1. OBJECTIVE

Write Python code to control Raspi and light up LED. Furtherly, using PWM to adjust the brightness of LED

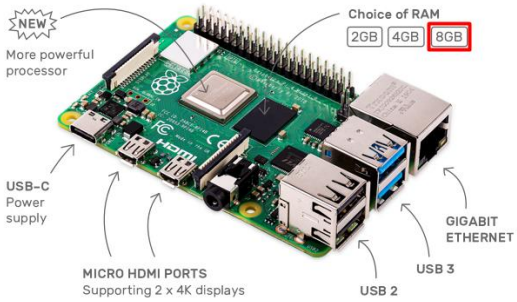
(1) Learn to control Raspi IO output port;

(2) Learn the basic Python programing skill;

(3) Use PWM to control LED.

2. COMPONENTS

(1) Raspi Board and Power;

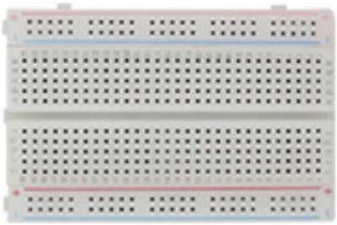


(2) A 16GB SD Card and a SD Card Reader;

(3) HDMI Screen, USB keyboard, USB mouse and HDMI Cable

(4) Basic Hardware (Extended IO Board, Wires, Bread Board)





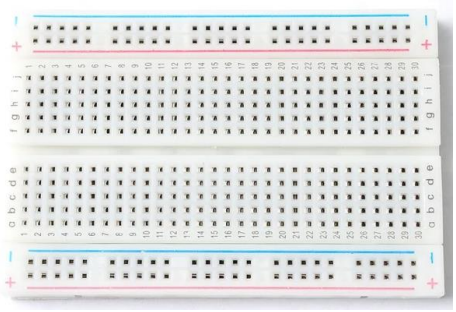
(5) Double color LED

(6) RGB-LED

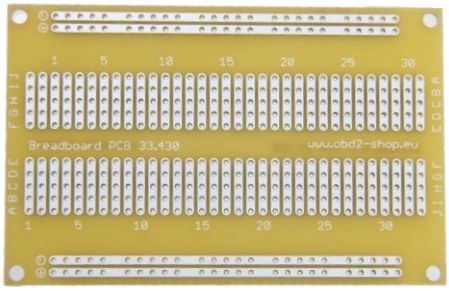
3. PRINCIPLE OF THE EXPERIMENT

3.1. Bread Board

Here is the bread board, which is a construction base for prototyping of electronics.

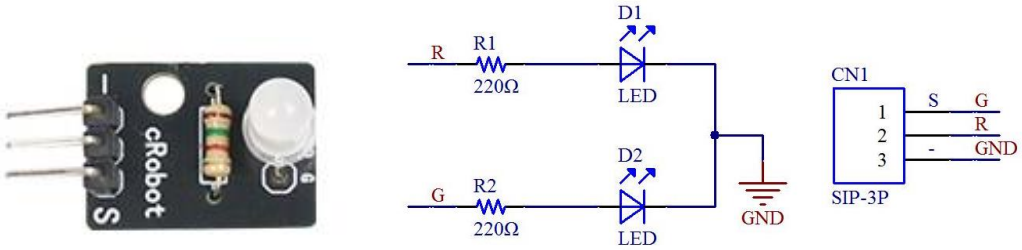


The internal connection is shown as below:



3.2. Double Color LED

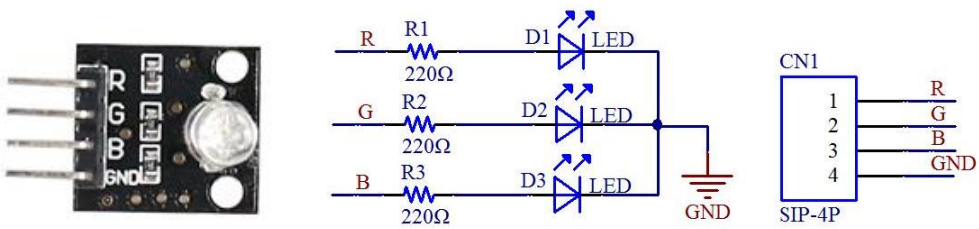
The hardware device and the corresponding schematic of double color LED is shown as below:



There are two LED luminous source, which emit red and green light, respectively. If Raspi outputs a high level to ‘R’ pin, the red LED on; If Raspi outputs a low level to ‘R’ pin, while a high level to ‘G’ pin, the green LED on. If both ‘R’ and ‘G’ pins are high level, only the red LED on.

