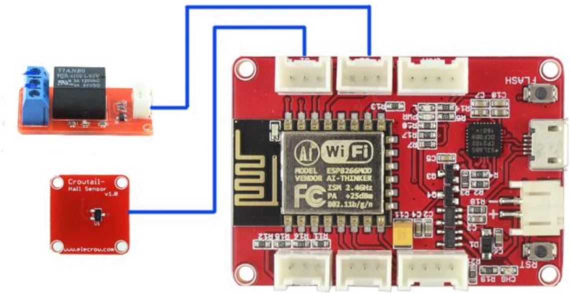
Lesson9: Hall Switch

In our daily life, we usually want to have an operation in certain situations, for example, the conveyor belt transmission for a particular item to a location and then remove it, or the water supply stop with the shower head back to the bracket when we are taking a shower, and so on. At this moment we can place a hall sensor in a specific location of the conveyor belt, and place an magnetic steel on the particular item package, so when the item is transferred to the specified location it can realize automatic operation. Here is no conveyor belt, just one hall switch to realize the automatic function.

Material:

Crowtail- ESP8266 NodeMCU x 1
Crowtail- Hall Sensor x1
Crowtail- Relay x1

Hardware Connection



Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P9_Hall_Switch -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Demo code:

```
while(1)          do
a=gpio.read(1)
print (b)
tmr.delay(2000000)          --delay 2s
if(a==1) then
      gpio.write(2,gpio.HIGH)
     else
      gpio.write(2,gpio.LOW)
     end
end
```

If there is a magnetic material approaching the hall sensor, the relay will start to work, on the contrary, the relay will be disconnected. Just like a hall switch.

10. Lesson10 Breathing Lamp

In this lesson, we through adjust the Rotary Angle Sensor to control the PWM output then control the brightness of LED， which could be a breathing lamp.

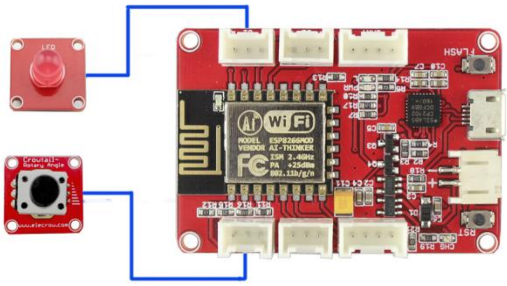
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- Rotary Angle Sensor

Crowtail- LED

Hardware Connection



Open the LuaLoader, click the “Upload File...”choose ESP8266 NodeMCU kit demo code -> P10_Breathing_Lamp -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Demo Code:

```
print("Breathing Lamp Start")
pwm.setup(1,1000,1023);     -- set pot 1 PWM mode frequency 1000hz    and the duty cycle 1023pwm.start(1);
tmr.alarm(2,100,1,function()    --Timer 2 using the timing 100ms Repeatability
local r=adc.read(0)
if(r<1024) then              -- the most analog read is 1024 but the most PWM’ suty cycle is 1023
pwm.setduty(1,r);       -- set the port 1 PWM’s duty cycle is “r”
else
pwm.setduty(1,1023);
end
  end)
```

The LED will lighten or darken with the Rotary Angle Sensor forward or reverse rotate.

11. Lesson11 Laser Pointer

This laser pointer or laser pen is a small handheld device with a laser diode emitting , it is frequently used in lecture halls and demonstrations to point at topics of interest on a presentation board. But please note that DO NOT Point this module to eyes directly.

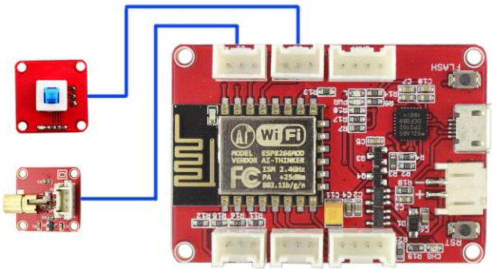
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- Switch x 1

Crowtail- Laser Pointer x 1

Hardware Connection



Open the LuaLoader, click the “Upload File...” to choose ESP8266 NodeMCU kit demo code -> P11_Laser_Pointer -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Demo code:

```
a=gpio.read(2)
while (1) do
if (a==1) then
     gpio.write(1,gpio.HIGH)
     else
```

```
    gpio.write(1,gpio.LOW)
end
end
```

Lesson12: One Wire Waterproof Temperature Sensor

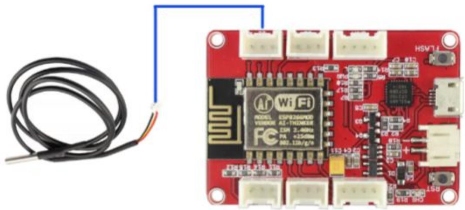
This sealed digital temperature probe lets you precisely measure temperatures in wet environments with a simple 1-Wire interface.

Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- One wire Waterproof Temperature Sensor x 1

Hardware Connection



Open the LuaLoader, click the "Upload File…" to choose ESP8266 NodeMCU kit demo code -> P12_DS18B20_One_Wire_Waterproof_Temperature_Sensor -> ds18b20.lua. After the upload completed then click "Download " to download the ds18b20.lua file. Then click the "Upload File…"choose the init.lua in the same folder, download it.

Demo Code:

Ds18b20.lua:

```
-- Set module name as parameter of require
local modname = ...
local M = {}
_G[modname] = M
--------------------------------------------------------------------------------
-- Local used variables
--------------------------------------------------------------------------------
-- DS18B20 dq pin
local pin = nil
-- DS18B20 default pin
local defaultPin = 1
--------------------------------------------------------------------------------
-- Local used modules
--------------------------------------------------------------------------------
-- Table module
local table = table
-- String module
local string = string
-- One wire module
```

```
local ow = ow
-- Timer module
local tmr = tmr
-- Limited to local environment
setfenv(1,M)
--------------------------------------------------------------------------------
-- Implementation
--------------------------------------------------------------------------------
C = 0
F = 1
K = 2
function setup(dq)
    pin = dq
    if(pin == nil) then
        pin = defaultPin
    end
    ow.setup(pin)
end
function addrs()
    setup(pin)
    tbl = {}
    ow.reset_search(pin)
    repeat
        addr = ow.search(pin)
        if(addr ~= nil) then
            table.insert(tbl, addr)
        end
        tmr.wdclr()
    until (addr == nil)
    ow.reset_search(pin)
    return tbl
end

function readNumber(addr, unit)
    result = nil
    setup(pin)
    flag = false
    if(addr == nil) then
        ow.reset_search(pin)
        count = 0
        repeat
            count = count + 1
            addr = ow.search(pin)
            tmr.wdclr()
        until((addr ~= nil) or (count > 100))
        ow.reset_search(pin)
    end
    if(addr == nil) then
        return result
    end
    crc = ow.crc8(string.sub(addr,1,7))
    if (crc == addr:byte(8)) then
        if ((addr:byte(1) == 0x10) or (addr:byte(1) == 0x28)) then
            -- print("Device is a DS18S20 family device.")
            ow.reset(pin)
            ow.select(pin, addr)
            ow.write(pin, 0x44, 1)
            -- tmr.delay(1000000)
            present = ow.reset(pin)
            ow.select(pin, addr)
            ow.write(pin,0xBE,1)
            -- print("P="..present)
            data = nil
            data = string.char(ow.read(pin))
            for i = 1, 8 do
                data = data .. string.char(ow.read(pin))
            end
            -- print(data:byte(1,9))
            crc = ow.crc8(string.sub(data,1,8))
            -- print("CRC="..crc)
            if (crc == data:byte(9)) then
                if(unit == nil or unit == C) then
                    t = (data:byte(1) + data:byte(2) * 256) * 625
                elseif(unit == F) then
                    t = (data:byte(1) + data:byte(2) * 256) * 1125 + 320000
                elseif(unit == K) then
                    t = (data:byte(1) + data:byte(2) * 256) * 625 + 2731500
                else
                    return nil
                end
                t1 = t / 10000
                t2 = t % 10000
                -- print("Temperature="..t1.."."..t2.." Centigrade")
                -- result = t1.."."..t2
                return t1, t2
            end
            tmr.wdclr()
        else
        -- print("Device family is not recognized.")
        end
    else
    -- print("CRC is not valid!")
    end
    return result
end
function read(addr, unit)
    t1, t2 = readNumber(addr, unit)
    if((t1 == nil ) or (t2 ==nil)) then
        return nil
    else
        return t1.."."..string.format("%04u", t2)
    end
end
-- Return module table
return M
```

Init.lua:

```
function getDSdata()
        pin=1 -- connect the signal wire to pin GPIO5
        t=require("ds18b20")
        t.setup(pin) -- gpio0
        addrs=t.addrs()
        node_id = node.chipid()
        print("Total Sensors.: "..table.getn(addrs).." ")
        print("Sensor Type...: "..node_id.." ")
        hex_format="%02X%02X%02X%02X%02X%02X%02X%02X"
        sensor_count=table.getn(addrs)
        if (sensor_count>0) then
            for i=1,sensor_count do
                sid=string.format(hex_format,string.byte(addrs[i],1,9))
                print("t"..i.." Unique ID    : "..sid.." ")
                tmr.wdclr()
            end
        end
            t1 = t.read(addrs[1],t.C)
            t5 = t.read(addrs[2],t.C)
            t6 = t.read(addrs[3],t.C)
            t2 = t.read(addrs[4],t.C)
            t3 = t.read(addrs[5],t.C)
            t4 = t.read(addrs[6],t.C)
            print("Temp t1 Lower.: "..t1.." deg C")
            print("Temp t2 Upper.: "..t2.." deg C")
        print("Temp t3 Middle: "..t3.." deg C")
        print("Temp t4 Top...: "..t4.." deg C")
        print("Temp t5 Flow..: "..t5.." deg C")
        print("Temp t6 Return: "..t6.." deg C")
-- Cleanup
      t = nil
      ds18b20 = nil
      package.loaded["ds18b20"]=nil
end
tmr.alarm(2, 1000, 1, function() getDSdata() end )
```

