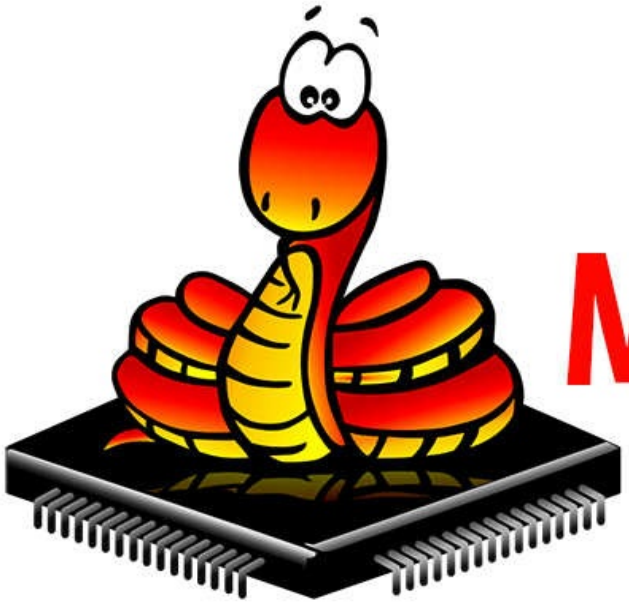


MicroPython for ESP8266 Development Workshop

Agus Kurniawan

Copyright

MicroPython for ESP8266 Development Workshop

Agus Kurniawan

1st Edition, 2016

Table of Contents

Preface

This book was written to help anyone want to get started with MicroPython development for ESP8266 boards. It describes the basic elements of MicroPython development.

Agus Kurniawan

Depok, November 2016

1. Preparing Development Environment

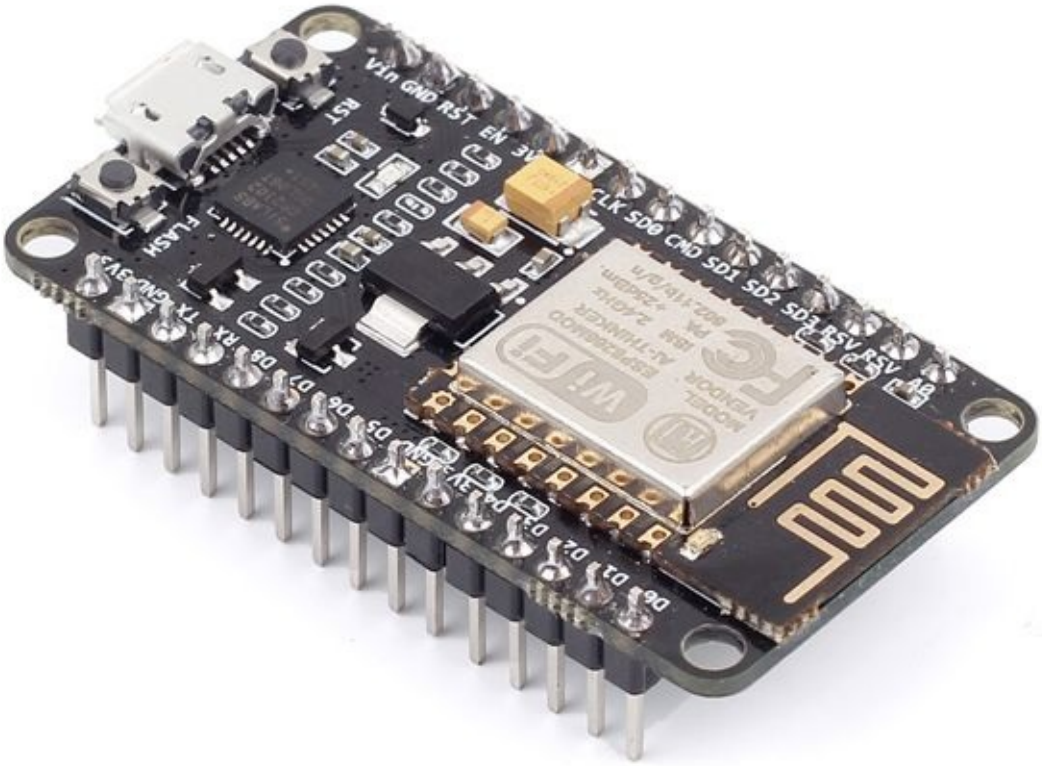
1.1 MicroPython Boards

MicroPython is a lean and efficient implementation of the Python programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.

We can deploy MicroPython on ESP8266 boards. For instance, you can deploy on these boards:

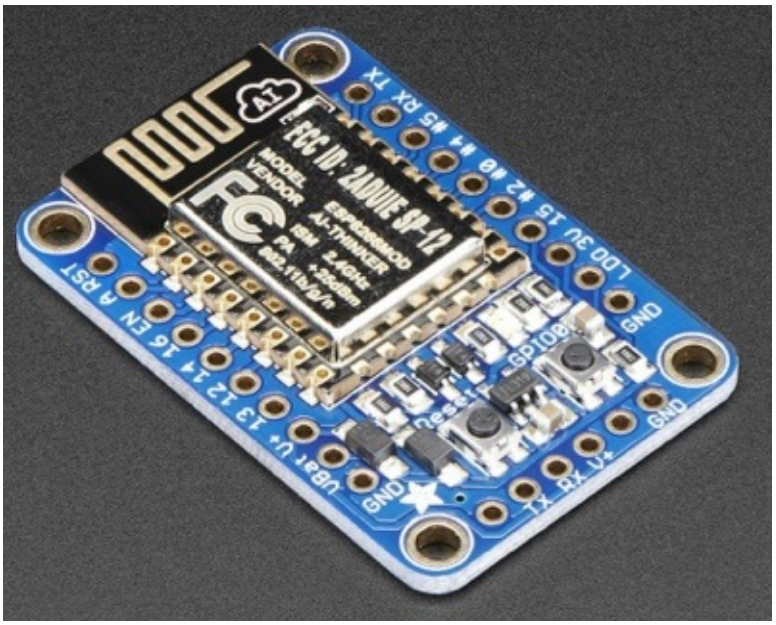
- NodeMCU, http://nodemcu.com/index_en.html
- SparkFun ESP8266 Thing, https://www.sparkfun.com/products/13231
- Adafruit HUZZAH ESP8266 Breakout, https://www.adafruit.com/product/2471
- Adafruit Feather HUZZAH with ESP8266 WiFi, https://www.adafruit.com/product/2821

NodeMCU v2 board:

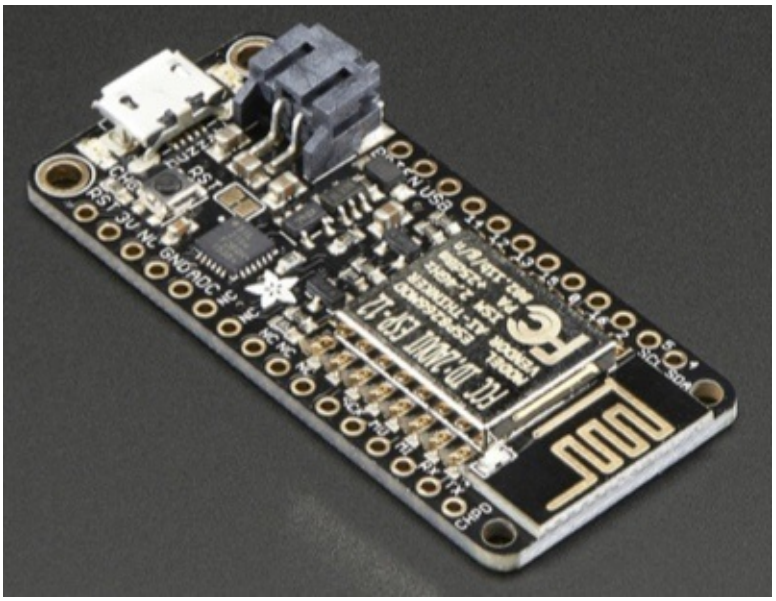


SparkFun ESP8266 Thing.

Adafruit HUZZAH ESP8266 Breakout.



Adafruit Feather HUZZAH with ESP8266 WiFi.


RST
CHPD
EN
RST
3V
NC
GND
ADC
NC
NC
SCL
SDA
RX
TX
CHPD

1.2 Electronics Components

We need electronic components to build our testing, for instance, Resistor, LED, sensor devices and etc. I recommend you can buy electronic component kit. We can use electronics kit from Arduino to be developed on MicroPython board. The following is a list of electronics kit which can be used in our case.

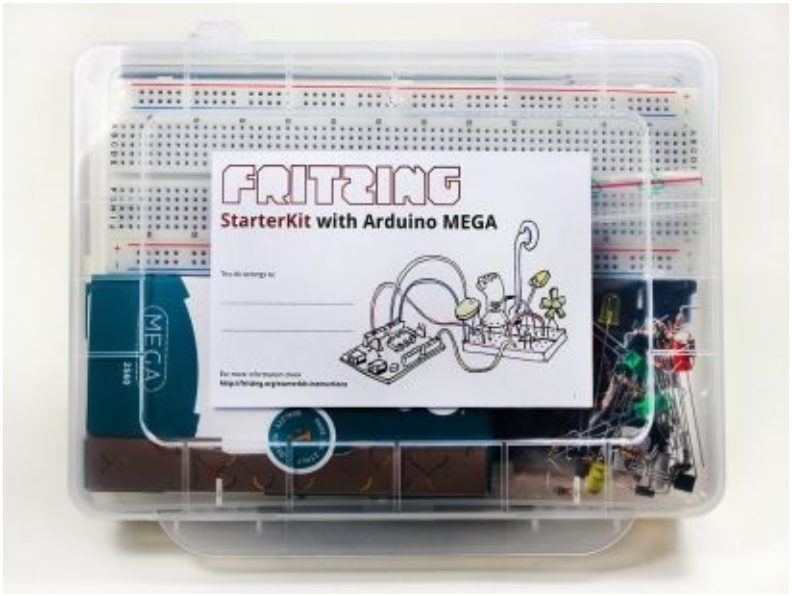
1.2.1 Arduino Starter Kit

Store website: http://arduino.cc/en/Main/ArduinoStarterKit

1.2.2 Fritzing

Store website: http://shop.fritzing.org/ .

You can buy Fritzing Starter Kit with Arduino UNO or Fritzing Starter Kit with Arduino Mega.



1.2.3 Cooking-Hacks: Arduino Starter Kit

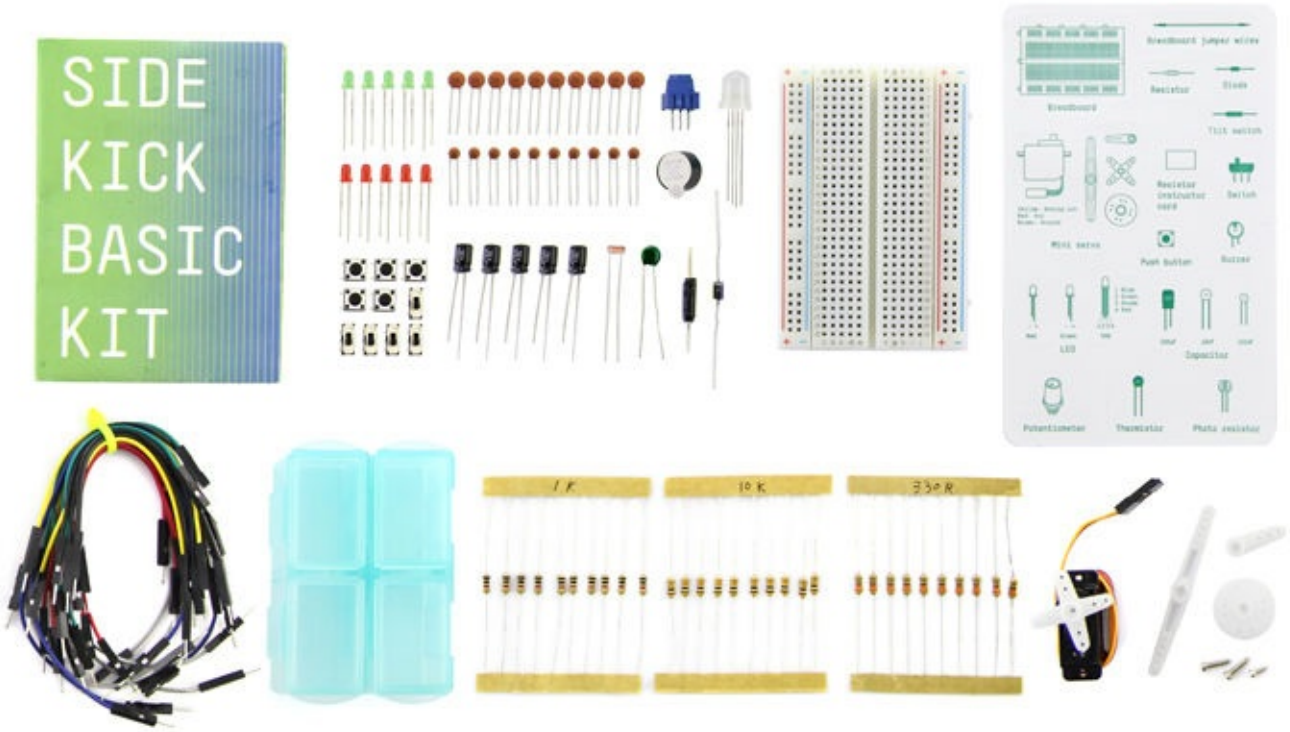
Store website: http://www.cooking-hacks.com/index.php/shop/arduino/starter-kits/arduino-starter-kit.html

1.2.4 Arduino Sidekick Basic kit v2

Store website: http://www.seeedstudio.com/depot/Sidekick-Basic-Kit-for-Arduino-V2-p-1858.html

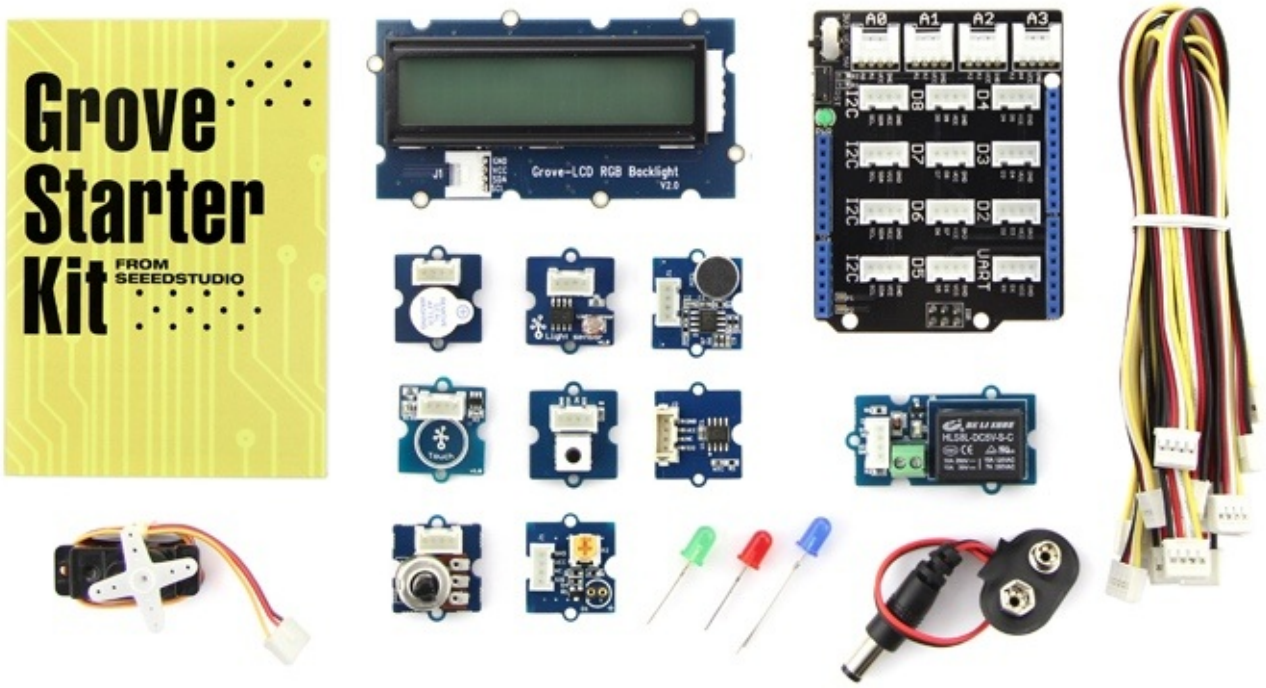
You also can find this kit on this online store.

http://www.exp-tech.de/seeed-studio-sidekick-basic-kit-for-arduino-v2



1.2.5 Grove - Starter Kit for Arduino

Another option, you can buy this kit on Seeedstudio, http://www.seeedstudio.com/depot/Grove-Starter-Kit-for-Arduino-p-1855.html .



1.2.6 DFRobot - Arduino Kit for Beginner v3

DFRobot provides Arduino kit too. You can buy it on the following website.

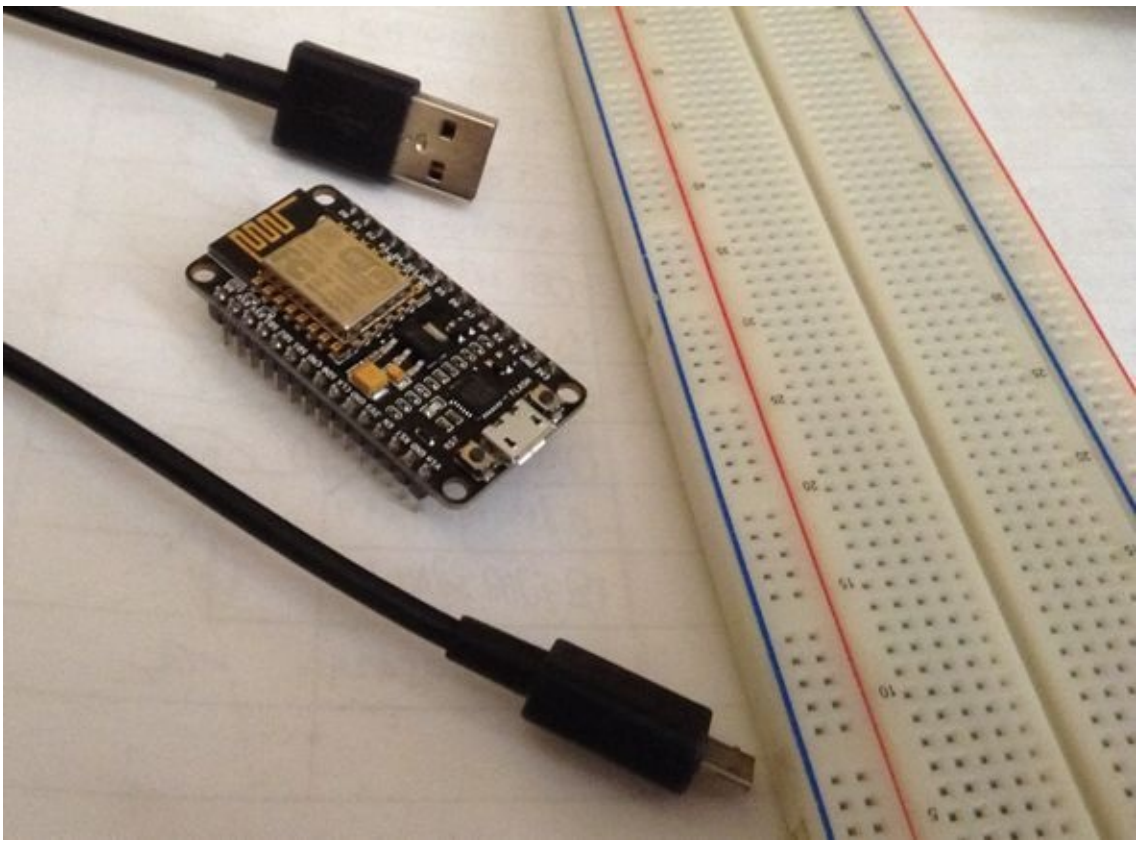
http://www.dfrobot.com/index.php?route=product/product&path=35_49&product_id=345

1.3 Development Tools

To develop app with MicroPython target, we can use any editor. You can learn how to install it on chapter 2.

1.4 Testing

For testing, I used NodeMCU v2 and SparkFun ESP8266 Thing boards for MicroPython on Windows, Linux and Mac.



I also used Arduino Sidekick Basic kit for electronic components and some sensor and actuator devices.

2. Setting Up MicroPython

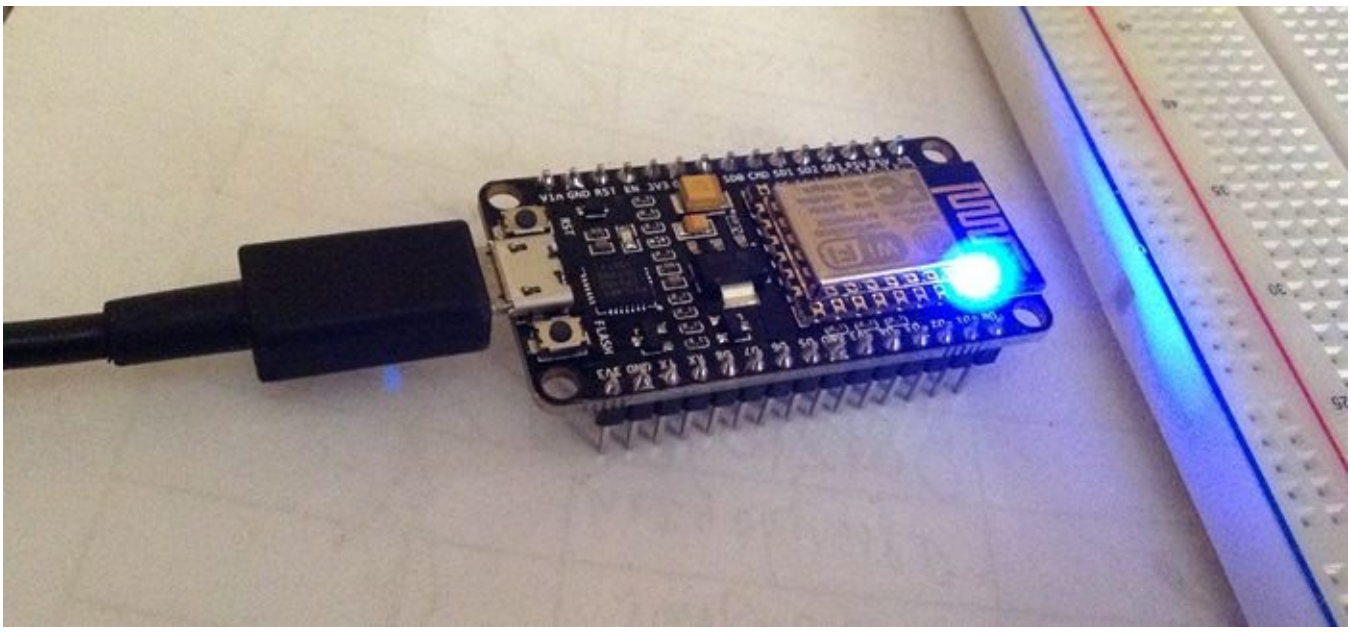
This chapter explains how to work on setting up MicroPython board.

2.1 Getting Started

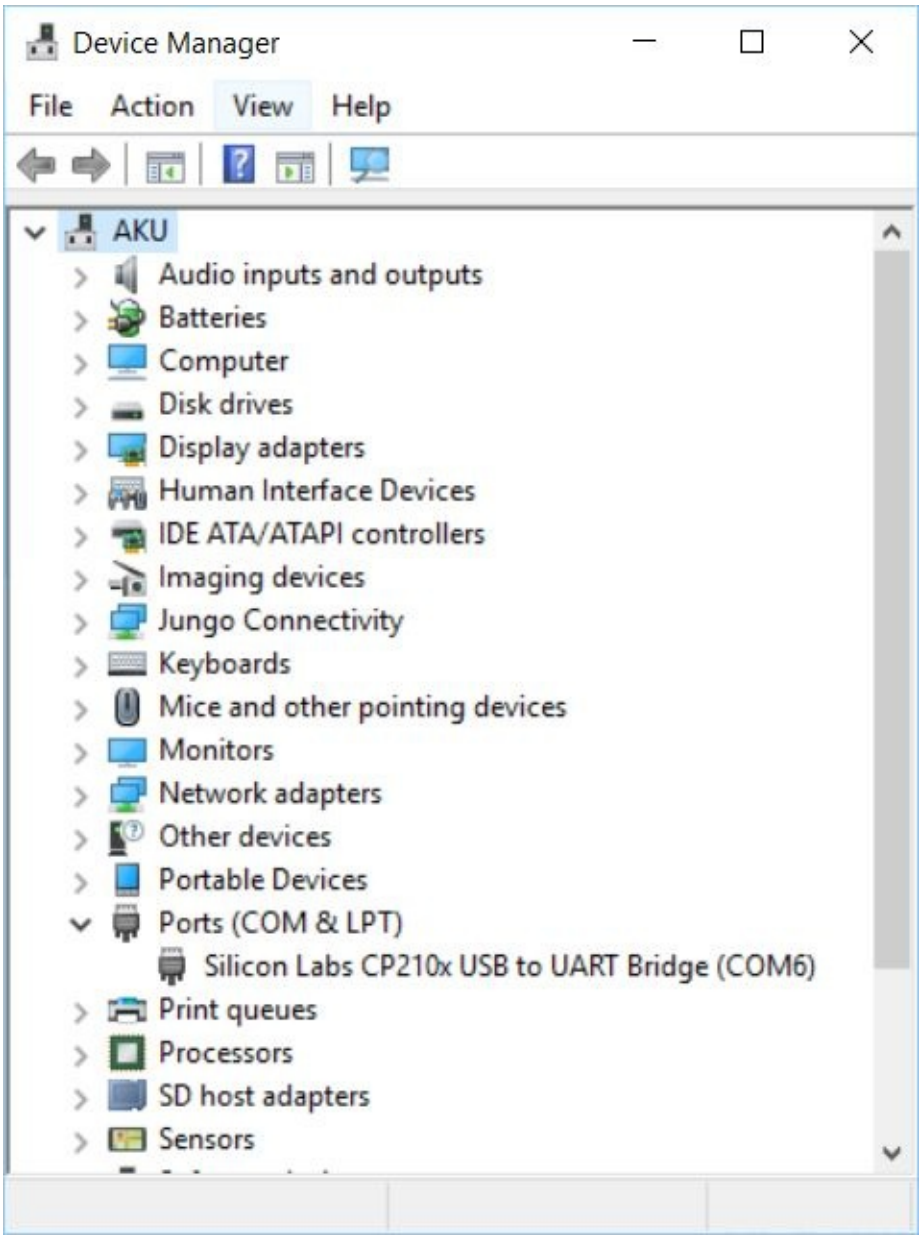
In this chapter, we learn how to get started with MicroPython board. We try to reflash the latest MicroPython firmware and then test some basic scripts. For testing, I use NodeMCU as sample for MicroPython board.

2.2 Connecting MicroPython Boards to Computer

Firstly, you connect MicroPython board to PC via USB/microUSB cable. After connected, you may get lighting on blue LED, for instance for, NodeMCU board.

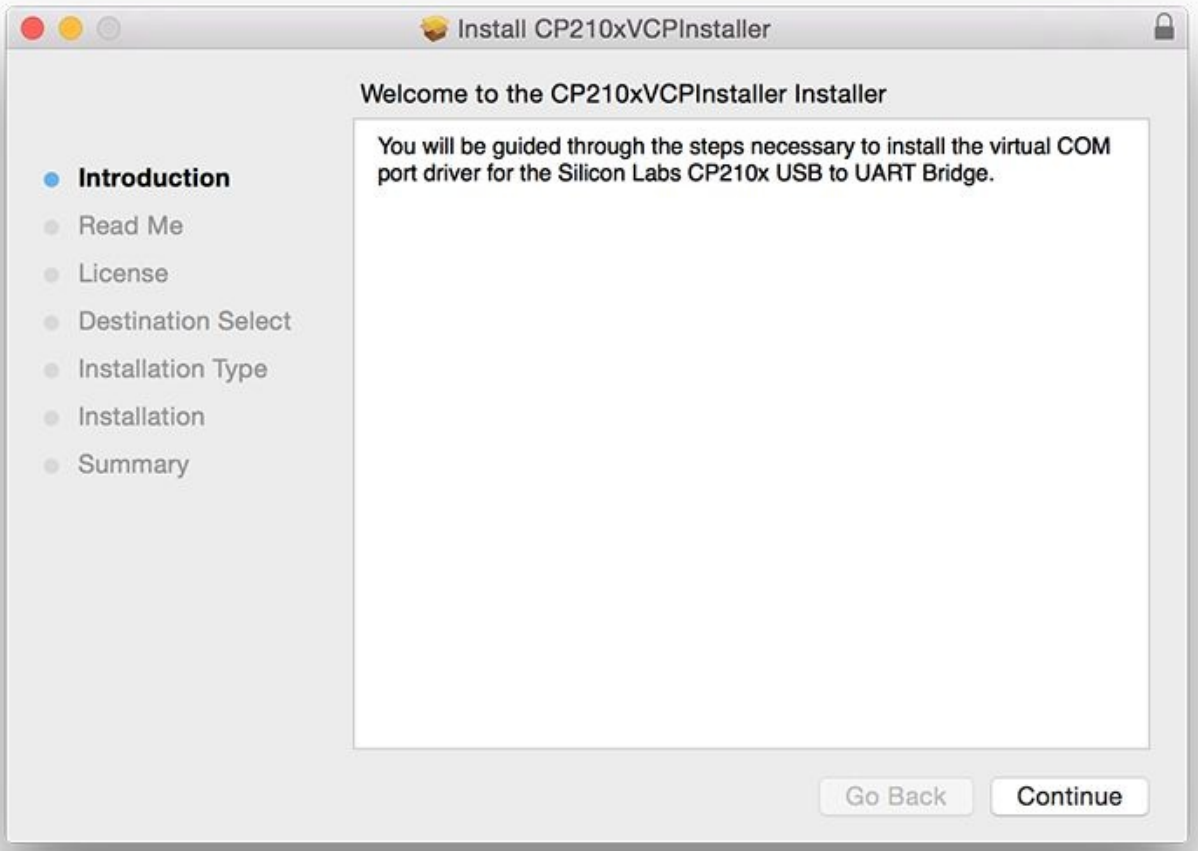


If you are working on Windows platform, open Device Manager, you should see NodeMCU board detected on Ports (COM & LPT) as Silicon Labs



However, if you get a problem about NodeMCU hardware driver, you update that driver. You also can install a driver from Silicon Labs because NodeMCU v2 uses this chip, from https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx . Please download it based on your platform.