3.3. RGB-LED

The hardware device and the corresponding schematic of RGB-LED is shown as below:

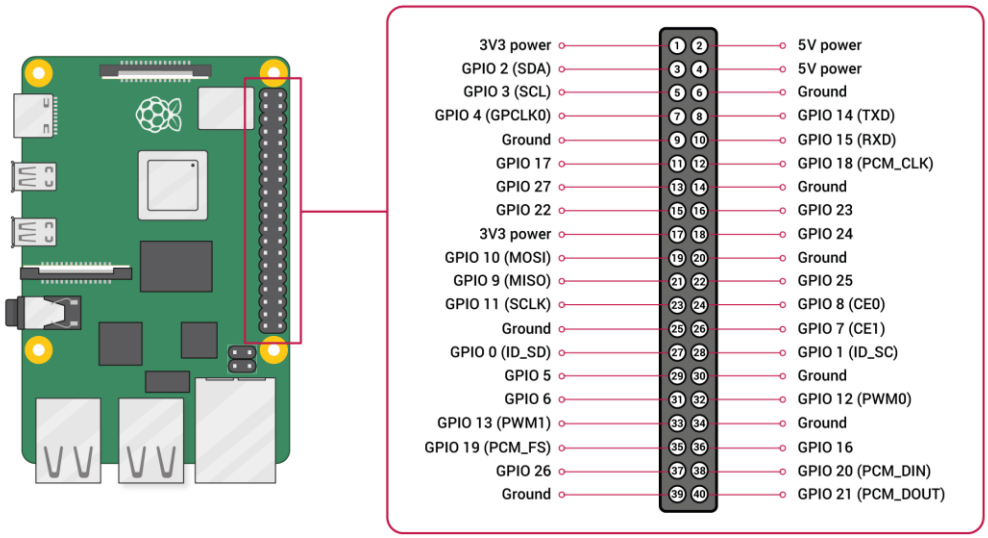


Compared with double color LED, the RGB-LED consists of three LEDs and multiple LEDs can be lit at the same time. As shown in the figure, three LEDs are controlled by the ‘R’, ‘G’, ‘B’ pins, respectively.

3.4. Pin guide

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. The GPIO is used to control LED and adjust the lightness of LED in this experiment.

There are three kinds of pins that are power (VCC), ground (GND) and GPIO. The function of each pin is shown as below:



Two 5V pins and two 3V3 (3.3V) pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all GPIO pins, which can be designated (in software) as an input or output pin and used for a wide range of purposes.

Note: The numbering of the GPIO pins is not in numerical order.

(1) Outputs Pin

A GPIO pin designated as an output pin can be set to high (3V3) or low (0V).

(2) Inputs Pin

A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and

GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, others on specific pins. For example:

- PWM (pulse-width modulation)

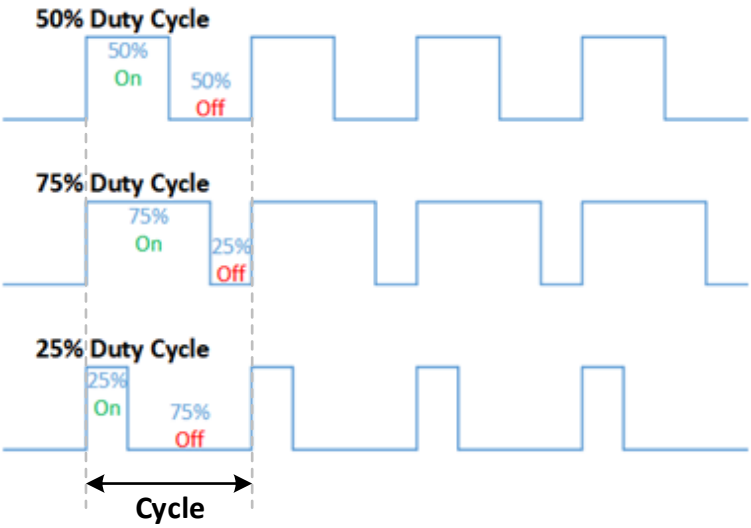
Software PWM available on all pins

Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19

3.5. PWM

The pulse-width modulation (PWM) is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load.

The term duty cycle of PWM describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on. When a digital signal is on half of the time and off the other half of the time, the digital signal has a duty cycle of 50% and resembles a "square" wave. When a digital signal spends more time in the on state than the off state, it has a duty cycle of >50%. When a digital signal spends more time in the off state than the on state, it has a duty cycle of <50%. Here is a pictorial that illustrates these three scenarios:

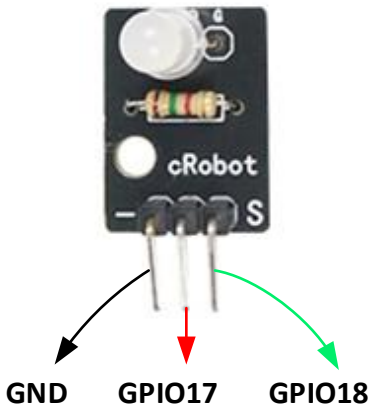


In this experiment, the PWM technology will be used to adjust the brightness of LED.