Lesson13: Rainbow Light

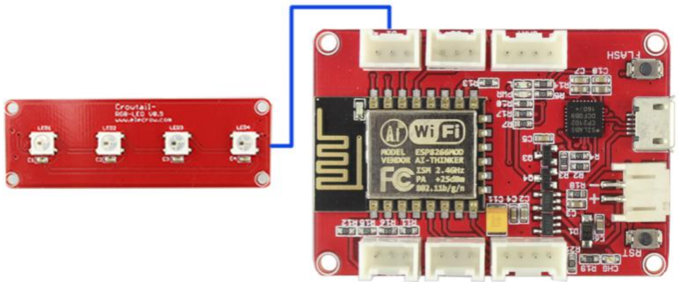
Do you want to make a RGB-LED like a colorful rainbow. Here is an example of how to make it. It is easy and only need one microcontroller pin. Now, let’s go into the colorful world.

Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- RGB LED   x 1

Hardware Connection



Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P13_Rainbow_Light -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

```
BRIGHT      = 1
ON          = BRIGHT * 255
LED_PIN     = 1          -- GPIO5
PIXELS      = 8
TIME_ALARM = 25          -- 0.025 second, 40 Hz
TIME_SLOW   = 500000   -- 0.500 second,   2 Hz

RED     = string.char( 0, ON,   0)
GREEN = string.char(ON,   0,   0)
BLUE    = string.char( 0,   0, ON)
```

```
WHITE = string.char(ON, ON, ON)
BLACK = string.char( 0,    0,    0)

function colourWheel(index)
  if index < 85 then
    return string.char(index * 3 * BRIGHT, (255 - index * 3) * BRIGHT, 0)
  elseif index < 170 then
    index = index - 85
    return string.char((255 - index * 3) * BRIGHT, 0, index * 3 * BRIGHT)
  else
    index = index - 170
    return string.char(0, index * 3 * BRIGHT, (255 - index * 3) * BRIGHT)
  end
end
rainbow_speed = 8
function rainbow(index)
  buffer = ""
  for pixel = 0, 7 do
    buffer = buffer .. colourWheel((index + pixel * rainbow_speed) % 256)
  end
  return buffer
end
if true then
  ws2812.write(LED_PIN, RED:rep(PIXELS))
  tmr.delay(TIME_SLOW)
  ws2812.write(LED_PIN, GREEN:rep(PIXELS))
  tmr.delay(TIME_SLOW)
  ws2812.write(LED_PIN, BLUE:rep(PIXELS))
tmr.delay(TIME_SLOW)
  ws2812.write(LED_PIN, WHITE:rep(PIXELS))
  tmr.delay(TIME_SLOW)
  ws2812.write(LED_PIN, BLACK:rep(PIXELS))
end
rainbow_index = 0
function rainbowHandler()
  while(1) do
    ws2812.write(LED_PIN, rainbow(rainbow_index))
    rainbow_index = (rainbow_index + 1) % 256
  end
end
 rainbowHandle
```

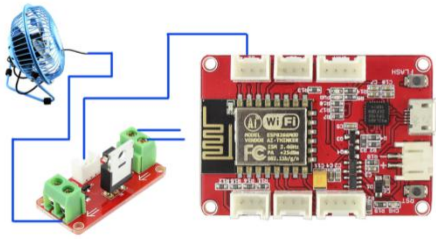
14.Lesson14 MOSFET Control Fan

The MOSFET is a kind of switch. It enables you to control higher voltage project with low voltage. There are two screw terminals on the board. One for input power source and the other for device you want to control. In this lesson we will teach you how to use the MOSFET to control a fan.

Material:

Crowtail- ESP8266 NodeMCU x 1
Crowtail- MOSFET x1

Hardware Connection



Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P14_MOSFET_Control_Fan -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

```
print("Crowtail- MOSFET Test")
while (1)     do
i=0
if (i==0) then
for i=0,1020,4 do
        pwm.setup(1,500,512)
        pwm.start(1)
        pwm.setduty(1,i)
        tmr.delay(20000)
if(i==1020)    then
for i=0,1020,4 do
        b=1020-i
        pwm.setup(1,500,512)
        pwm.start(1)
        pwm.setduty(1,b)
        tmr.delay(20000)      -- delay 20ms
end
end
end
end
end
```

After successful download the file you will see the fan rotating faster and faster, when it reach the fastest speed, it will become rotating slower and slower, and circulating.