For instance, you download the driver for mac. Just follow instructions.

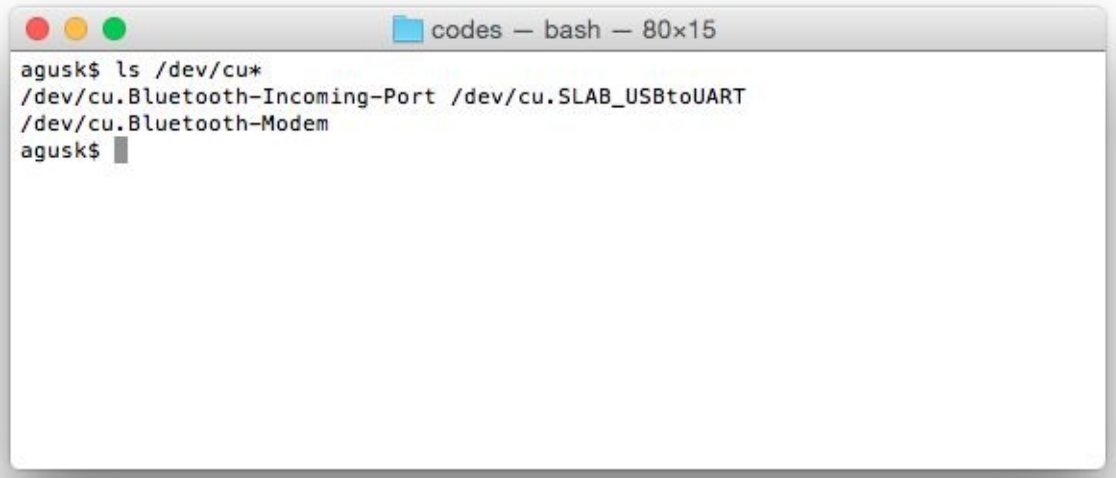


After installed, you verify it. For instance, on my OS X it has detected. Type this command.

```
ls /dev/cu*
```

For Linux, you can use ls /dev/tty* .

Then, you see a list of driver. This board is recognized as /dev/cu.SLAB_USBtoUART .



If you use SparkFun ESP8266 Thing, you need additional tool such as serial USB tool to communicate with ESP8266.

2.3 Flashing The Latest MicroPython Firmware

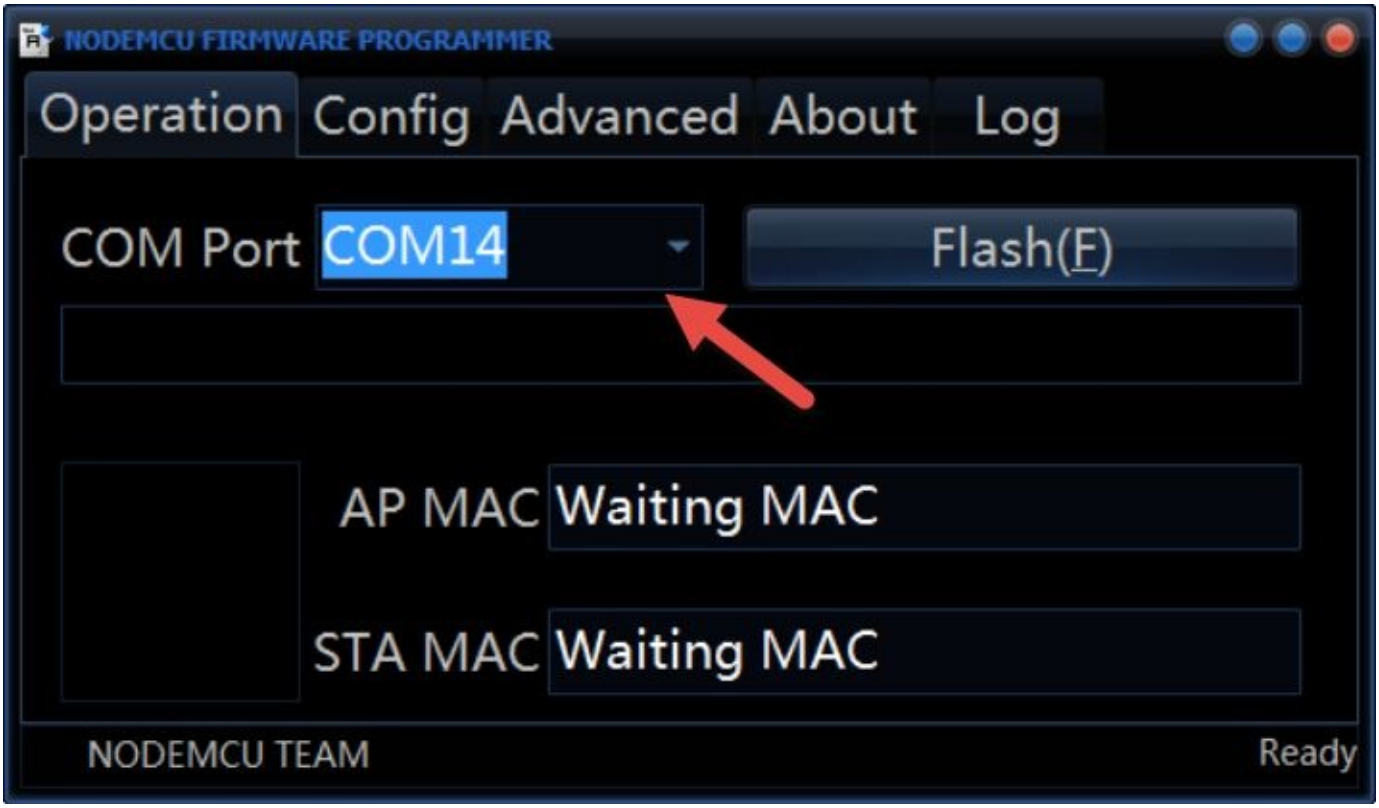
In this section, we try to flash the latest MicroPython firmware. You can get the latest MicroPython firmware https://micropython.org/download#esp8266. Download *.bin file.

In this scenario, I try to flash MicroPython firmware on Windows, OSX and Linux.

2.3.1 Windows Platform

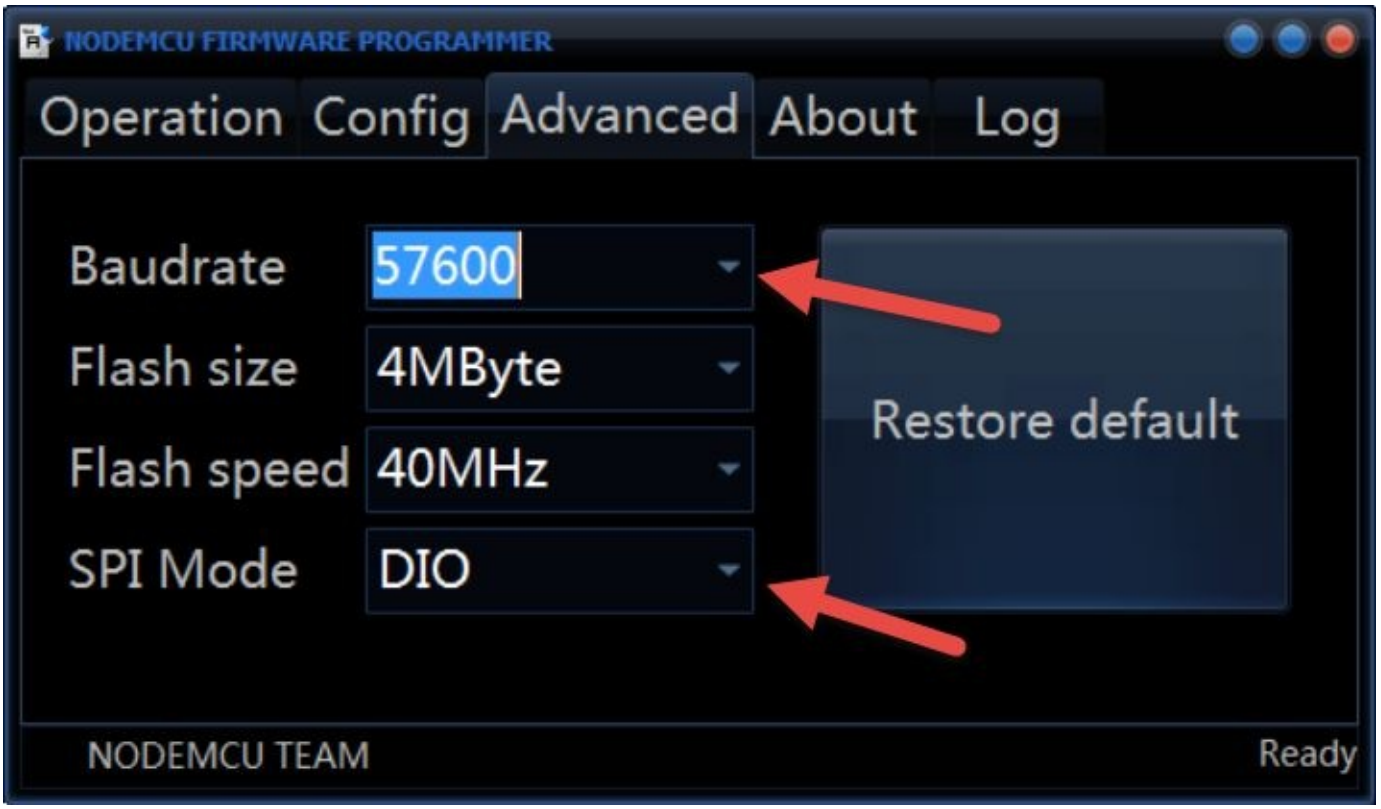
In this section, I show you how to flash it on Windows platform.

To flash firmware, you can use NodeMCU flasher. Please download NodeMCU Flasher on https://github.com/nodemcu/nodemcu-flasher . There are two binary platforms, x86 and x64. Open the program so you get the dialog as below.

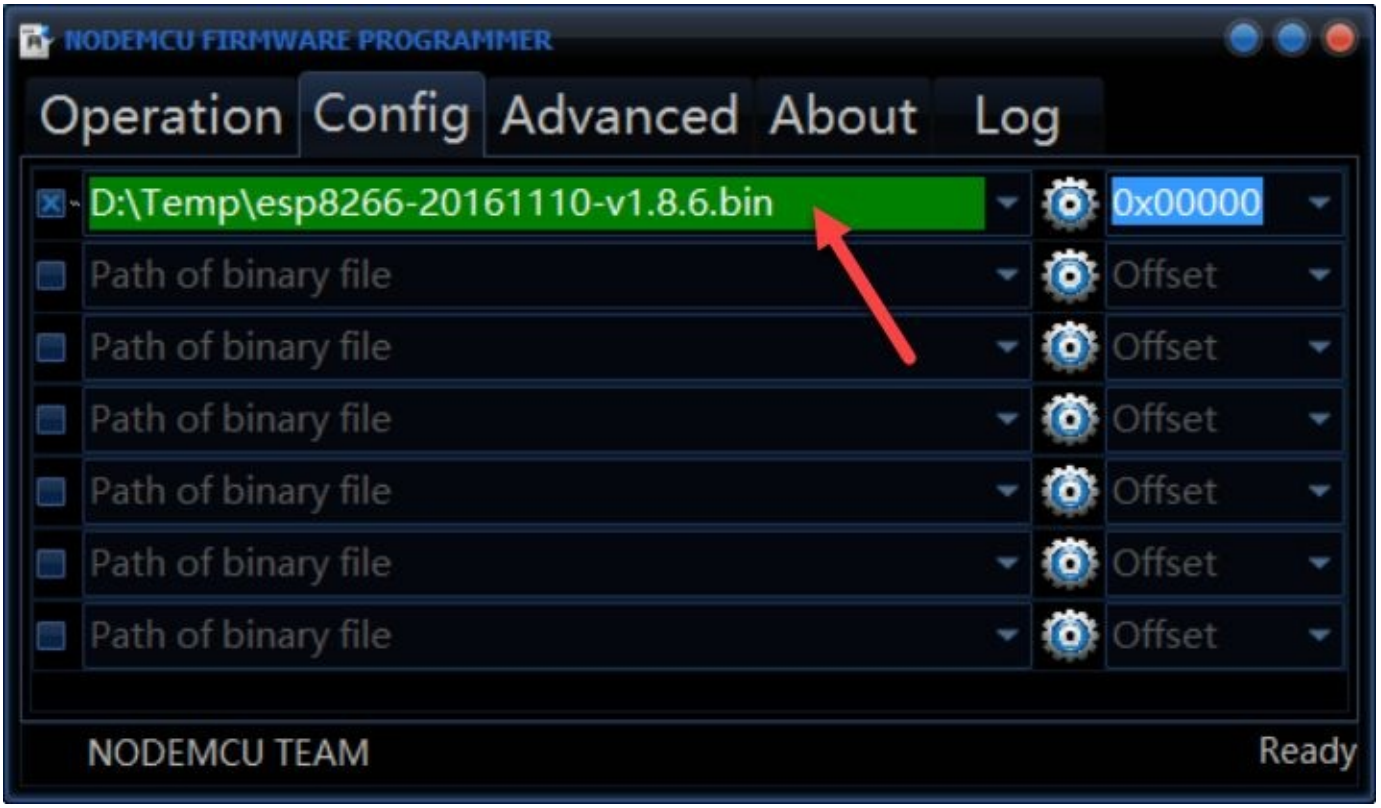


Select COM Port where NodeMCU was attached.

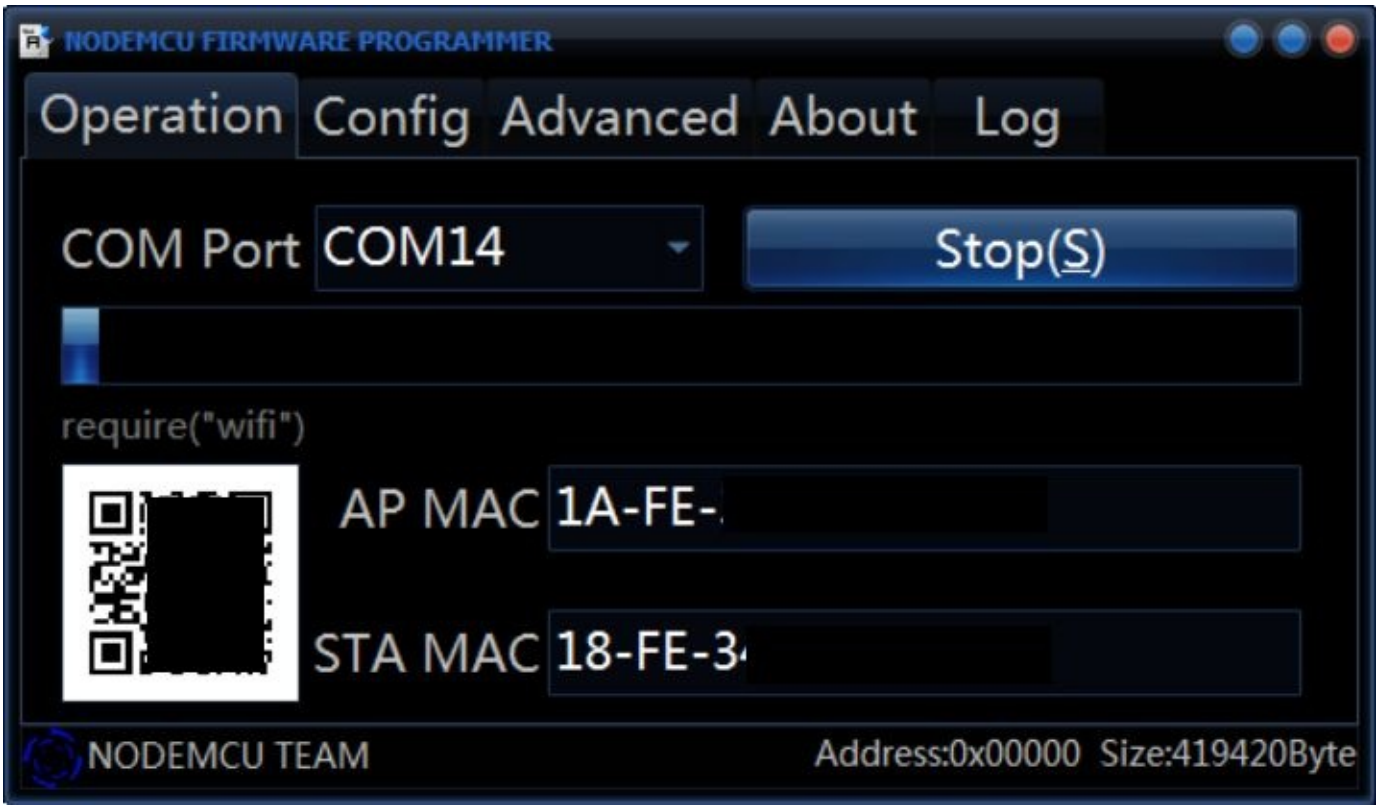
Then, click **Advanced** tab. Select Baudrate 5700 and SPI Mode is DIO. You also can change baudrate to 115200 if you get a problem on flashing MicroPython firmaware.



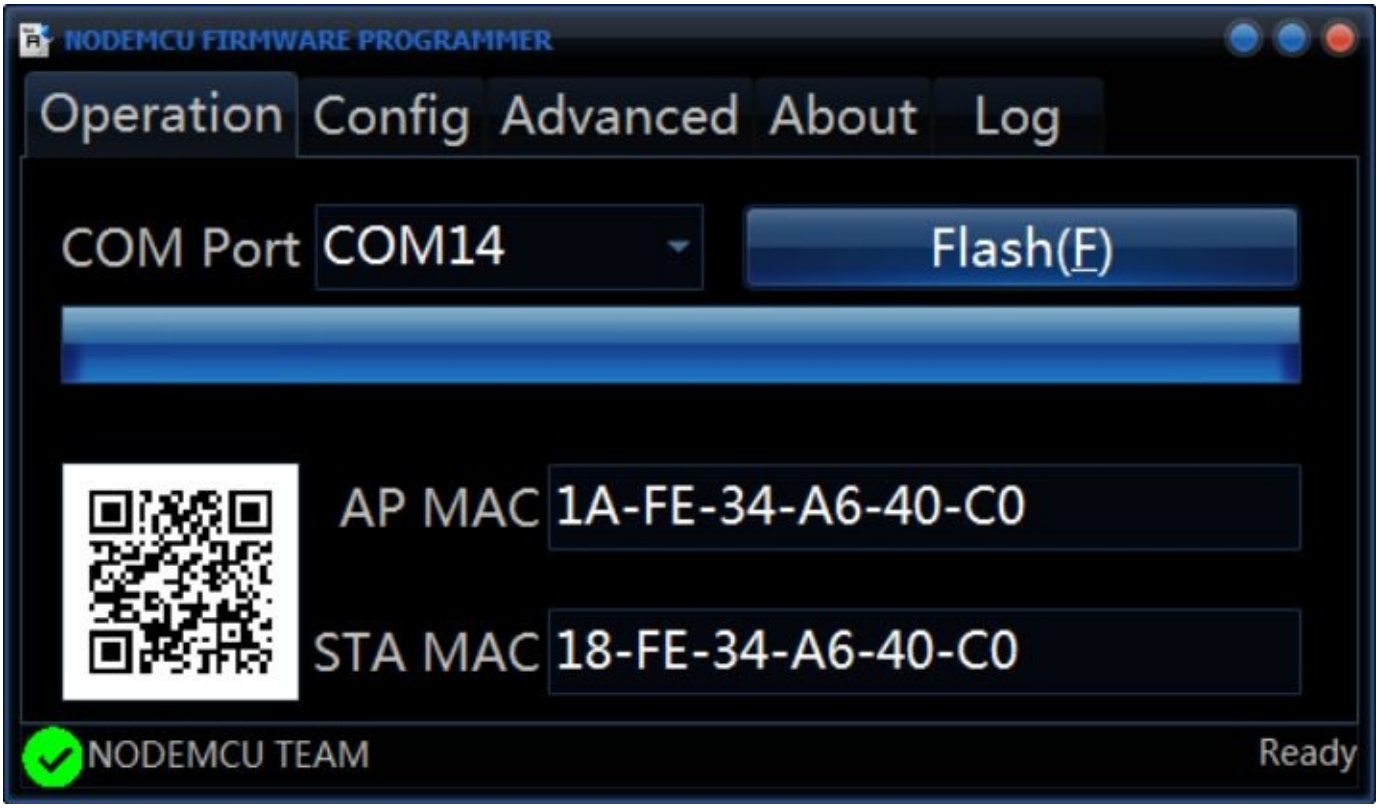
Click **Config** tab and select firmware file.



On Operation tab, you start by clicking **Flash** button.

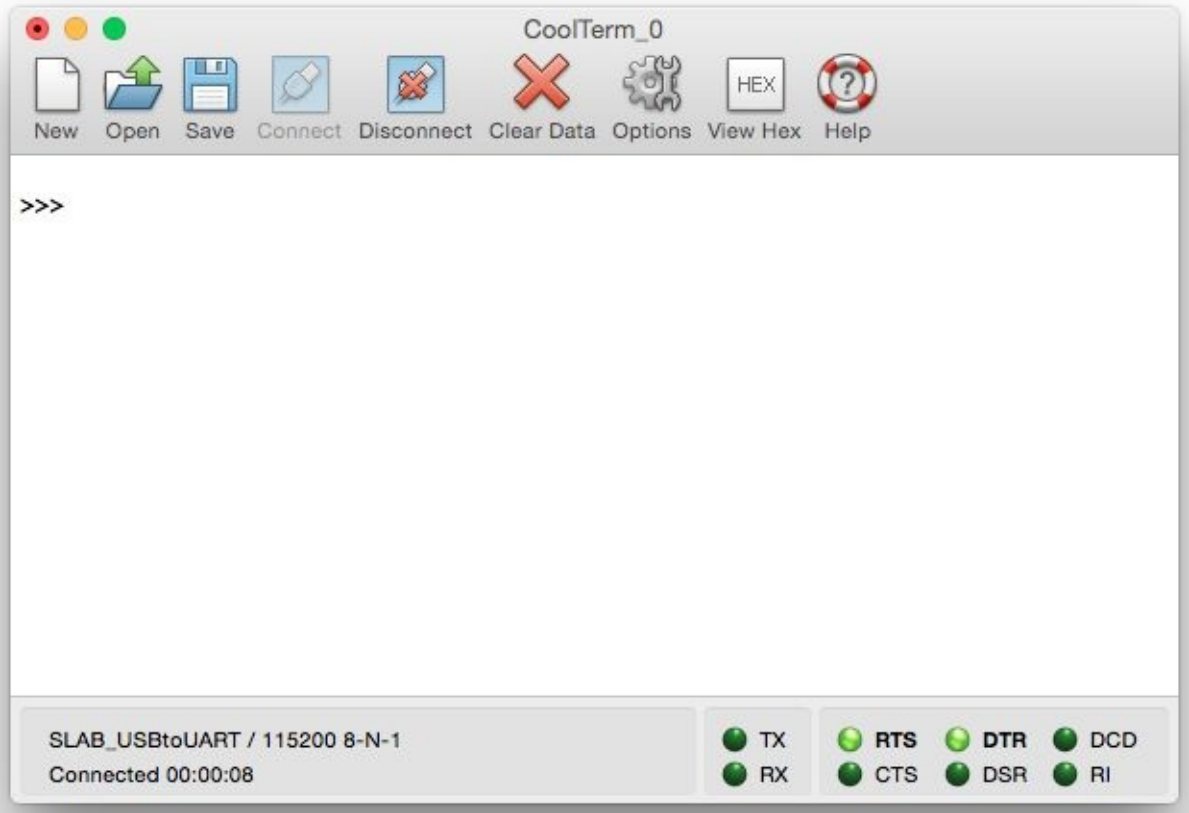


If done, you will get a notification.

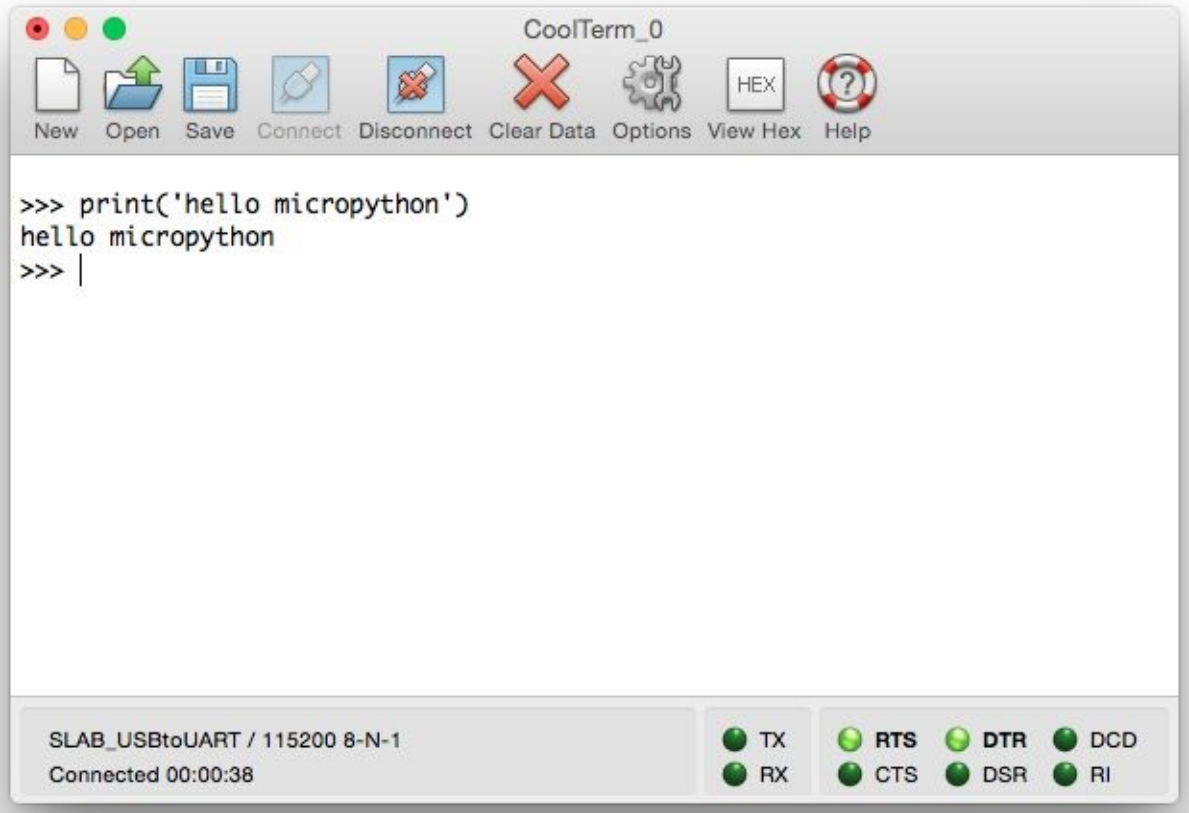


To test, you can use serial/UART tool and connect to NodeMCU board. Set NodeMCU serial port and set baudrate 115200.

Then, connect to the board. If success, you should see NodeMCU shell.



Try to write a simple script print(‘hello micropython’) so you will get a response.



2.3.2 Linux and OS X Platforms

If you're working on Linux or Mac, you can use esptool.py tool from https://github.com/espressif/esptool. This tool also is recommended for Windows platform. You can install it via pip.

