IoT exercises – Week 1

In the first week, you will start playing with NodeMCU/ESP8266, a low-cost and efficient platform, creating your own project using PyFlasher/Esplorer and configuring the environment for your project.

-If you are in Portsmouth, you should have collected the basic components. If not yet, please come to my office 1.12 BK, 1300-1700 on working days to collect the hands-on hardware components. (Just in case that I am not in the office, please drop me an email before you come to the campus).

-For those who stay in the UK but outside Portsmouth, you will be contacted for your address hopefully you will receive the device by the end of this week through post.

-For those who are in their home countries outside the UK, you should have received the mail regarding the equipment and need to purchase the basic components yourself (no more than 15-20 GBP in total). Please do contact Dalin for more details.

First you will need to connect the NodeMCU/ESP8266 with the PC and then you will need to attempt some exercises as shown below:

1) Check the GPIO pins and light the LED, 2) Make the LED blink by adding a timer function

(optional) **3) Make the LED blink in predefined interval patterns (eg. On for 1000ms, then Off for 500ms, then On for 2000ms, then Off for 100ms, …)**

If you are experienced with either Raspberry Pi or Arduino practice, you are encouraged to explore more modules available on NodeMCU/ESP8266: 1) you can try your own modules or even build your own firmware, 2) you can bring your own device of Raspberry Pi or Arduino Kit and burn the firmware to the NodeMCU/ESP8266. DON'T limit yourself to the unit materials.

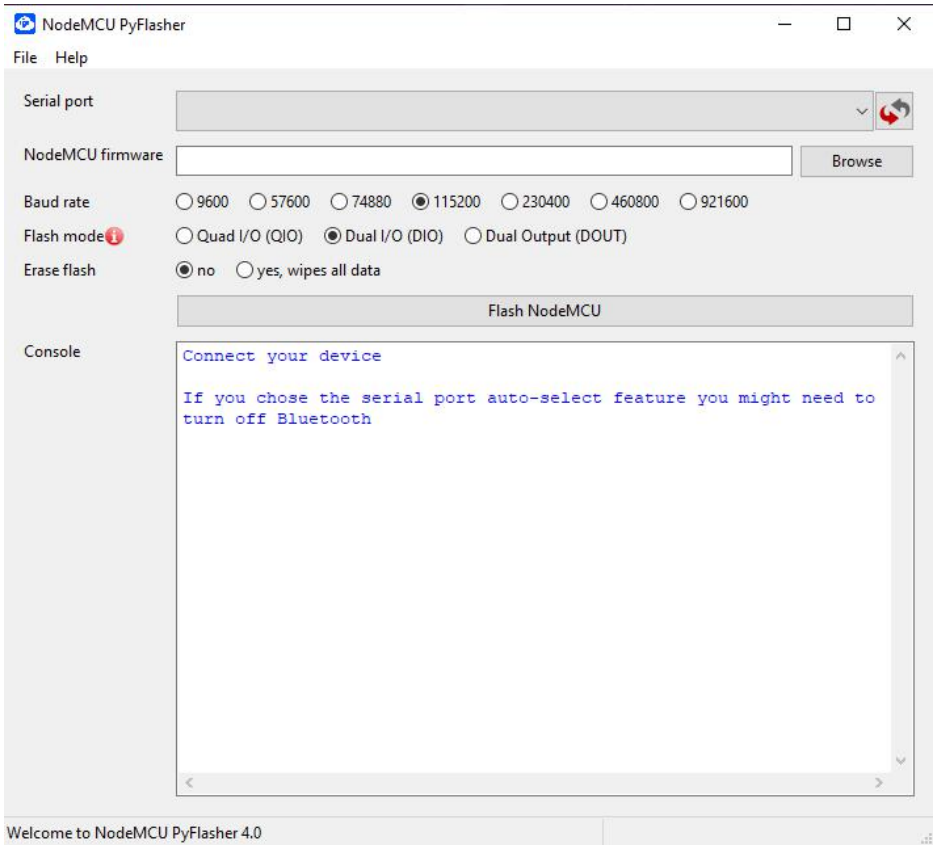
Step 1:

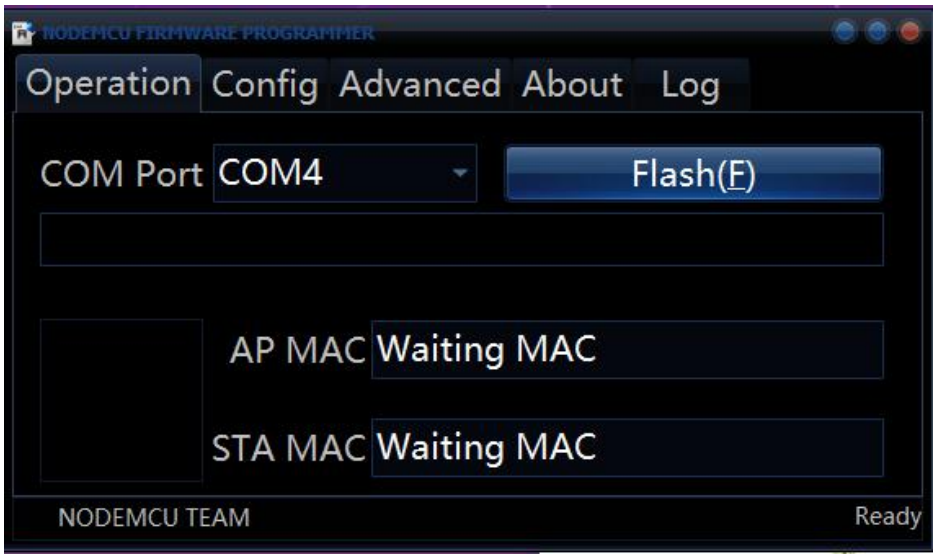
In this step, you need to download and install in the Windows OS or Mac or Linux from Moodle in the folder of IoT Practical Drivers and Software.

)

1. **the USB-UART drivers for either CP2102 or CH341 (installed already on the PCs on campus) For this TB, particularly, the delivery is likely to be all online. You need to install the driver yourself if it is not in your PC yet.**
2. **the** PyFlasher **or the** ESP8266Flasher **to burn firmware to your NodeMCU/ESP8266.**
3. **the firmware "**nodemcu-19-modules-firmware-float.bin**" of 19 modules.**

- Connect your NodeMCU/ESP8266 through the USB of PC.
- Open either the PyFlasher or the ESP8266Flasher, choose the Serial Port and the firmware .bin document you just downloaded.
- Set the Baud rate at 115200 or larger (OK to try a lower one), either erase the flash or not, then flash the NodeMCU.





- After the burning, you can reset the NodeMCU by pressing the RST button.

Step 2:

So far, you have burned the firmware in your own NodeMCU/ESP8266. In the step 2, you will need to compile and upload the code through Esplorer (if you are using Linux, esptool is recommended for using the command line)

1. **Download the zipped file of** Esplorer **from Moodle**
2. **The Esplorer is executed through the JAR file after unzipping. JAR requires you to add the Java Runtime Environment no later than** JRE8.

During the pandemic online delivery, you might need to set your local machine (Particularly Win10 PC). Thanks to Mathew, relatively clear steps are given as follow for you to get the JRE8 ready.