Lesson15: Read Data From ADXL345 Digital Accelerometer Sensor

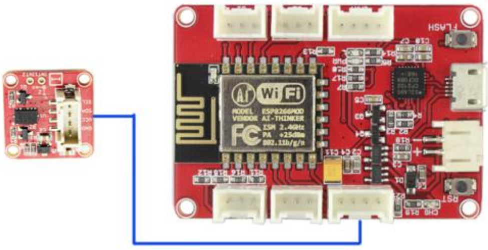
In this lesson ESP8266 is used to read data from ADXL345 Digital Accelerometer Sensor, It easy to use and obviously observe the result from the debug window. There is also some application of ADXL345 accelerometer in our daily life, likes free-hall detection and so on.

Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- 3-Axis Digital Accelerometer

Hardware Connection

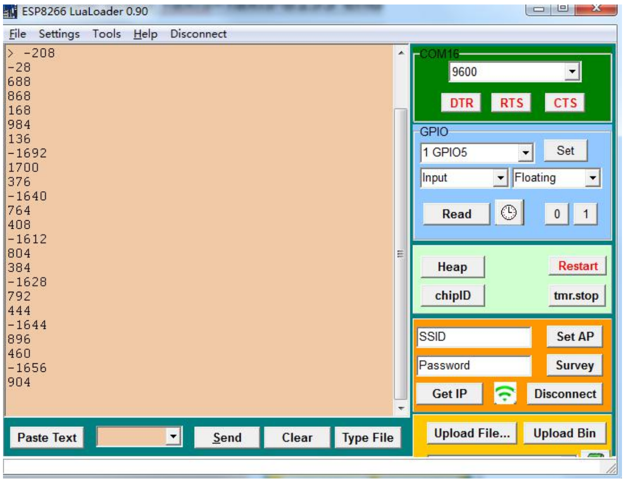


Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P15_Read_Data_From_ADXL345 -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

```
sda=5 --SDA connected to pin 5
scl=6 --SCL connected to pin 6
dev_addr=0x53    --Address of accelerometer

-- initialize i2c, set pin1 as sda, set pin2 as scl
i2c.setup(id,sda,scl,i2c.SLOW)

i2c.start(id)
i2c.address(id, dev_addr,i2c.TRANSMITTER)
i2c.write(id,0x2D) --Power control register
i2c.write(id,0x00) --Activate standby mode to configure device
i2c.stop(id)

i2c.start(id)
i2c.address(id, dev_addr,i2c.TRANSMITTER)
i2c.write(id,0x31) --Data format register
i2c.write(id,0x0B) --Set g range to 16, Full res
i2c.stop(id)

i2c.start(id)
i2c.address(id, dev_addr,i2c.TRANSMITTER)
```

```
i2c.write(id,0x2C) --BW rate register
i2c.write(id,0x0A) --Data rate 100Hz
i2c.stop(id)

i2c.start(id)
i2c.address(id, dev_addr,i2c.TRANSMITTER)
i2c.write(id,0x2D) --Power control register
i2c.write(id,0x08) --Activate measure mode
i2c.stop(id)

-- user defined function: read from reg_addr content of dev_addr
function read_reg(reg_addr)
  i2c.start(id)
  i2c.address(id, dev_addr,i2c.TRANSMITTER)
  i2c.write(id,reg_addr)
  i2c.start(id)
  i2c.address(id, dev_addr,i2c.RECEIVER)
  c=i2c.read(id,1)
  i2c.stop(id)
  return c
end

function adxl()
  X0 = read_reg(0x32)
  X1 = read_reg(0x33)
  Y0 = read_reg(0x34)
  Y1 = read_reg(0x35)
  Z0 = read_reg(0x36)
  Z1 = read_reg(0x37)

  --Combine 2 bytes to get a single 16bit number
  Xtemp = bit.lshift(string.byte(X1), 8)
  Xaxis = bit.bor(Xtemp, string.byte(X0))
  Ytemp = bit.lshift(string.byte(Y1), 8)
  Yaxis = bit.bor(Ytemp, string.byte(Y0))
  Ztemp = bit.lshift(string.byte(Z1), 8)
  Zaxis = bit.bor(Ztemp, string.byte(Z0))

  --Clear 3 sign extended MSB bits
  Xaxis=bit.band(0x1FFF, Xaxis)
  Yaxis=bit.band(0x1FFF, Yaxis)
  Zaxis=bit.band(0x1FFF, Zaxis)

  --Check if number is negative
  Xn=bit.band(0x1000, Xaxis)
  Yn=bit.band(0x1000, Yaxis)
  Zn=bit.band(0x1000, Zaxis)

  --If negative, convert twos complement number to decimal(-8193=-8192-1)
  if Xn==4096 then Xaxis=Xaxis-8193 end
  if Yn==4096 then Yaxis=Yaxis-8193 end
  if Zn==4096 then Zaxis=Zaxis-8193 end

  --4mg/LSB, multiply by 4, should divide by 1000
  Xaxis=Xaxis*4
  Yaxis=Yaxis*4
  Zaxis=Zaxis*4
  print(Xaxis)
  print(Yaxis)
  print(Zaxis)
end

tmr.alarm(2, 2000, 1, function() adxl() end )
```