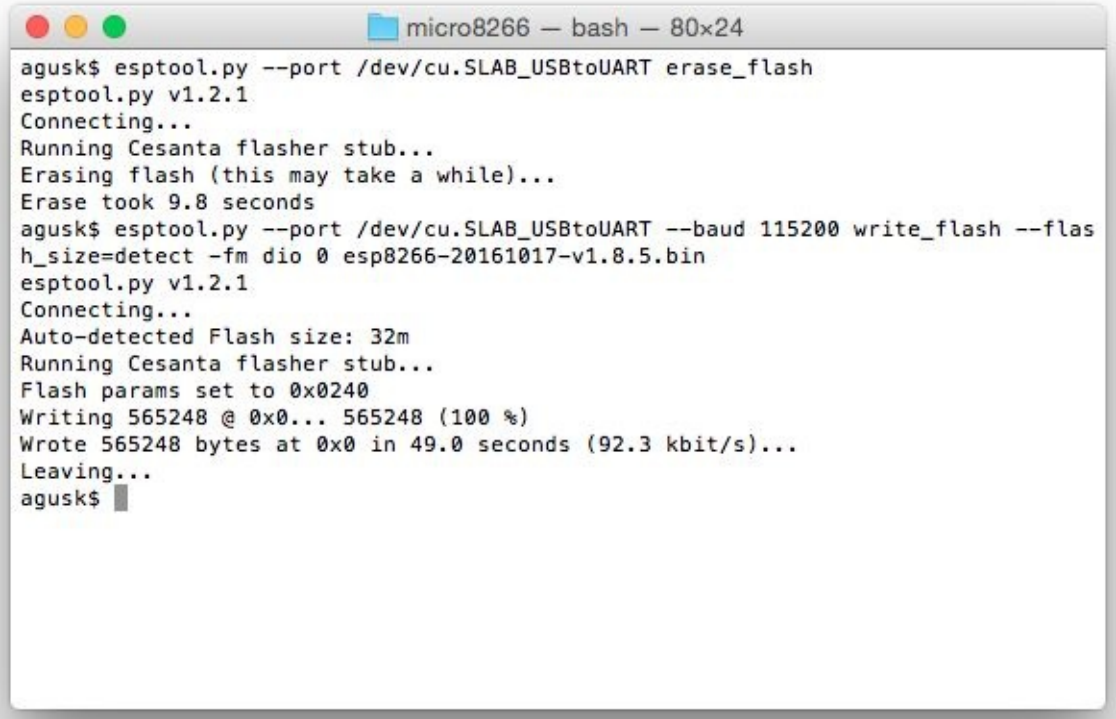
```
$ pip install esptool
```

You can download the latest MicroPython for ESP8266 on https://micropython.org/download#esp8266. Don't download the daily build firmware. Firstly, we clear the existing firmware on ESP8266 board. Then, we flash MicroPython.

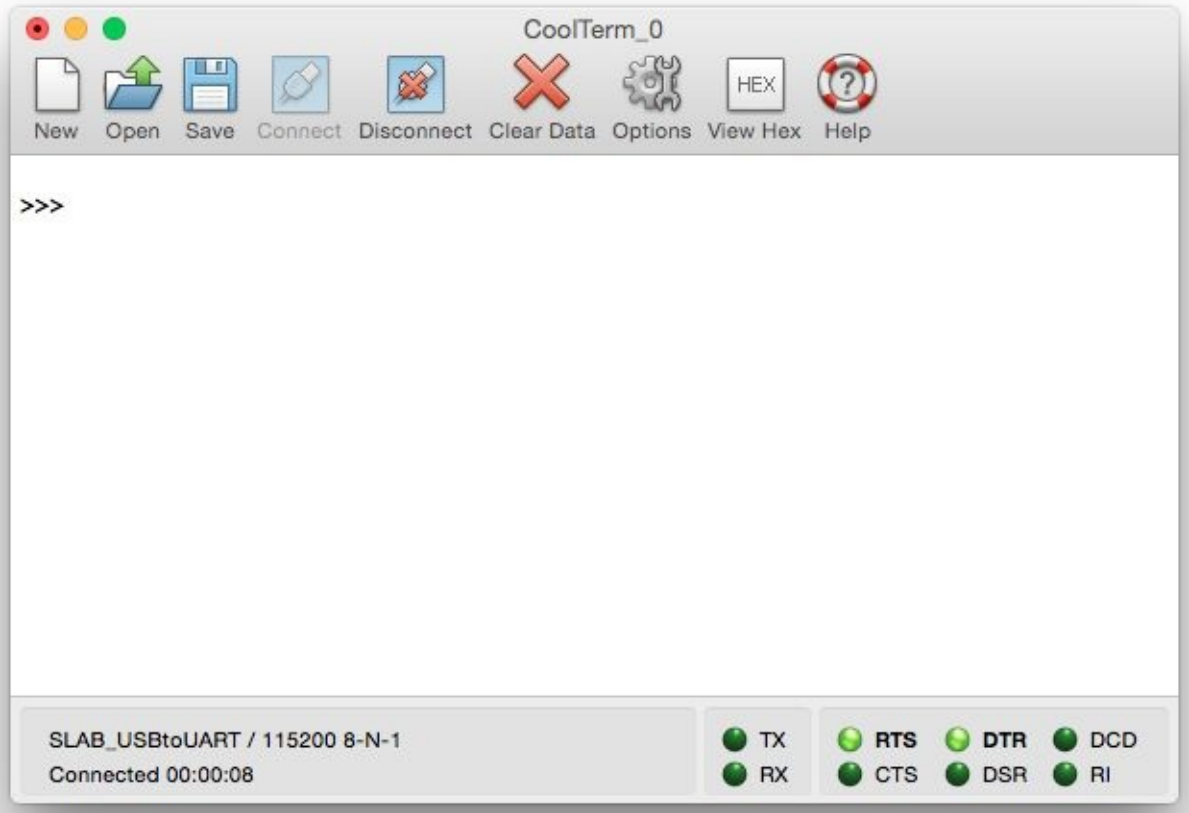
For instance, I flash my NodeMCU on port /dev/cu.SLAB_USBtoUART and MicroPython firmware, esp8266-20161017-v1.8.5.bin. The flash baudrate is 115200.

```
$ ./esptool.py --port /dev/cu.SLAB_USBtoUART erase_flash
$ ./esptool.py --port /dev/cu.SLAB_USBtoUART --baud 115200 write_flash -
```

Some ESP80266 boards may not use "-fm dio" parameter.



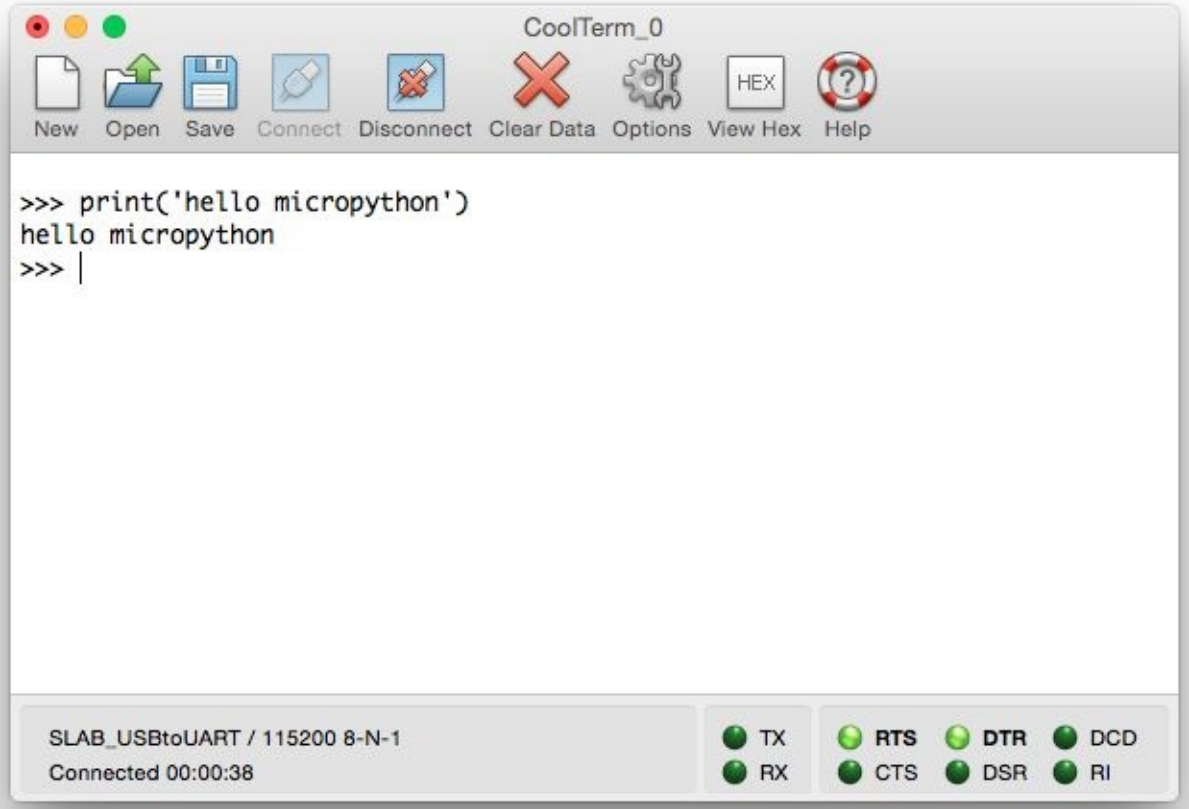
After completed flashing, you reset your board. Then, using a serial tool, such as CoolTerm, you should connect to ESP8266 board. Don’t forget to use baudrate 115200. After connected, you should see “>>>”. If you don’t see it, please press ENTER on your keyboard.



For testing, you can type this command.

```
>>> print('hello micropython')
```

You should see “hello micropython” response from MicroPython board.


CoolTerm_0
New
Open
Save
Connect
Disconnect
Clear Data
Options
HEX
View Hex
Help
>>> print('hello micropython')
hello micropython
>>>
SLAB_USBtoUART / 115200 8-N-1
Connected 00:00:38
TX
RX
RTS
CTS
DTR
DSR
DCD
RI

2.4 Development Tools

In this case, I show you several development tools you can use it in MicroPython boards based ESP8266 development.

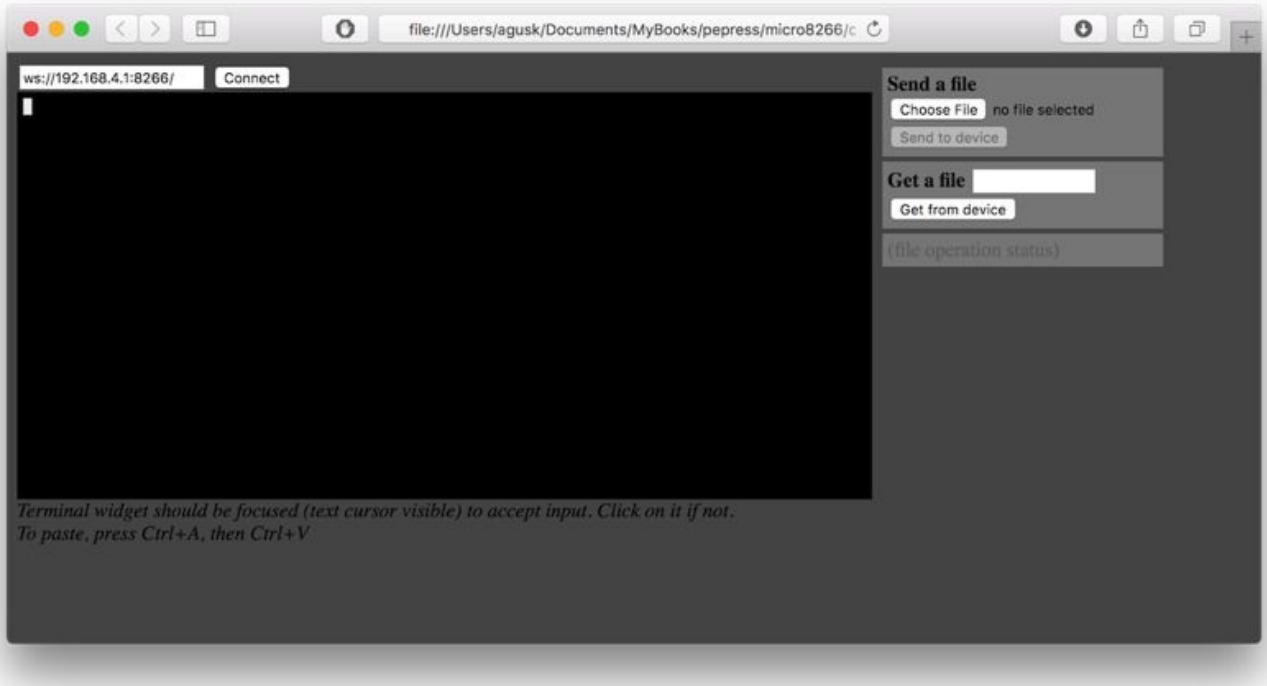
2.4.1 Serial/UART Tool

By default, you can use Serial/UART tools such as CoolTerm (Mac) or Putty (Windows) or screen (linux) .

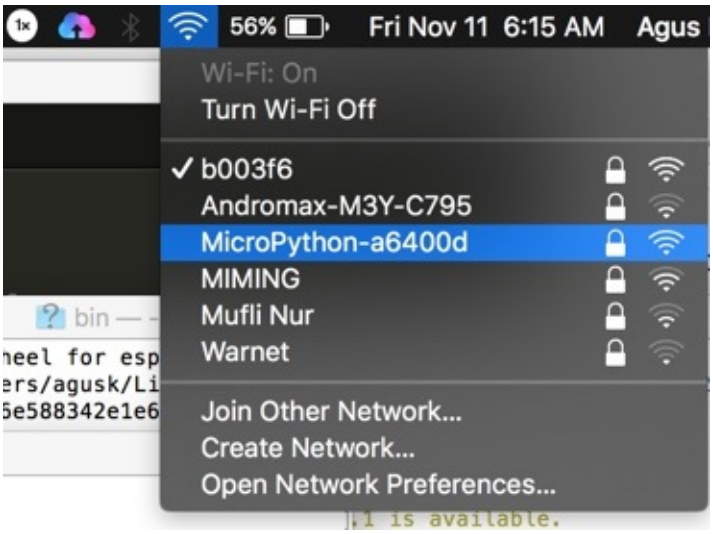
You also can use Realterm, http://realterm.sourceforge.net/ and Tera Term, http://ttssh2.osdn.jp/index.html.en .

2.4.2 WebREPL

WebREPL is a tool to access MicroPython remotely. We can upload our program file into the board. You can download and extract it on https://github.com/micropython/webrepl. Open webrepl.html file and you should see the form as follows.

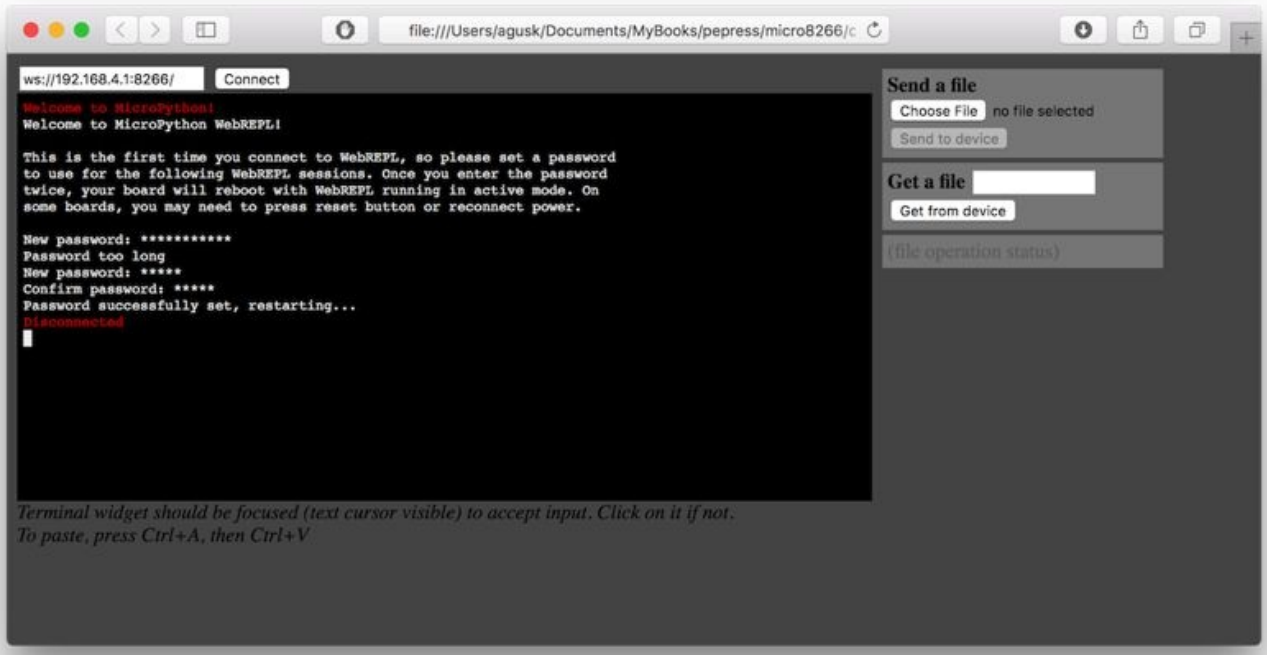


To access, you should join the WiFi network from MicroPython. You should see WiFi "MicroPython-xxxx". The default password/pin is **micropythoN**.



I use this tool to upload and execute MicroPython program.

After connected, click Connect on WebREPL. If this is the first time, you should be asked to change password. Please change password for MicroPython.



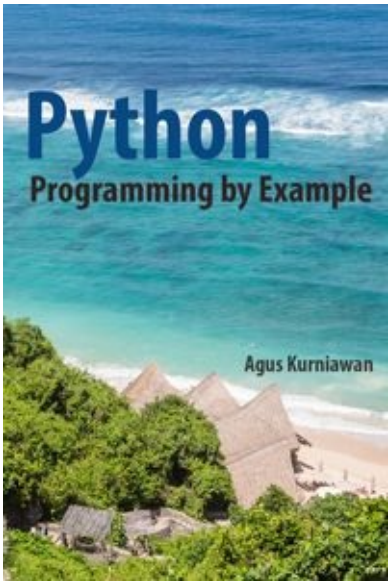
If done, you can perform Python shell on this web.


ws://192.168.4.1:8266/
Disconnect
Welcome to MicroPython!
Welcome to MicroPython WebREPL!
This is the first time you connect to WebREPL, so please set a password
to use for the following WebREPL sessions. Once you enter the password
twice, your board will reboot with WebREPL running in active mode. On
some boards, you may need to press reset button or reconnect power.
New password: **********
Password too long
New password: *****
Confirm password: *****
Password successfully set, restarting...
Disconnected
Welcome to MicroPython!
Password:
WebREPL connected
>>> a=3
>>> b=4
>>> a*b
12
>>>
Terminal widget should be focused (text cursor visible) to accept input. Click on it if not

2.5 Python programming

You should have basic knowledge about Python programming to develop MicropPython program. I recommend you read Python tutorial on books or online tutorial.

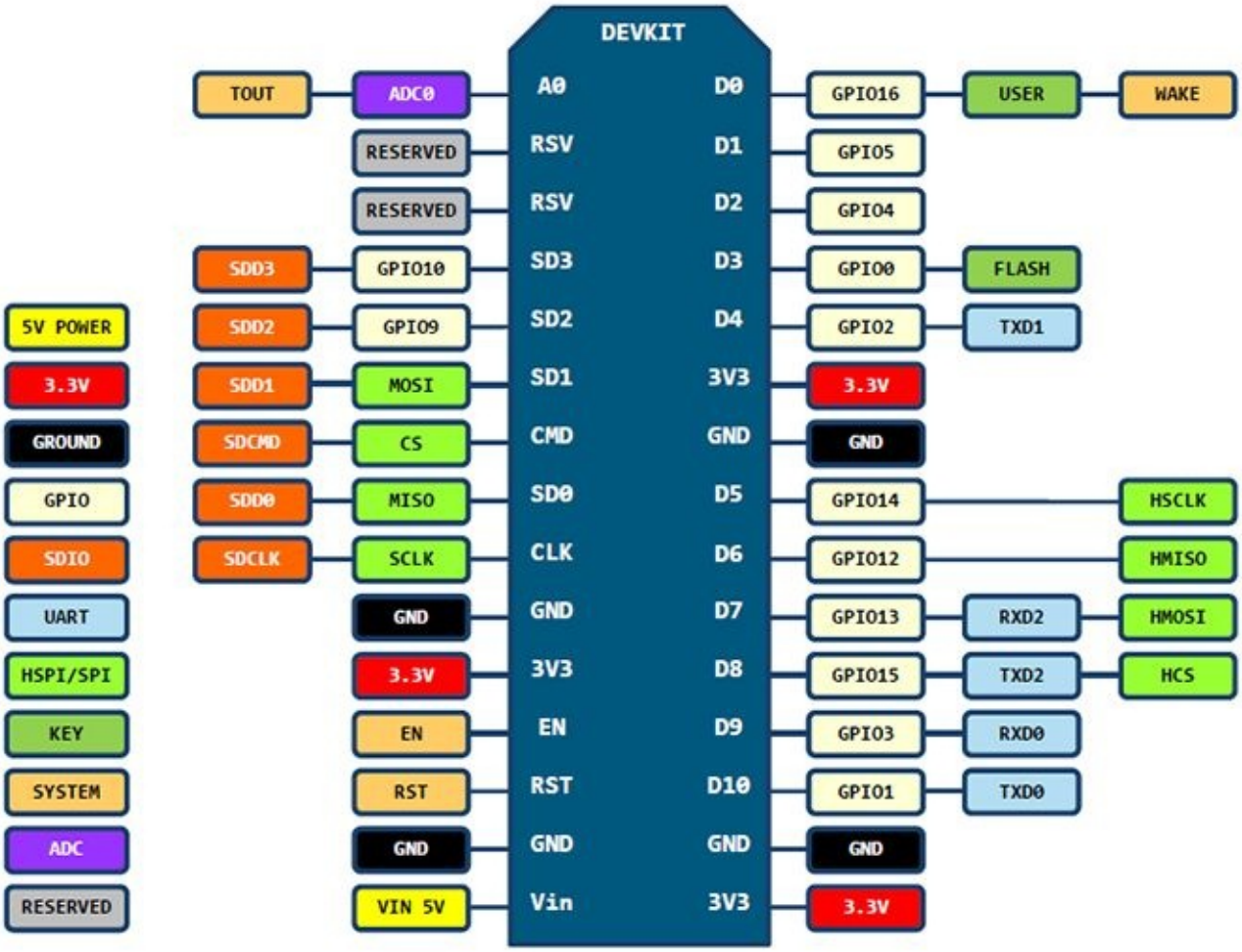
I also write a book about Python, with titled **Python Programming by Example**. You can review it on http://blog.aguskurniawan.net/post/pythonbook01.aspx.

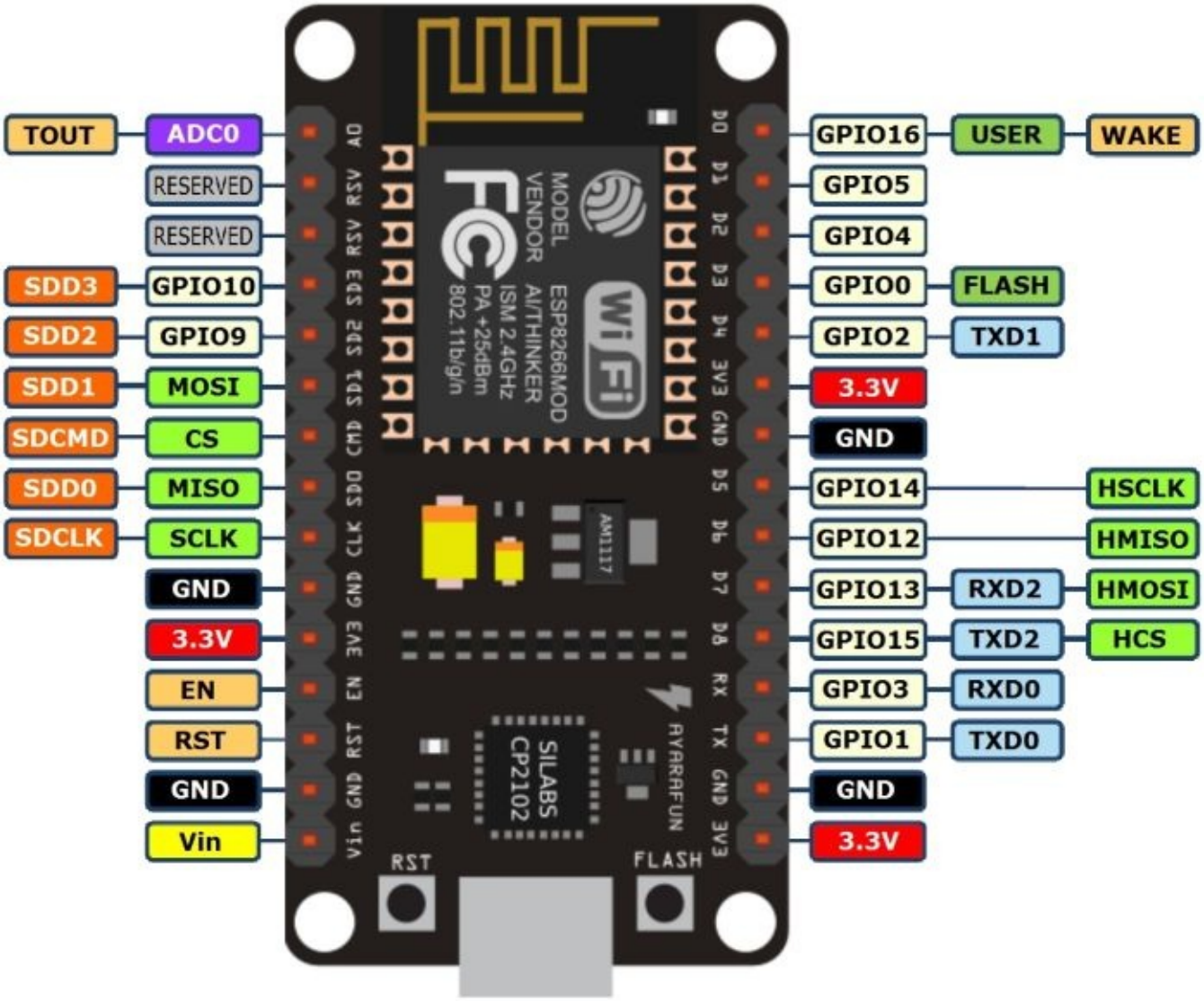


2.6 Hello MicroPython: Blinking LED

In this section, we build a blinking LED program via Python shell via MicroPython firmware. We use serial tool to write a program.

Firstly, you must know MicroPython board layout, for instance ESP8266 with NodeMCU, you can see NodeMCU v2 GPIO layout as follows.

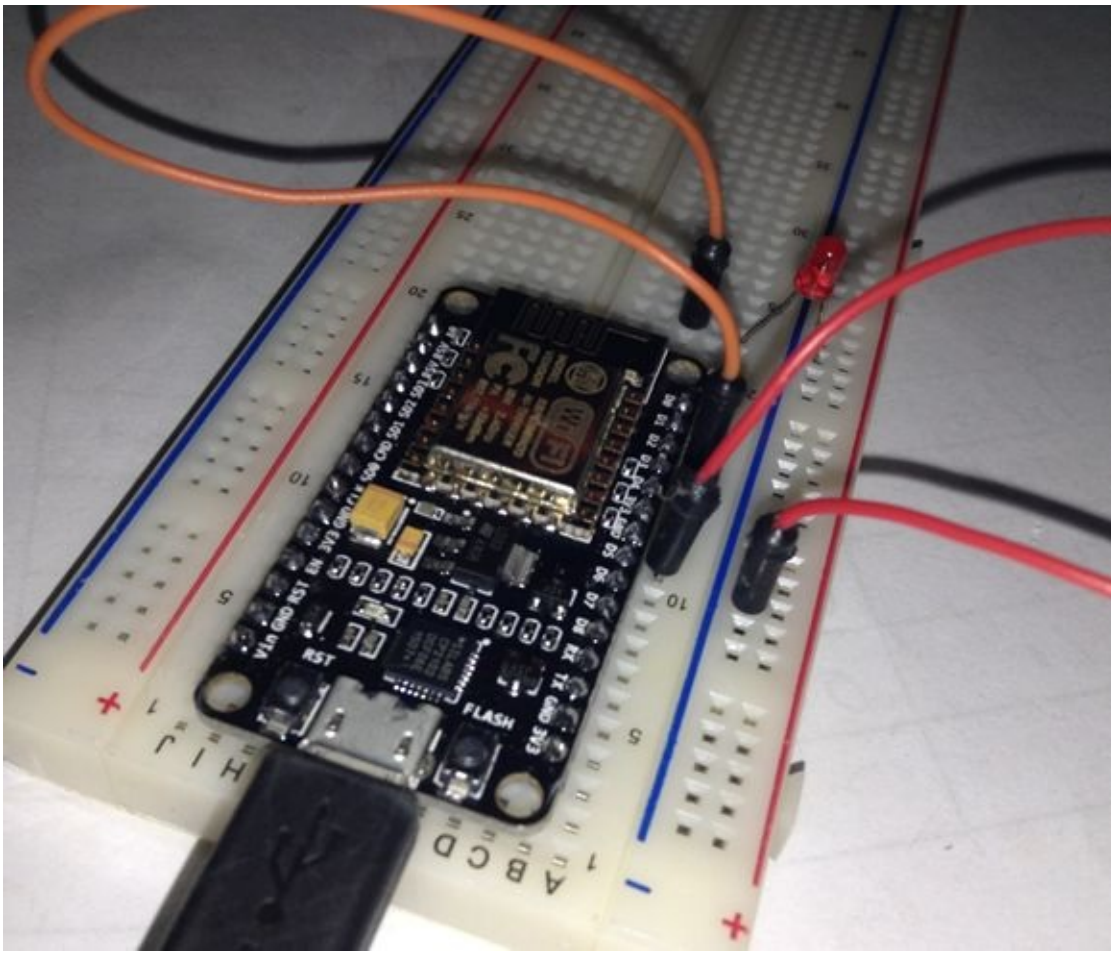




You can use these pins 0, 2, 4, 5, 12, 13, 14, 15, and 16.

