IoT exercises – Week 4

This week, we will look at ADC in NodeMCU/ESP8266 and connect it with the photoresistor (light sensitive resistor). Particularly, their combination will be useful when you want to use the illumination as a hint for control (eg. the power supply management of your mobile phone screen). The adc module will be used in this practical.

Besides, if you have problem with the button based ON/OFF control last week, a solution is given here. But you are expected to implement an improved version. (The button detection function is trivial but the concept of how to solve it is critical!)

Furthermore, if you are interested, please implement the information fusion algorithms (eg. Kalman filter) in NodeMCU/ESP8266 as an option

.

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 last week's lecture we have talked about the Analog-to-digital Converter (ADC). This week we will see how it can be used to track the voltage change and use it as the numerical condition to control our SoC or sensors.

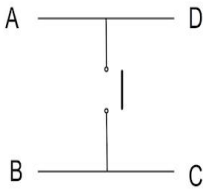
The details of adc module can be found in

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

The Kalman Filter we learned in the lecture is a classic algorithm for estimation and has been applied in multi-sensor fusion. It is optional for you to either implement the Kalman Filter by yourself or refer to well established libraries to try it.

(Optional) Exercise 0:

In this exercise, you will need to optimize the Button Push Release detection function.



- Here the example used 2 timers to detect the push and release respectively.
- Can you think of any pitfalls? How can you solve it?

==========

```
buttonPin = 7

gpio.mode(buttonPin, gpio.INPUT)

gpio.write(buttonPin, gpio.LOW)

pushed = 0

mytimerPush = tmr.create()

mytimerRelease = tmr.create()

mytimerPush:register(100, 1, function()

if gpio.read(buttonPin)==1 and pushed ==0 then

    pushed = 1

end

end)

mytimerRelease:register(100, 1, function()

if gpio.read(buttonPin)==0 and pushed == 1 then

      pushed = 0

      print("Button Push detected")

end

end)
```

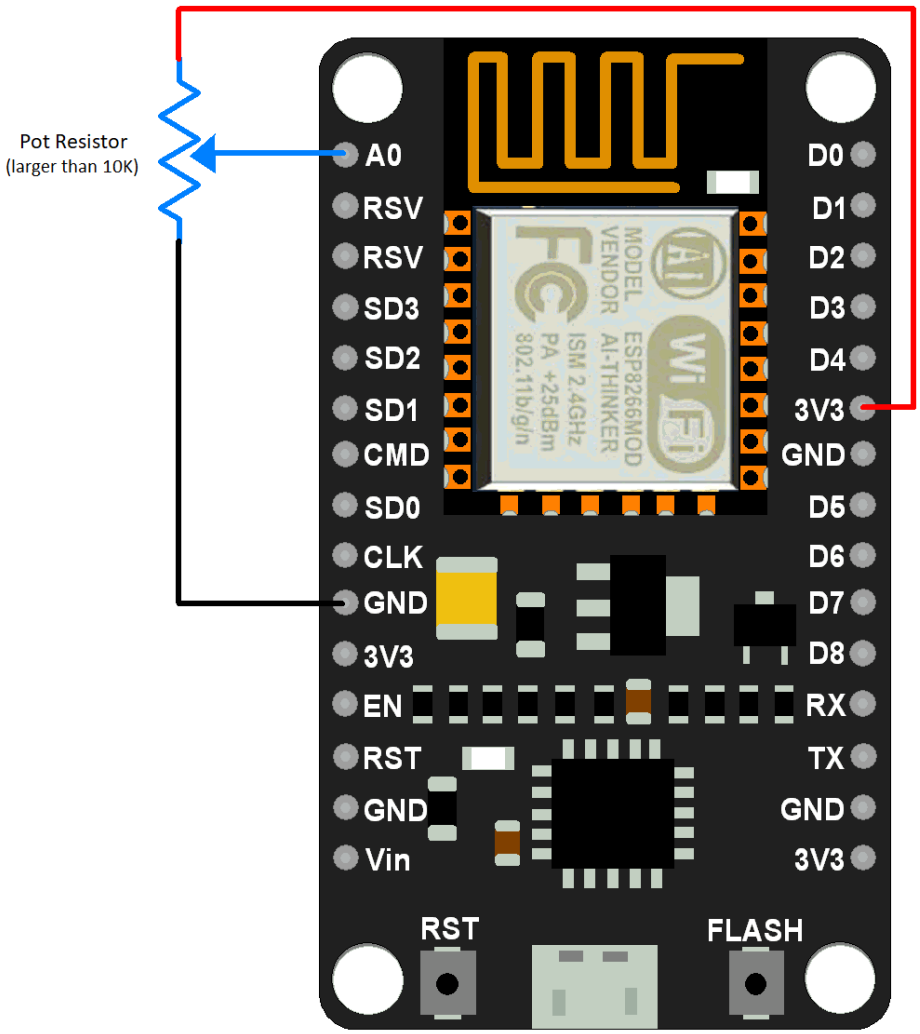
```
mytimerPush:start()

mytimerRelease:start()
```

==========

Exercise 1:

In this exercise, you will need to use the ADC to read voltage using the adc module.

- Connect the 3.3V out pin, the GND pin, the Pot Resistor (ideally larger than 10K, here we use 10K (103) or 20K (203)), and the ADC pin (A0).



==========

```
pinADC = 0

--A0 pin

mytimer:register(500, 1, function()
```

```
    digitV = adc.read(pinADC)
    --the maximum for NodeMCU ADC is 1.1v
    --it is a 10-bit ADC, can represent 0-1023
    --1023 represent 1.1v or larger
    print(digitV)
end
mytimer:start()
```

==========

- Now adjust the resistance of Pot Resistor, and see the change of ADC readings.
- Please note that the ADC of NsodeMCU/ESP8266 works for a range between 0-1.1V.

Exercise 2:

Use the ADC input as the control variable and forward it to the PWM to control the lightness of the LED.

Now you have a lantern with adjustable lightness.

Exercise 3:

Replace the pot resistor with photoresistor (light sensitive resistor, here we use a GL5537) and a fixed resistance (be careful about the total resistance, here we use a 10K resistor), check how the illumination will affect the resistance of your resistor.

Think of how you can use this in your platform for a smart LED lantern control.

Exercise 4:

- Now combine what we have practiced on PWM adjustable control, button ON/OFF control and the ADC we just played with.
- You are expected to build a local platform – a smart LED lantern.
  - With one button, the LED can be switched ON/OFF
  - With another or the same button, the brightness of the LED can be switched

between 1-5 for 5 levels.

- With the ADC and photoresistor, the brightness of the LED will adjust to the environment change. (The brighter the environment, the less brightness of the LED)

- If you don't know how to do it, please let me know or refer to previous exercises.
- Should you need any resources, please let me know it.

Now you have your own local smart LED lantern, plus the temperature/humidity meter you have created before. They will be the local IoT hardware before you connect them to the Internet.

(Optional) Exercise 5:

In this exercise, you can explore the algorithm of Kalman Filter. You can use Python, MATLAB or any other platforms to create it. If you are fine with the concept and calling it from an established library, try to build one Kalman Filter in our NodeMCU/ESP8266 platform and apply on two DHT11 sensors to output a fused result.

Let me know it should you need any other resources.