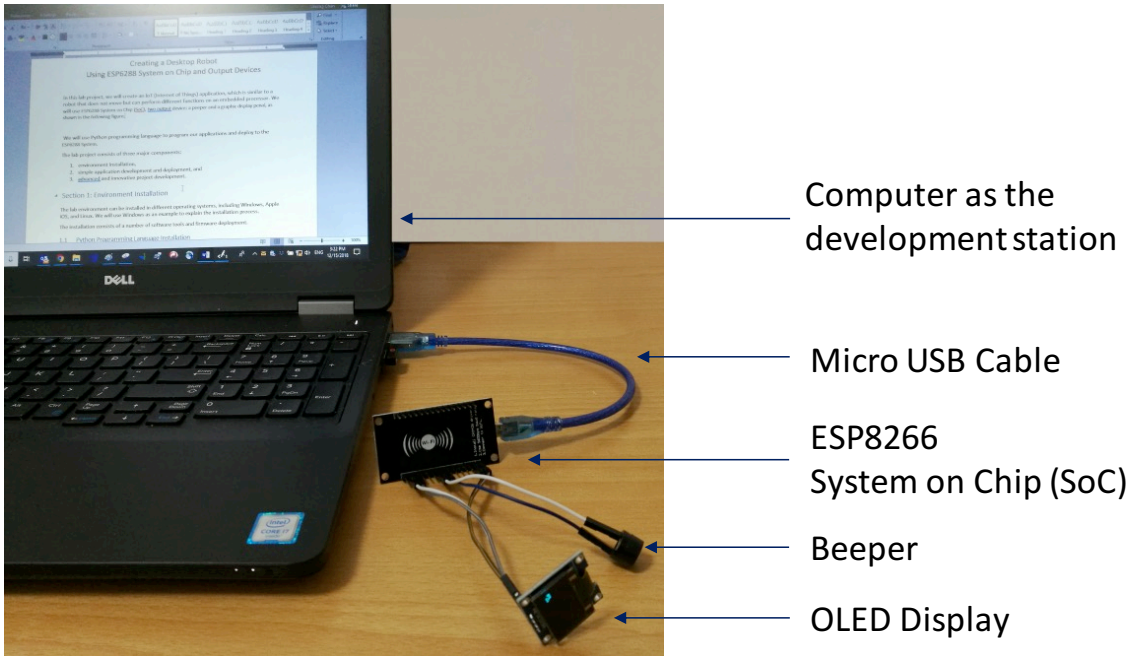
Creating an Innovative Robot/IoT Application Using ESP6288 System on Chip and Output Devices

In this lab project, you will create a Robot/IoT (Internet of Things) application, which is similar to a robot that does not move but can perform different functions on an embedded processor. You will use ESP6288 System on Chip (SoC), two output devices: a beeper and a graphic display penal, as shown in the following figure.



You will use Python programming language to program our applications and deploy to the ESP6288 System.

The lab project consists of three major components:

1. Environment installation,
2. Simple application development and deployment, and
3. Innovative project development.

Table of Contents

Section 1: Environment Installation

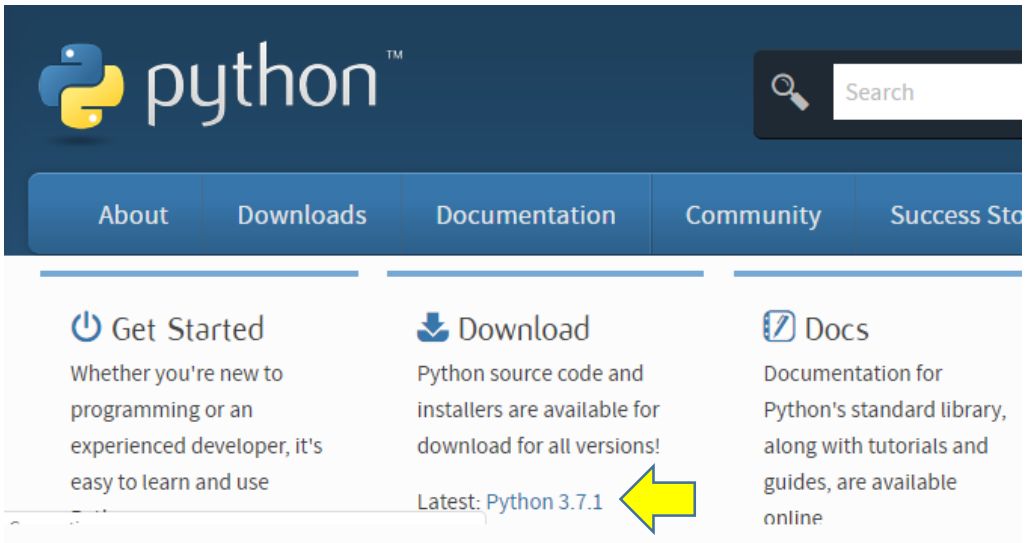
The lab environment can be installed in different operating systems, including Windows, Apple iOS, and Linux. We will use Windows as an example to explain the installation process.

The installation consists of a number of software tools and firmware deployment.

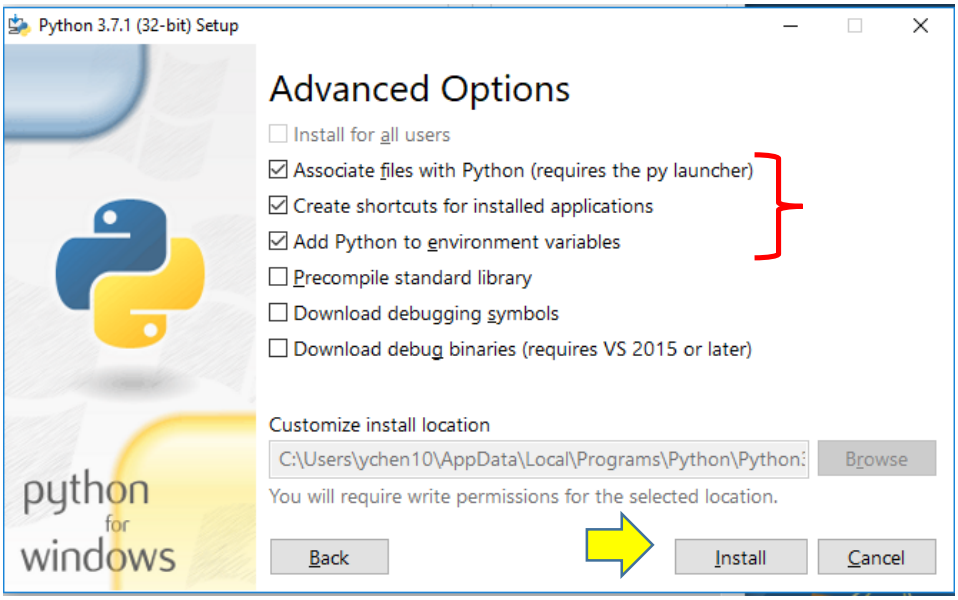
1.1 Python Programming Language Installation and Exercises

Download the latest Python environment from the official website:

www.python.org



The following options must be selected, as shown in the figure:



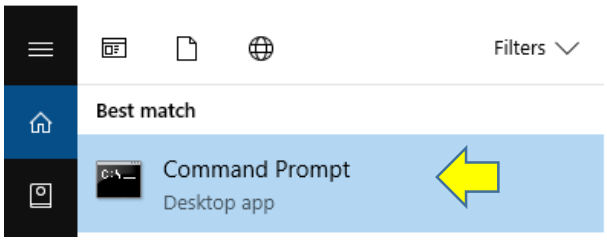
1.2 Install esptool.py Python Package

Next, you will install. There are two different ways to install the tool:

Method 1: Running CMD Command

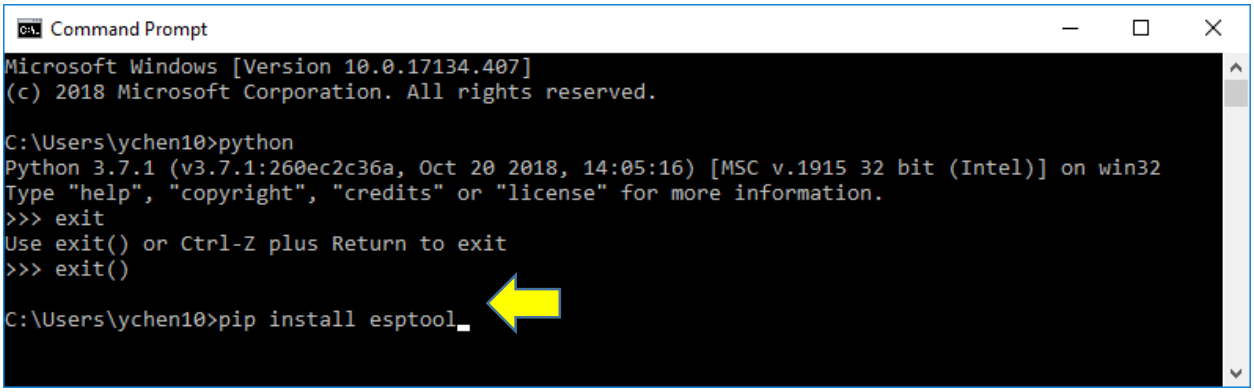
This is a better way and recommended way to install the package. You just need one command, which installs and takes care of the dependency:

Step 1: Type cmd in your Windows “Search” bar and run Command Prompt:

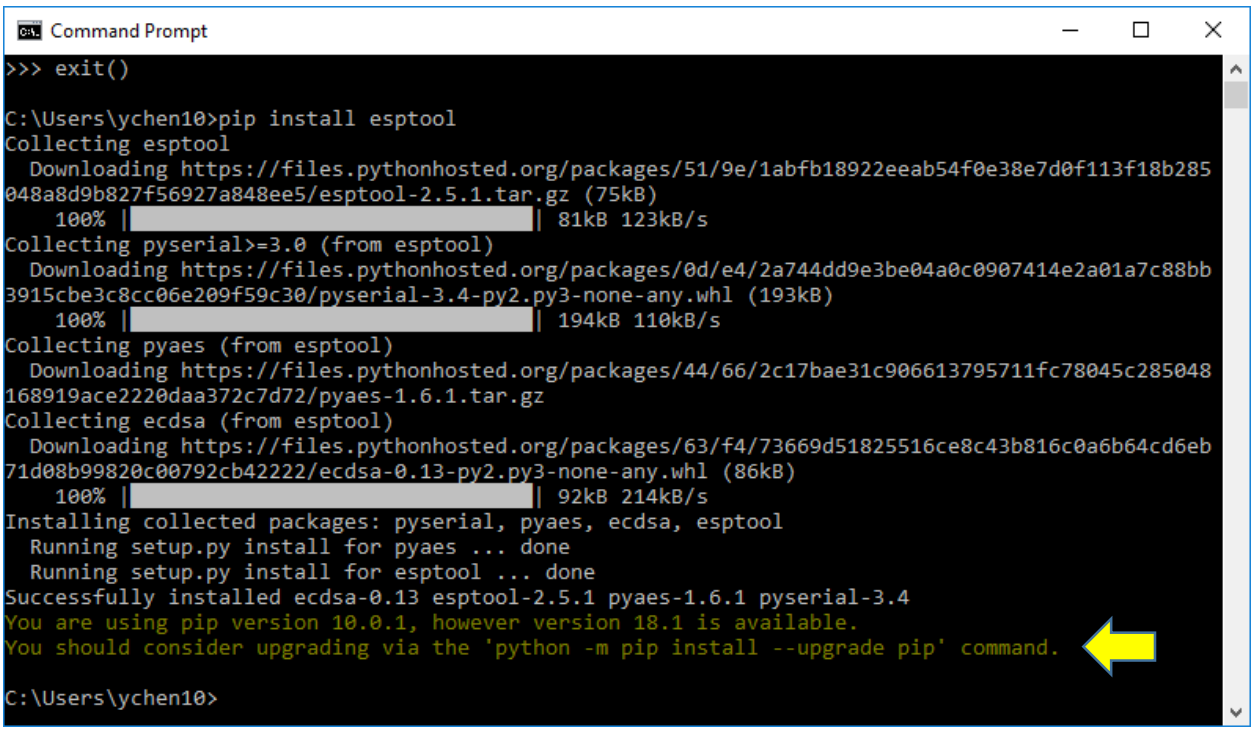


Step 1: In the command console window, type the following command:

pip install esptool



It takes a minute or two to complete the downloading process, as shown in the following screenshot.