1. Download and install Java SDK8.

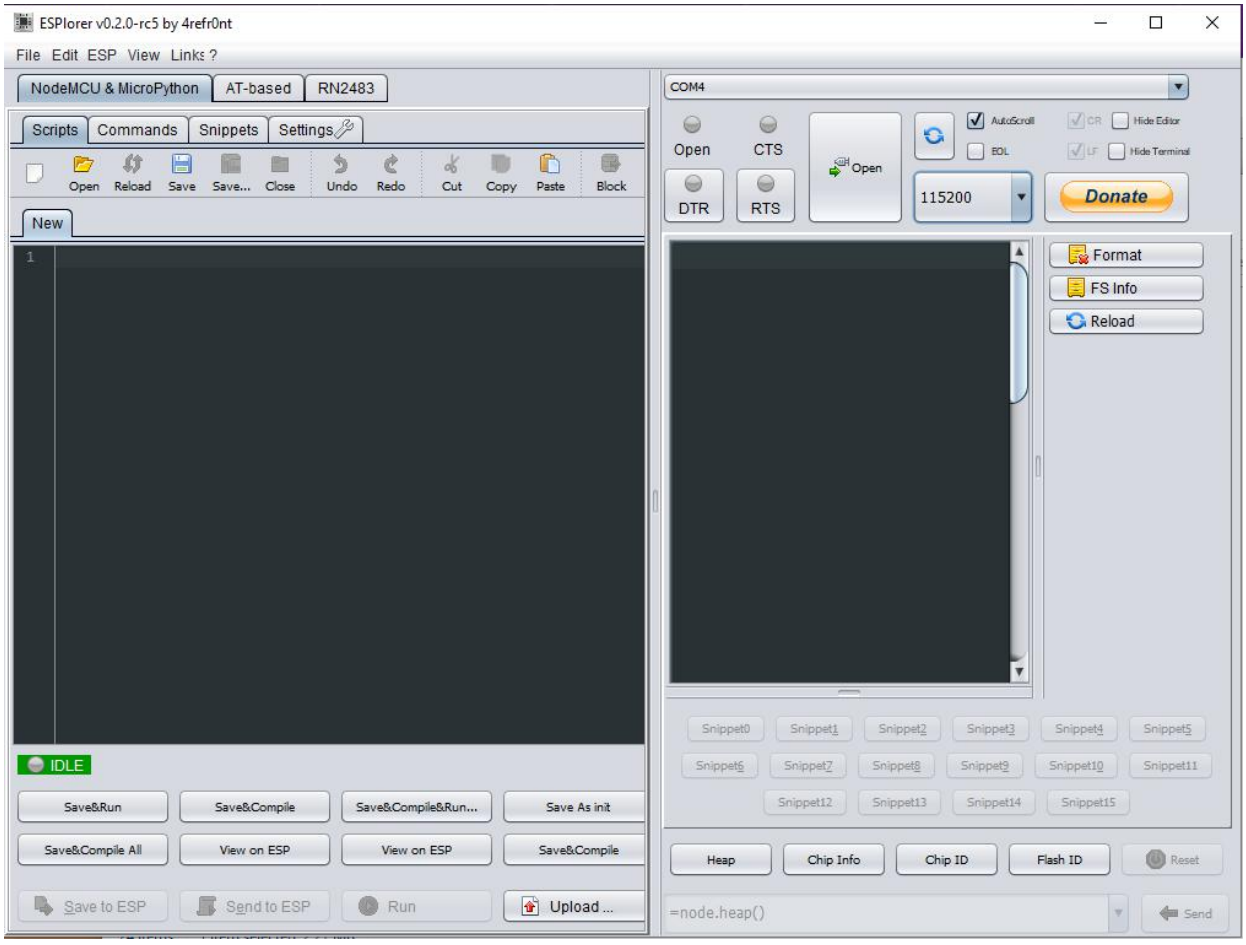
2. Make sure JAVA_HOME exists in System environment models (search Edit system environment models in windows) if it doesn't add it https://confluence.atlassian.com/doc/setting-the-java_home-variable-in-windows-8895.html

- Now you can run the ESPlorer.jar
- Before you start coding, choose the same right Serial Port and the correct Baud

rate, press Open (If the NodeMCU is not detected, reset it by pressing RST button on the chip).

- If the flash is erased, the ESPlorer will format the files. After the 30-second formatting, you will see the NodeMCU version and the "lua: cannot open init.lua".
- Init.lua is the initial file to be executed.
- The same old step of "HelloWorld": type in and save the following line of printing.

  print("HelloWorld")
- Click "Save to ESP" or "Ctrl+S", you will see the output.



If the previous step is unclear, please refer to the following link of official documents for a more detailed guidance of configuration:

https://nodemcu.readthedocs.io/en/master/getting-started/

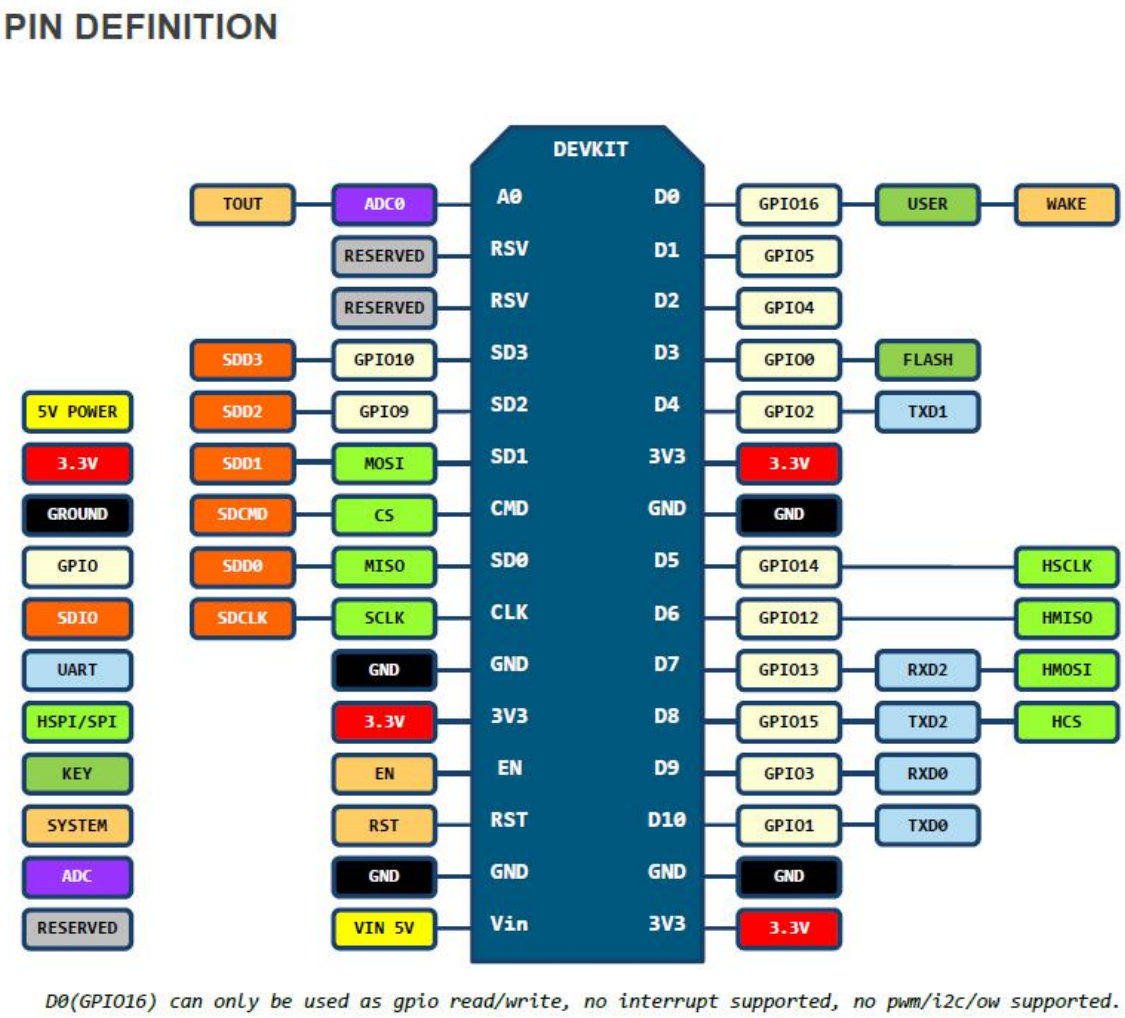
So you have successfully burned the firmware and upload the init.lua file of "HelloWorld" to the NodeMCU/ESP8266. Now please familiarize yourself with

the operations and further develop functions for the system.

Exercise 1:

In this exercise, you will need to check the GPIO pins and light the LED.