2.6.1 Wiring

Connect LED on NodeMCU GPIO5 (D1). Other LED pin is be connected to NodeMCU GND. Now you connect NodeMCU board to Computer via USB cable.

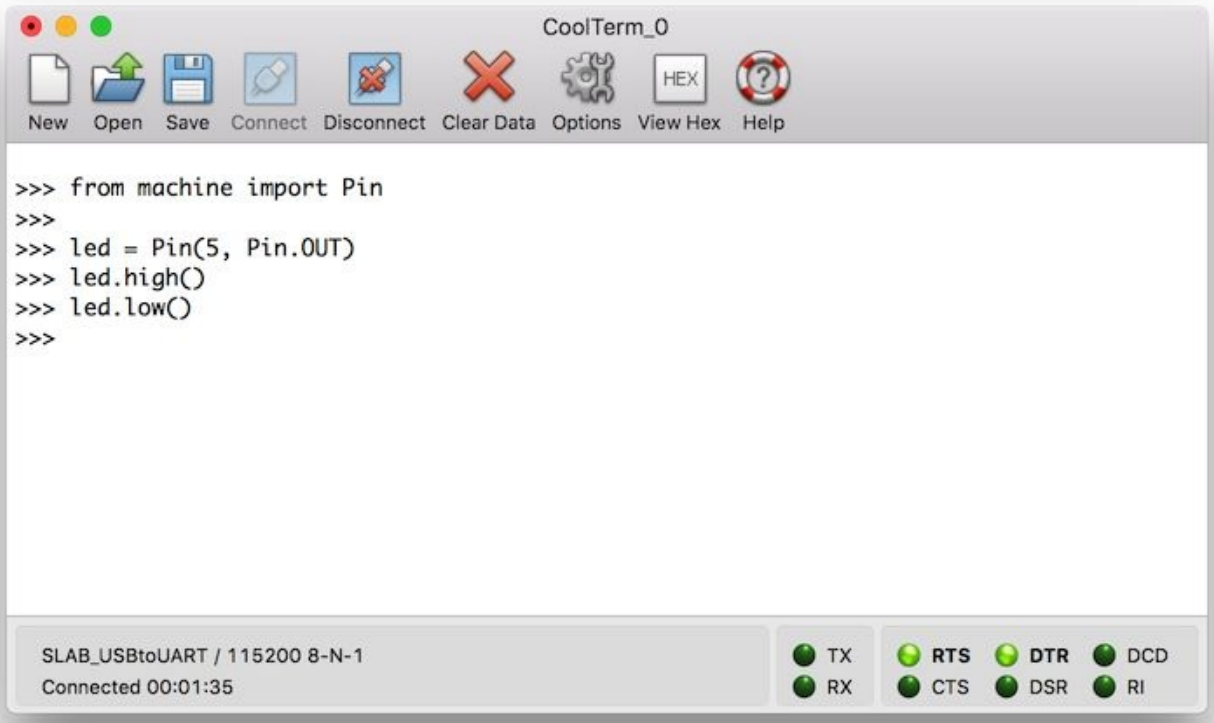


2.6.2 Writing Program Using Serial/UART Tool

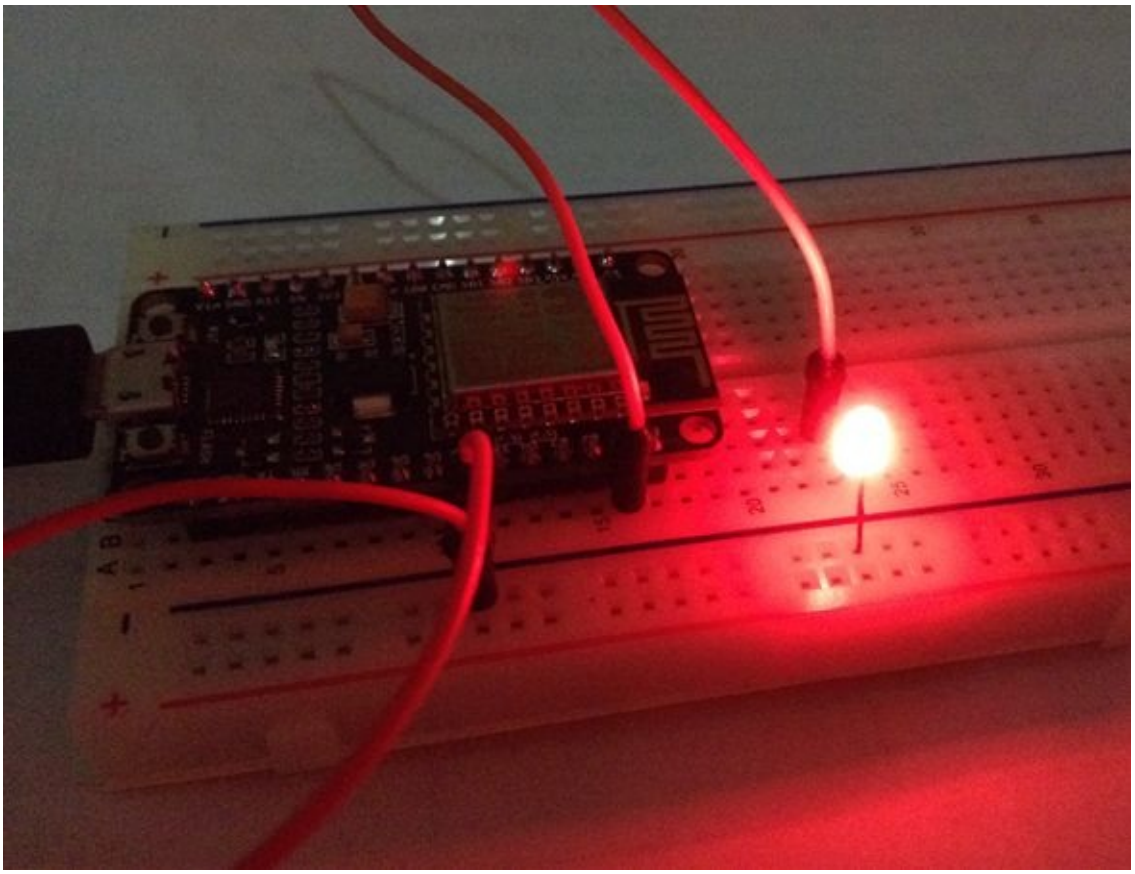
The first demo is to blink a LED. Type this script per lin on Serial/UART tool.

```
from machine import Pin

led = Pin(5, Pin.OUT)
led.high()
led.low()
```

This script will turn on a LED on D1 and then turn off it.



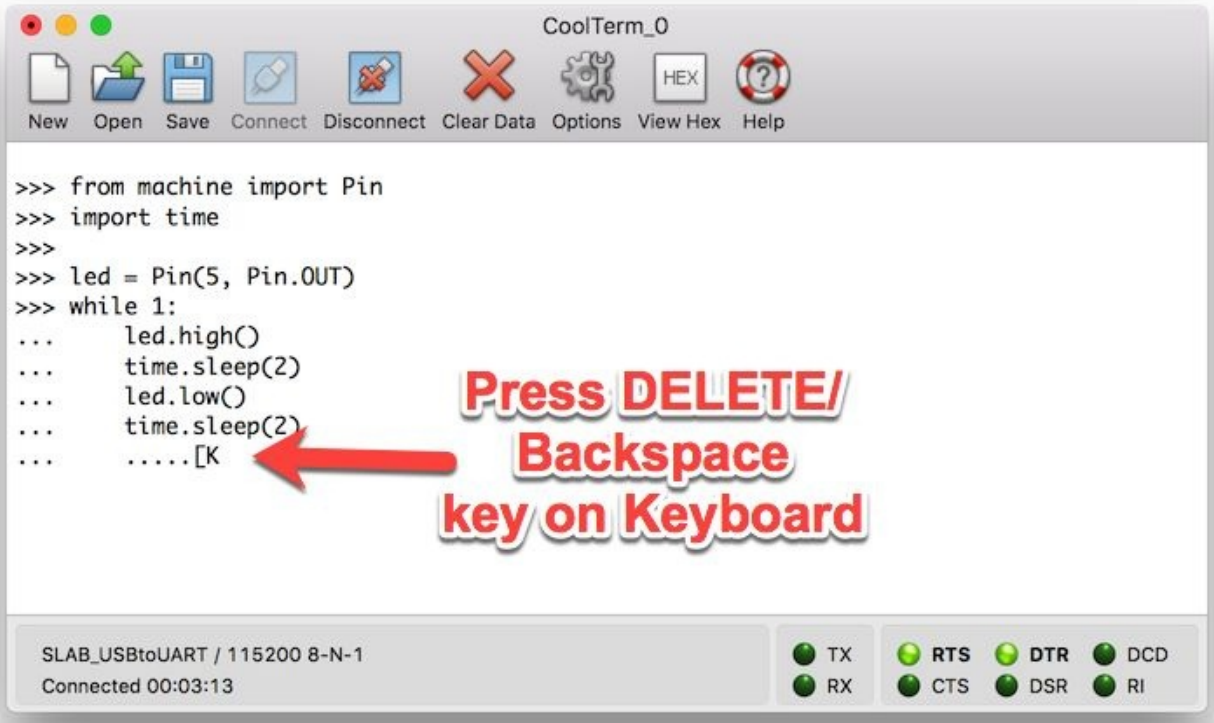
If you should see lighting LED after executed gpio.write(pin, gpio.HIGH).



We want to run blinking continuously. Now we try to build a blinking LED. Type this script.

```
from machine import Pin
import time

led = Pin(5, Pin.OUT)

while 1:
    led.high()
    time.sleep(2)
    led.low()
    time.sleep(2)
```

The last script, press backspace/delete key keyboard to exit from while looping.



You should see a blinking LED. If you want to stop, you can press RST button on NodeMCU board.

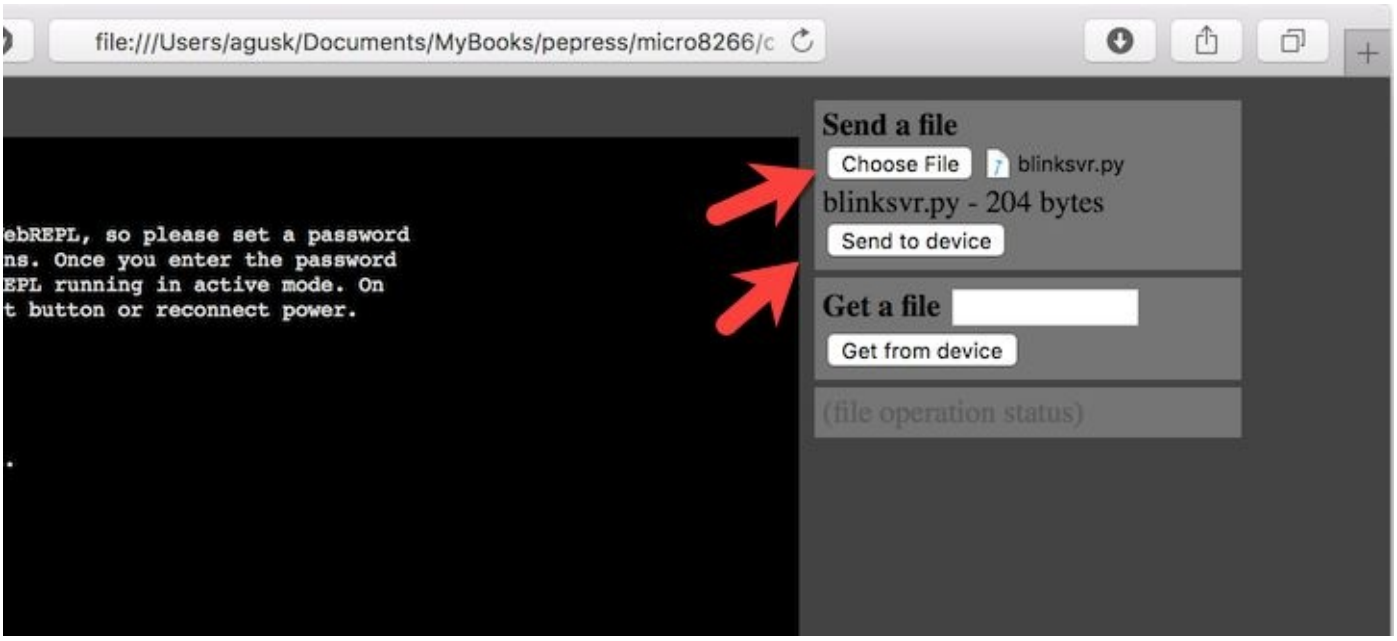
2.7 Uploading Python Script File to MicroPython Board

In this section, I show you how to upload our Python file to MicroPython board such as NodeMCU board and then execute it. We use WebREPL too. Firstly, we use the same wiring on previous section. Then, we create a file, called blinksvr.py, and write these scripts.

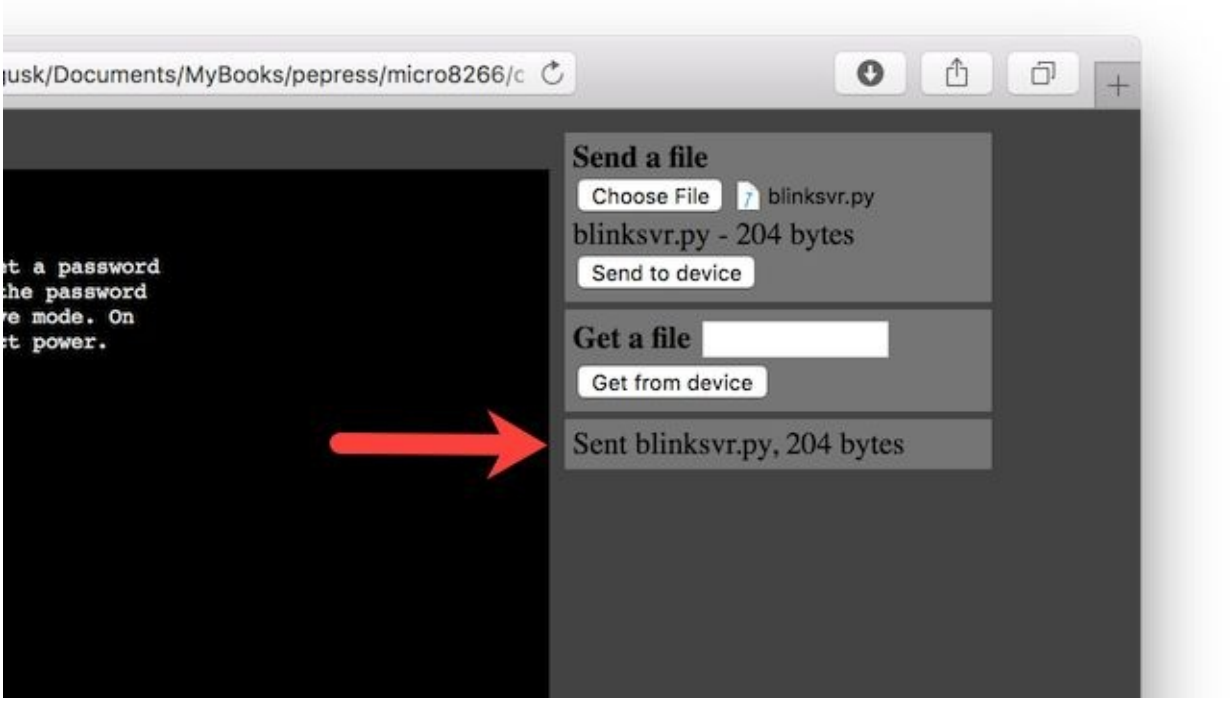
```
from machine import Pin
import time


def run():
    led = Pin(5, Pin.OUT)
    while 1:
        led.high()
        time.sleep(2)
        led.low()
        time.sleep(2)
```

To upload this file, open WebREPL. Connect to your MicroPython board.

Click **Choose File** button and select that file. If done, click **Send to device**.



If succeed, you should see the confirmation.



Now you can run by typing these command on WebREPL terminal.

```
>>> import blinksvr
>>> blinksvr.run()
```

You should blinking LED.

As we know, your MicroPython board may has limited storage. If you want to delete the file, you can type this command, for instance, deleting blinksvr.py.

```
>>> import os
```

```
>>> os.remove('blinksvr.py')
```

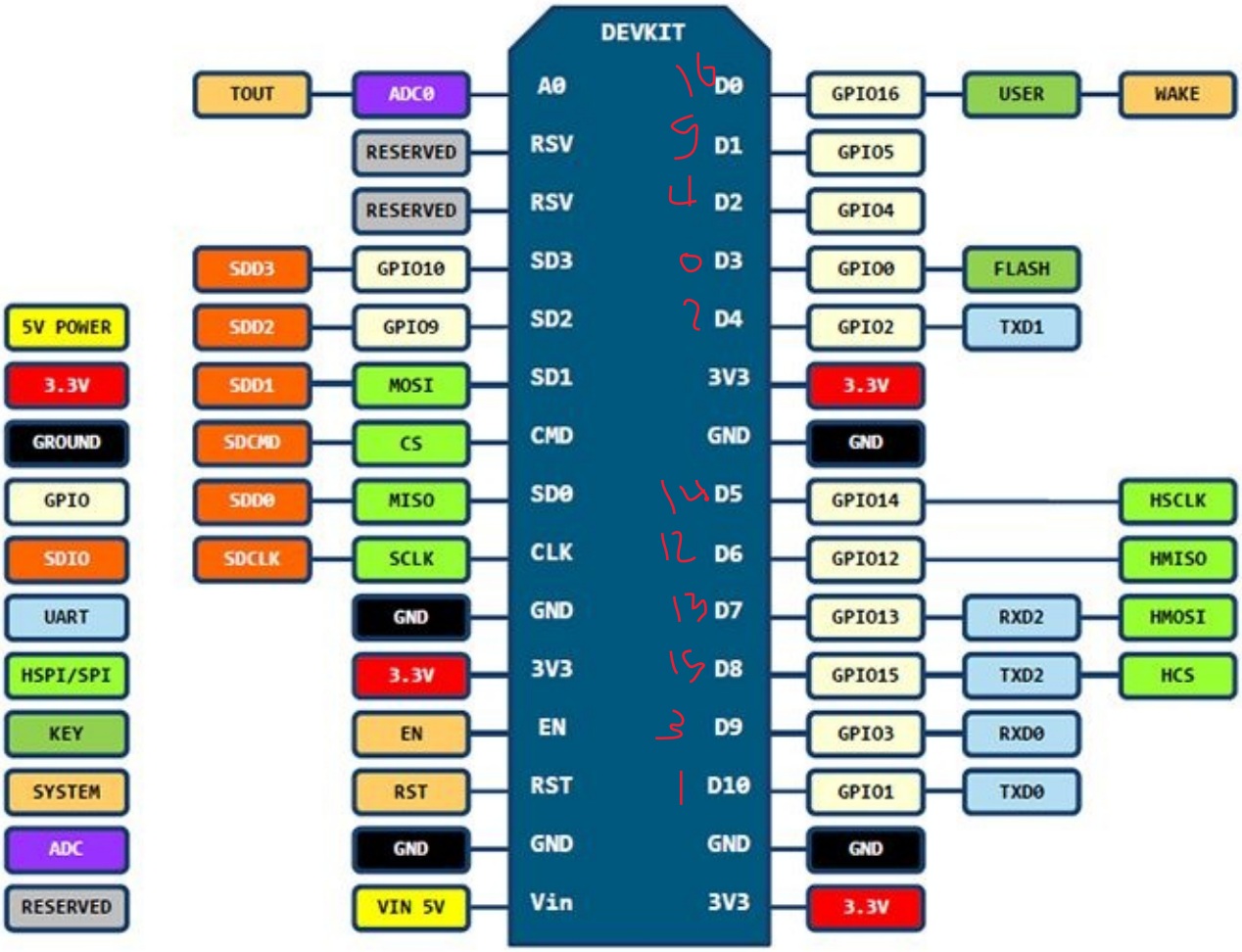
If your program is still running, please reset your board and then delete the file.

3. GPIO Programming

In this chapter I’m going to explain how to work with GPIO on MicroPython.

3.1 Getting Started

In general, GPIO can be used to control digital I/O on MicroPython boards. For MicroPython board-based ESP8266, there are available pins: 0, 1, 2, 3, 4, 5, 12, 13, 14, 15, 16, which correspond to the actual GPIO pin numbers of ESP8266 chip.

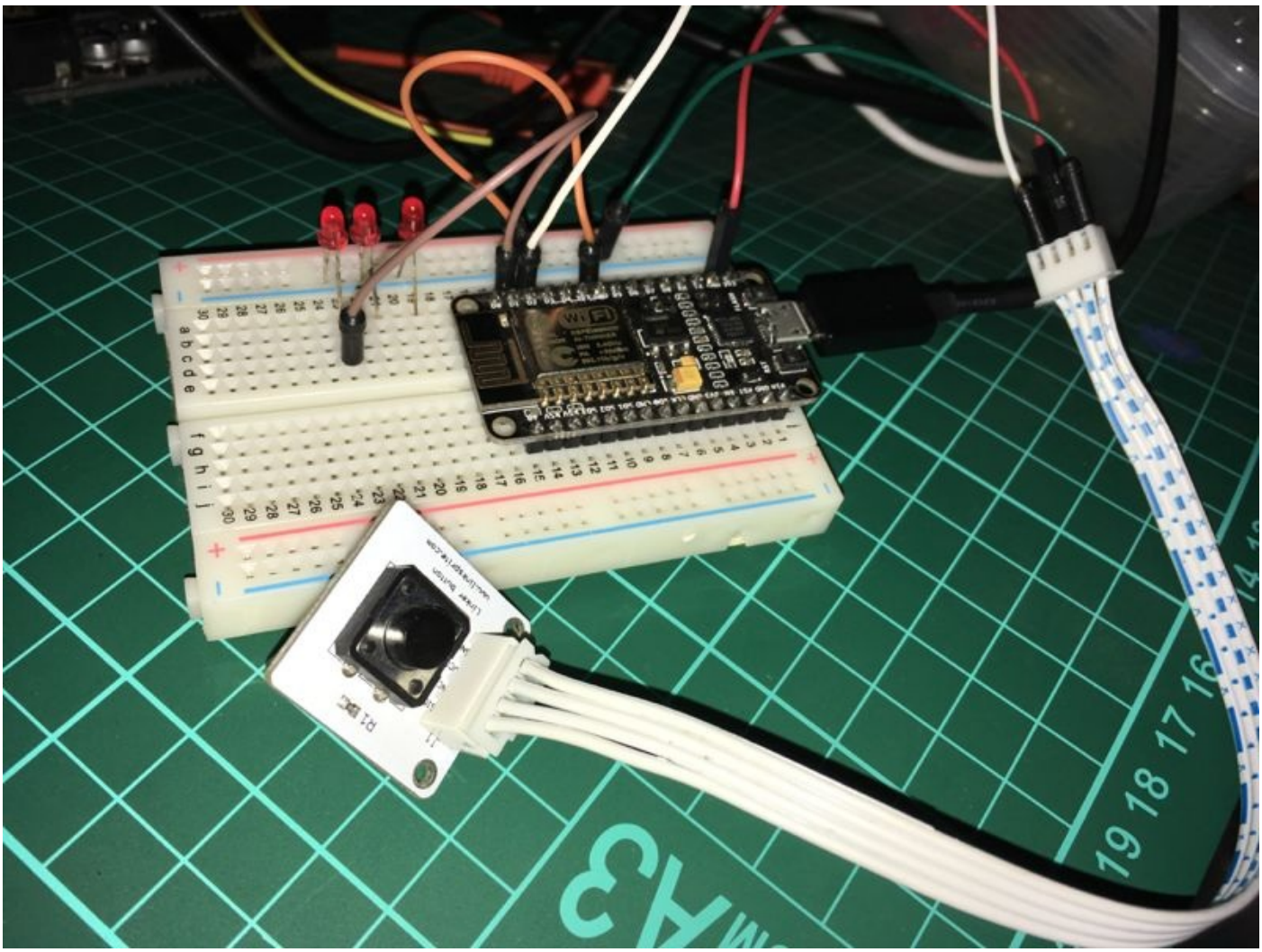


In this chapter, we build a program to illustrate how MicroPython GPIO work. We need a LED and a pushbutton. You can read about GPIO on MicroPython based ESP8266 on http://docs.micropython.org/en/latest/esp8266/library/machine.Pin.html. For testing, I used NodeMCU board as MicroPython board.

Let’s start!.

3.2 Wiring

Connect LED to GPIO5 (D1) on the board and pushbutton to GPIO4 (D2). The following is a sample of wiring.



3.3 Writing a Program

To create a program, we just create a new Python file, called **ledbutton.py**. Then, write these scripts.

```
from machine import Pin


def run():
    print('demo digital I/O')
    led = Pin(5, Pin.OUT)    # create output pin on GPIO16
    button = Pin(4, Pin.IN)   # create output pin on GPIO5
    while 1:
        state = button.value()
        if state > 0:
            led.high()
        else:
            led.low()
```

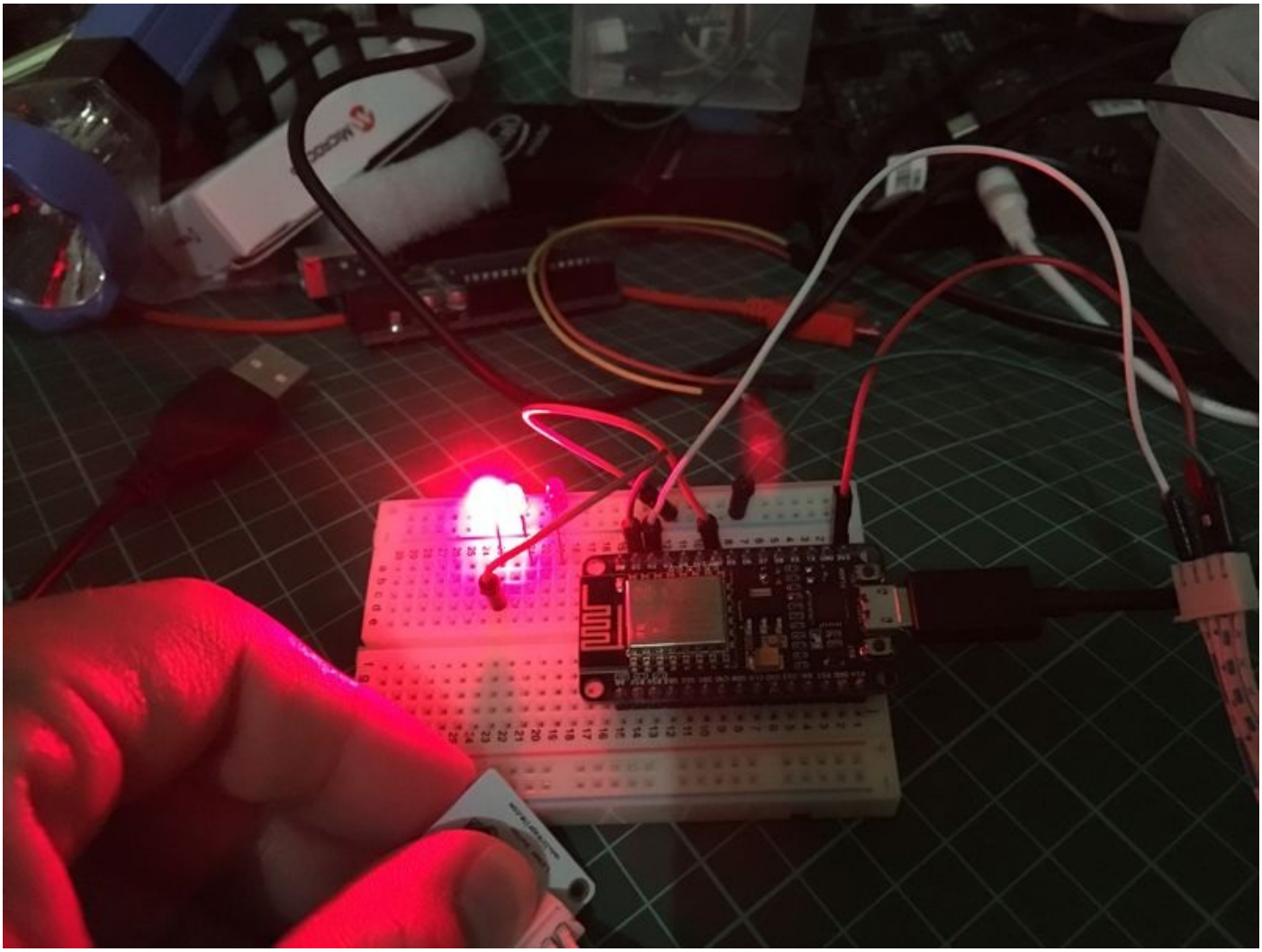
Save these scripts.