Method 2: Website download

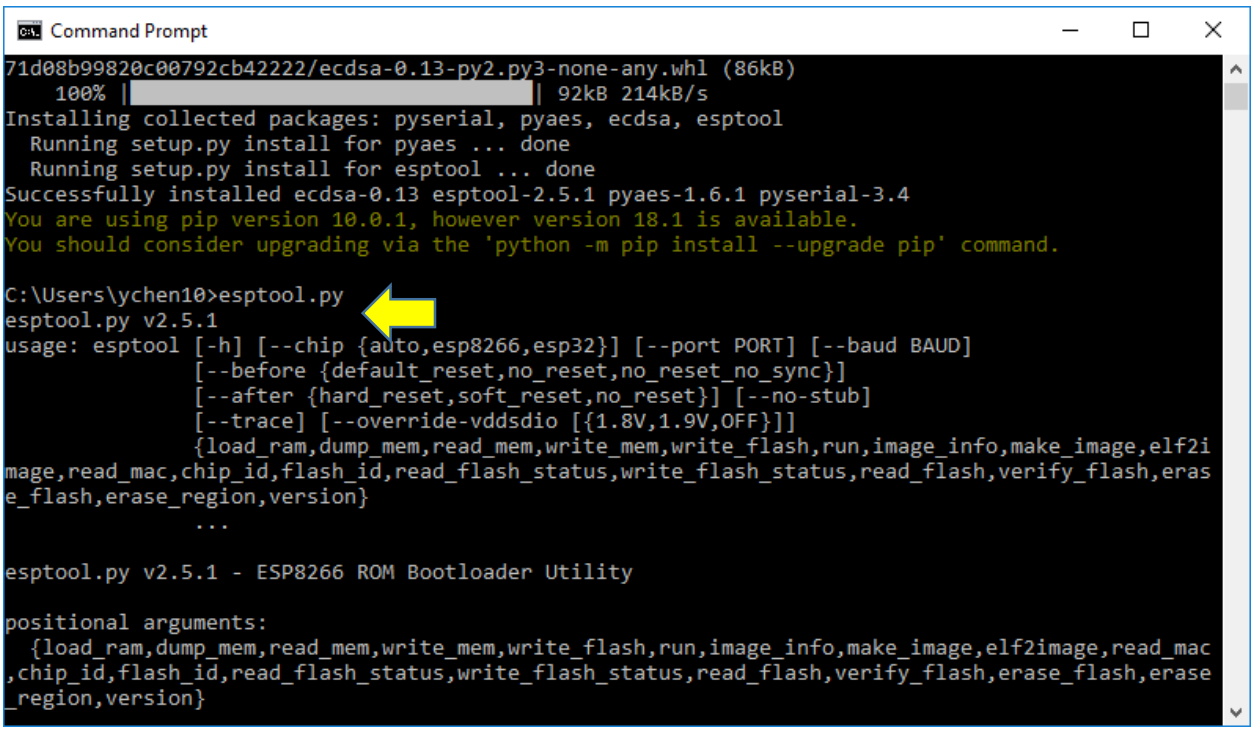
If Method 1 does not work for you, you can go to Website:
https://github.com/espressif/esptool/

Read the “Manual Installation” Section, you will download the file esptool.py. However, you need to take care of the dependency issue by also installing pySerial version 3.0 or newer for serial communication with the target device.

After have method 1 or method 2 completed, now, you can run the esptool.py program by typing the tool’s name at the prompt:

esptool.py

As can be seen in the following screenshot, the tool is running correctly.



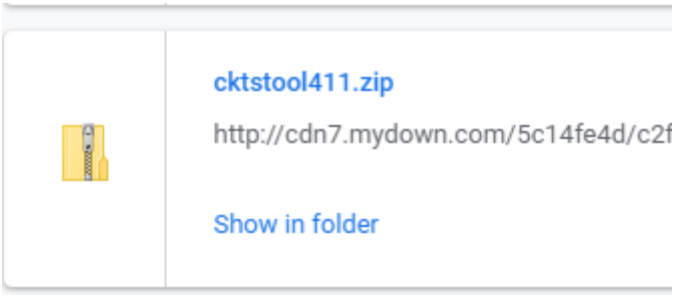
1.3 Install Serial Debugging Tool

A Serial Debugging Assistant (串行调试助手) tool is needed for the communication between your PC and the ESP8266 processor.

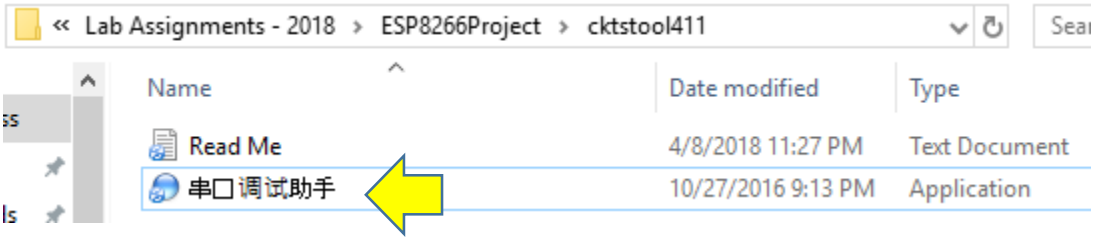
You use Baidu to search for the tool. It finds a number of tools. A simple one is sufficient, and you can choose the smallest one in size.

Click the link to enter the download page:

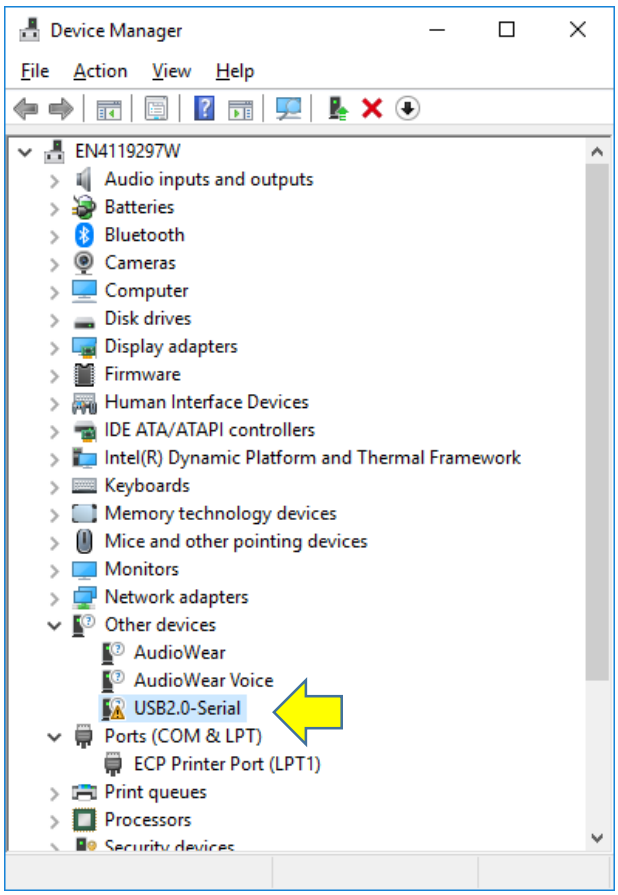
It will download a .zip file:



Save this zip file into your ESP8266 project folder and unzip the file:



After downloading completed, you can check in the PC's Device Manager to see if the Serial tool is installed correctly. Open your PC computer's Control Panel and open the Device Manager. In the list, expand the "Other Devices" folder to find the newly installed Serial tool.

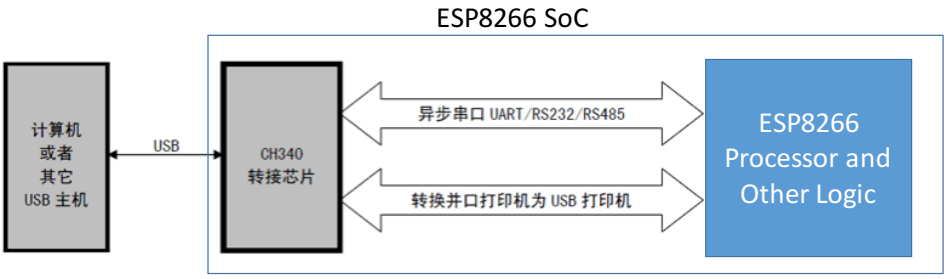


Now, your PC can communicate with any serial communication devices. But you do not see the COM port for the ESP8266 device yet.

1.4 Install Device Driver for your PC

Before installing the driver, make sure that your ESP8266 is connected to your PC via a USB cable.

Your PC can communicate with any serial communication devices using the Serial Debugging Assistant tool. You still need a device driver on your PC for ESP8266, in order for your PC to recognize the ESP8266 device. ESP8266 is a Serial Device, but USB uses a parallel communication protocol, and thus, a parallel to serial conversion is needed. ESP8266 SoC uses USB-Serial Conversion Chip CH340, as shown in the following figure.



Thus, you need download device driver for USB-Serial Chip CH340 conversion chip. The website is: http://wch.cn/products/CH340.html

The Website lists the device drivers for different operating systems.