After successfully download this file you will see the measuring accelerometer of x,y,z direction in the debug window of LuaLoader.



```
> -208
-28
688
868
168
984
136
-1692
1700
376
-1640
764
408
-1612
804
384
-1628
792
444
-1644
896
460
-1656
904
```

Lesson16: Servo Control

In this lesson, we will learn how to realize 9G servo control, and how to let it rotate. 9G servo need a 50HZ PWM, and the HIGH level’ duty cycle of this 20ms PWM, will decide the direction of the rotation.

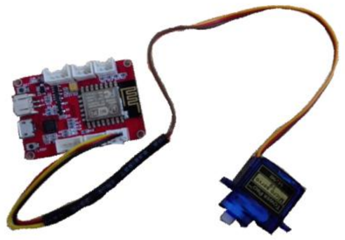
Material:

Crowtail- ESP8266 NodeMCU x 1

Crowtail- 9G Micro servo 9G x 1

Smart Phone x1

Hardware Connection



Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P16_Servo_Remote_Control -> init.lua. After the upload completed then click “Download ” to download the init.lua file.

Forward rotation:

```
for i=1,100,1 do
   tmr.alarm(0,20,1,function ()     -- 50Hz
        gpio.write(1, gpio.HIGH)
        tmr.delay(2000)              --Forward
        gpio.write(1, gpio.LOW)
end)
end
```

Reversion

```
for i=1,100,1 do
  tmr.alarm(0,20,0,function () -- 50Hz
        gpio.write(1, gpio.HIGH)
        tmr.delay(500)          --Reversion
        gpio.write(1, gpio.LOW)
     end)
end
```

After the successful uploading, you will see the 9G servo auto rotate in two different direction.

Lesson17: TWO NodeMCU Communication

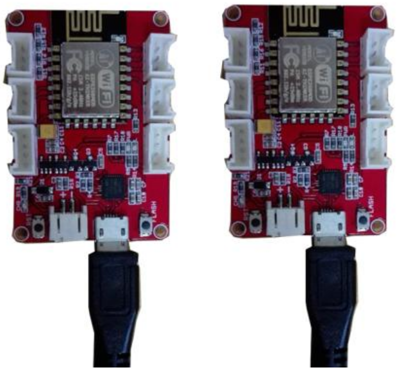
The ESP8266 NodeMCU has three kinds of working mode which are “AP” 、“STA”、“AP+STA”.

In this lesson we will use two NodeMCU to conmmunicate with each other. One NodeMCU to be set as “AP” mode and another to be set as “STA” mode, then establish TCP service between them.

Material:

Crowtail- ESP8266 NodeMCU x 2

Hardware Connection



Open the LuaLoader, set one serial port which connect the NodeMCU ("AP mode") and connecting , then click the "Upload File…" after connecting successful, choose ESP8266 NodeMCU kit demo code -> P17_NodeMCU_Communication ->TCP Server-> init.lua. After the upload completed then click "Download " to download the init.lua file.

The following, set another serial port which connect the NodeMCU ("STA mode") and connecting , then click the "Upload File…" after connecting successful, choose ESP8266 NodeMCU kit demo code -> P17_NodeMCU_Communication ->TCP Client-> init.lua. After the upload completed then click "Download " to download the init.lua file.

