INTRODUCTION TO IoT

SPRING 2020

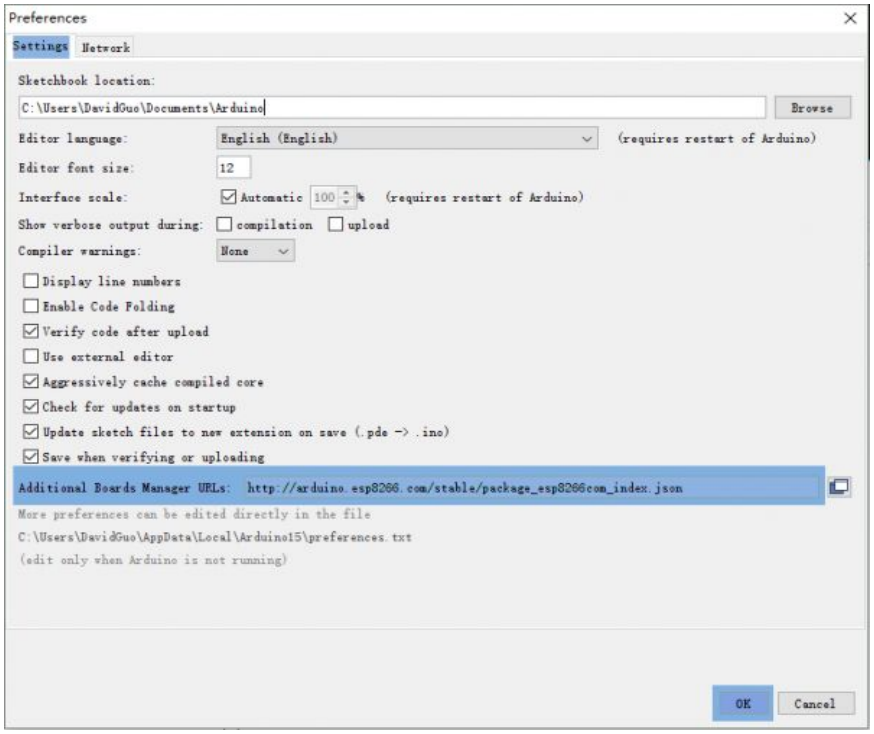
LAB-1

Introduction

NodeMCU comes pre-programmed with Lua interpreter, but you don't have to use it! Instead, you can use the Arduino IDE which may be a great starting point to familiarize themselves with the technologies surrounding the IoT. Note that when you use the NodeMCU board with the Arduino IDE,it will write directly to the firmware, erasing the NodeMCU firmware.So if you want to go back to Lua SDK,use the "flasher" to re-install the firmware.The NodeMCU programming can be as easy as in Arduino,the main difference is the distribution of pins in the nodemcu board.

Setup for LAB experiments:

- Install the COM/Serial port driver
    - The new version NodeMCUv1.0(we use this) comes with the CP2102 serial chip,you can download and install the driver here.
- Download Arduino IDE from Arduino.cc
- Install the ESP8266 Board Package
    - Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional Board Manager URLs field in the Arduino v1.6.4+ preferences (Open Arduino IDE–>File–>Preferences–>Settings).