3.4 Testing

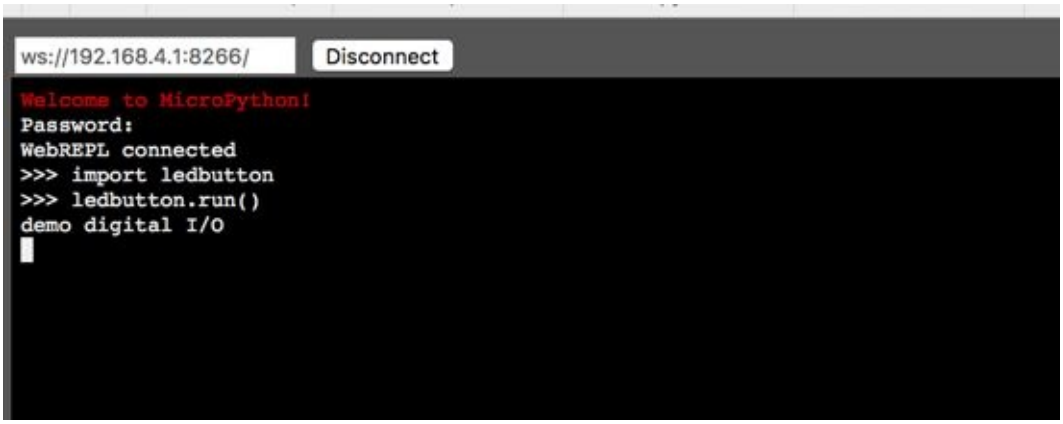
Now you can upload and run this program to MicroPython board via WebREPL. Now you can run it using WebREPL terminal. Type these command

```
>>> import ledbutton
>>> ledbutton.run()
```

For testing, try to press pushbutton. You should see a lighting LED.



You also see the program output if you connect to MicroPython board via Serial app. Or you can see it on WebREPL.

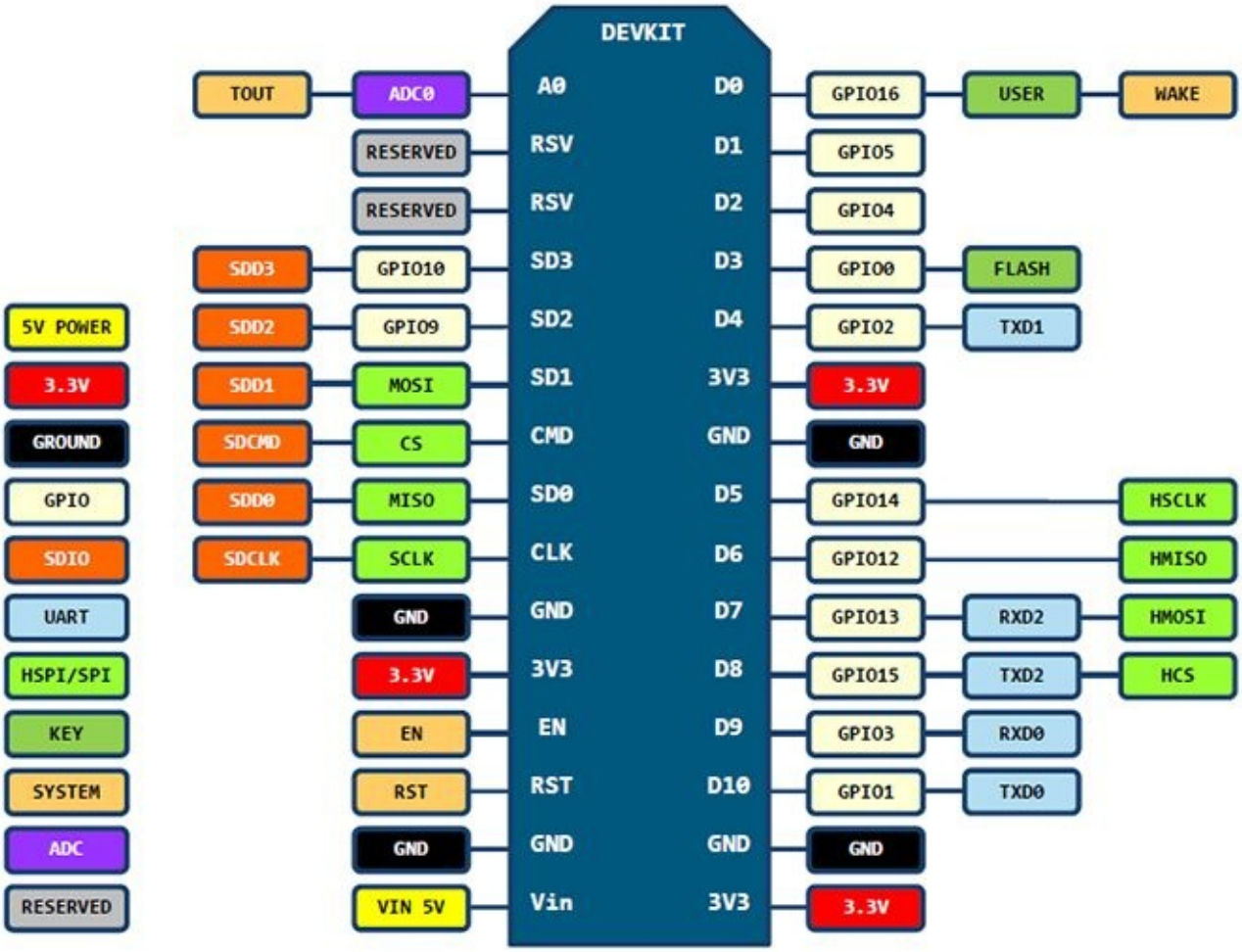

ws://192.168.4.1:8266/
Disconnect
Welcome to MicroPython!
Password:
WebREPL connected
>>> import ledbutton
>>> ledbutton.run()
demo digital I/O

4. PWM and Analog Input

This chapter explains how to work with MicroPython board based ESP8266 Analog I/O.

4.1 Getting Started

In this chapter, we learn how to work with PWM and Analog Input. For testing, I use NodeMCU board as MicroPython board. On the ESP8266 board, we can use PWM pins 0, 2, 4, 5, 12, 13, 14 and 15 all support PWM.



In this chapter, we try to access MicroPython Analog I/O using MicroPython program. There are two scenarios for our cases:

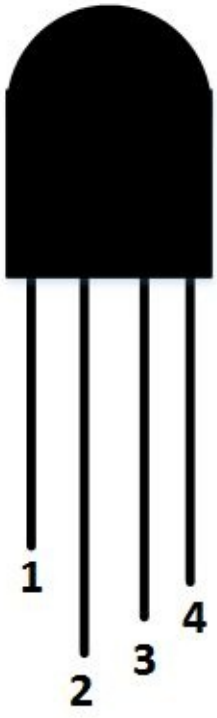
- Controlling RGB LED
- Reading Analog input using Potentiometer

Let’s start.

4.2 Demo Analog Output (PWM) : RGB LED

In this scenario we build a MicroPython program to control RGB LED color using MicroPython Analog output (PWM). RGB LED has 4 pins that you can see it on Figure below.

To understand these pins, you can see the following Figure.



Note:

- Pin 1: Red

- Pin 2: Common pin
- Pin 3: Green
- Pin 4: Blue

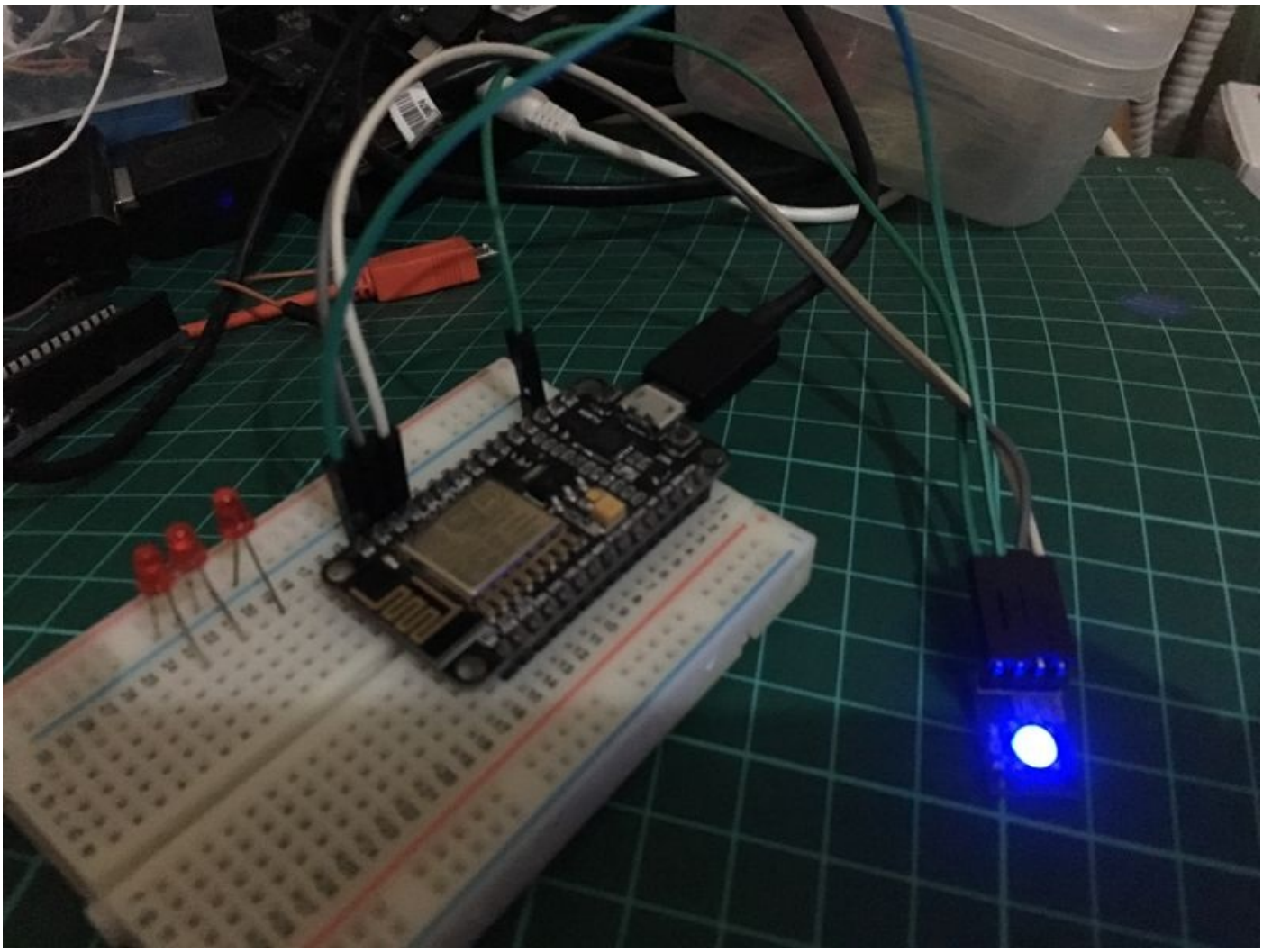
Now we can start to build a MicroPython application and hardware implementation.

4.2.1 Wiring

For our testing, we configure the following PWM pins.

- RGB LED pin 1 (red) is connected to NodeMCU GPIO5 (D1)
- RGB LED pin 2 is connected to NodeMCU 3V3 (VCC +3.3V)
- RGB LED pin 3 (green) is connected to NodeMCU GPIO4 (D2)
- RGB LED pin 4 (blue) is connected to NodeMCU GPIO0 (D3)

Here is a sample implementation with NodeMCU and RGB Led



4.2.2 Writing Program

To display a certain color, we must combine colors from red, green, blue. NodeMCU provides API for PWM which can set a value from 0 to 1023 using PWM library.

Let”s start to build a program. Firstly, create a file, called **pwmdemo.py**. Then, write these scripts.

```
from machine import Pin, PWM
import time


gpio_red = 5
gpio_green = 4
gpio_blue = 0


def set_rgb(red, green, blue):
    pwm_red = PWM(Pin(gpio_red), freq=1000, duty=red)
    pwm_green = PWM(Pin(gpio_green), freq=1000, duty=green)
    pwm_blue = PWM(Pin(gpio_blue), freq=1000, duty=blue)

    time.sleep(2)
    pwm_red.deinit()
    pwm_green.deinit()
    pwm_blue.deinit()


def run():
    print('print PWM with RGB led')

    while 1:
        print('red')
        set_rgb(1023, 0, 0)
        print('green')
        set_rgb(0, 1023, 0)
        print('blue')
        set_rgb(0, 0, 1023)
        print('yellow')
        set_rgb(1023, 1023, 0)
        print('purple')
        set_rgb(323, 0, 323)
        print('aqua')
        set_rgb(0, 1023, 1023)
```

This program will generate six colors: red, green, blue, yellow, purple, and aqua.

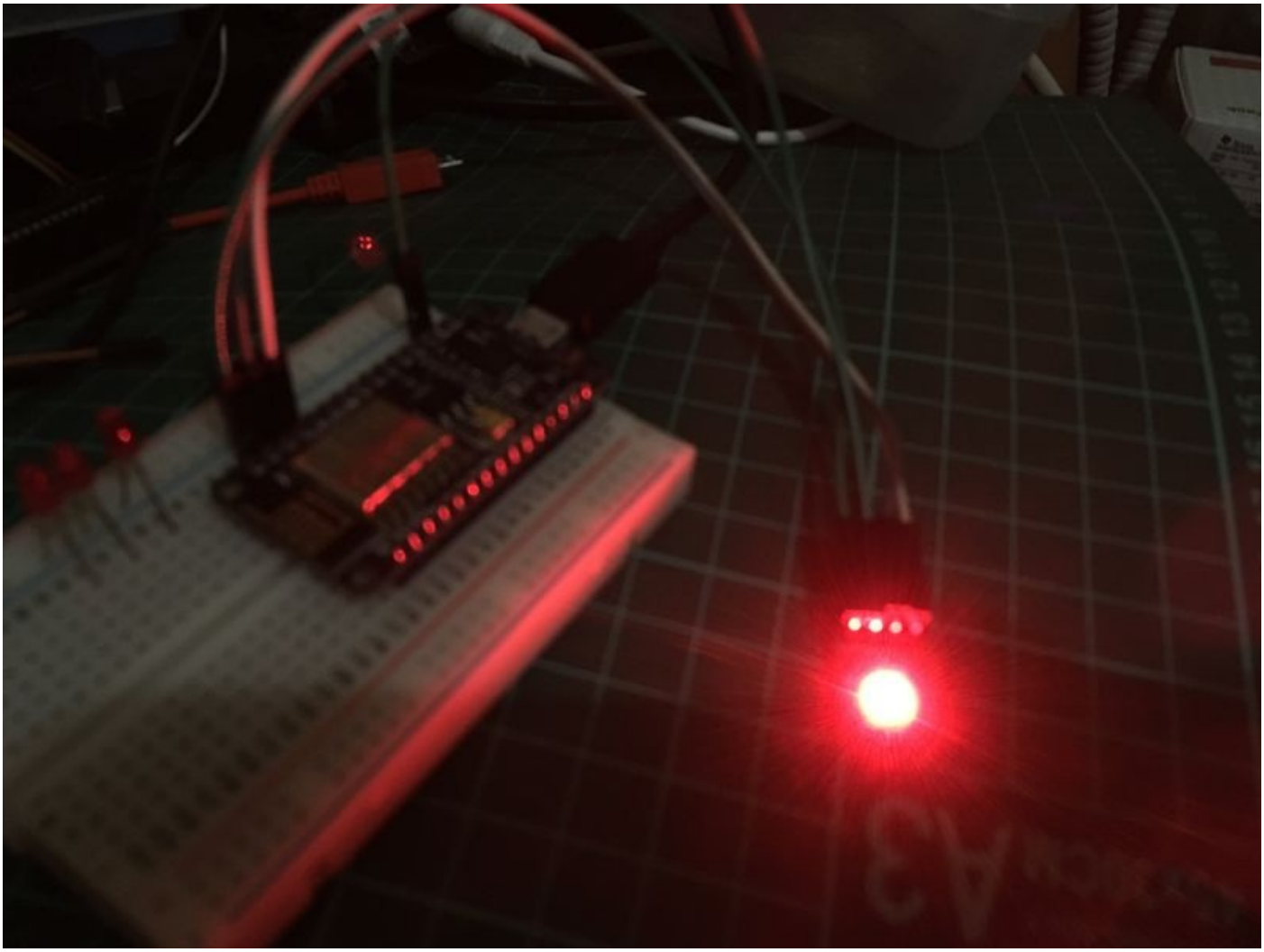
Save this file.

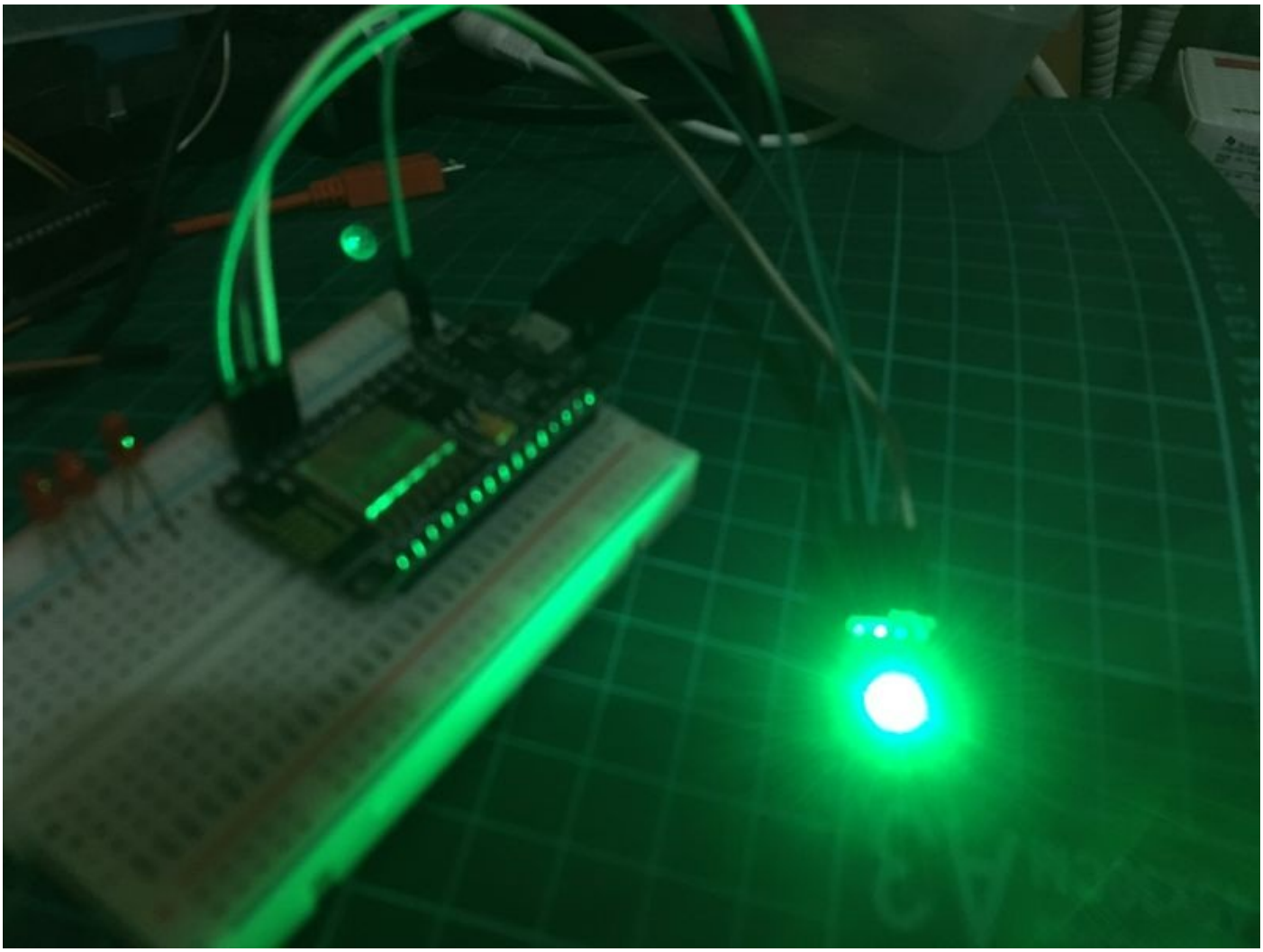
4.2.3 Testing

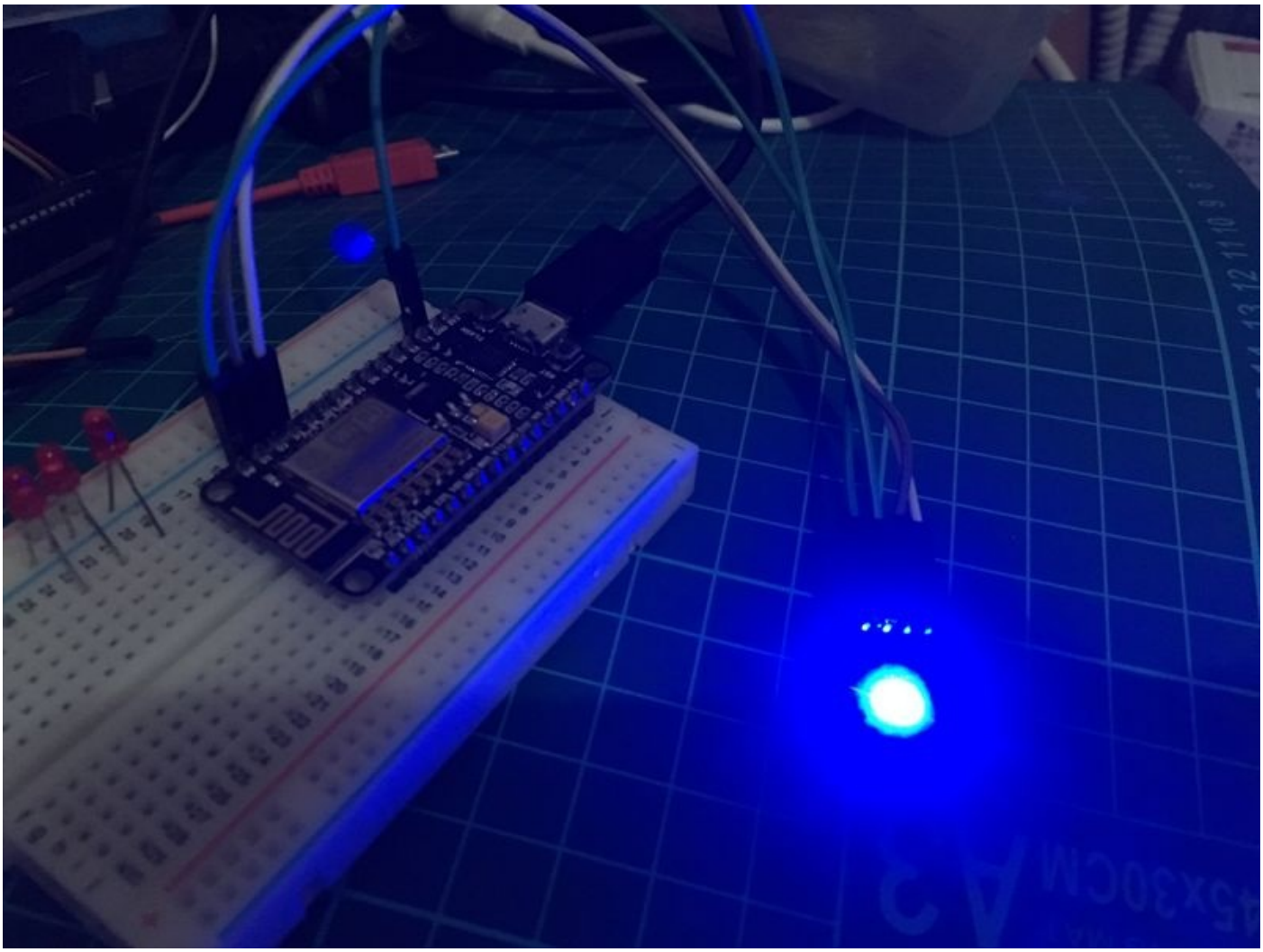
Upload and run the program. Then, run the program as follows.

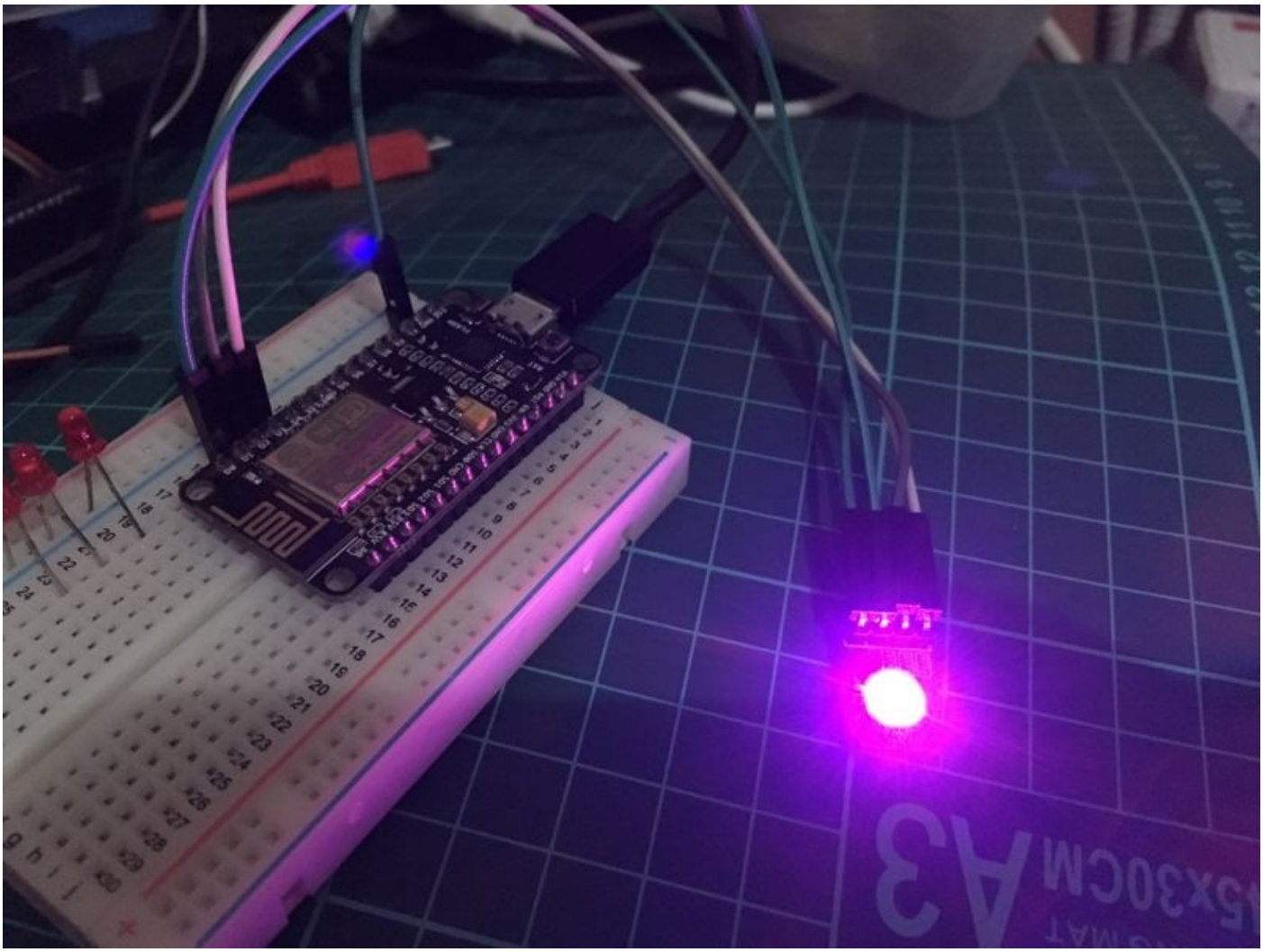
```
>>> import pwmdemo
>>> pwmdemo.run()
```

You should see several color on RGB LED. The following is a sample demo on RGB LED.