Demo code

AP:

```
print("ESP8266 Server")
wifi.setmode(wifi.STATIONAP);
wifi.ap.config({ssid="test",pwd="12345678"});
print("Server IP Address:",wifi.ap.getip())

sv = net.createServer(net.TCP)
sv:listen(80, function(conn)
    conn:on("receive", function(conn, receivedData)
        print("Received Data: " .. receivedData)
    end)
    conn:on("sent", function(conn)
      collectgarbage()
    end)
end)
```

STA:

```
print("ESP8266 Client")
wifi.sta.disconnect()
wifi.setmode(wifi.STATION)
wifi.sta.config("test","12345678") -- connecting to server
wifi.sta.connect()
print("Looking for a connection")
i=0
tmr.alarm(1, 2000, 1, function()
     if(wifi.sta.getip()~=nil) then
```

```
          tmr.stop(1)
          print("Connected!")
          print("Client IP Address:",wifi.sta.getip())
          cl=net.createConnection(net.TCP, 0)
          cl:connect(80,"192.168.4.1")
          tmr.alarm(2, 5000, 1, function()
          if(wifi.sta.getip()==nil) then
                wifi.sta.disconnect()
                wifi.sta.config("test","12345678") -- connecting to server
                 wifi.sta.connect()
                 print("Looking for a connection")
                 tmr.start(1)
           end
            cl:send("Hello World!")
           cl:send(i)
           i=i+1
          end)
      else
         print("Connecting...")
      end
end)
```

We can according the print information display in the Lualoader debug window to judge the two NodeMCU Whether have established a connection. If the “AP mode ” NodeMCU receive a “Hello World!” from the “STA mode” NodeMCU that means the communication establish successful.

Lesson18: DHT11 Temperature Display System

In this lesson you will know how to use the DHT11 temperature sensor, and display it on the OLED, It is easy to make us always know the current temperature. And we will use two ESP8266 NodeMCU for the sake of knowing the temperature of the DHT11 that placed anywhere.

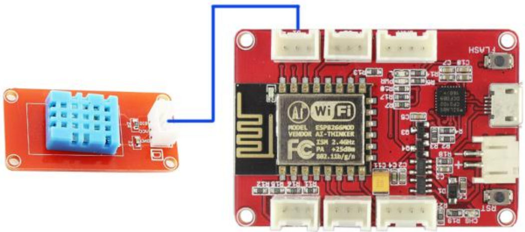
Material:

Crowtail- ESP8266 NodeMCU x 2

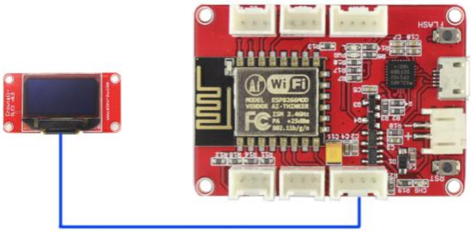
Crowtail- OLED x 1

Crowtail- Temperature & humidity Sensor x 1

Hardware Connection



“AP Mode” And Temperature measurement



“STA Mode” And Temperature Diplay

Open the LuaLoader, click the “Upload File…”choose ESP8266 NodeMCU kit demo code -> P18_DHT11_Temperature_OLED_Display-> TCP Server->init.lua. After the upload completed then click “Download ” to download the init.lua file. Then Click “Upload File”, Choose the”main.lua” in the same folder, download it.

The following, set another serial port which connect the NodeMCU (“AP mode”) and connecting , then click the “Upload File…” after connecting successful, choose ESP8266 NodeMCU kit demo code -> P18_DHT11_Temperature_OLED_Display ->DHT11-> init.lua. After the upload completed then click “Download ” to download the init.lua file. Then Click “Upload File”, Choose the” dht11.lua” in the same folder, download it.

TCP Serve:

```
-- init.lua --
--tcp server--
print("ESP8266 Server")
wifi.setmode(wifi.STATIONAP);
wifi.ap.config({ssid="test2",pwd="12345678"});
print("Server IP Address:",wifi.ap.getip())
-- Run the main file
dofile("main.lua")

-- main.lua --
function init_OLED(sda,scl) --Set up the u8glib lib
     sla = 0x3c
     i2c.setup(0, sda, scl, i2c.SLOW)
     disp = u8g.ssd1306_128x64_i2c(sla)
     disp:setFont(u8g.font_6x10)
     disp:setFontRefHeightExtendedText()
     disp:setDefaultForegroundColor()
     disp:setFontPosTop()
end
-- Connect--
print('\nAll About Circuits main.lua\n')
init_OLED(5,6) --Run setting up
i=0
 -- Start a simple http server
if srv~=nil then
```

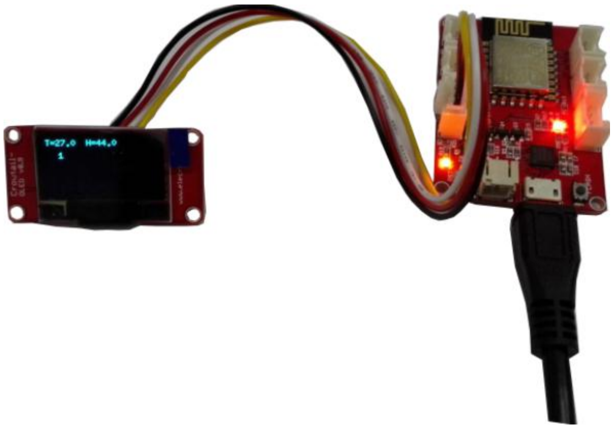
```
  srv:close()
end
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
  conn:on("receive",function(conn,payload)
    print(payload)
    i=i+1
     disp:firstPage()
    repeat
        disp:drawStr(0,0,payload) --Print the data out
         disp:drawStr(15,15,i) --Print the data out
    until disp:nextPage() == false
    conn:send("<h1> Hello, NodeMCU!!! </h1>")
  end)
  conn:on("sent",function(conn) conn:close() end)
end)
```

