

IoT exercises – Week 2

This week, you will go on familiarising yourself with NodeMCU/ESP8266, the low-cost and efficient platform, creating your own project using PyFlasher/Esplorer and using the bespoke built-in modules. If you have not finished the exercises for Week 1, please go on with them. Should you need further support, please just let me know it during the practical.

The official documentation is provided here again for your reference. Please remember, when you are building your own IoT project in the future, always refer to the bespoke built-in modules and read their **documentation** first, which will be helpful!!!

<https://nodemcu.readthedocs.io/en/master/>

In the lecture we have talked about the control of SoC and categorise them into adjustable control and ON/OFF control. This week we will mainly practice on the adjustable control using **Pulse Width Modulation (PWM)**. The PWM control is provided by the NodeMCU/ESP8266 platform in the **pwm** module, which has been included in your firmware bin file.

The ON/OFF control with **button click** detection will be practiced on in later weeks together with other application. The **relay** will not be included in our exercises because it is most commonly used in the 220V environment. Please be aware of **SAFETY** when you work with relay to control the sockets or any appliances at home.

Exercise 1:

In this exercise, you will need to connect the LED to the NodeMCU/ESP8266 using a breadboard and Dupont wires. Please note that the LED is a diode, which can only be properly lighted up by current flow through one single direction.



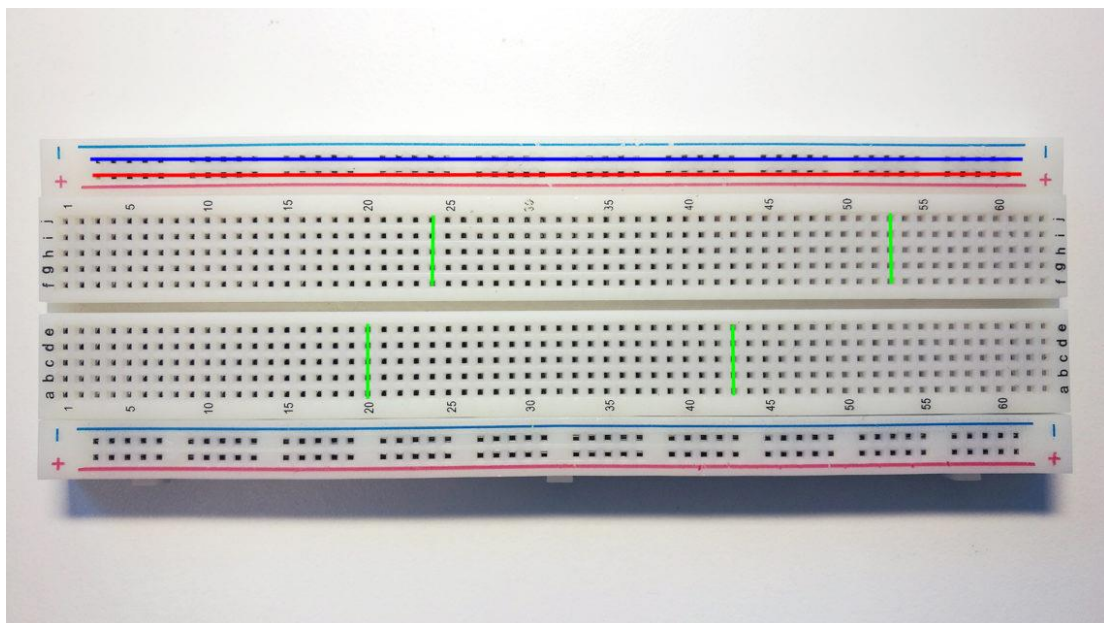
- You are given with either a 2pin LED or a 3pin LED.
 - For 2pin LED, one pin is connected to the ground pin ('GND' as shown on

your NodeMCU/ESP8266), another pin is connected to the 3V3 output pin. You might need a resistor within the loop of LED and SoC to lower the current flow to avoid breaking/burning the LED.

- For 3pin LED, the middle pin is connected to the ground pin, while you can use either one of the rest pins to connect to the '3V3' output pin. You can also use both to connect to different '3V3' output pins to use it as 2 LEDs.



- 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.



- Now light up the LED and swap the pin holes to see if it is still lighted.

Exercise 2:

In this exercise, you will need to use the **pwm** module built in the NodeMCU/ESP8266 to dim the LED.

- Now connect your LED to GND and GPIO pin 3
 - A resistor of 10K could be used in the loop (optional)
- **pwm** module is used here:

```
=====

dc=1023

pinDim = 3

pwm.setup(pinDim,1000,dc)

pwm.start(pinDim)

mytimer = tmr.create()

mytimer:alarm(200,tmr.ALARM_AUTO,function()

    dc=dc-10

    print(dc)

    pwm.setduty(pinDim,dc)

    if (dc<10) then

        dc=1023

    end

end

)
```

- ```
=====
```
- Save them as the **init.lua** file to **NodeMCU** through **Explorer**. Check the LED dimming. (Reset after every time you save files to the chip to run them automatically.)
  - For more about **pwm** module, refer to <https://nodemcu.readthedocs.io/en/master/modules/pwm/>
  - Now you can change the **rate (1000)** and the **duty cycle (dc)** to see their effect on the

PWM control.

- Please note that we no longer set the pin with GPIO commands. They are built in in the PWM.

### Exercise 3:

In this exercise, you will need to make the LED dim in a cycle of “Top-down and Bottom-up”, ie. to control the LED with `pwm` from the light to dark, then dark to light. Please note that in the previous exercise, you were doing the “Top-down” from the light to dark.

### Exercise 4:

In this exercise, you will need to combine the `PWM` and `Timer` to create a traffic lights system (`Green` light and `Red` light are ON consecutively with `Yellow` light ON between the switch as a buffer). Use 3 LEDs to represent the `Red`, `Green`, `Yellow`. Both `pwm` and `tmr` modules will be used.

- Please consider the following local function to control the intensity (ON/OFF in our case) of light.

=====

```
function trafficLight(dc_r,dc_g,dc_y)
```

```
 pwm.setduty(pinR,dc_r)
```

```
 pwm.setduty(pinG,dc_g)
```

```
 pwm.setduty(pinY,dc_y)
```

```
end
```

=====

- For example, the `Green` light is ON for 4 seconds, then OFF and `Yellow` light is ON for 0.5 seconds, then `Red` light is ON for 2 seconds, then OFF and `Yellow` light is ON for 0.5 seconds. (`Red`, `Yellow`, `Green`, `Yellow`, `Red`, ...)

### **(Optional) Exercise 5:**

In this exercise, you will need to combine the [optional exercise in Week 1](#) and [pwm](#) module to build a blinking dimming LED

- Dimming of the LED will be combined with an ON/OFF state.
- [Duty Cycle \(dc\)](#) keeps changing at its own rate (eg. 1000).
- ON/OFF follow another cycle (eg. ON for 500ms, OFF for 300ms, then ON for 500ms, OFF for 300ms, ...).