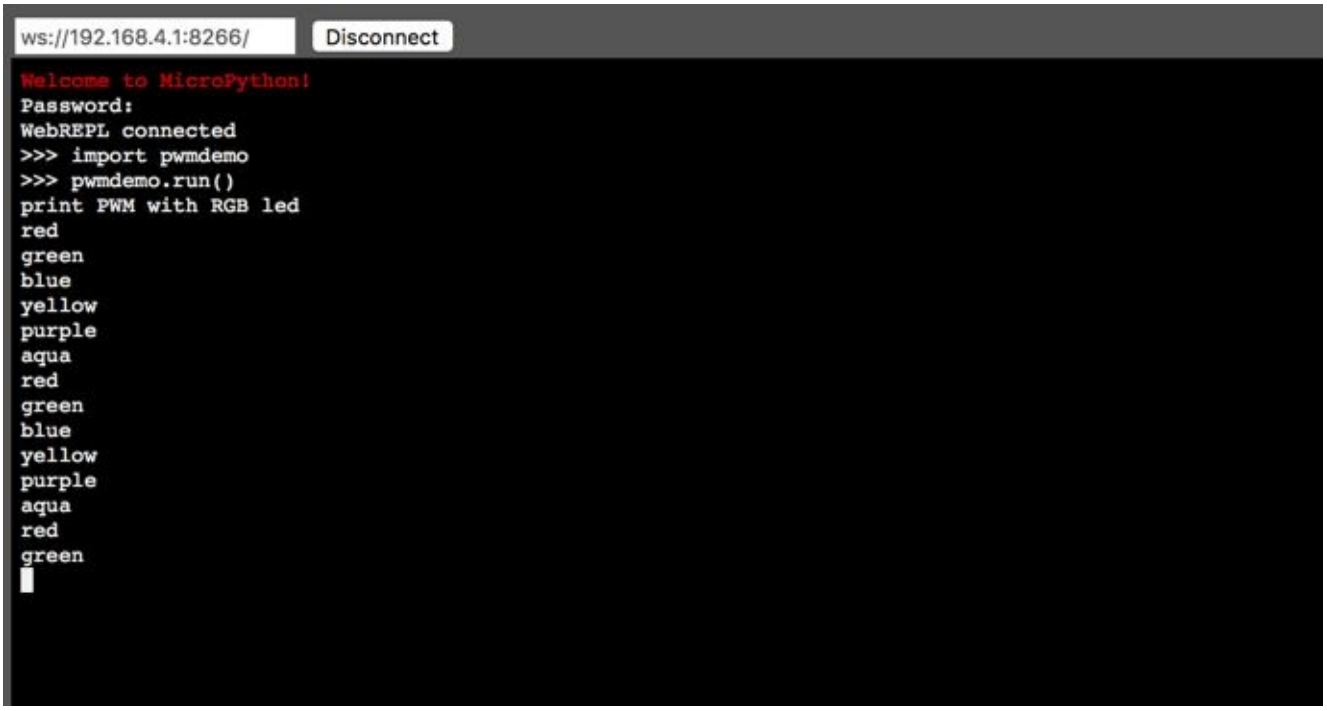








If you connect to NodeMCU board via Serial app or WebREPL, you should get program output, shown in Figure below.



4.3 Demo Analog Input: Working with Potentiometer

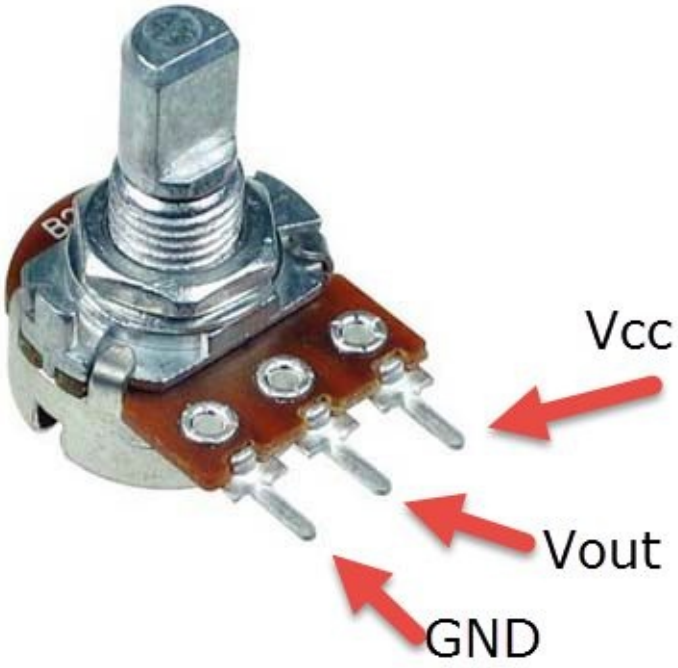
In this section, we learn how to read analog input on MicroPython board. For illustration, I use Potentiometer as analog input source. Our scenario is to read analog value from Potentiometer. Then, display it on Lua shell.

NodeMCU v2 only has one ADC on A0. If you want to work with many analog input, you must expand it using ICs based ADC. In this section, we are working on NodeMCU ADC on A0.

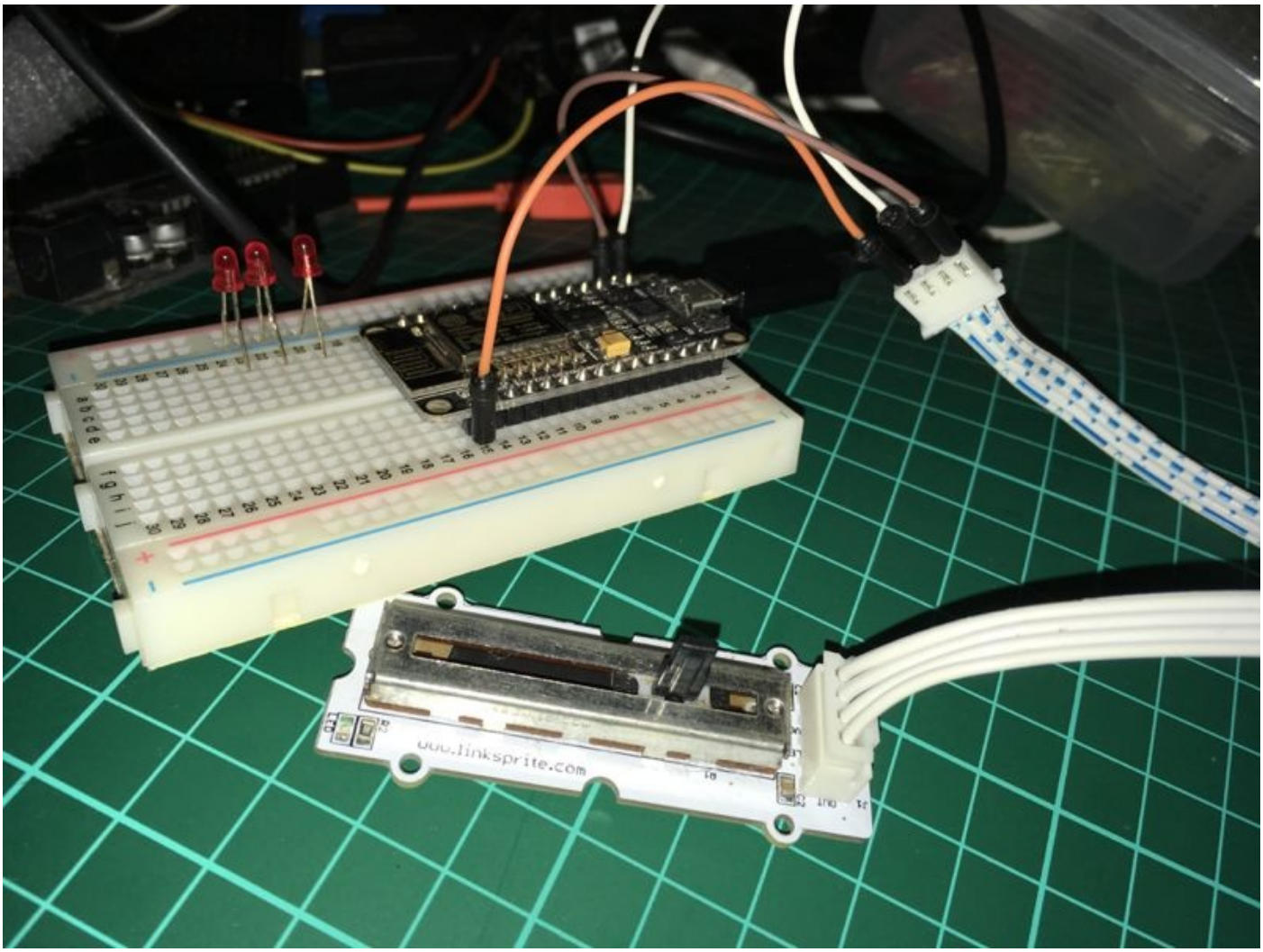
Let’s start!.

4.3.1 Wiring

To understand Potentiometer, you see its scheme in Figure below.



You can connect VCC to NodeMCU board on 3V3 pin (VCC +3.3V). Vout to NodeMCU board Analog input A0. In addition, GND to NodeMCU board GND. The following is hardware implementation. I use slide potentiometer.



4.3.2 Writing Program

Firstly, create a file, called **adcdemo.py**. To read analog input, we can use adc.read() function. Ok, Let's write these scripts.

```
from machine import ADC
import time


def run():
    print('ADC demo')

    while 1:
        adc = ADC(0)
        print('ADC: ' + str(adc.read()))
        time.sleep(2)
```

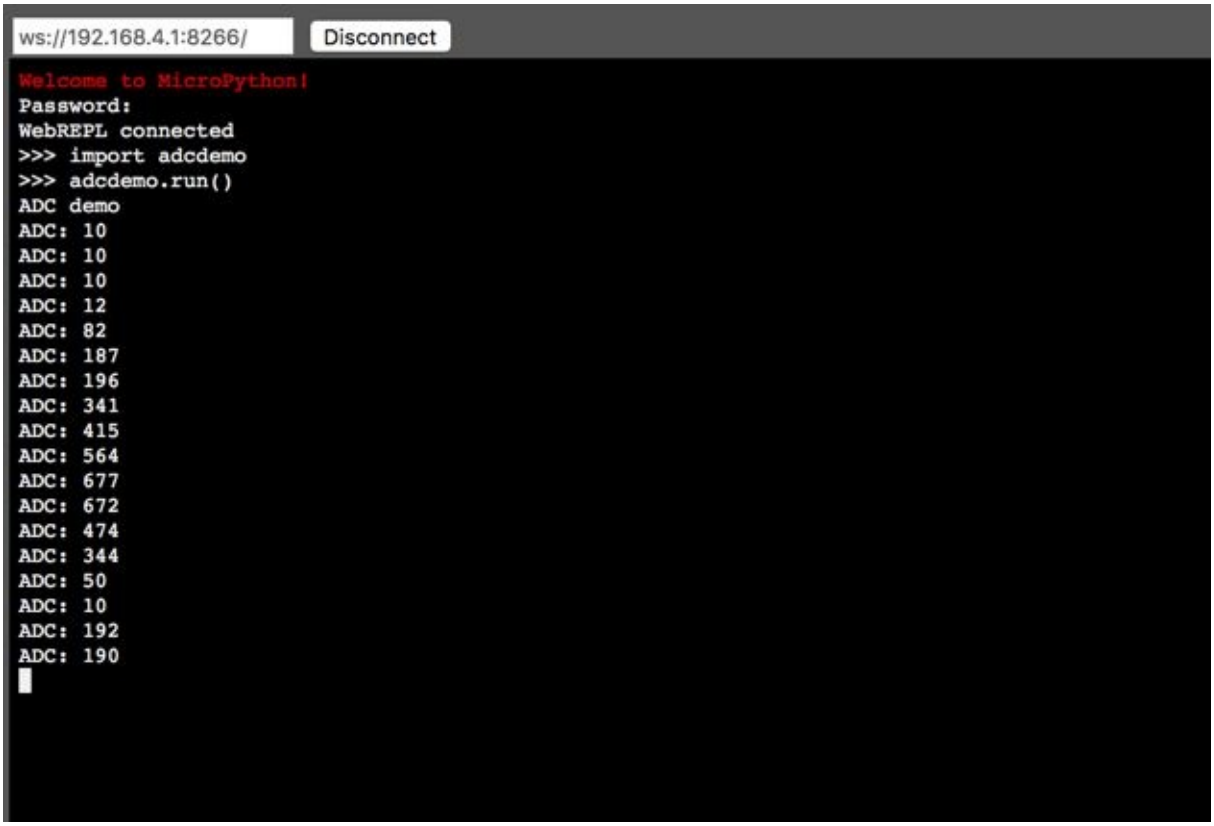
Save this code.

4.3.3 Testing

Upload and run this program. If succeed, you can run the program.

```
>>> import adcdemo
>>> adcdemo.run()
```

You should see the output on WebREPL terminal.



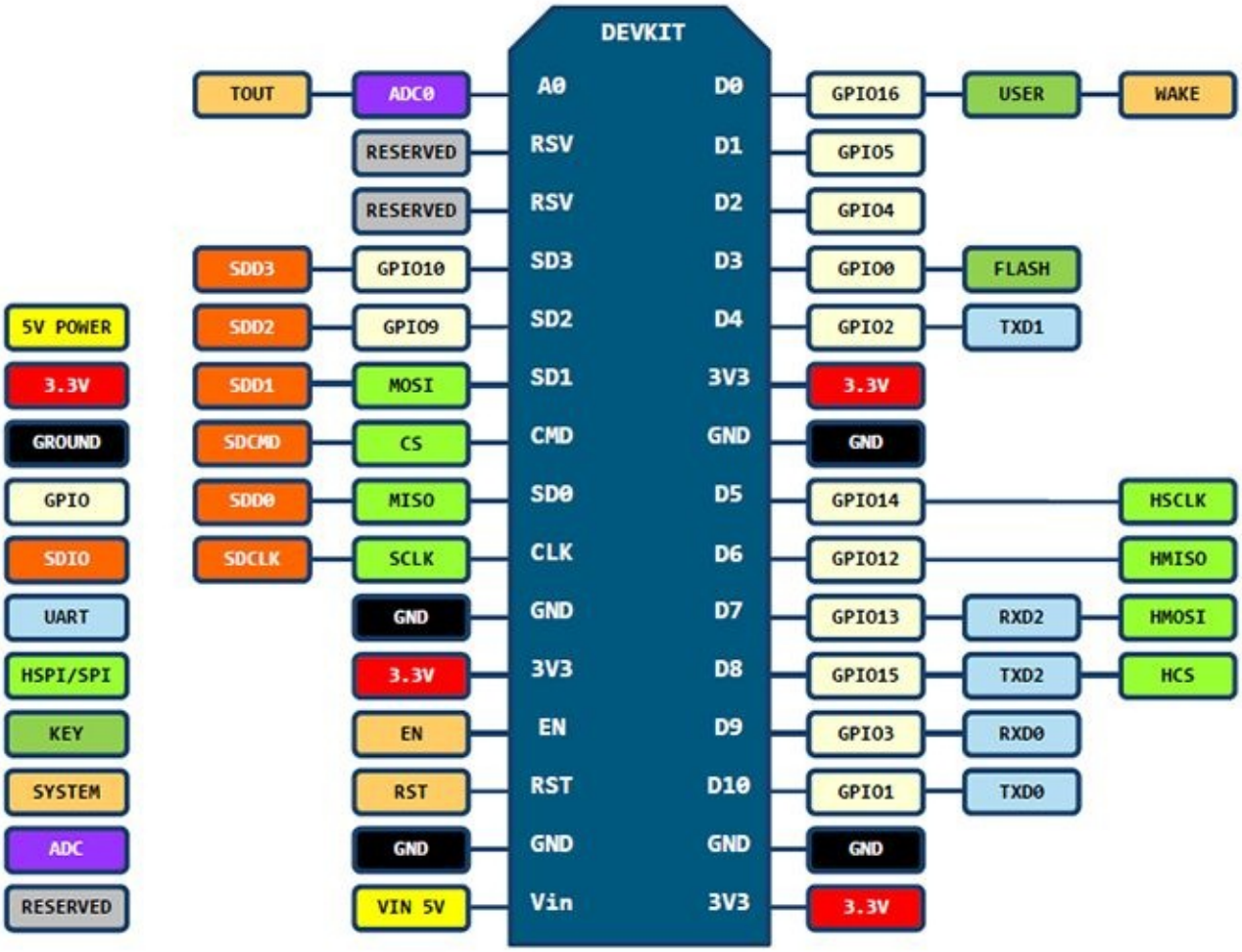
5. Working with I2C

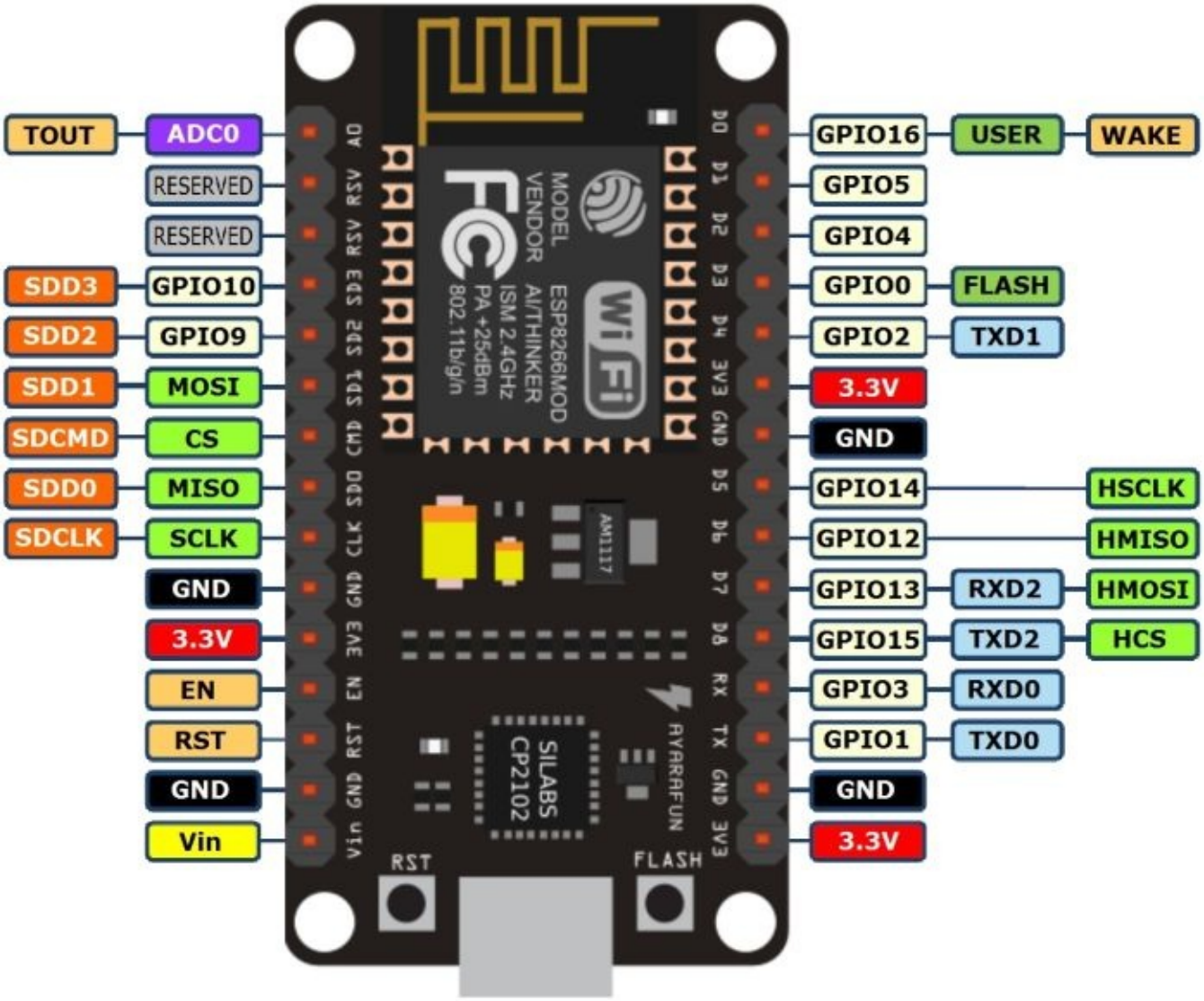
In this chapter we learn how to work with I2C on MicroPython board.

5.1 Getting Started

The I2C (Inter-Integrated Circuit) bus was designed by Philips in the early ’80s to allow easy communication between components which reside on the same circuit board. TWI stands for Two Wire Interface and for most marts this bus is identical to I²C. The name TWI was introduced by Atmel and other companies to avoid conflicts with trademark issues related to I²C.

I2C bus consists of two wires, SDA (Serial Data Line) and SCL (Serial Clock Line). MicroPython supports all pins for I2C software.

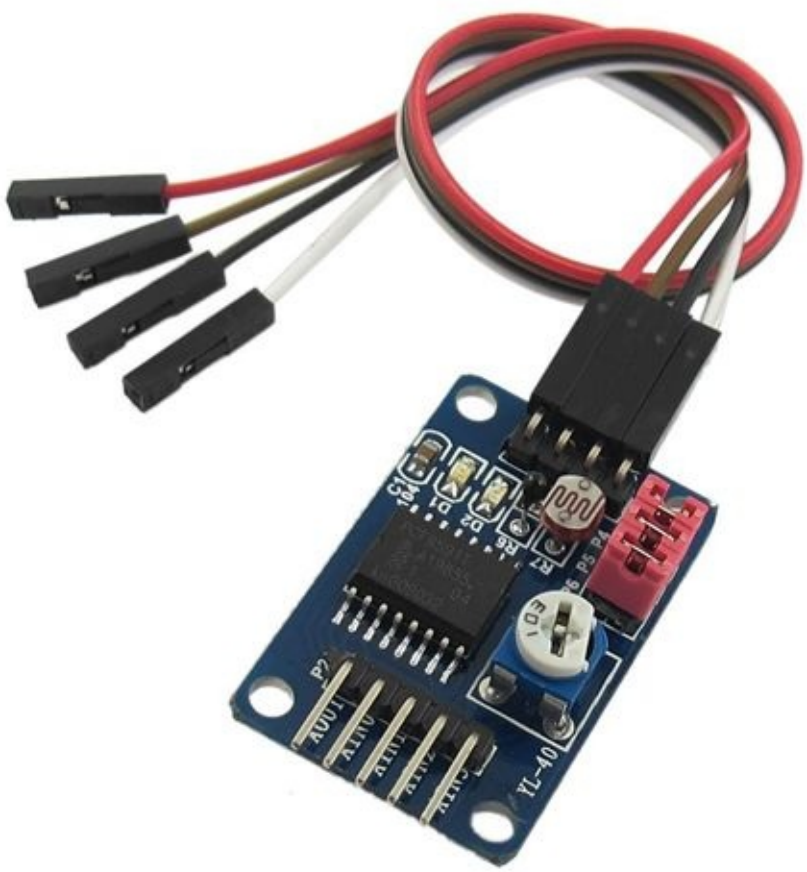




For testing, I used PCF8591 AD/DA Converter module with sensor and actuator devices. You can find it on the following online store:

- Amazon, http://www.amazon.com/PCF8591-Converter-Module-Digital-Conversion/dp/B00BXX4UWC/
- eBay, http://www.ebay.com
- Dealextreme, http://www.dx.com/p/pcf8591-ad-da-analog-to-digital-digital-to-analog-converter-module-w-dupont-cable-deep-blue-336384
- Aliexpress, http://www.aliexpress.com/

In addition, you can find this device on your local electronics store/online store.



This module has mini form model too, for instance, you can find it on Amazon, http://www.amazon.com/WaveShare-PCF8591T-Converter-Evaluation-Development/dp/B00KM6X2OI/ .

This module use PCF8591 IC and you can read the datasheet on the following URLs.

- http://www.electrodragon.com/w/images/e/ed/PCF8591.pdf
- http://www.nxp.com/documents/data_sheet/PCF8591.pdf

For testing I2C on MicroPython, I use PCF8591 AD/DA Converter module and NodeMCU module. To access I2C, we can use I2C library, http://docs.micropython.org/en/latest/esp8266/library/machine.I2C.html.

Let’s start.

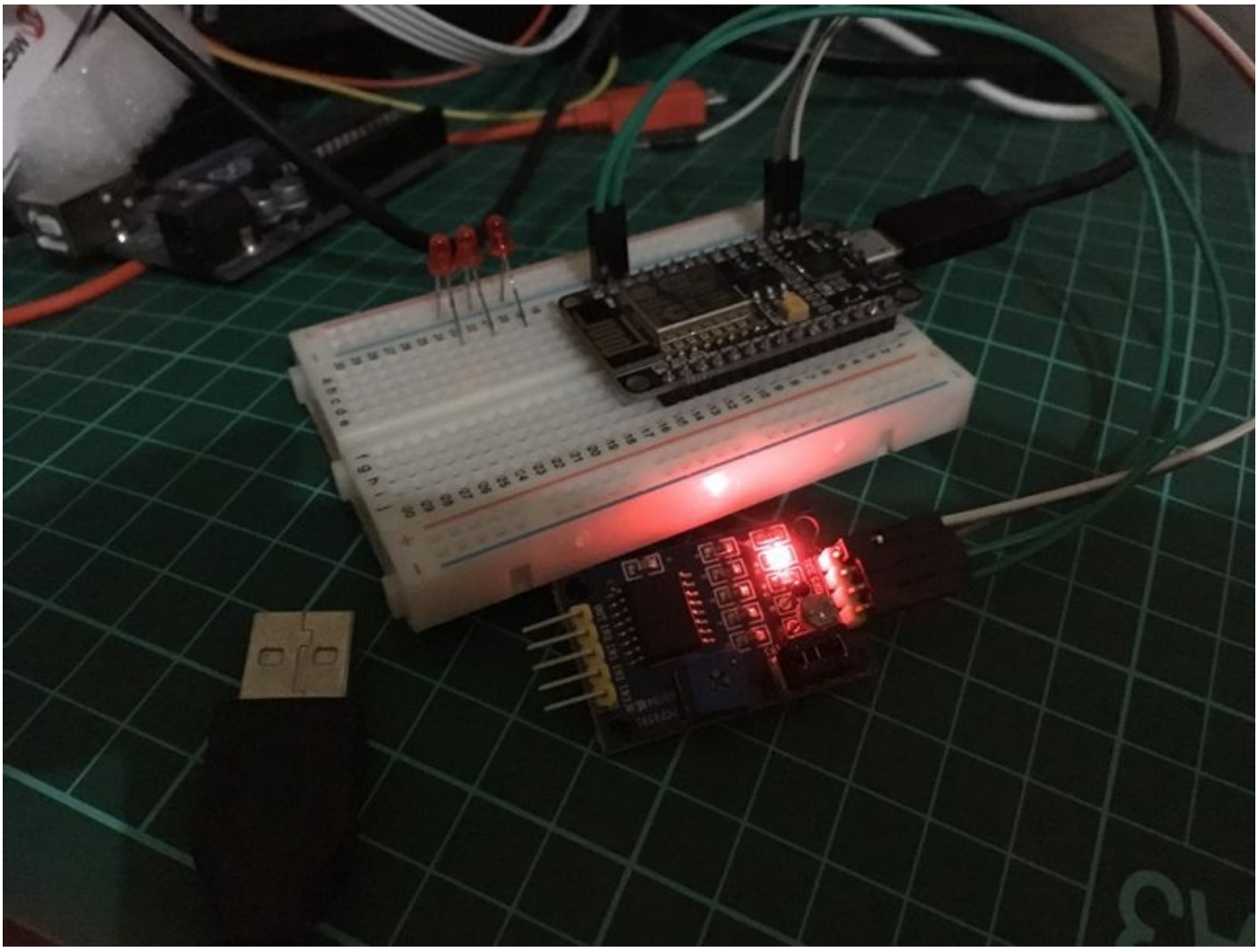
5.2 Writing Program

We connect PCF8591 AD/DA Converter module to NodeMCU board directly.

The following is our wiring lab:

- PCF8591 AD/DA Converter module SDA (A4) —> NodeMCU SDA (D1)
- PCF8591 AD/DA Converter module (A5) —> NodeMCU CLK (D2)
- PCF8591 AD/DA Converter module VCC —> NodeMCU VCC
- PCF8591 AD/DA Converter module GND —> NodeMCU GND

Hardware implementation can be shown in Figure below.