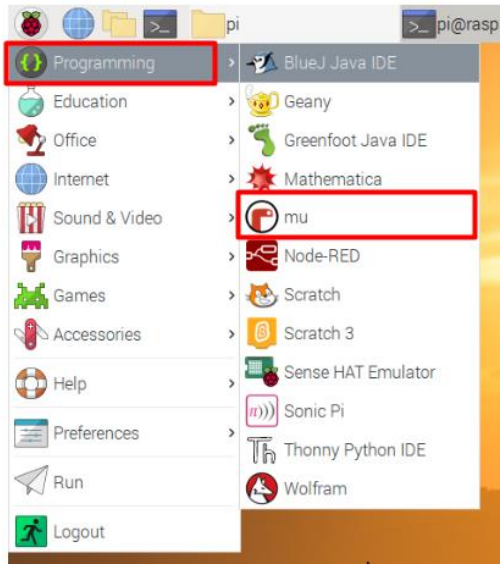
4. Experiment Steps

1) Connect Raspi and Breadboard using the 40-wires cable as below.

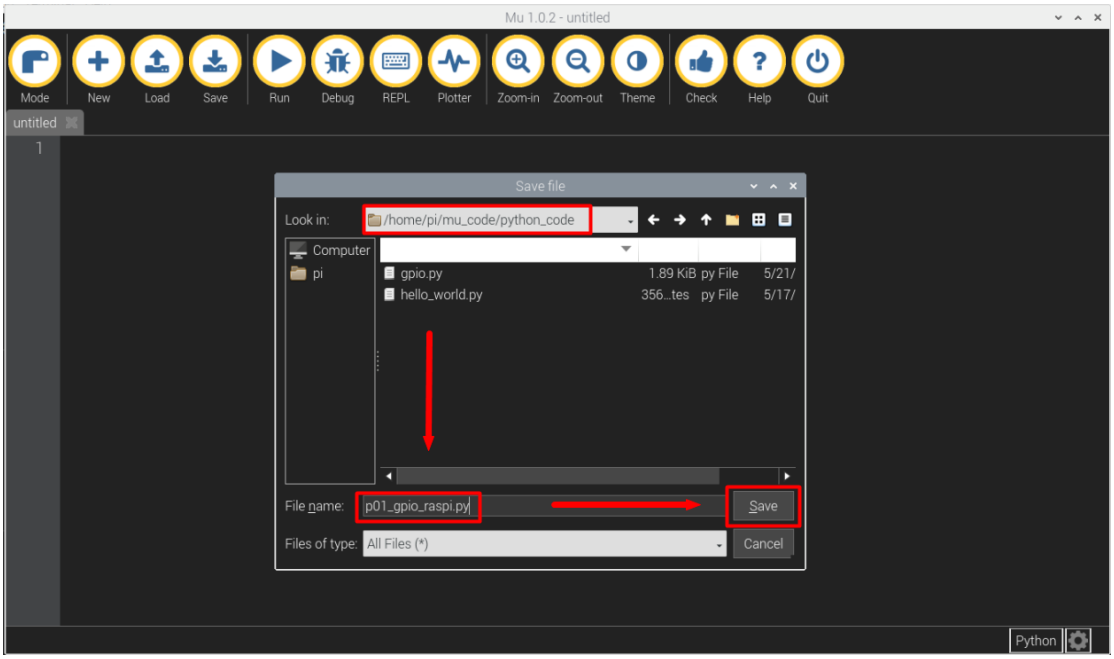
2) As shown below, connect the middle 'R' pin of the Double Color LED with GPIO 17, the 'G' pin with GPIO 18, and the left pin with GND.



3) Power on Raspi and start-up Mu from the Raspi menu: **Menu → Programming → mu**, as below



4) Click the **New** button, then before typing anything, click the **Save** button and enter the "python_code" folder, and give your file the name, "p01_gpio_raspi.py", then click **Save** button.



5) Type the following Python code into the text area to import the useful python libraries:

```
from gpiozero import *
from time import sleep
```

6) Define two variables to correlate GPIO 17 and GPIO 18.

```
gpio_red = 17   # define the gpio number controlling red LED
```

```
gpio_gre = 18   # define the gpio number controlling green LED
```

7) Use function `LED()` to initialize the red LED and green LED.

```
led_red = LED(gpio_red, active_high=True, initial_value=False)
led_gre = LED(gpio_gre, active_high=True, initial_value=False)
```

Note: *From the schematic of double colour LED shown in 3.2. , we can get that LED is turned on when GPIO output high voltage, so we set* `active_high=True`. *We set* `initial_value=False`, *as we hope the initial state of LED is off. For more information about* `LED()` *function, please visit* **this site**.