- The GPIO Pins and their indices are shown as below



A0: ADC, Pin Index 0

D1-D8: Pin Index 1-8

RX: Pin Index 9

TX: Pin Index 10

SD2-SD3: Pin Index 11-12

D0 Default gpio.High

D1-D2 Default gpio.Low

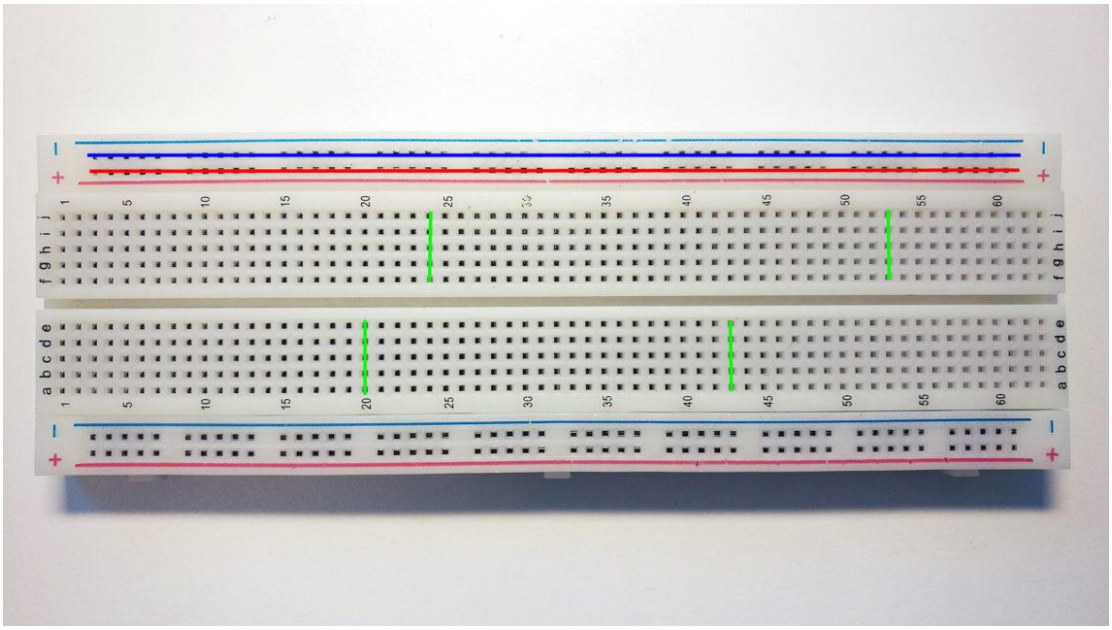
D3-D8 Default gpio.High

RX, TX Default gpio.High

SD1 Default gpio.Low

SD2-SD3 Default gpio.High

- In case that you have not used the breadboard before, the 5 pin holes in a row (denoted by Green) are connected, the Blue/Red pin holes are connected respectively.



- There are 2 LEDs on the chip controlled by the D0 and D4 pins. They are lighted when their mode and value are set as OUTPUT and LOW.
  - **When pin 4 is low, LED near ESP8266 is lighted**
  - **When pin 0 is low, LED near CP2102/CH340G is lighted**
- gpio module is used here:

==========

```
pinLED1=4
```

```
gpio.mode(pinLED1, gpio.OUTPUT)

gpio.write(pinLED1, gpio.LOW)

pinLED2=0

gpio.mode(pinLED2, gpio.OUTPUT)

gpio.write(pinLED2, gpio.HIGH)
```

==========

- Save them as the init.lua file to NodeMCU through Esplorer. Check the LED. (Reset after every time you save files to the chip to run them automatically.)
- For more about gpio module, refer to

https://nodemcu.readthedocs.io/en/master/modules/gpio/

- Now you can check the rest of the GPIO Pins as output or input and their voltages, a multi-meter is available for you to use.

Thanks to Sam, an advice is added here: please remember to include a resistor in your circuit before you power the led to avoid burning it.

Exercise 2:

In this exercise, you will need to make the LED blink by adding a timer in your function.

- tmr module is used here:

==========

```
mytimer = tmr.create()

mytimer:register(2000, 1, function()

    --the function inside the object of tmr (named mytimer is executed every 2000ms)

    --put your function here

end)

mytimer:start()
```

==========

- For more about tmr module, refer to

https://nodemcu.readthedocs.io/en/master/modules/tmr/

- Once you have blinked 1 of the LED, blink them sequentially (ie, when LED1 is on, LED2 is off; when LED1 is off, LED2 is on).
- Notice that NodeMCU/ESP8266 provides up to 7 independent timers to run at the same time. Try to combine more to enhance your application.

(Optional) Exercise 3:

- Using gpio and tmr module, make a LED blink following a predefined time interval. For example:
  - When the input is {1000,500,2000,200}, the LED is on for 1000ms, then off for 500ms, then on for 2000ms, then 200ms, then repeat the cycle.
  - When the input is {1000,500}, the LED is on for 1000ms, then off for 500ms, then repeat the cycle.
  - When the input is either 0 or 1, the LED is set always on or off