5.3 Writing Program

Now you can start to write a MicroPython program for NodeMCU. Create a file, called i2cdemo.py and write these scripts.

```
from machine import Pin, I2C
import time


def run():
    print('read sensor from i2c protocol')
    PCF8591 = 0x48          # I2C bus address
    PCF8591_ADC_CH0 = '\x00'  # thermistor
    PCF8591_ADC_CH1 = '\x01'  # photo-voltaic cell
    PCF8591_ADC_CH3 = '\x03'  # potentiometer

    # construct an I2C bus
    gpio_scl = Pin(5)
    gpio_sda = Pin(4)
    i2c = I2C(scl=gpio_scl, sda=gpio_sda, freq=100000)

    while 1:
        # read thermistor
        i2c.writeto(PCF8591, PCF8591_ADC_CH0)
        i2c.readfrom(PCF8591, 1)
        data = i2c.readfrom(PCF8591, 1)
        print('Thermistor: ' + str(ord(chr(data[0]))))

        # photo-voltaic cell
        i2c.writeto(PCF8591, PCF8591_ADC_CH1)
        i2c.readfrom(PCF8591, 1)
        data = i2c.readfrom(PCF8591, 1)
        print('photo-voltaic: ' + str(ord(chr(data[0]))))

        # potentiometer
        i2c.writeto(PCF8591, PCF8591_ADC_CH3)
        i2c.readfrom(PCF8591, 1)
        data = i2c.readfrom(PCF8591, 1)
        print('potentiometer: ' + str(ord(chr(data[0]))))

        time.sleep(2)
```

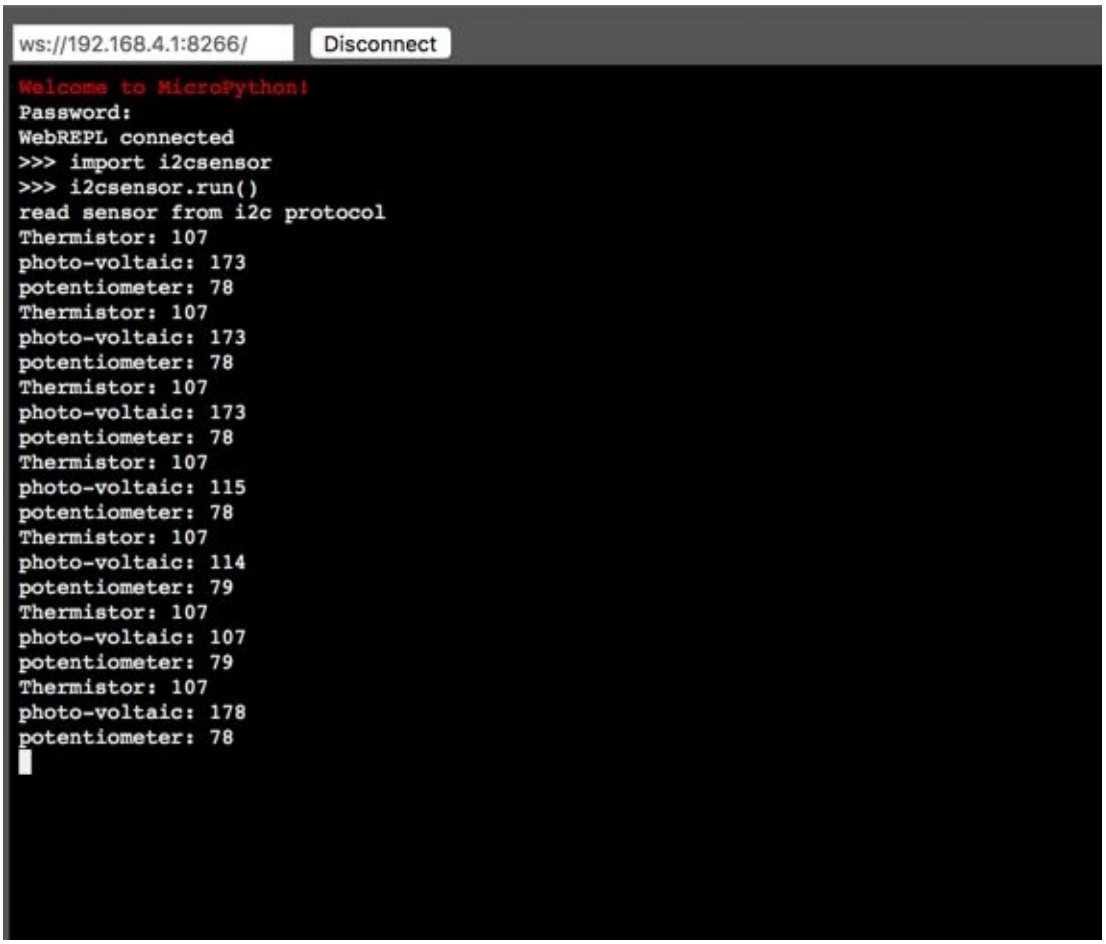
Save this code.

5.4 Testing

Now you can upload and run the MicroPython program to NodeMCU board. You can run this command in WebREPL.

```
>>> import i2csensor
>>> i2csensor.run()
```

If success, you should see the program output on WebREPL. The following is a sample output.

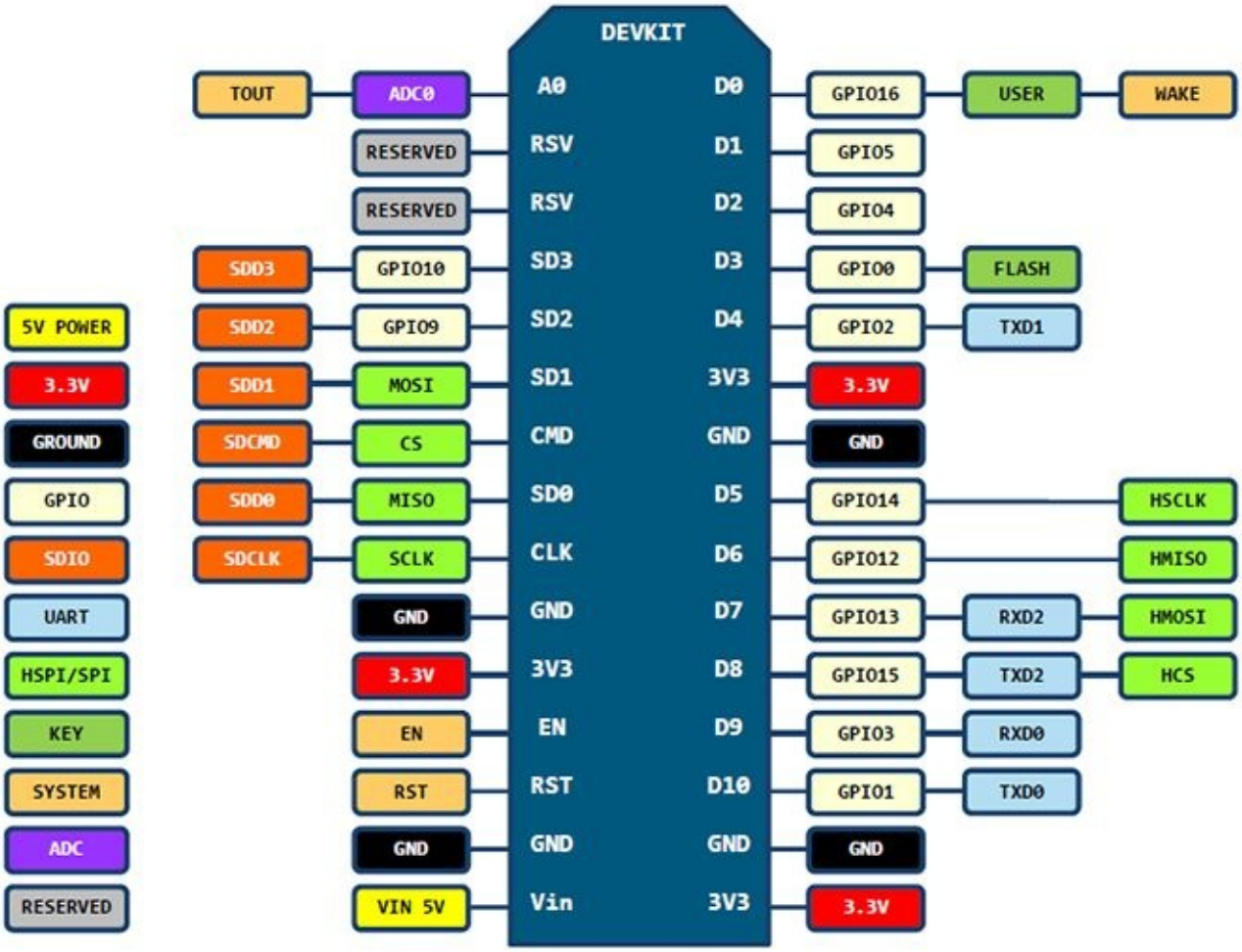


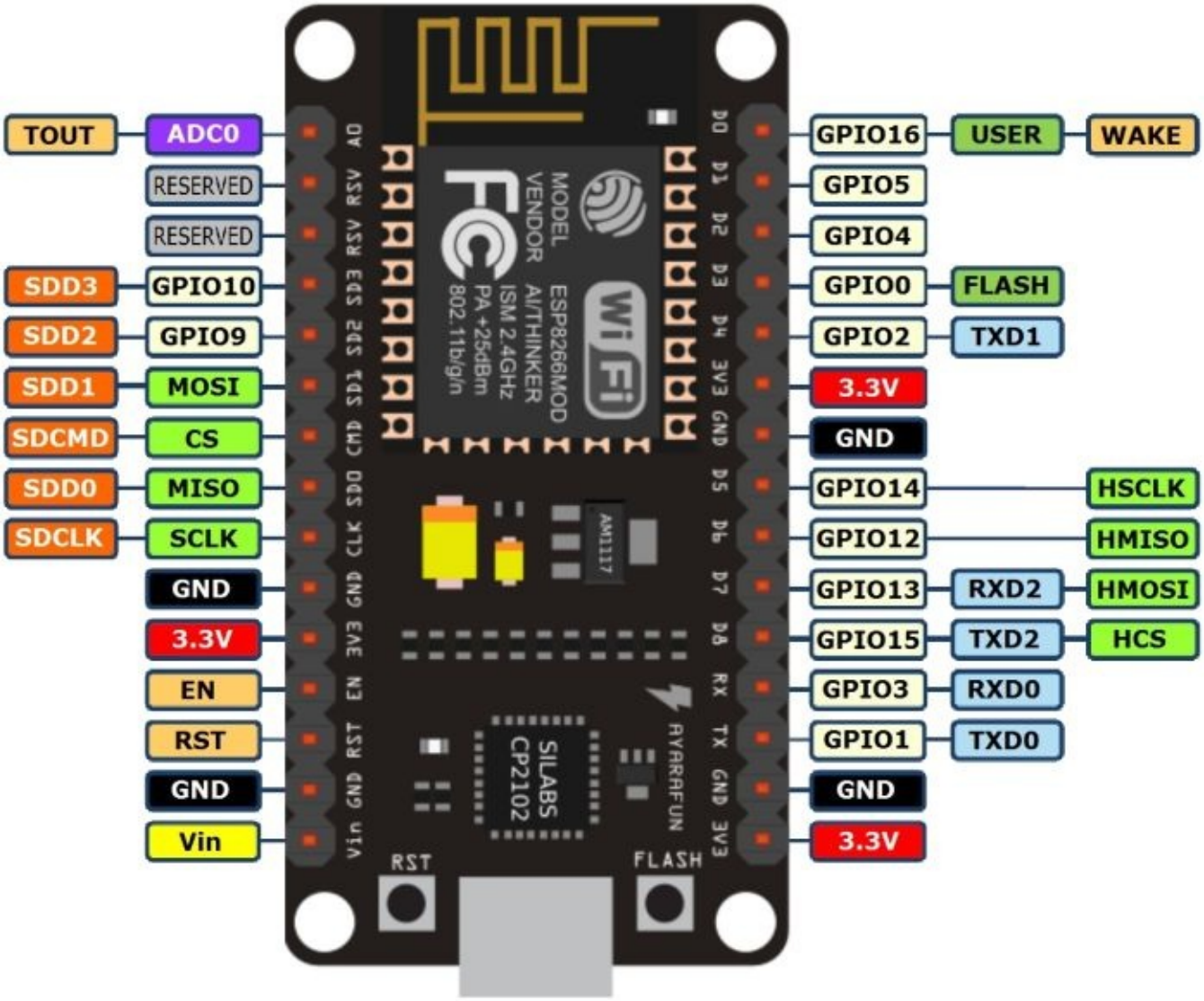
6. Working with UART

In this chapter I’m going to explain how to access UART on MicroPython board.

6.1 Getting Started

NodeMCU v2 provides three UARTs. You can see them on TXD0, RXD0, TXD1, TXD2, RXD2.





We can access UART using UART library, http://docs.micropython.org/en/latest/esp8266/library/machine.UART.html.

In this chapter, I use Arduino board as UART source. We read incoming message from UART.

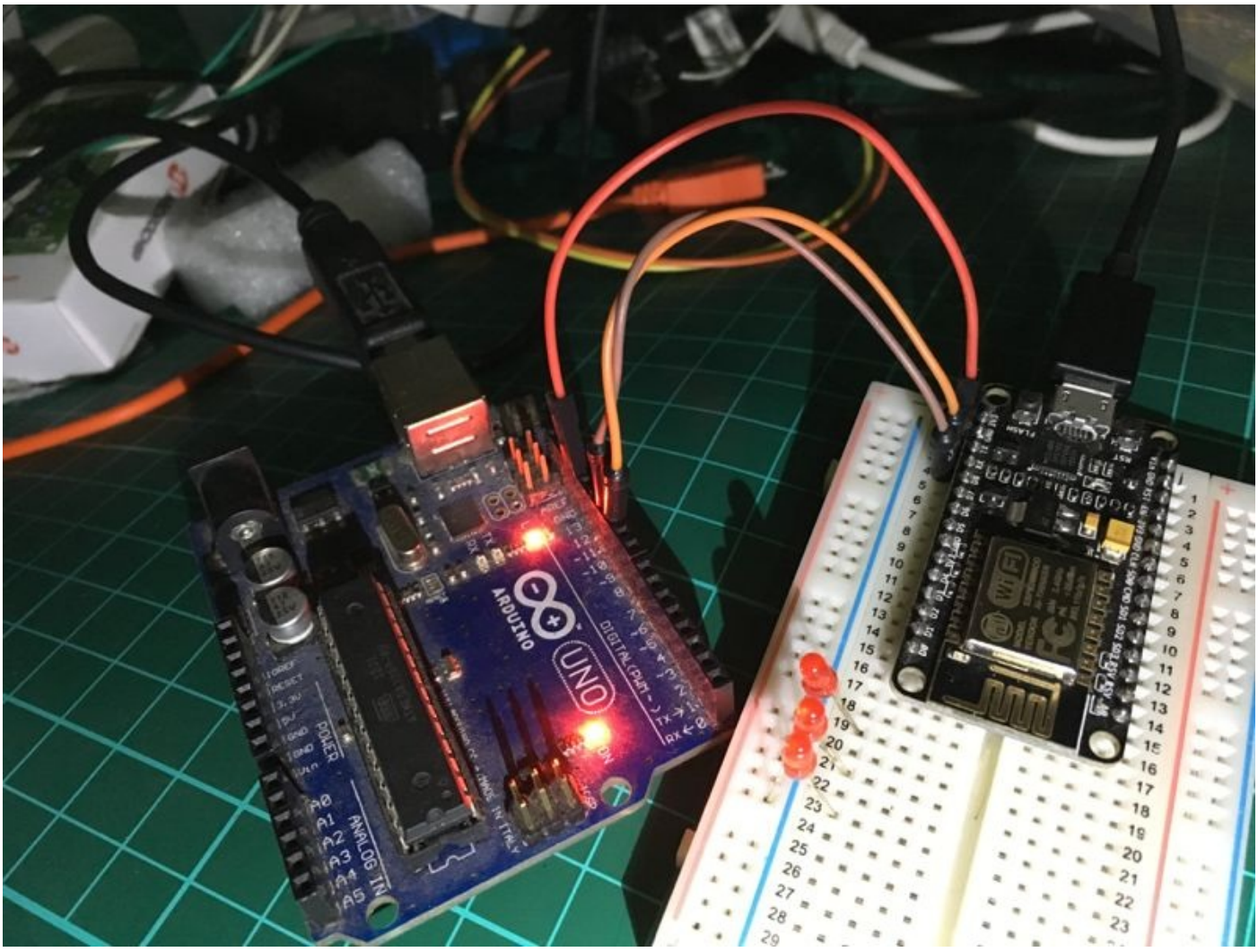
Let’s start!.

6.2 Wiring

In this scenario, I use Arduino Uno which is connected to NodeMCU board. We use UART0 on NodeMCU which is connected to Arduino board. We should connect RX pin to TX pin and TX pin to RX pin. The following is our wiring.

- NodeMCU D10 (TXD0) is connected to Arduino Digital 10 (RX)
- NodeMCU D9 (RCD0) is connected to Arduino Digital 11 (TX)
- NodeMCU GND is connected to Arduino GND

My wiring implementation can be seen in Figure below.

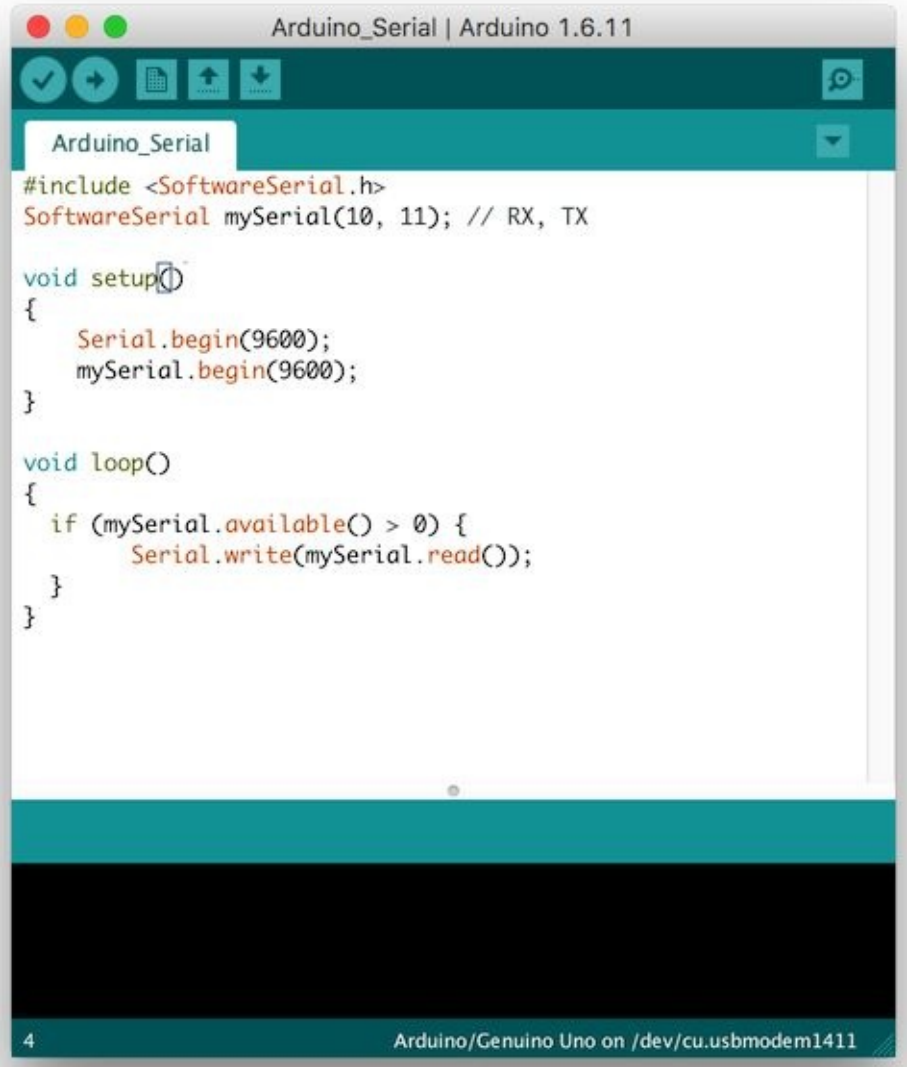


6.3 Writing a Program

Firstly, we write a program for Arduino using Arduino IDE. We use SoftwareSerial to access Serial on Digital 10 and 11. This program will wait incoming UART data and then send to Arduino UART on 0 and 1 pins.

Write this program.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
    Serial.begin(9600);
    mySerial.begin(9600);
}

void loop()
{
  if (mySerial.available() > 0) {
        Serial.write(mySerial.read());
  }
}
```



Save this program. Then, upload it to Arduino board. Before uploading, please make sure Arduino UART (digital 0, 1, 10, and 11 pins) doesn't connect to any board.

The next step is to write a program for NodeMCU board. Create a file, called **uartdemo.py**. Write these scripts.

```
from machine import UART
import time


def run():
    print('demo UART')

    uart = UART(0, baudrate=9600)
    counter = 50
    while 1:
        uart.write(str(counter) + '\r\n')
        time.sleep(2)
```

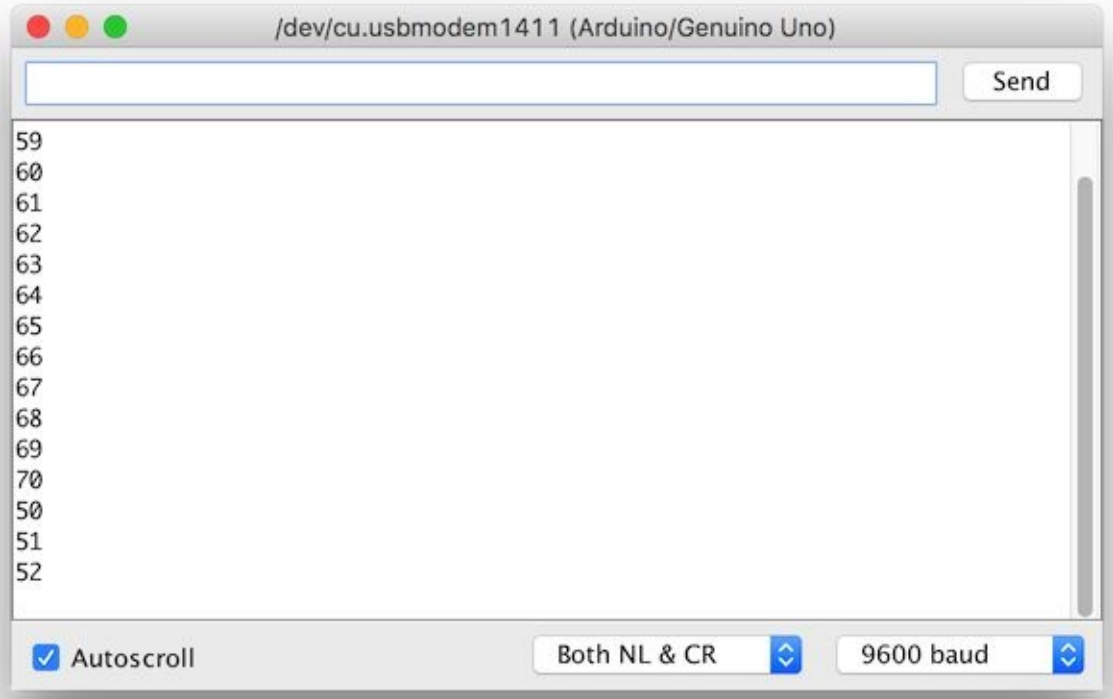
```
        counter += 1
        if counter > 70:
            counter = 50
```

Save this file.

6.4 Testing

Now you can upload and run MicroPython program via WebREPL. If done, connect NodeMCU UART to Arduino UART (Digital pins: 10 and 11).

To see the UART output, open Serial Monitor tool from Arduino IDE. Set baud 9600. You should see the UART output.



The following is program output on WebREPL.

7. Working with SPI

In this chapter I’m going to explain how to work with SPI on MicroPython board.

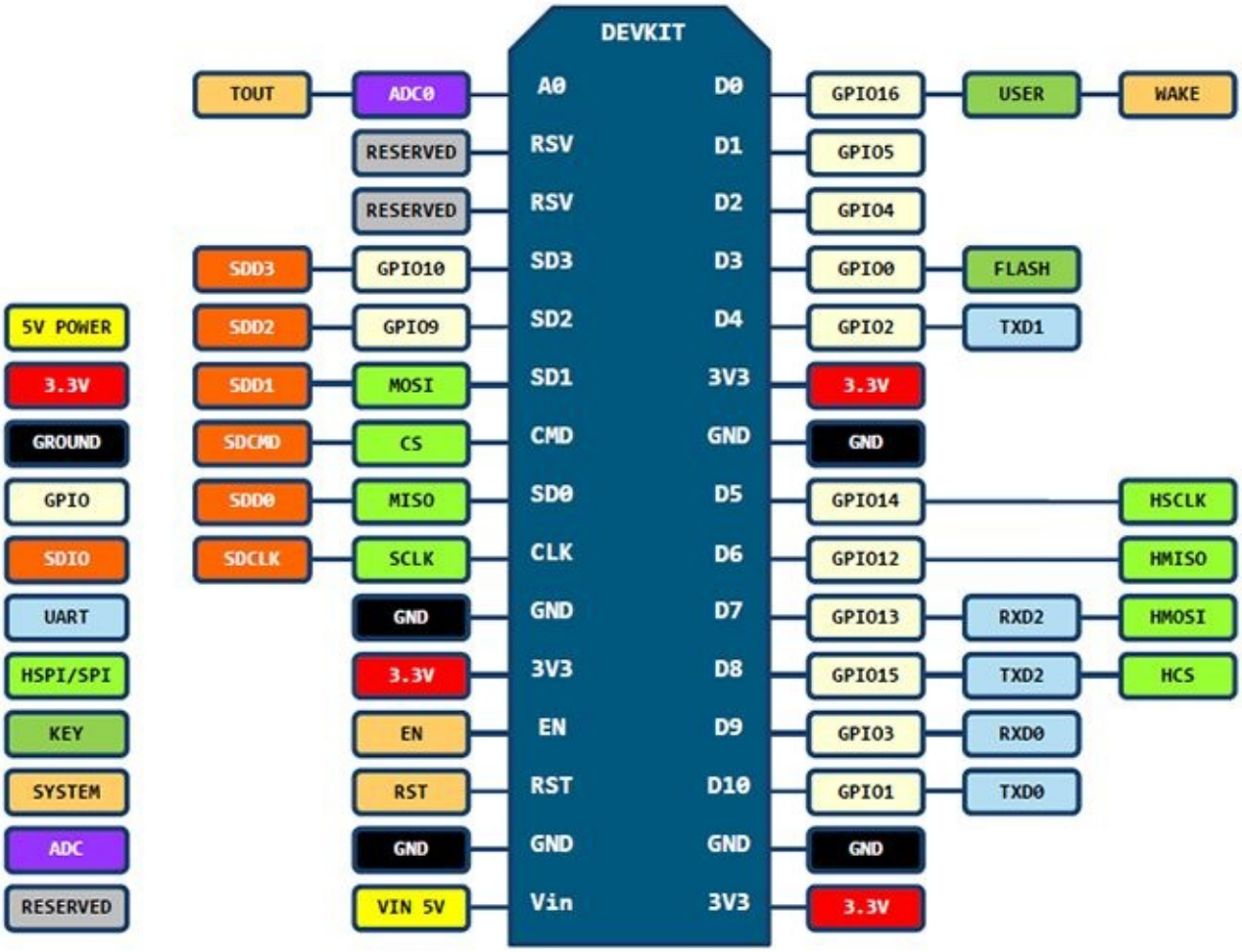
7.1 Getting Started

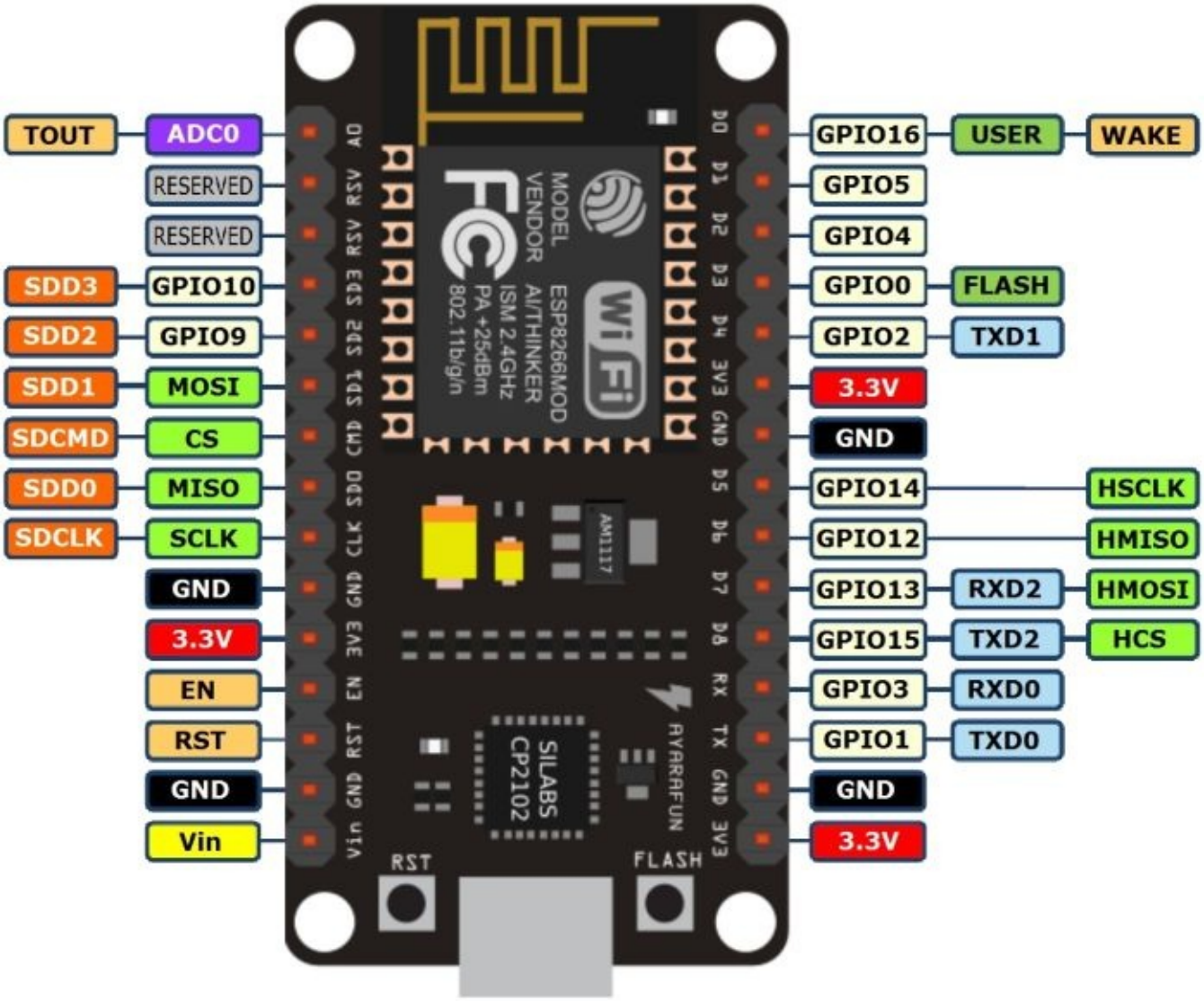
The Serial Peripheral Interface (SPI) is a communication bus that is used to interface one or more slave peripheral integrated circuits (ICs) to a single master SPI device; usually a microcontroller or microprocessor of some sort.

SPI in NodeMCU board can be defined on the following pins:

- MOSI
- MISO
- SCK

You can see these pins on SPI NodeMCU board, shown in Figure below.





We can use all pins for SPI. To access SPI, we can use SPI library http://docs.micropython.org/en/latest/esp8266/library/machine.SPI.html.

In this chapter, I use Arduino UNO as SPI source which communicates with MicroPython board.

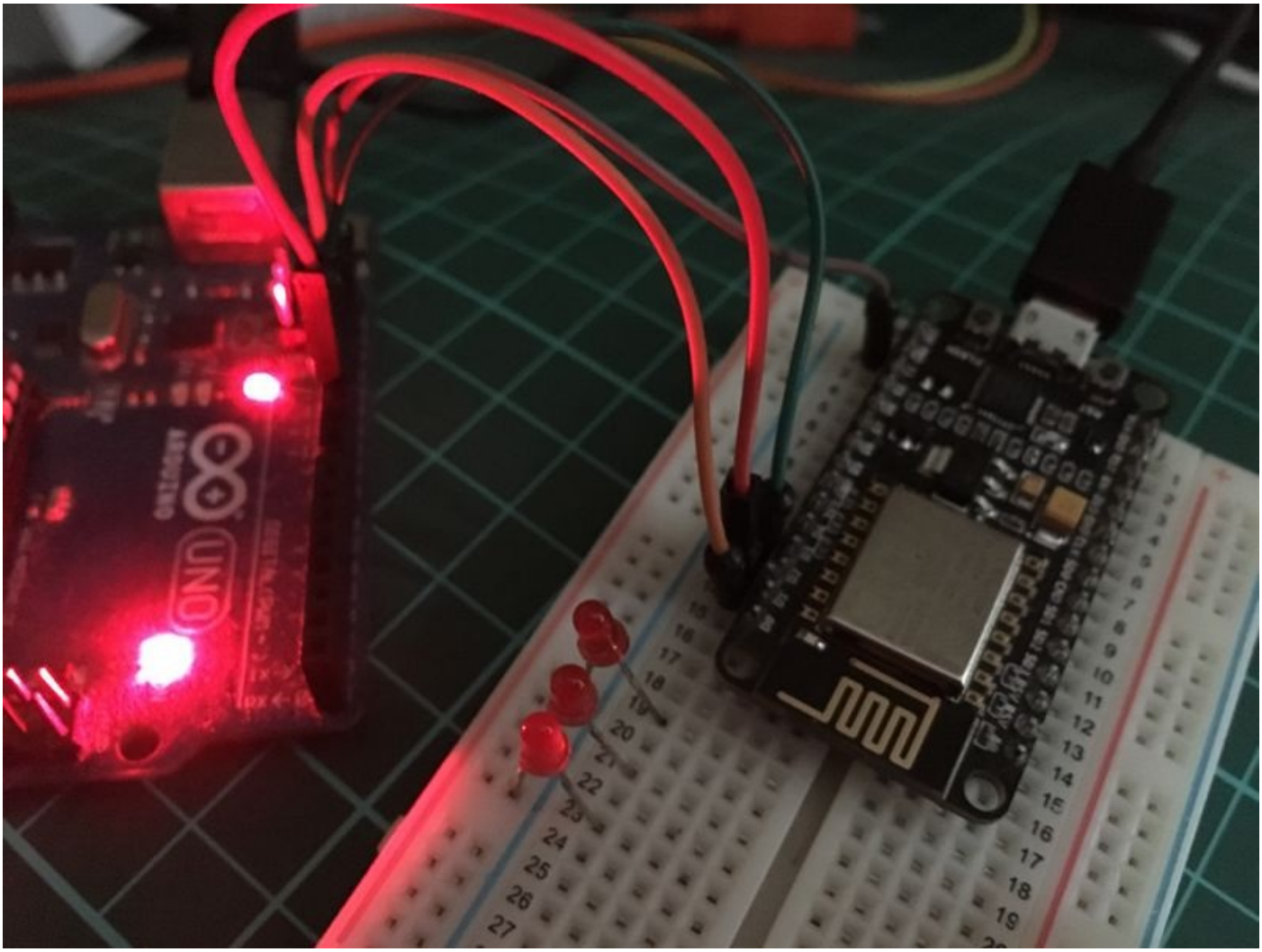
Let’s start!.

7.2 Wiring

For testing, I use Arduino board. Do wiring as follows.

- NodeMCU MOSI GPIO4 (D2) pin is connected to Arduino MOSI (Digital 11) pin
- NodeMCU MISO GPIO4 (D3) pin is connected to Arduino MOSI (Digital 11) pin
- NodeMCU SCLK GPIO5 (D1) pin is connected to Arduino SCK (Digital 13) pin
- NodeMCU GND pin is connected to Arduino GND pin

The following is a sample of wiring.

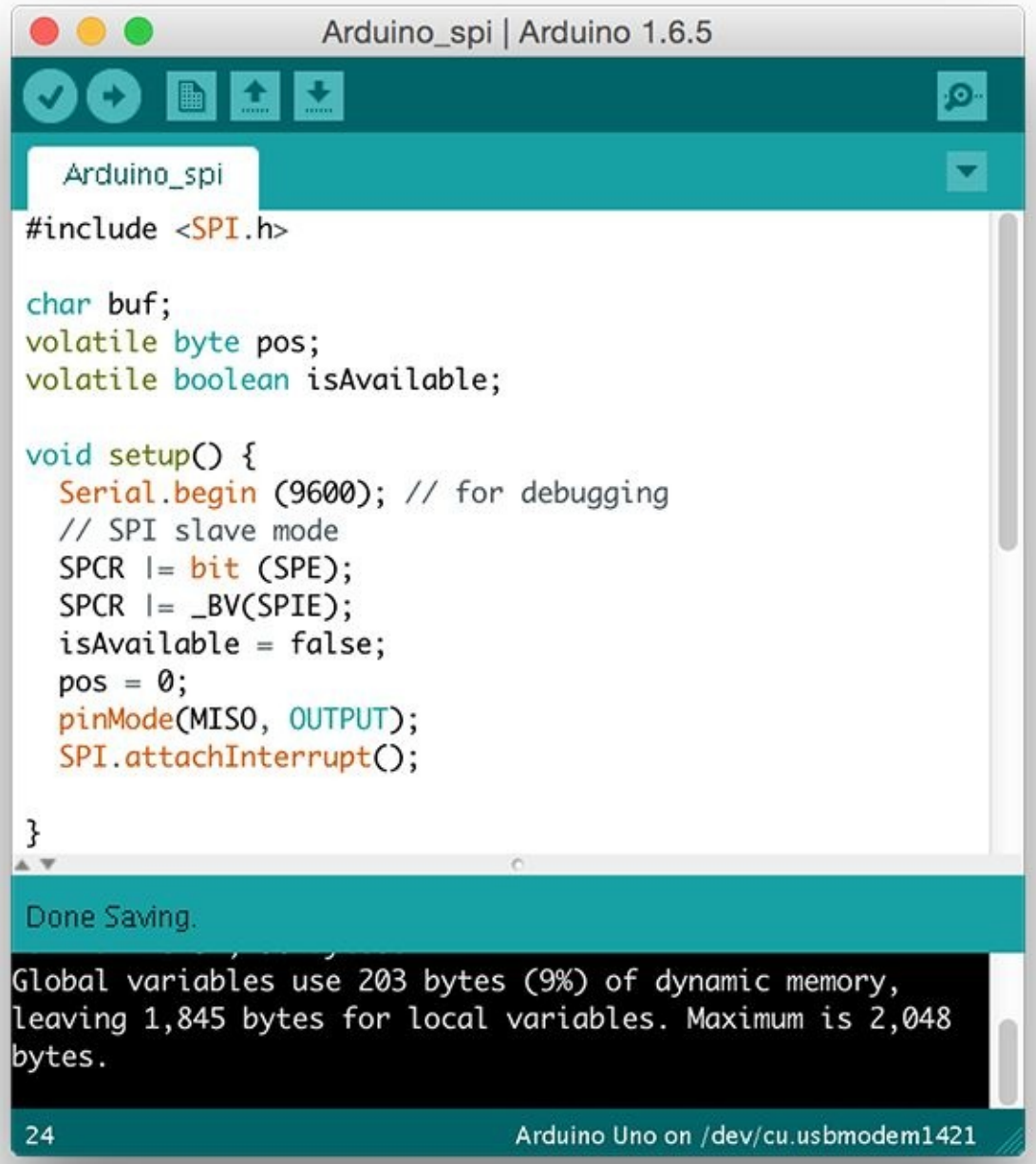


7.3 Writing a Program

Firstly, we write a program for Arduino. Write these codes on Arduino IDE.