DHT11:

```
--init.lua
-- tcp client
print("ESP8266 Client1")
wifi.sta.disconnect()
tmr.delay(100)
wifi.setmode(wifi.STATION)
wifi.sta.config("test2","12345678") -- connecting to server
wifi.sta.connect()
i=0
tmr.alarm(0,2000, 1, function()
    if wifi.sta.getip()== nil then
        print("IP unavaiable, Waiting...")
        i=i+1
        if(i>10) then
            print("restart nodeMCU")
            node.restart()
        end
        wifi.sta.disconnect()
        wifi.sta.connect()
    else
        tmr.stop(0)
        print("Config done, IP is "..wifi.sta.getip())
        dofile("dht11.lua")
    end
end)
```

```
--dht11.lua--
-- Measure temperature, humidity and post data to thingspeak.com
-- 2014 OK1CDJ
-- DHT11 code is from esp8266.com
---Sensor DHT11 is conntected to GPIO0
pin = 1
Humidity = 0
HumidityDec=0
Temperature = 0
TemperatureDec=0
Checksum = 0
ChecksumTest=0

function getTemp()
Humidity = 0
HumidityDec=0
Temperature = 0
TemperatureDec=0
Checksum = 0
ChecksumTest=0

--Data stream acquisition timing is critical. There's
--barely enough speed to work with to make this happen.
--Pre-allocate vars used in loop.
bitStream = {}
for j = 1, 40, 1 do
        bitStream[j]=0
end
bitlength=0
gpio.mode(pin, gpio.OUTPUT)
gpio.write(pin, gpio.LOW)
tmr.delay(20000)      --default 20000
--Use Markus Gritsch trick to speed up read/write on GPIO
gpio_read=gpio.read
gpio_write=gpio.write
gpio.mode(pin, gpio.INPUT)
--bus will always let up eventually, don't bother with timeout
while (gpio_read(pin)==0 ) do end
c=0
while (gpio_read(pin)==1 and c<100) do c=c+1 end
--bus will always let up eventually, don't bother with timeout
while (gpio_read(pin)==0 ) do end
c=0
while (gpio_read(pin)==1 and c<100) do c=c+1 end
```

```
--acquisition loop
for j = 1, 40, 1 do
    while (gpio_read(pin)==1 and bitlength<10 ) do
        bitlength=bitlength+1
    end
    bitStream[j]=bitlength
    bitlength=0
    --bus will always let up eventually, don't bother with timeout
    while (gpio_read(pin)==0) do end
end

--DHT data acquired, process.

for i = 1, 8, 1 do
    if (bitStream[i+0] > 2) then
        Humidity = Humidity+2^(8-i)
    end
end
for i = 1, 8, 1 do
    if (bitStream[i+8] > 2) then
        HumidityDec = HumidityDec+2^(8-i)
    end
end
for i = 1, 8, 1 do
    if (bitStream[i+16] > 2) then
        Temperature = Temperature+2^(8-i)
    end
end
for i = 1, 8, 1 do
    if (bitStream[i+24] > 2) then
        TemperatureDec = TemperatureDec+2^(8-i)
    end
end
for i = 1, 8, 1 do
    if (bitStream[i+32] > 2) then
        Checksum = Checksum+2^(8-i)
    end
end
ChecksumTest=(Humidity+HumidityDec+Temperature+TemperatureDec) % 0xFF

print ("Temperature: "..Temperature.."."..TemperatureDec)
print ("Humidity: "..Humidity.."."..HumidityDec)
print ("ChecksumReceived: "..Checksum)
print ("ChecksumTest: "..ChecksumTest)
end

--- Get temp and send data to thingspeak.com
function sendData()
getTemp()
tmr.stop(2)
-- conection to thingspeak.com
print("Sending data to TCP Server")
conn=net.createConnection(net.TCP, 0)
conn:on("receive", function(conn, payload) print(payload) end)
-- api.thingspeak.com 184.106.153.149
conn:connect(80,'192.168.4.1')
conn:send("T="..Temperature.."."..TemperatureDec.."    H="..Humidity.."."..HumidityDec.."\r\n")
conn:on("sent",function(conn)
                                    print("Closing connection")
                                    conn:close()
                              end)
conn:on("disconnection", function(conn)
                                    print("Got disconnection...")
   end)
end
-- send data every X ms to thing speak
tmr.alarm(2, 300, 1, function() sendData() end ) --60000
```