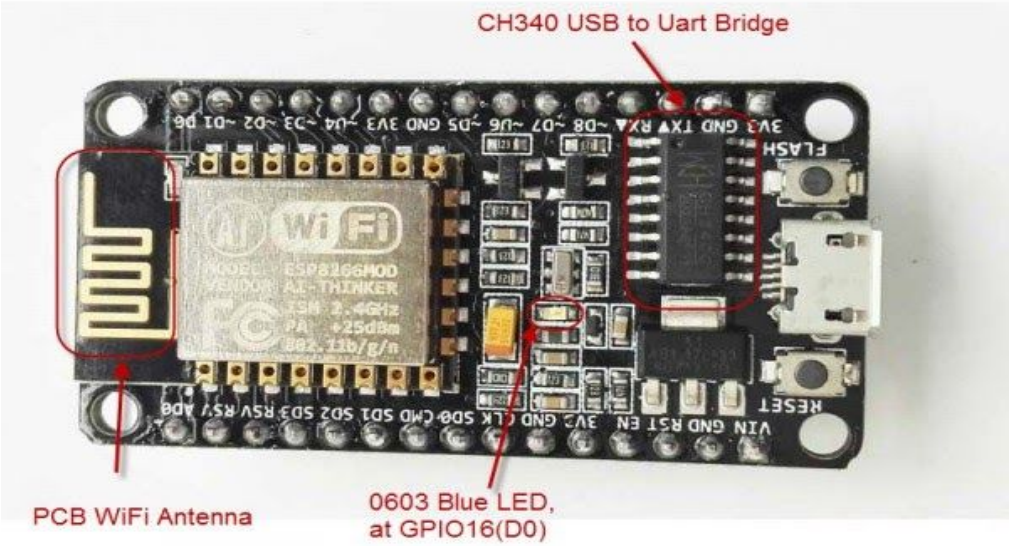


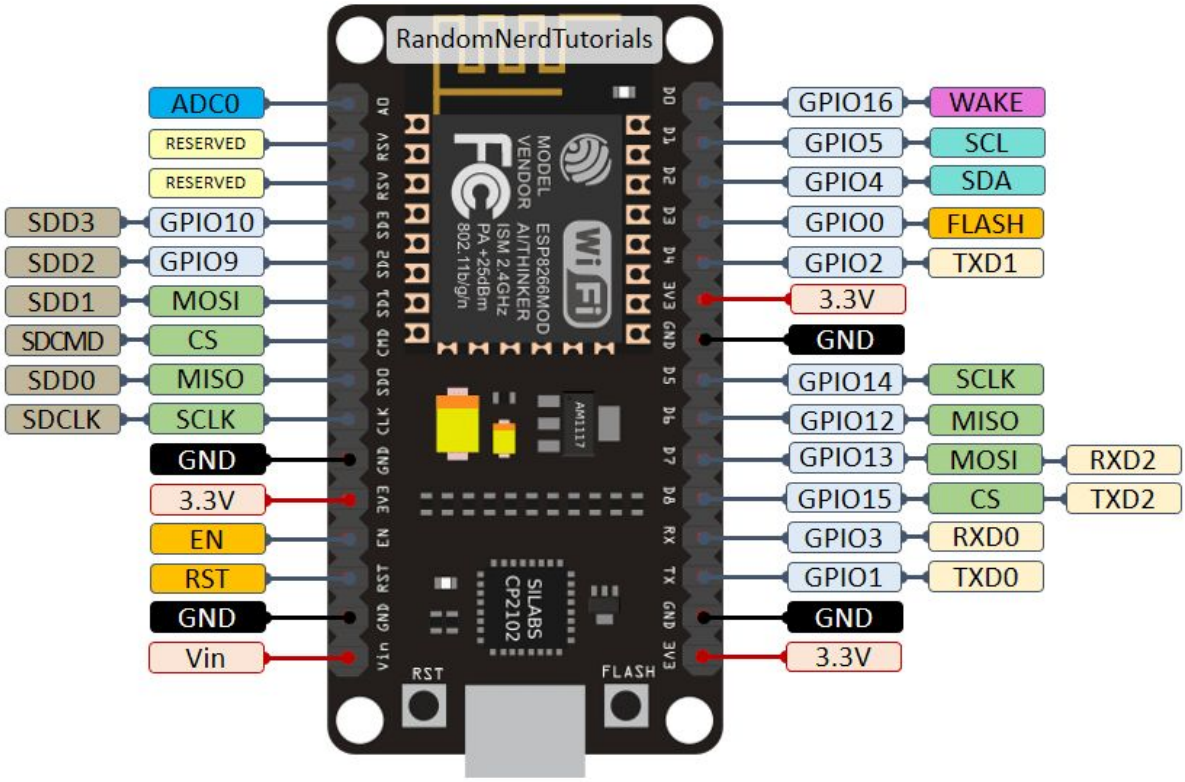
- Next, use the Tools =>Board=>Board Manager to install the ESP8266 package.

- Note: You'd better close the Arduino IDE and restart it again.
- When you've restarted, select NodeMCU 1.0 from the Tools->Board dropdown
- Config the Board menu and choose the right Port for your device.
    - CPU Frequency：80MHz,
    - Flash Size：4M（3M SPIFFS），
    - Upload Speed：115200