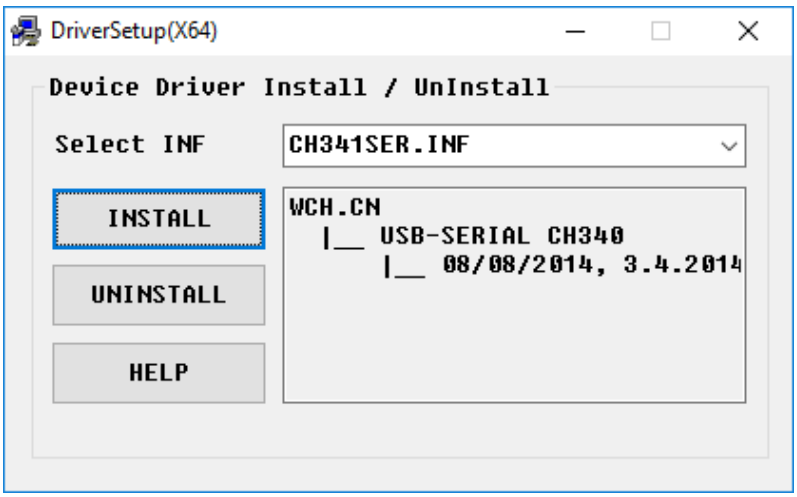
- For MacOS: you will download: /dev/cu.wchusbserial1420
- For Windows: you will download: COM port, for example, COM3

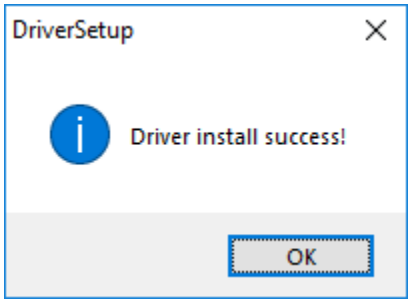
Now, you enter the website: http://wch.cn/products/CH340.html

Follow the link, you enter the download page:

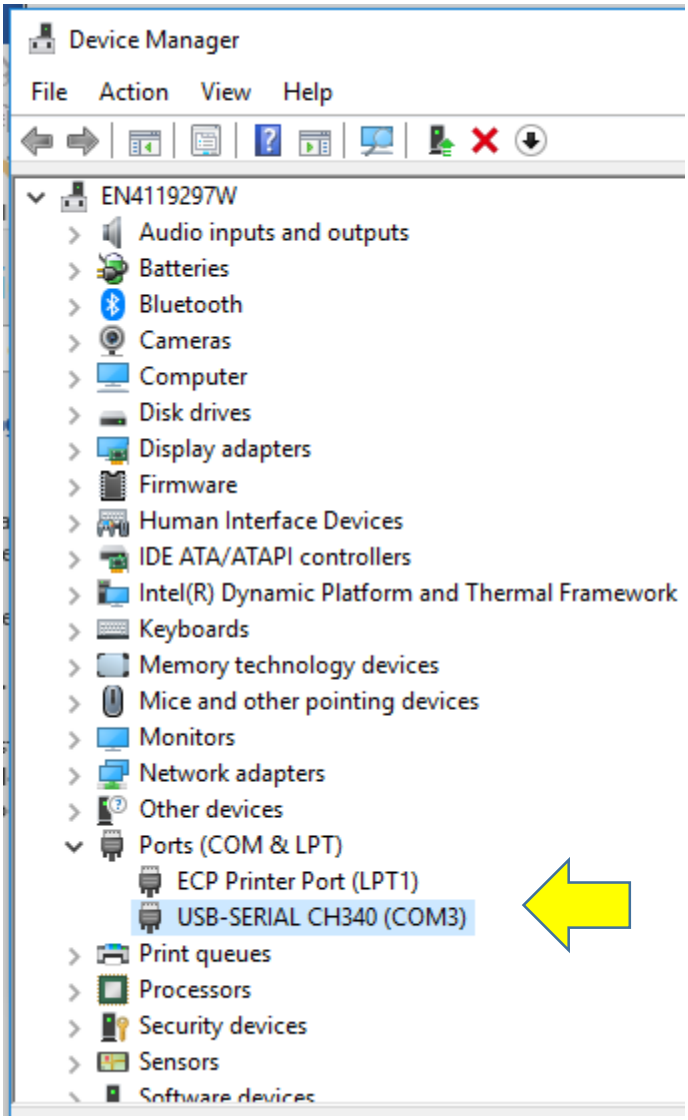
After downloading, you run the downloaded executable file:

This file will perform the installation:





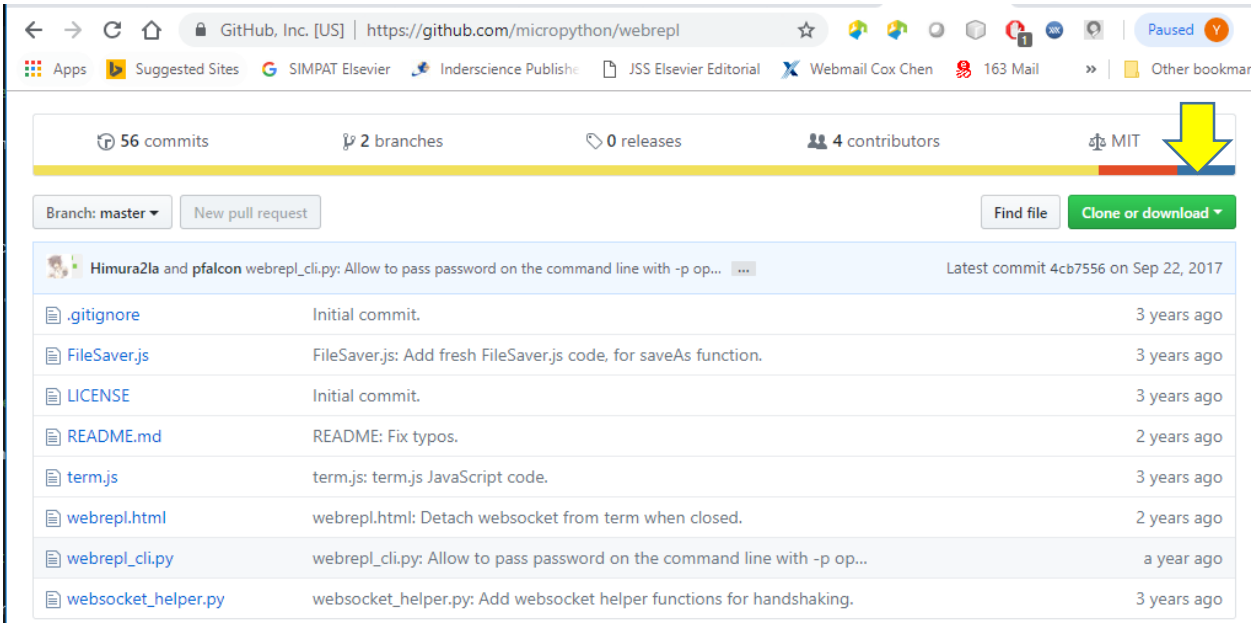
After the installation of the driver, you open the Device Manager again. Now, you see that CH340 device is installed on COM3 port:



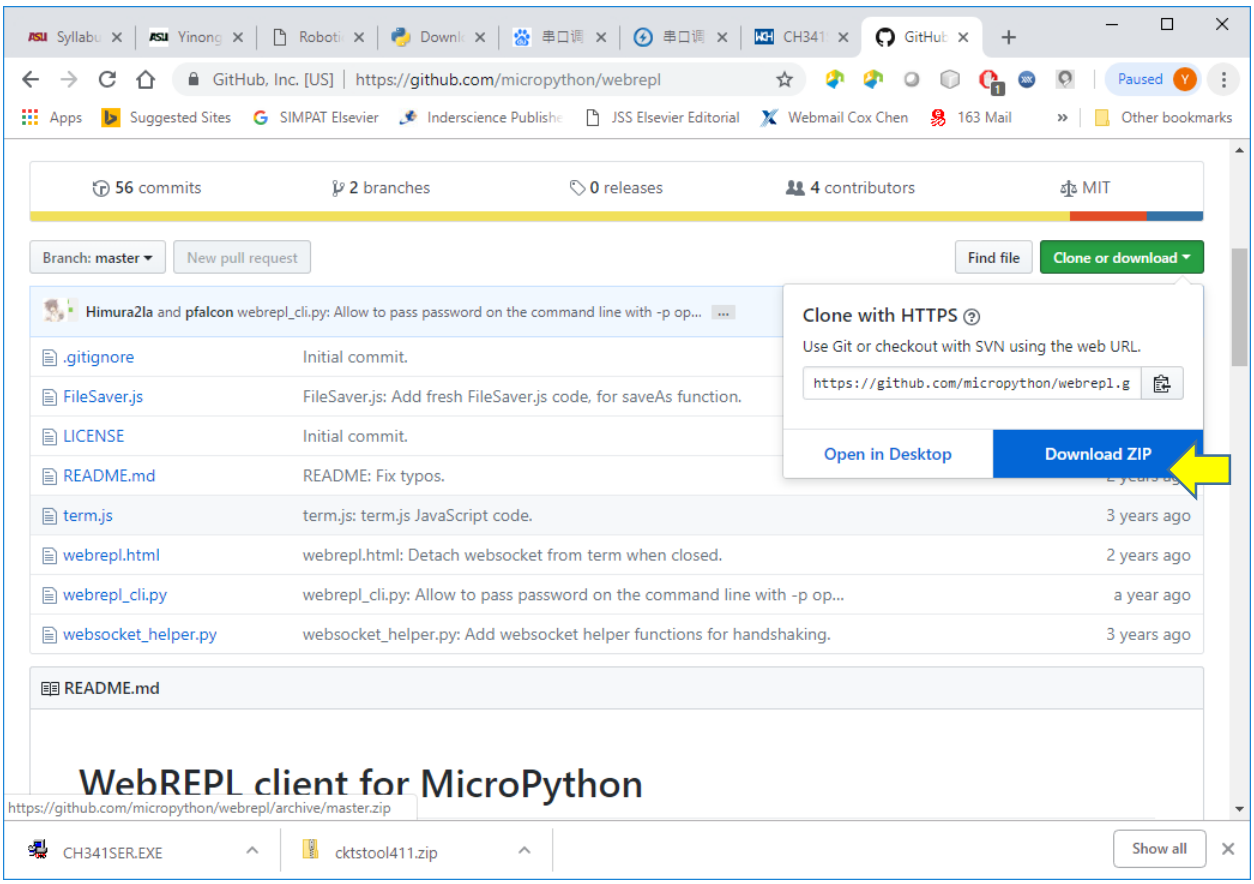
1.5 Install Webrepl Web Programming Interface for Python

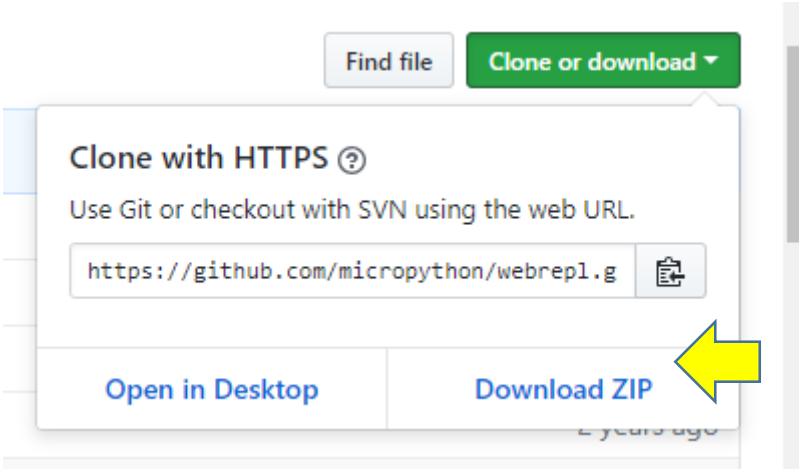
The Serial Debugging Assistant tool can send text and short Python code to the ESP8266 to execute. However, it is not suitable for typing long Python code in this tool. You need a better Python editor for this purpose. You will use Webrepl, which is a Web client for Micro Python on ESP8266. It can be downloaded at the following github address:

`https://github.com/micropython/webrepl`

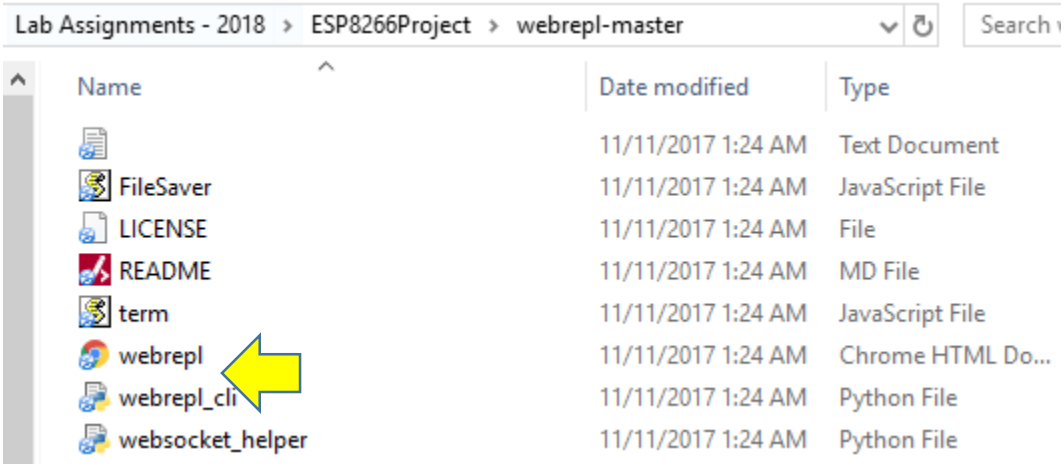


In this page, click Clone or download, and choose download Zip file.





After downloaded, move the zip file into your ESP8266 project folder and unzip the file. The unzipped files are shown in the following folder.



Click the HTML file, it will open in a web browser and you will write the Python code using this Web tool. Before you open and use this web tool, you still need to download deploy the Python environment into ESP8266 chip, in the form of Firmware.

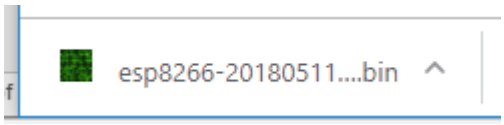
1.6 Flash ESP8266 and Install Firmware

Before you flash and install firmware to ESP8266, make sure that your ESP8266 is connected to your PC via a USB cable.

You can download the firmware in the following web site:

`http://micropython.org/resources/firmware/esp8266-20180511-v1.9.4.bin`

It will download a bin file:



Move this file into a directory on your local disk, for example: **C:\Users\ychen10**

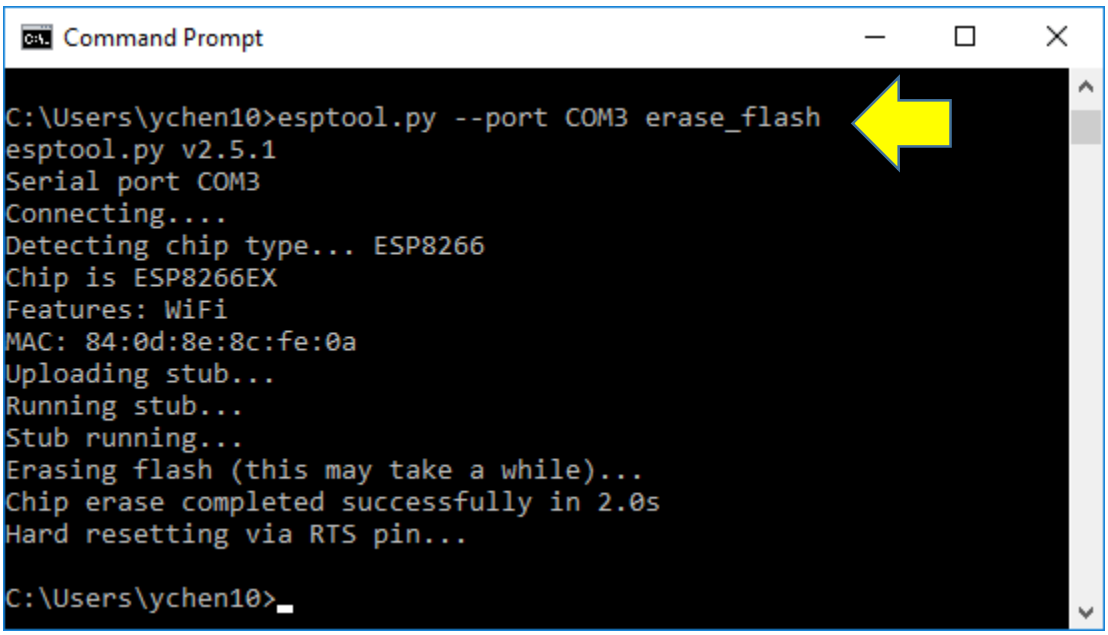
Then, you go back to the Command Prompt. Use the following command in CMD console:

cd C:\Users\ychen10

This command will change the directory to the folder, where your firmware bin file is copied into. The, you use the following command erase and flash the ESP8266 memory.

esptool.py --port COM3 erase_flash

The command is shown in the following screenshot.



After the memory of the ESP8266 is cleared, you can copy new firmware into the ESP 8266 Chip. You use the following long command:

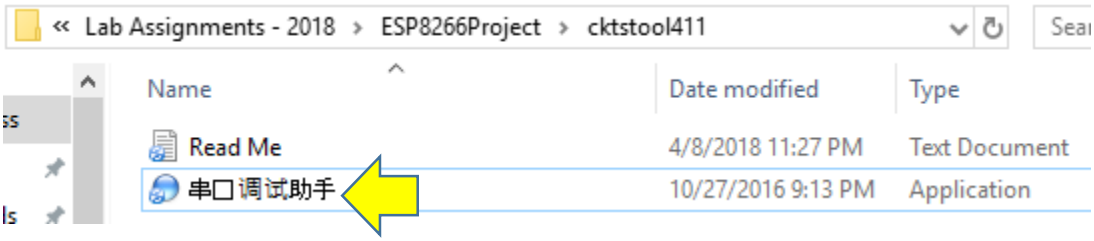
```
esptool.py --port COM3 --baud 460800 write_flash
--flash_size=detect -fm dio 0 esp8266-20180511-v1.9.4.bin
```

Note, the command is long. If you copy and paste, you must copy the first line and then append the second line to the end of the first line. You cannot copy two lines together, as it may add a new line between the two lines.

The command is shown in the following screenshot.

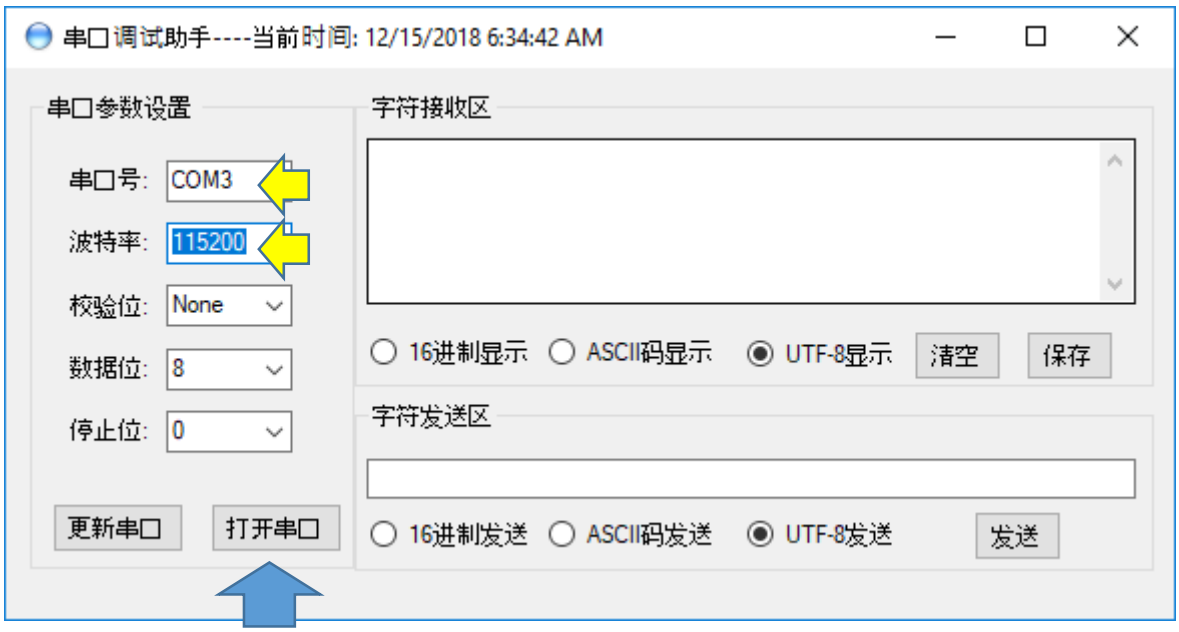
Before you can use the new firmware, you need to reboot the device: Unplug the USB device from your computer, and then, plug it in again.

Now, you go back the Serial Debugging Assistant tool folder:

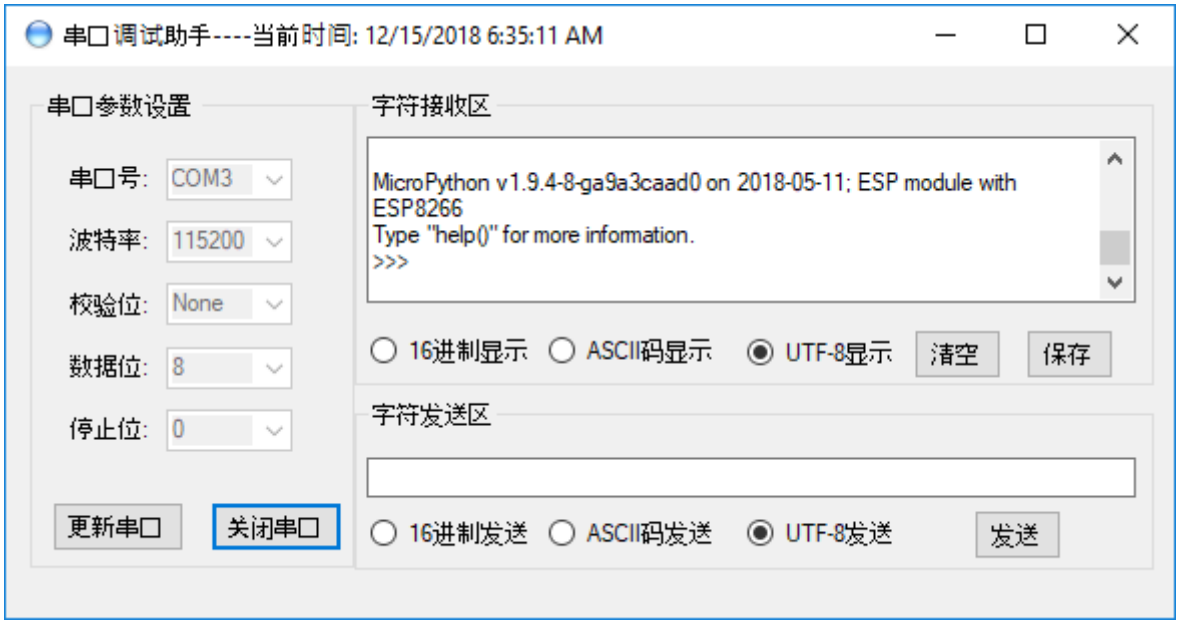


Open the Serial Debugging Assistant tool. The following window is open.