```
#include <SPI.h>

char buf;
volatile byte pos;
volatile boolean isAvailable;

void setup() {
  Serial.begin (9600); // for debugging
  // SPI slave mode
  SPCR |= bit (SPE);
  SPCR |= _BV(SPIE);
  isAvailable = false;
  pos = 0;
  pinMode(MISO, OUTPUT);
  SPI.attachInterrupt();

}
ISR (SPI_STC_vect)
{
  byte c = SPDR;
    if(c>64 && c<91){
      buf = c;
      isAvailable = true;
    }

}


void loop() {
  if(isAvailable){
    Serial.println (buf);
    isAvailable = false;
  }
}
```

This program does wait incoming data from SPI master and then display the received data to Arduino Serial so we can see it on Serial Monitor.



Save this program as Arduino_spi. Build and upload the program to Arduino board.

The next step is to write a program for MicroPython board. Create a file, called **spidemo.py**, and write these scripts.

```
from machine import Pin, SPI
import time


def run():
    print('demo spi')
    gpio_sck = Pin(5)
    gpio_mosi = Pin(4)
    gpio_miso = Pin(0)
```

```
    spi = SPI(-1, baudrate=100000, polarity=1, phase=0, sck=gpio_sck, mo
    val = 65
    while 1:
        spi.write(chr(val))     # write 1 byte to spi
        print('write spi: ' + str(val))

        time.sleep(2)
        val += 1
        if val > 90:
            val = 65
```

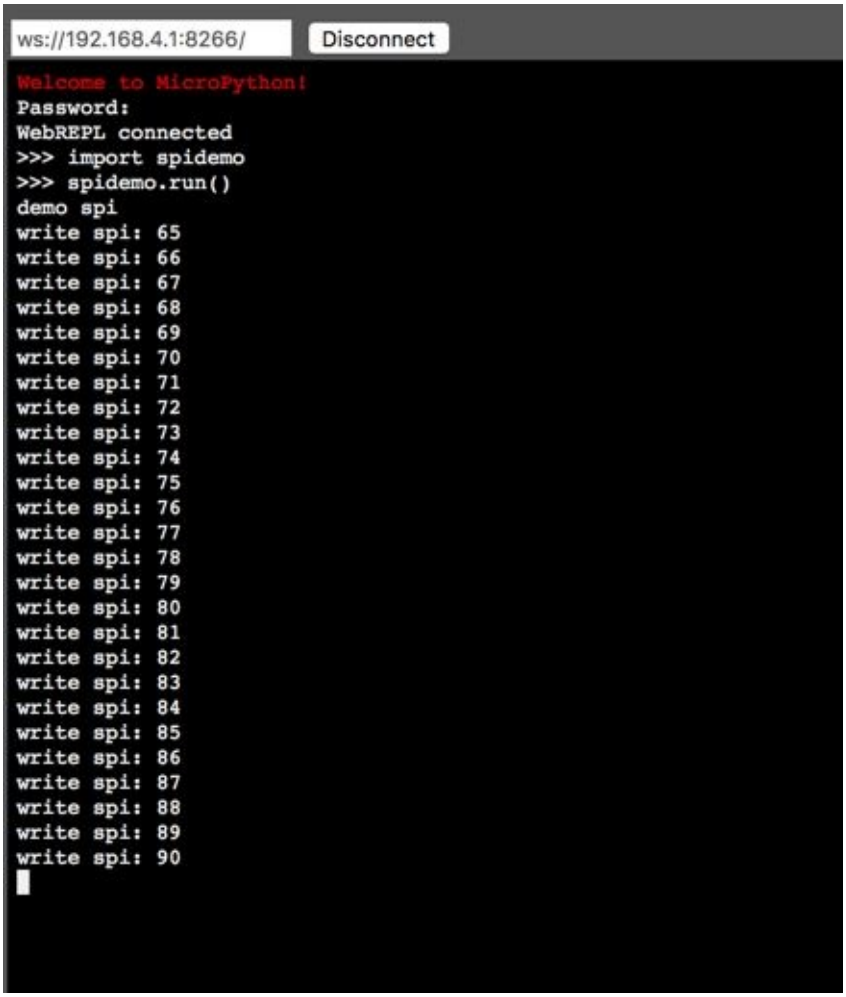
Save this code.

7.4 Testing

Now you can upload MicroPython program to MicroPython board using WebREPL. If done, you can run the program.

```
>>> import spidemo
>>> spidemo.run()
```

You should see received data from SPI.



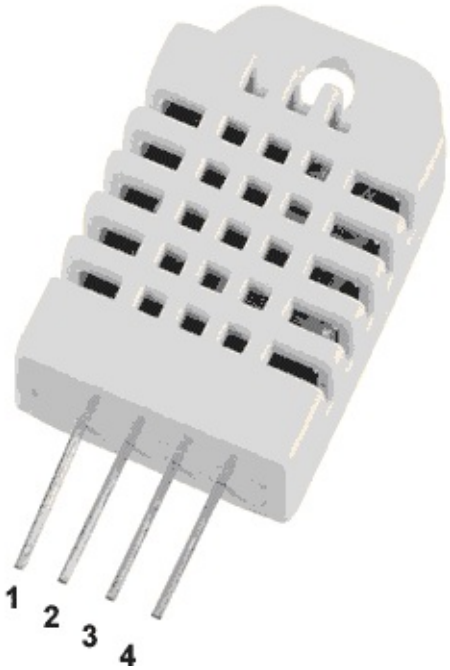
8. Working with DHT Module

In this chapter I’m going to explain how to work with DHT module on MicroPython boards.

8.1 Getting Started

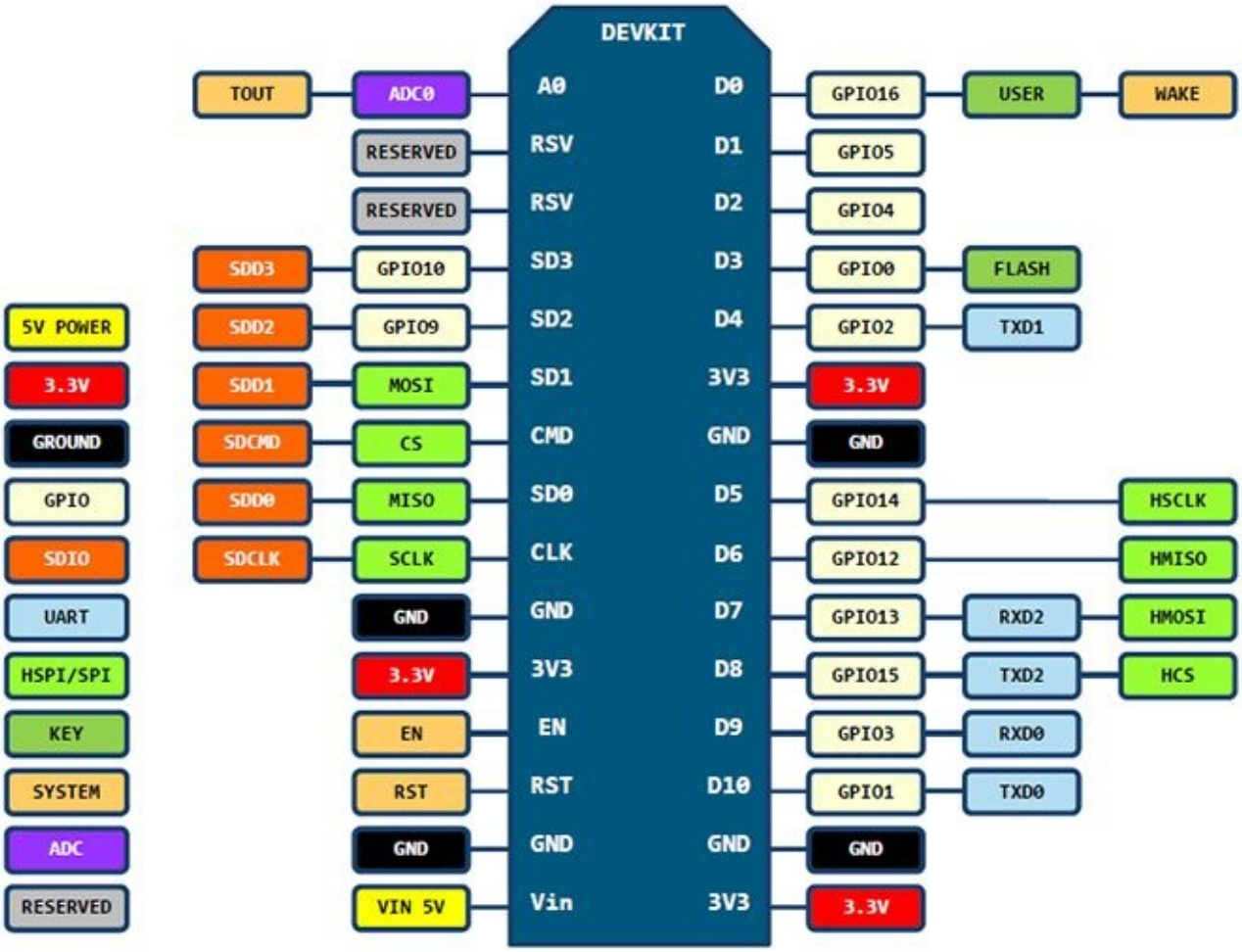
In this chapter, we try to develop a simple application to access DHT module. This module can sense temperature and humidity. It’s easy to find in electronic stores. You can see DHT22 layout in Figure below.

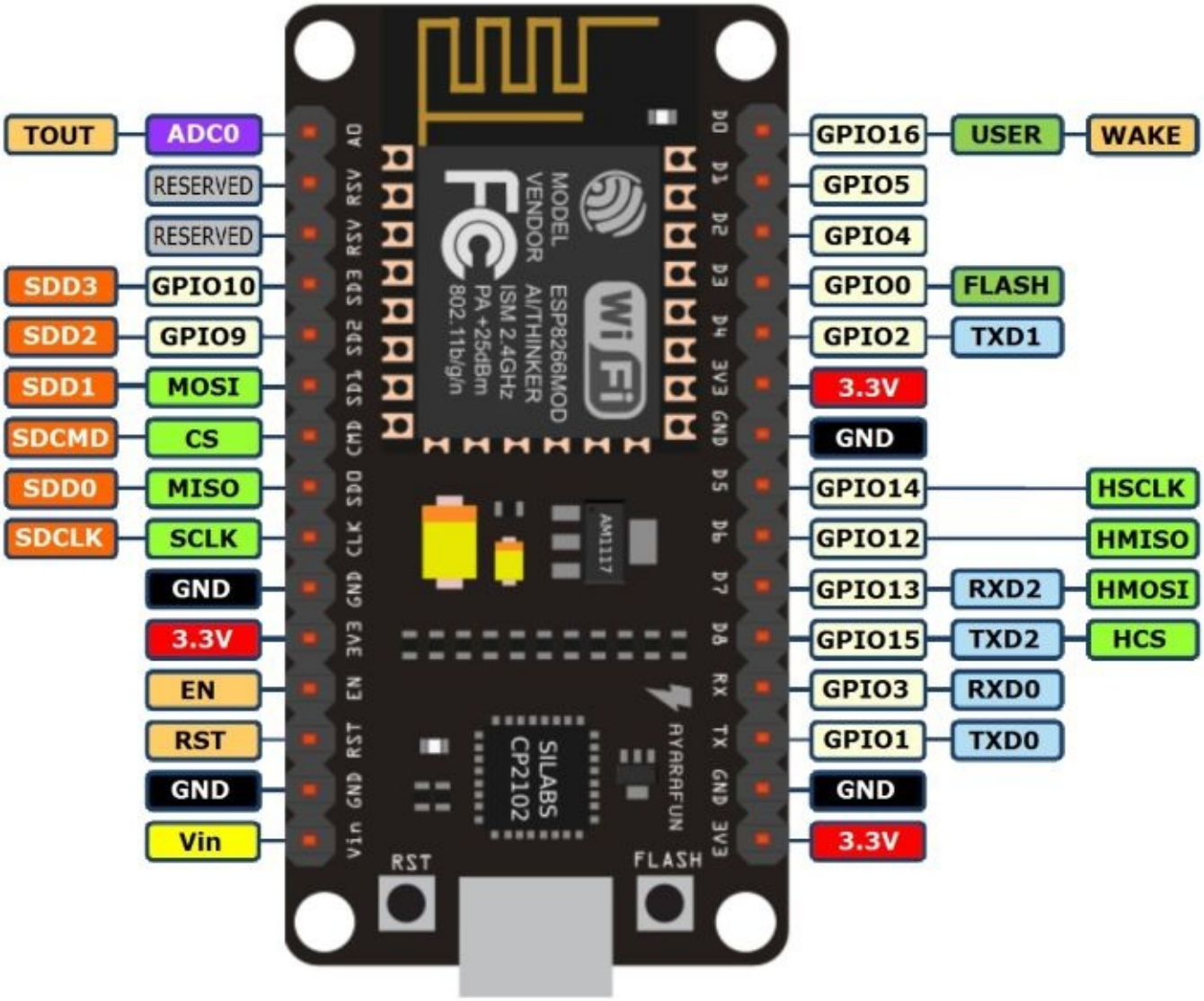
DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



8.2 Wiring

I use NodeMCU for MicroPython board. The following is NodeMCU layout.

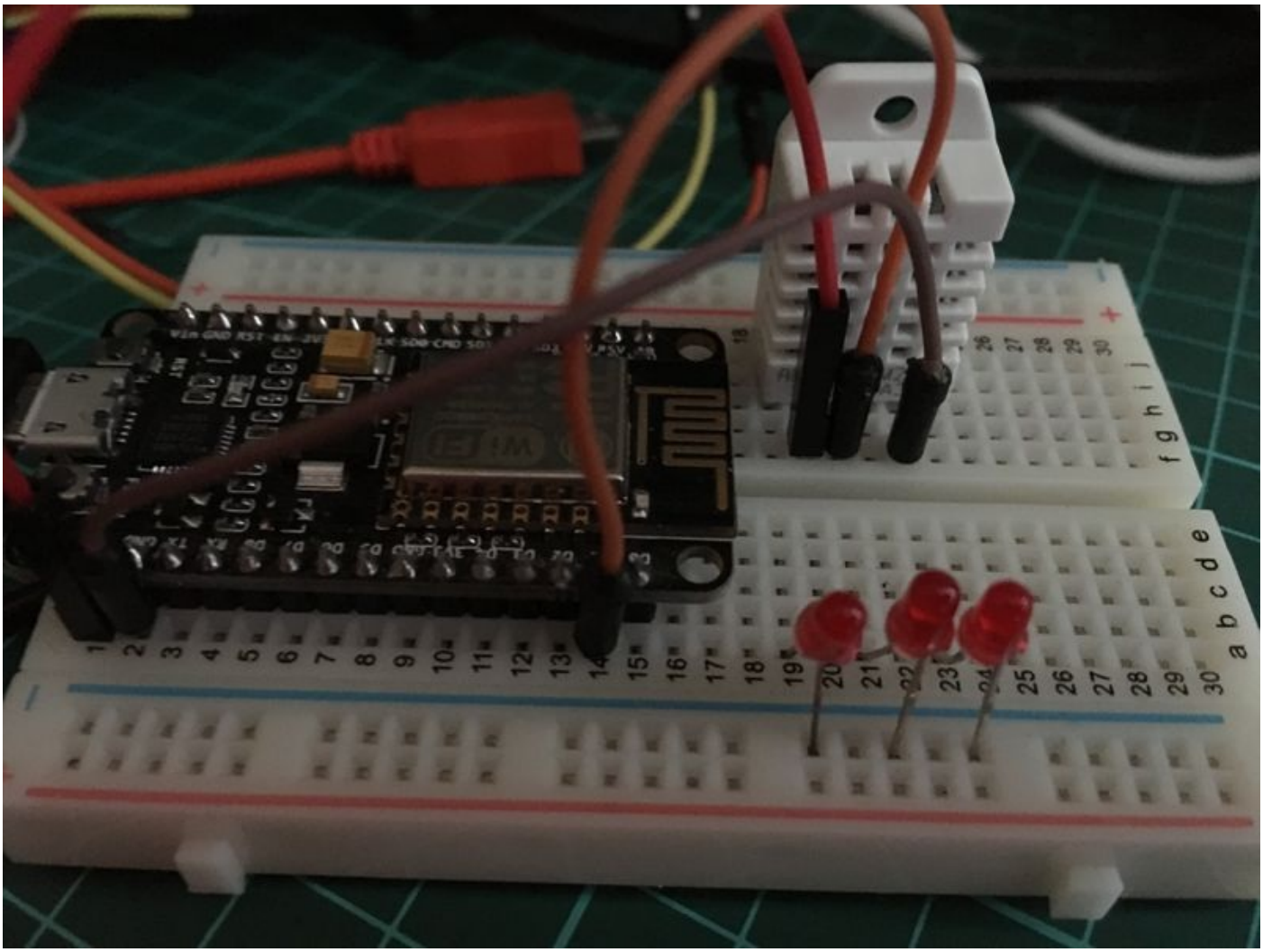




Our wiring for DHT22 and NodeMCU as follows:

- DHT VCC is connected to NodeMCU 3.3V
- DHT GND is connected to NodeMCU GND
- DHT Data is connected to NodeMCU GPIO5 (D1)

The following is our implementation.



8.3 Writing MicroPython Program

Now we can access DHT using dht module from MicroPython. Open editor and write these scripts.

```
from machine import Pin
import dht
import time


def run():
    print('dht module demo')

    gpio_dht = Pin(5)
    d = dht.DHT22(gpio_dht)
    while 1:
        d.measure()
        temperature = d.temperature()
        humidity = d.humidity()

        print('Temperature: ' + str(temperature) + ' Celsius')
        print('Humidity: ' + str(humidity) + ' % RH')
        time.sleep(2)
```

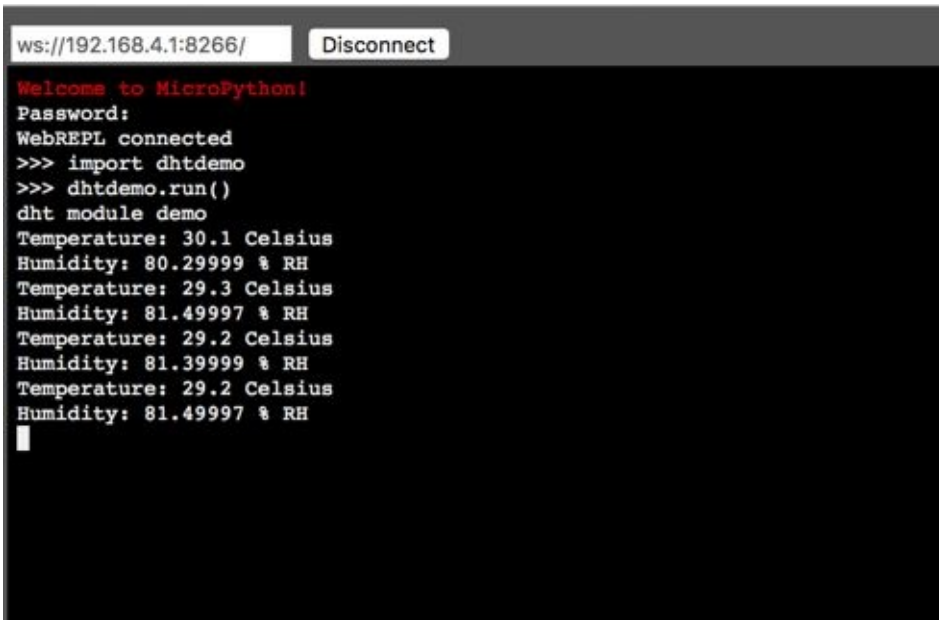
Save this program as dhtdemo.py.

8.4 Testing

Now you can upload dhtdemo.py to MicroPython board via WebREPL. Then, run the program.

```
>>> import dhtdemo
>>> dhtdemo.run()
```

You should see temperature and humidity on Terminal.



Source Code

You can download source code on http://www.aguskurniawan.net/book/micro82667.zip .

My Books for ESP8266 Development

I wrote two books related to ESP8266 development.

NodeMCU Development Workshop, http://blog.aguskurniawan.net/post/nodemcu.aspx.

SparkFun ESP8266 Thing Development Workshop, http://blog.aguskurniawan.net/post/sparkfun8266.aspx.

Contact

If you have question related to this book, please contact me at aguskur@hotmail.com . My blog: http://blog.aguskurniawan.net