8) Type the following code to turn on and turn off LEDs.

```
led_red.on()
sleep(0.5)  # delay 0.5 second
led_red.off()
sleep(0.5)
led_gre.on()
sleep(0.5)
led_gre.off()
sleep(0.5)

led_red.close()
led_gre.close()
```

9) Enter the Debug mode by clicking the **Debug** button to test our code. And click **Step Over** button and see the changes of the "debug inspector" area on the right of the window. When `led_red.on()` and `led_red.off()` are executed, we can observe if the red LED can be turned on and off, respectively.

Note: *If LED can't be turned on, we should check if the hardware connection is well.*

10) As LED is derived by GPIO, so we can use function `DigitalOutputDevice()` to realize the same function.

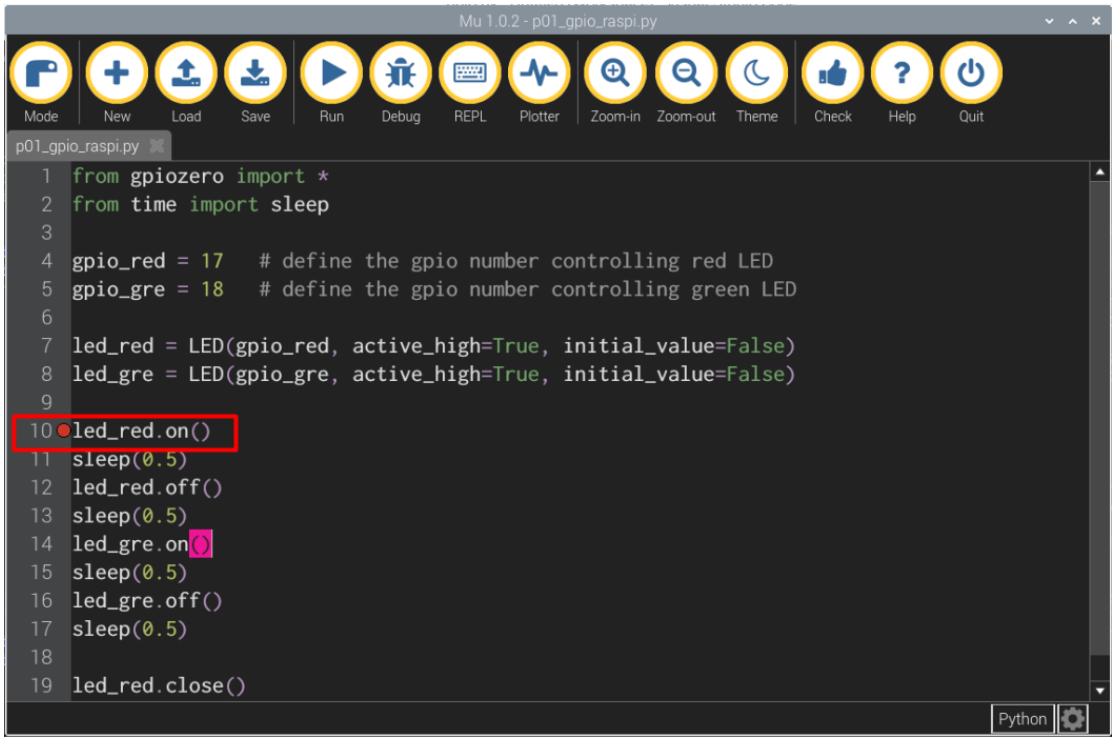
```
led_red.on()
sleep(0.5)
led_red.off()
sleep(0.5)
led_gre.on()
sleep(0.5)
led_gre.off()
```

```
sleep(0.5)

led_red.close()
led_gre.close()

# using gpio function
led_red = DigitalOutputDevice(gpio_red, active_high=True, initial_va
lue=False)
led_gre = DigitalOutputDevice(gpio_gre, active_high=True, initial_va
lue=False)
```

11) Click the column number of `led_red.on()` to insert a break print



12) Continue to click **Step Over** button until the highlighted bar moves to the last line. Every time you click the **Step Over** button, pay attention to the "debug inspector" area and the "output" area.

*Note: **Step Over** - runs the next line of code in your program; **Step In** - if the next line of code is a function, it will 'step into' the function and run it; **Step Out** - if the program is currently running a function, it will 'step out' of the function and return to the line of code that called the function.*

Note: As not all problems with code are syntax errors (which Mu usually recognises). Some errors in your code will be bugs, meaning your program runs fine, but it doesn't do what you want it to do. So the Debug function is usually used to find out bugs in the

program.

13) Click **Stop** to stop debug.

14) Click **Quit** to quit Mu

*Note: For more documents about Raspi, please click **here**; For more knowledge about Python language, please click **here**; For more tutorial about Mu IDE, please click **here**.*

End of Tutorial 1