LAB EXPERIMENT- 1_a

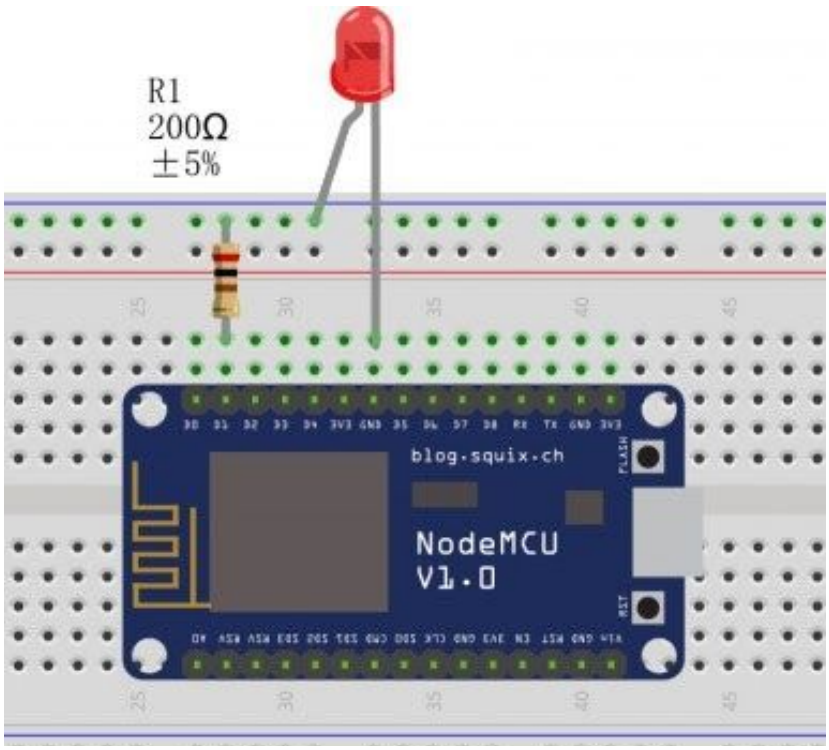
- In this lab session,we will introduce how to control an external device like LED and BUZZER using the pins on NodeMCU.
- **Hardware Required**
    - NodeMCU x 1
    - LED x 1
    - 200 ohm resistor x 1
    - Micro USB cable x 1
    - PC x 1
- Blink the onboard LED connected to D0(GPIO16) on the nodemcu.
- Follow the above mentioned board config to use NodeMCU 1.0 from Tools->Board dropdown
- Use the below given PINOUT diagram of the NodeMCU





- Pseudocode:
    - Setup block
        - Declare the led pin D0 as output using pinMode(<pin number>,<pin type>) function
    - Loop block

        - Make the LED blink using the functions digitalWrite(<pin number>,<output>) and delay(<time in milliseconds>) with appropriate delay in milliseconds.
- Blink an external LED
    - Connect one end of the resistor 200 ohms to D1(or any digital pin) and the other end to LED long leg(positive leg / anode). Connect the short leg(negative leg/cathode) to GND.
    - The value of the resistor in series with the LED may be of a different value than 200 ohm; the LED will lit up also with values up to 1K ohm.



- Pseudocode
    - Same as internal LED except for LED pin now is D1(or any other digital pin).

LAB EXPERIMENT- 1_b

- In this lab session,we will introduce how to print the output of the sensor data on to serial monitor using the micro USB cable.
- **Hardware Required**
    - NodeMCU x 1
    - Micro USB cable x 1
    - PC x 1
- Connect your NodeMCU to the PC and put below code to your Arduino IDE .

- Pseudocode
    - Use Serial.begin(<baud rate>) to begin the serial communication
    - Use Serial.println(<data>) to print on the serial monitor in a new line.
    - The data in the experiment can be a randomly generated number using random(<min>,<max>)

LAB EXPERIMENT- 1_c

- In this lab session,we will introduce how to read the inputs using Serial.read from the serial monitor and controlling the buzzer's frequency using the tone function.
- **Hardware Required**
    - NodeMCU x 1
    - Micro USB cable x 1
    - PC x 1
    - Passive Buzzer MH-FMD
- Difference between **active and passive buzzer**:
    - An active buzzer will generate a tone using an internal oscillator, so all that is needed is a DC voltage. A passive buzzer requires an AC signal to make a sound. It is like an electromagnetic speaker, where a changing input signal produces the sound, rather than producing a tone automatically.
    - With a "passive" device or speaker, you have to send it an AC "sound signal" and you can control the sound. The Arduino needs to generate the "tone". With enough hardware & software you could make it play music. etc. The only downside (besides a bit of extra programming) is that it takes processor-time to generate the sound.
- Pseudocode:
    - Setup block:
        - Declare the output pin for the buzzer.
    - Loop block:
        - Check if the characters are available using Serial.available()
        - Read the string and save as frequency variable
        - Use the tone() function to play the sound with the given frequency variable.
        - Use appropriate delay and repeat above steps for every new input.