You need to choose Port number to be COM3 and the baud rate to be 115200, and then click “Open the Serial Port”, as shown in the following figure.



Now, the port is open:

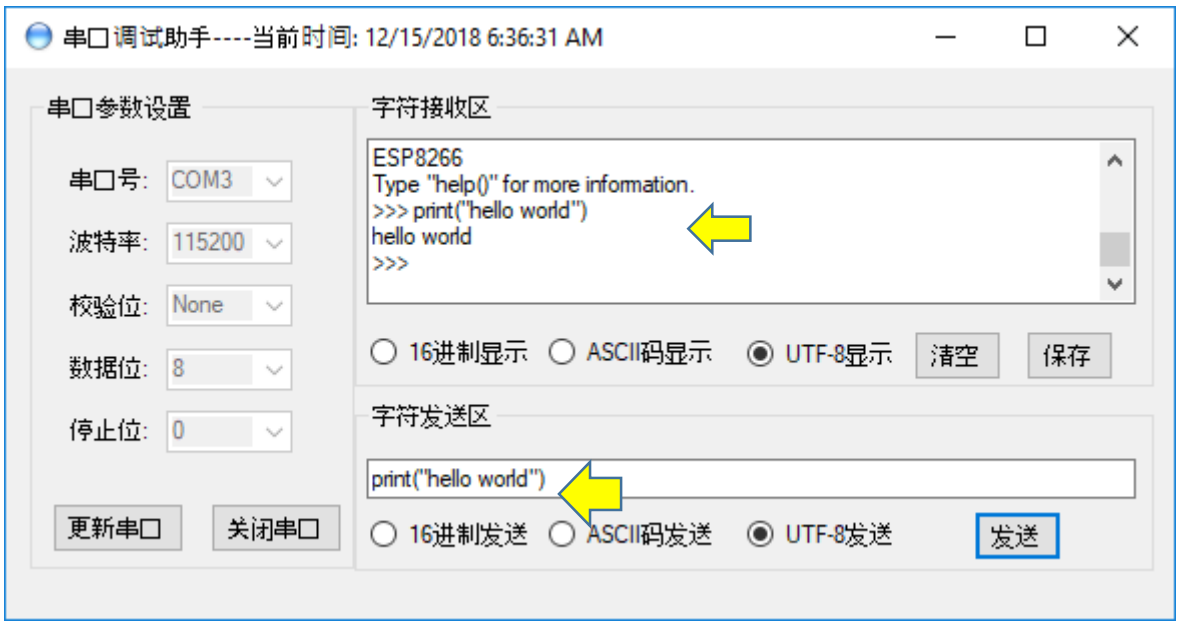


Now, you can enter text and Python code in the Character Sending Area, which will be sent to the ESP8266 processor.

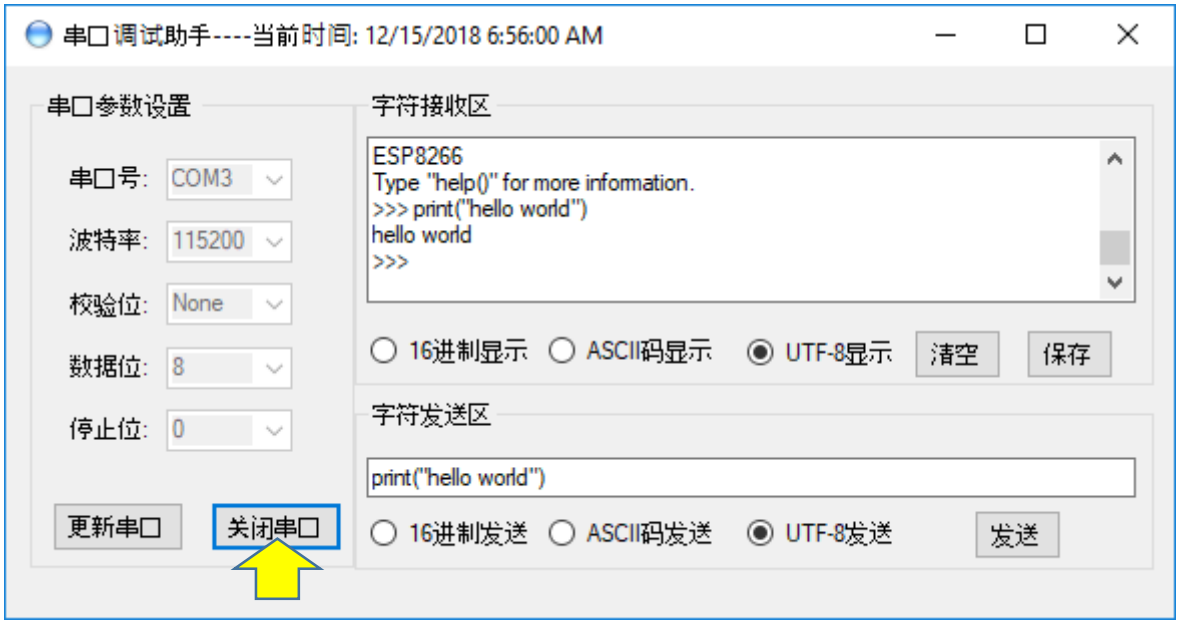
For example, if you enter Python code:

Print(“hello world”)

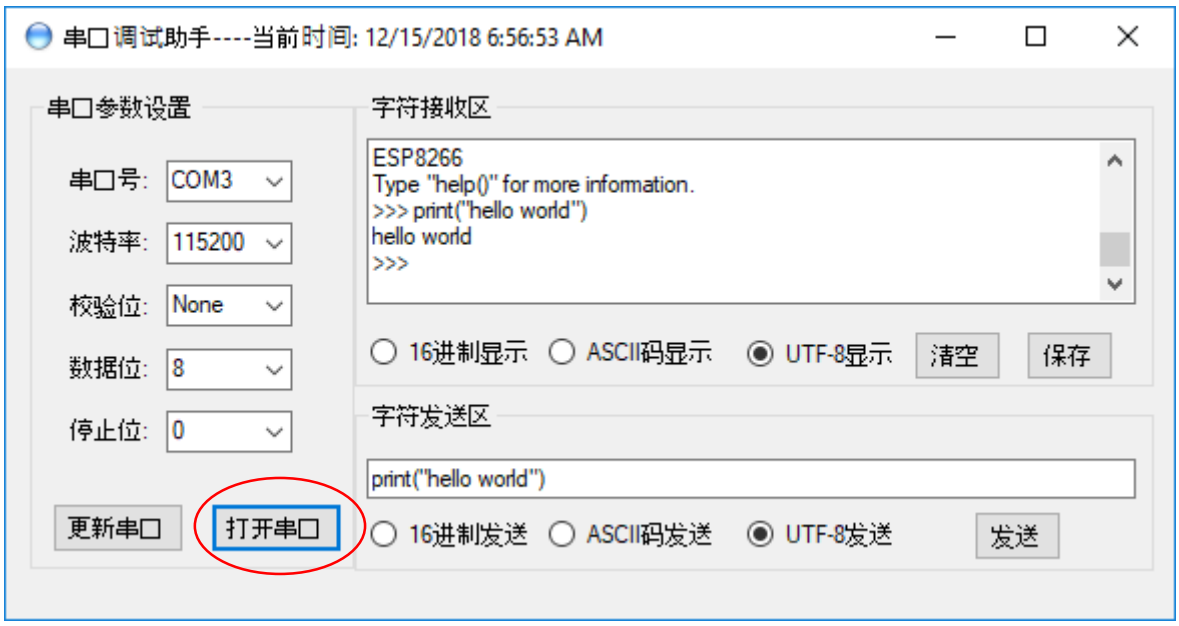
Then, click “Send”, the Python code will sent to the ESP8266 processor and be executed. The result is shown Character Receiving Area.



After using the tool, always click: Close the Serial Port:



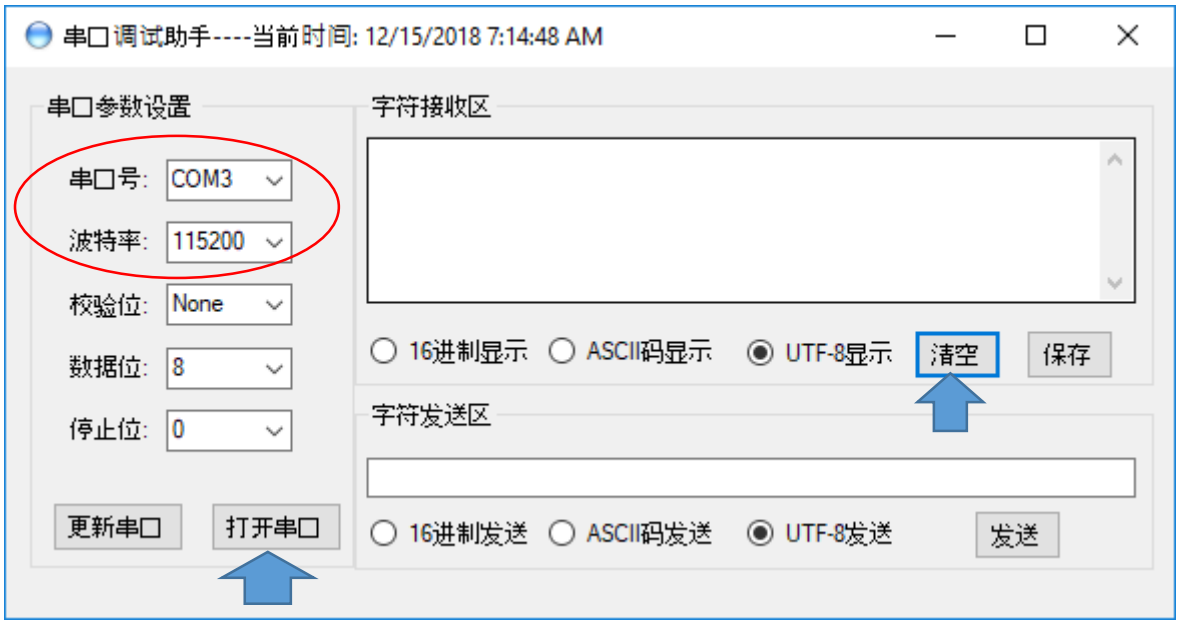
After clicking Close the Serial Port, the window looks like the following:



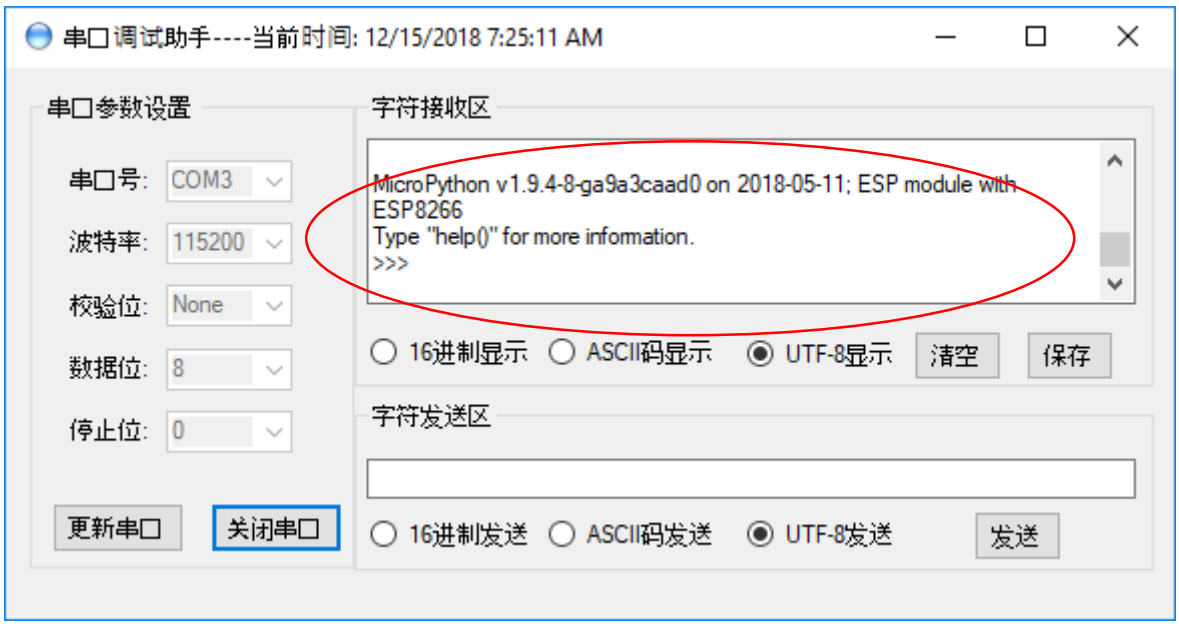
1.7 Setup Webrepl

Make sure that you have connected you ESP8266 to your PC using a USB cable.

Run serial Debugging Assistant tool and choose the correct port number and baud rate. Open the Serial Port.



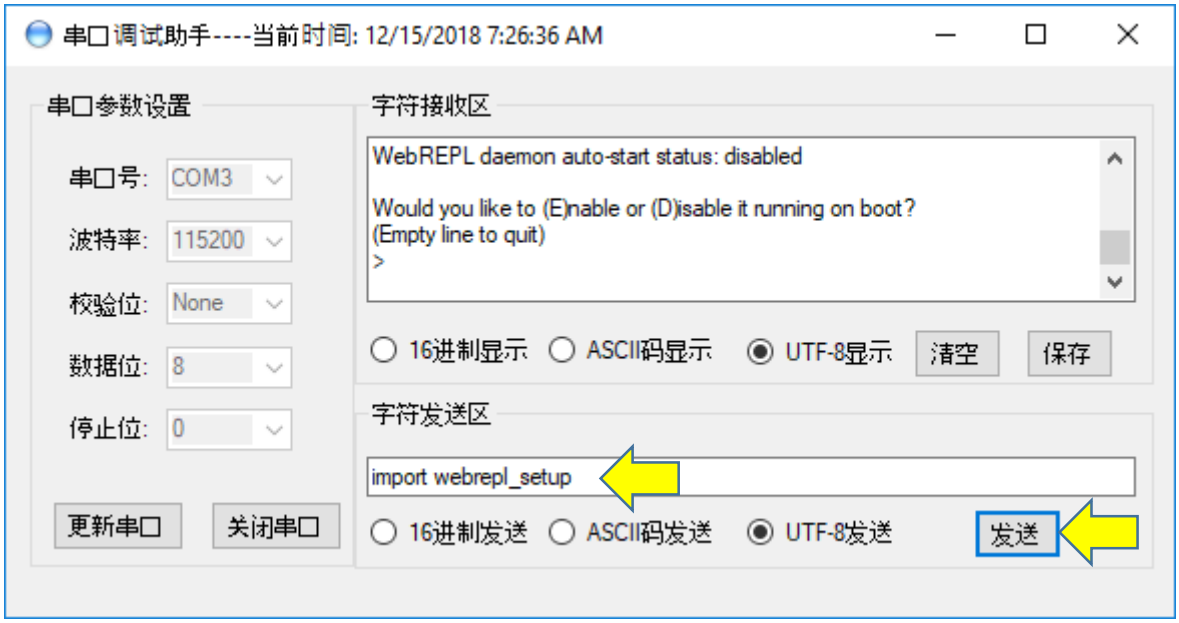
The following window shows that the port is open.



Enter the following command in the Character Sending Area:

```
import webrepl_setup
```

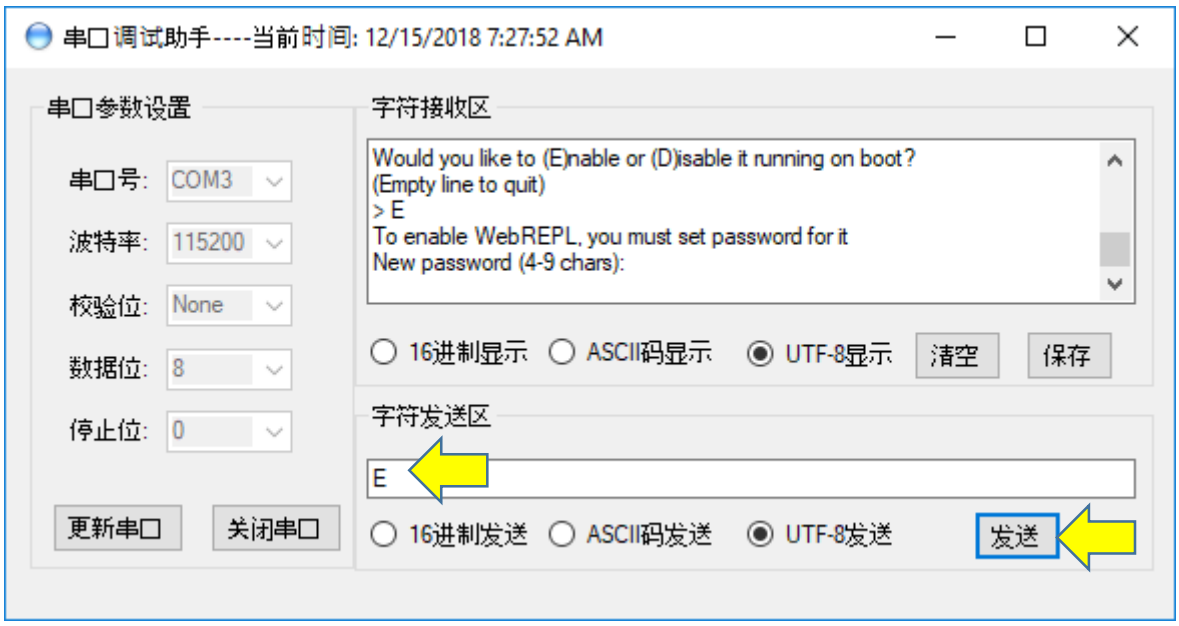
as shown in the following diagram:



The Character Receiving Area shows the options: E or D. Enter E for enable:

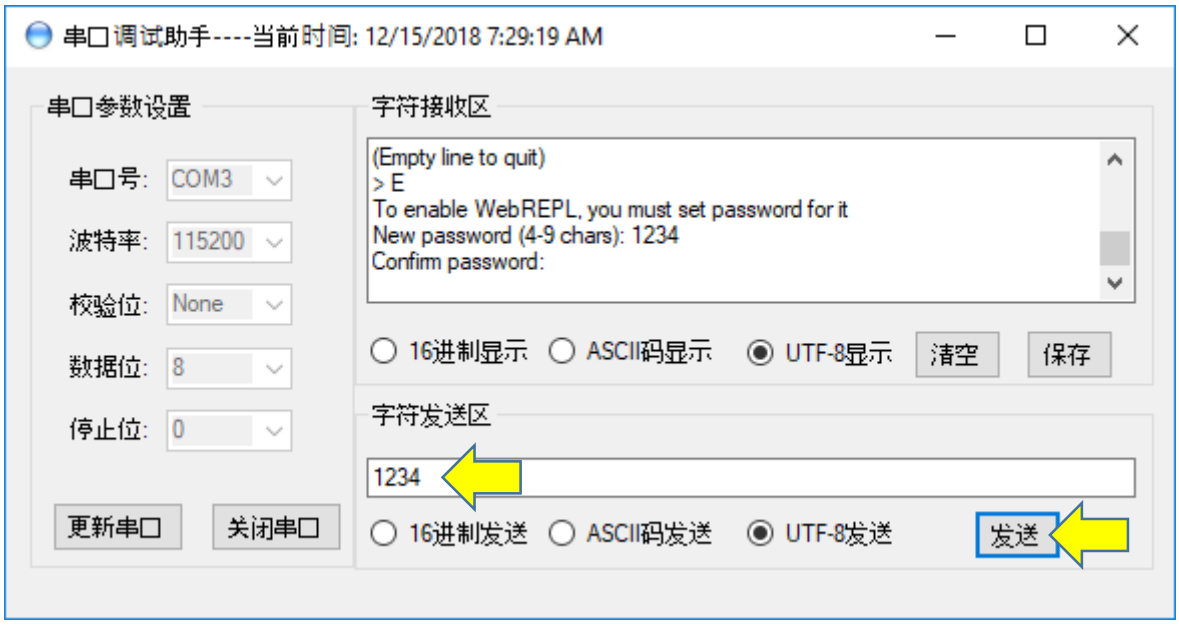
```
E
```

as shown in the following diagram:

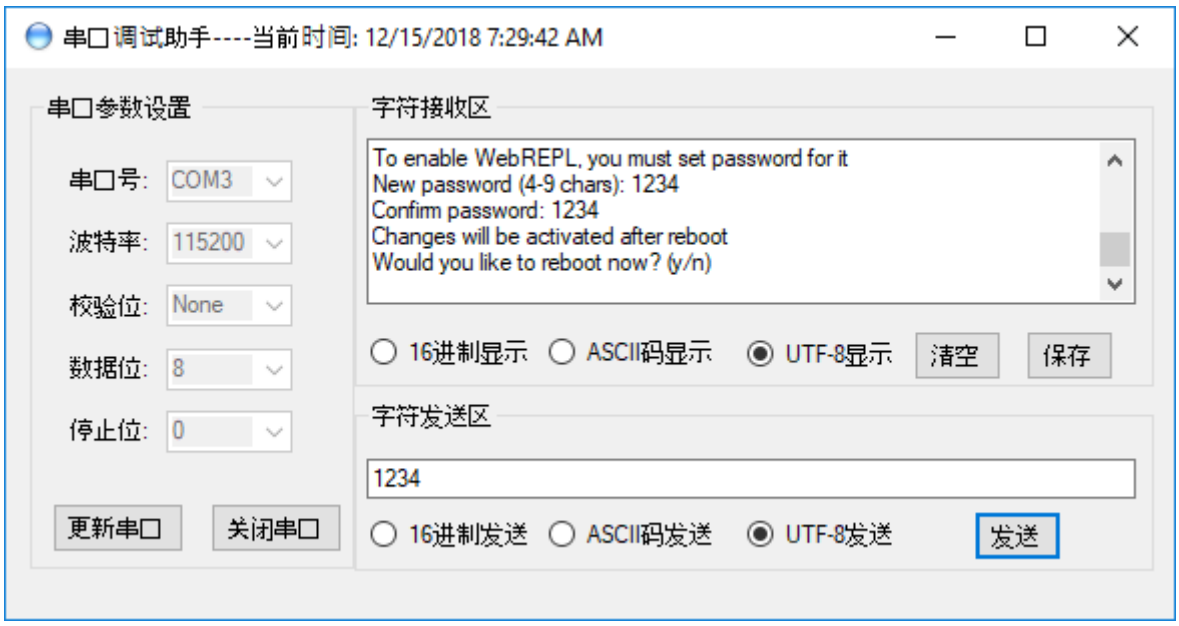


The Character Receiving Area shows to create new password between 4 and 9 characters.

You choose a simple password 1234. You must choose your own password:

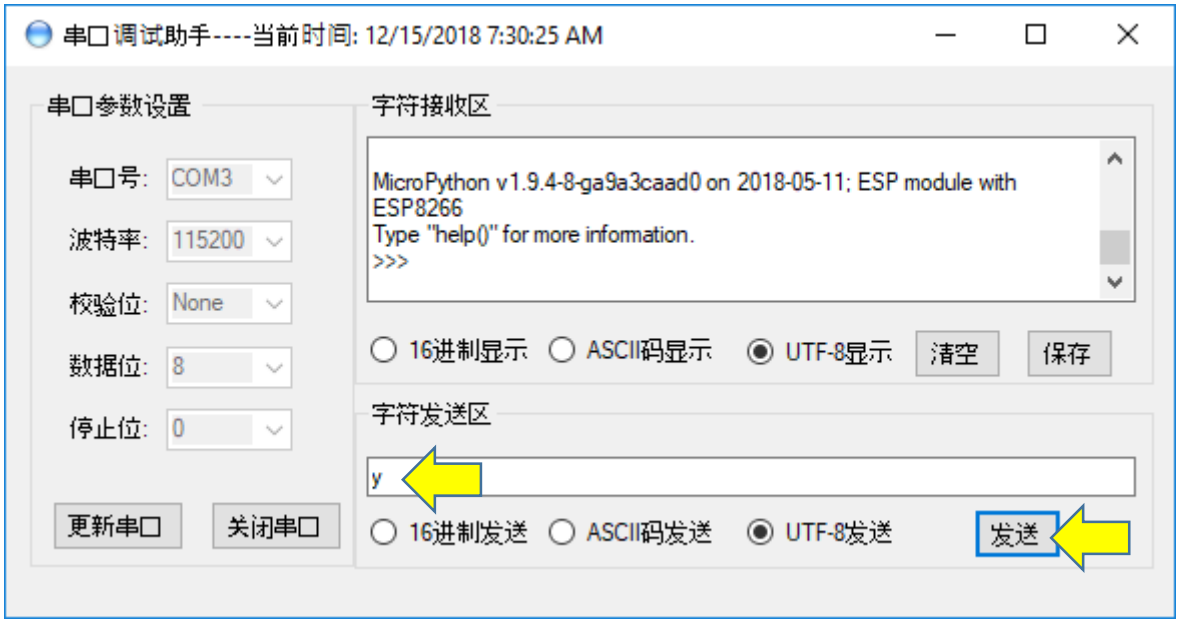


The system will ask you to re-enter the same password: Enter 1234 again:



The system ask you if you want to reboot the system (y/n)? Yes, you must reboot the system by enter:

```
y
```

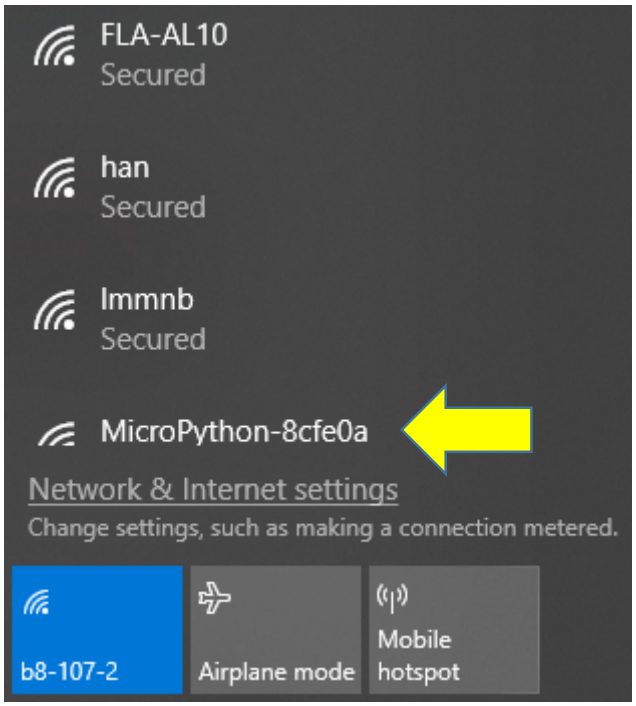


1.8 Setup WiFi

Webrepl is a web client. It will connect to ESP8266 through WiFi. You use USB cable for installing and setting up the system in ESP8266. Once you have completed the installation and setup, you no longer need the SUB cable. Instead, you use WiFi. You need keep the USB cable plugged in for the power supply.

After installation, setting up, and boot, the device configures itself as a WiFi Access Point (AP) which you can connect to. The ESSID is of the form `MicroPython-xxxxxx` where the x's will be replaced with part of the MAC address of your device. The number will be the same every time you connect, but it will be different from all ESP8266 chips. The password for the WiFi is micropythoN. Note the upper-case N at the end. Its IP address will be 192.168.4.1 once you connect to its network.

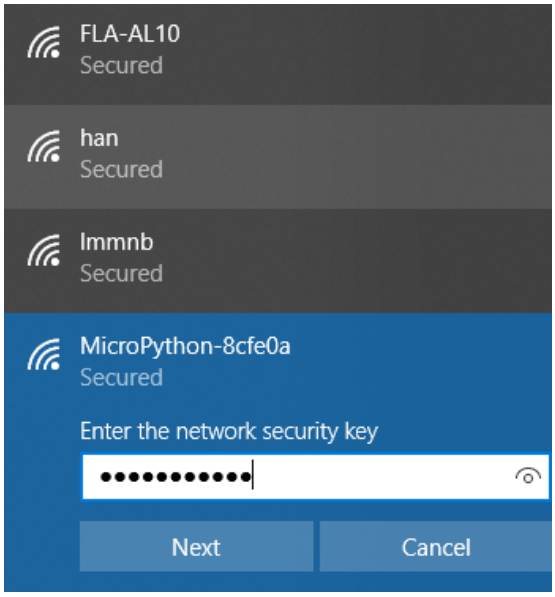
Now, you change your PC's WiFi connection the ESP8266



To find your WiFi Access Point (AP), you can use the following Python code to print the AP address:

```
import network
import ubinascii
mac = ubinascii.hexlify(network.WLAN().config('mac'),':').decode()
print (mac)
```

Click the WiFI Access Point, and it asks for the password:



The password is: micropythoN

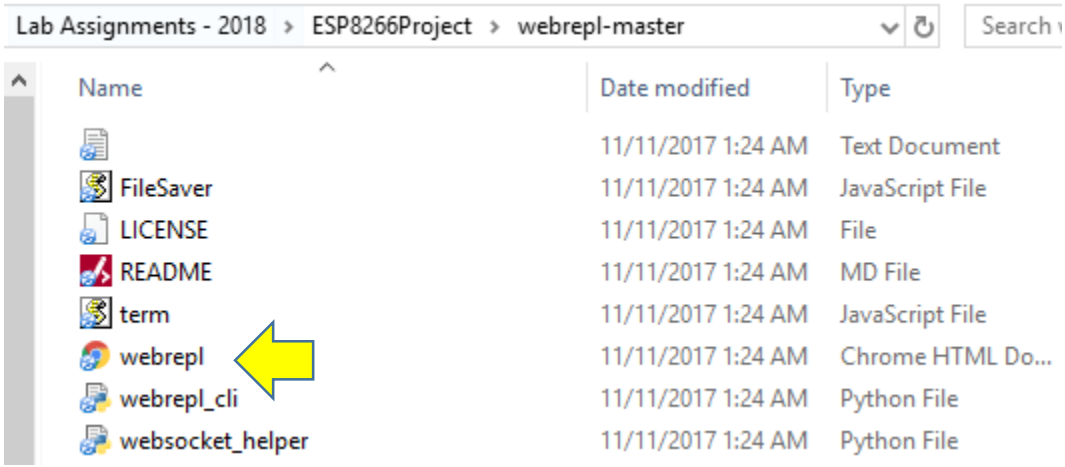
Now, you can start to use Webrepl to write Python program and send to ESP8266 for execution.

Section 2: Getting Started Running Python Programs on ESP8266

In this section, you will send python program from Webrepl to ESP8266. You can type program in Webrepl, or upload Python code file into Webrepl and send to ESP8266.