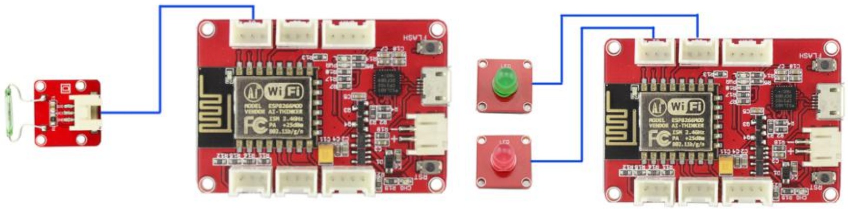
Lesson19: Wireless Toliet Led Indicator

When we sit on the seat of office, sometimes want to go the toilet, but do not know the toilet there is not occupied. This lesson is designed primarily a remote indicator toilet, do not need to go to the toilet can also know whether someone is using the toilet, this greatly facilitates our daily life. Of course, this design idea can be applied in many other occasions of life.

Material:

Crowtail- ESP8266 NodeMCU x 2
Crowtail- Magnetic Switch x1
Crowtail- LED x1

Hardware Connection



Open the LuaLoader, click the "Upload File…"choose ESP8266 NodeMCU kit demo code -> P19_Wireless_Toliet_LED_Idicator-> Client->init.lua. After the upload completed then click "Download " to download the init.lua file.

Note!!! This init.lua file uploaded should choose the NodeMCU which connect with Magnetic Switch.

Then open the LuaLoader, choose the correct serial port for the NodeMCU which connect with two indicator LED click the "Upload File…"choose ESP8266 NodeMCU kit demo code -> P19_Wireless_Toliet_LED_Idicator-> Server->init.lua. After the upload completed then click "Download " to download the init.lua file.

Client:

```
-- ESP8266 Client
print("ESP8266 Client1")
wifi.sta.disconnect()
wifi.setmode(wifi.STATION)
wifi.sta.config("test","12345678") -- connecting to server
wifi.sta.connect()
print("Looking for a connection")
pin=1
gpio.mode(pin, gpio.INPUT,gpio.FLOAT)
pin_value=0

tmr.alarm(1, 2000, 1, function()
        if(wifi.sta.getip()~=nil) then
                tmr.stop(1)
               tmr.start(2)
                print("Connected!")
                print("Client IP Address:",wifi.sta.getip())
```

```
          cl=net.createConnection(net.TCP, 0)
          cl:connect(80,"192.168.4.1")
          tmr.alarm(2,1000, 1, function()
            if(wifi.sta.getip()==nil) then
                   wifi.sta.disconnect()
                   wifi.sta.config("test","12345678") -- connecting to server
                 wifi.sta.connect()
                 print("reconnection")
                 tmr.start(1)
                    tmr.stop(2)
              end
            pin_value=gpio.read(pin)
              if(pin_value==0) then
            cl:send("10")
              print("send 10")
              else
              cl:send("11")
              print("send 11")
              end
          end)
      else
         print("Connecting...")
      end
end)
```

Server:

```
-- ESP8266 Server
print("ESP8266 Server")
wifi.setmode(wifi.STATIONAP);
wifi.ap.config({ssid="test",pwd="12345678"});
print("Server IP Address:",wifi.ap.getip())
pin_value=0
pin=1
pin1=2
gpio.mode(pin, gpio.OUTPUT)
gpio.mode(pin1, gpio.OUTPUT)
gpio.write(pin, gpio.LOW)     --Red LED
gpio.write(pin1, gpio.HIGH)   --Green LED
sv = net.createServer(net.TCP)
sv:listen(80, function(conn)
    conn:on("receive", function(conn, receivedData)
        print("Received Data: " .. receivedData)
          pin_value=receivedData
          tmr.alarm(0, 100, 1, function()
              print("inter tmr")
              if(pin_value=="10") then
              gpio.write(pin, gpio.HIGH)
              gpio.write(pin1, gpio.LOW)
              elseif(pin_value=="11") then
                   gpio.write(pin, gpio.LOW)
                   gpio.write(pin1, gpio.HIGH)
              end
         end)
    end)
    conn:on("sent", function(conn)
      collectgarbage()
    end)
end)
```

The lessons are over here, and you have learned the basic application of ESP8266 NodeMCU, but the usage of ESP8266 is not only these. There are many other examples of the ESP8266 in the folder files, you can try them by yourself And if you need other Crowtail modules you can visit:http://www.elecrow.com/ . Hope you have a enjoy time.