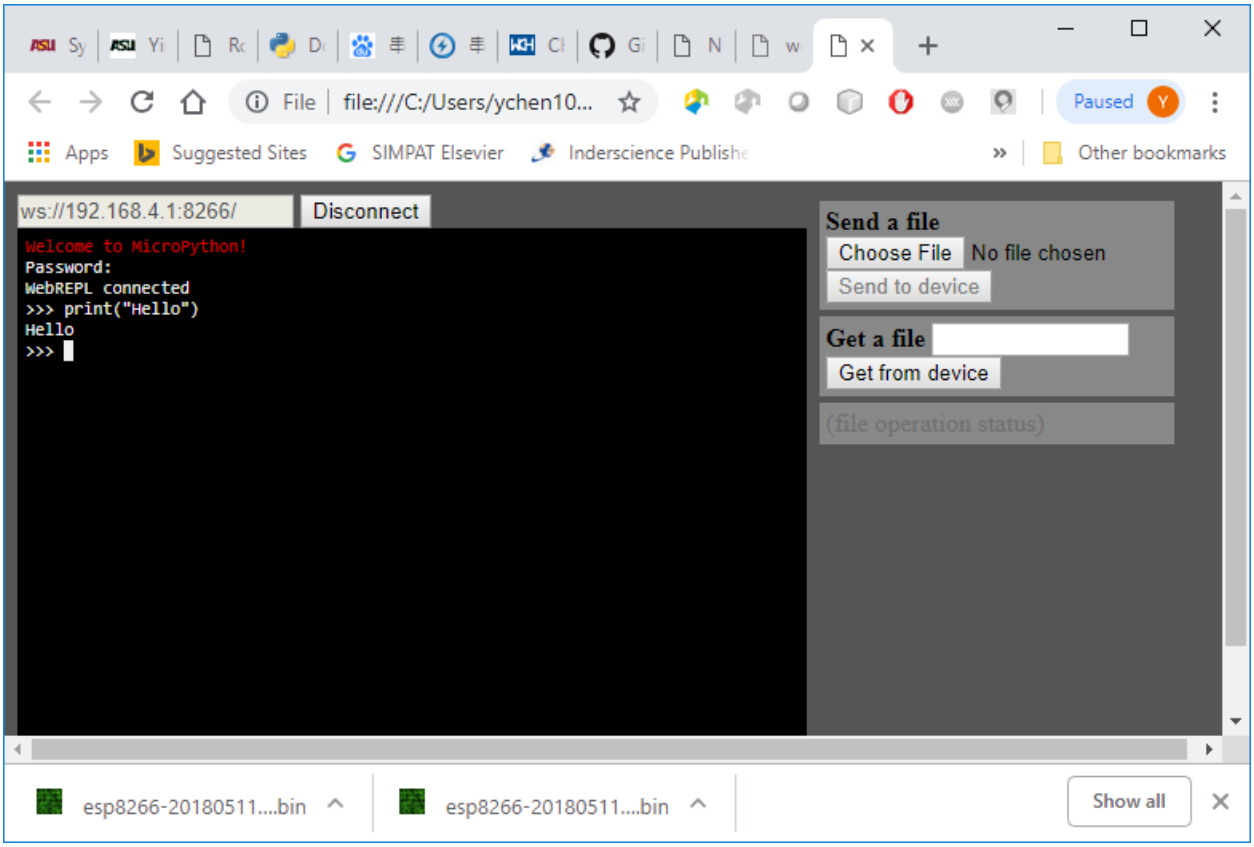
2.1 Writing Python program in Webrepl

Now, go back the folder where you have created in Section 1.4. Double click the HTML file webrepl to open it:



Webrepl opens in a Web browser, as shown in the following figure.

Note: use a good browser, such as Chrome. Edge may not work.



Now, you can run various Python commands. For example:

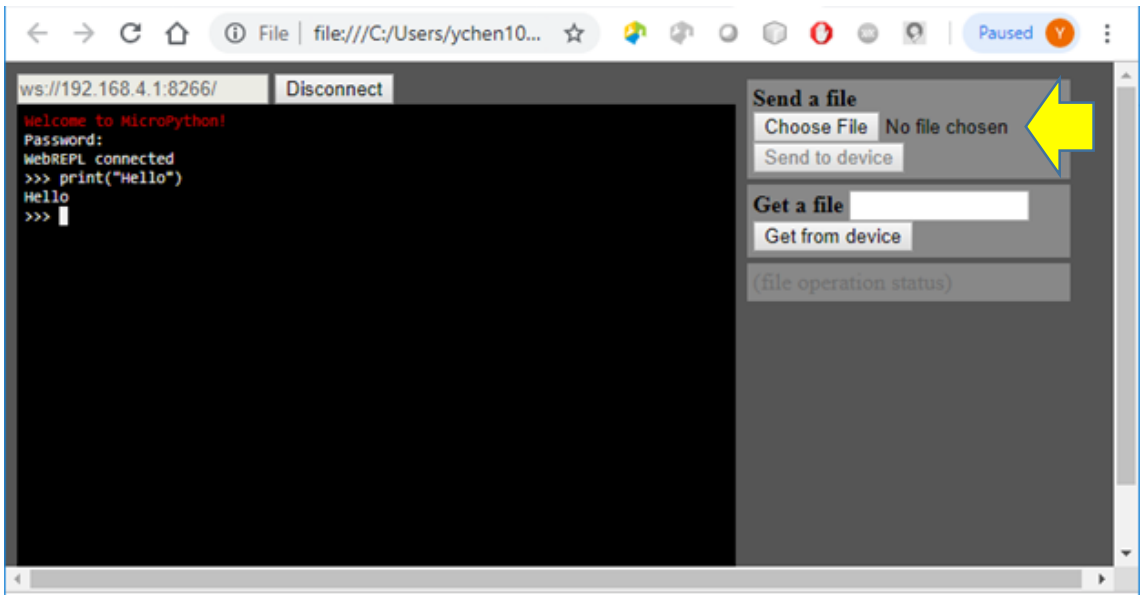
>>> print(“hello!”)

It will print Hello. In this example, Webrepl is running Python code in Webrepl, not in ESP8266 chip.

2.2 Uploading Python program into Webrepl and sending to ESP8266

Webrepl also allows us to upload a Python file into the Webrepl and then send the code to ESP8266. It means that you can use a normal Python programming environment, as downloaded and installed in Section 1.1 on our PC, to write normal Python code. You can also use VIPLE to write Python code!

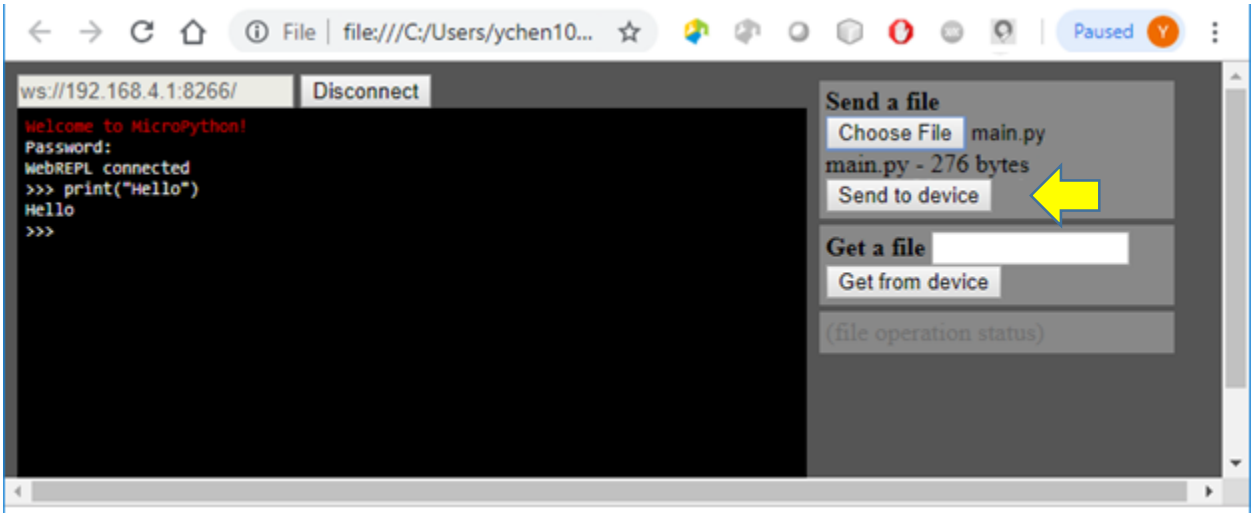
To upload a file, you click Choose File, as shown in the following figure.



The file you upload must be called main.py. If you have many Python programs, you must use different names for the program. When you want to upload a program, you can make a copy of the program and change the copy to main.py.

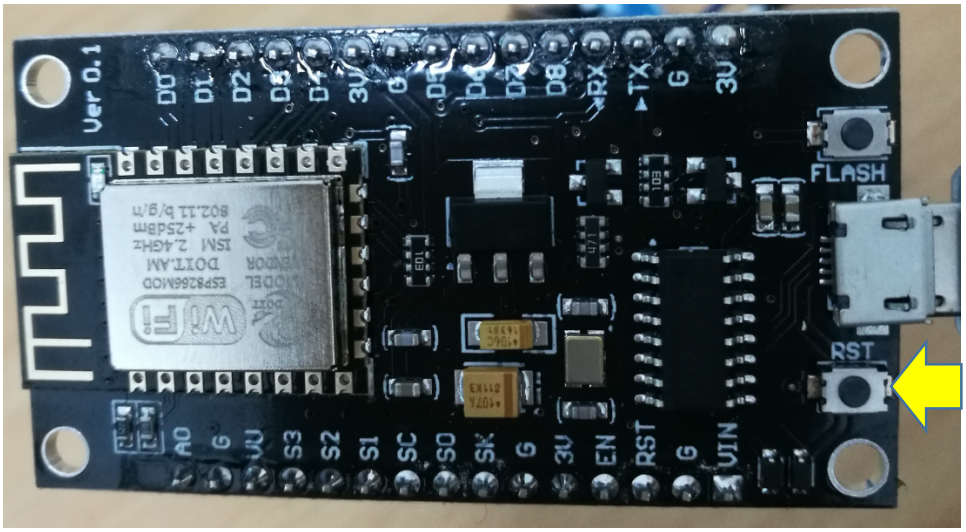
You can click connect or disconnect to ESP8266 in Webrepl. You must make sure it is conneted before you send the code to ESP8266 for execution.

After selected the main.py, click Send to device, which will send the file to ESP8266 via WiFi.



To run a new program on ESP8266, you need to reset the ESP8266.

There are two different ways to reset ESP8266. (1) By press the RST button on ESP8266, or unplug and re-plug the USB cable.



You can also use CMD command to send the Python file to ESP8266:

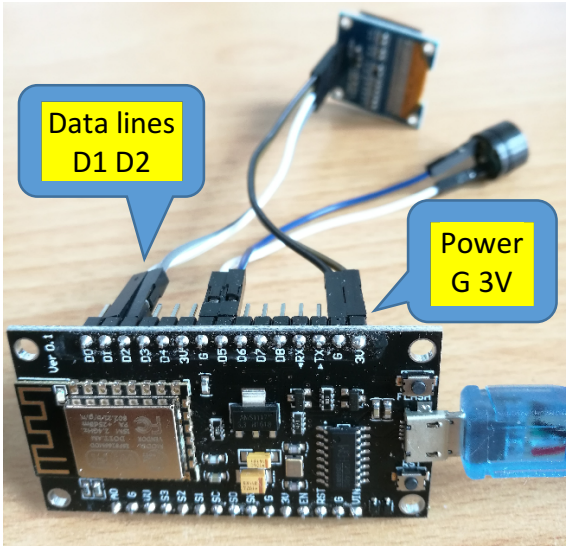
./webrepl/webrepl_cli.py -p 1234 main.py 192.168.4.1:main.py

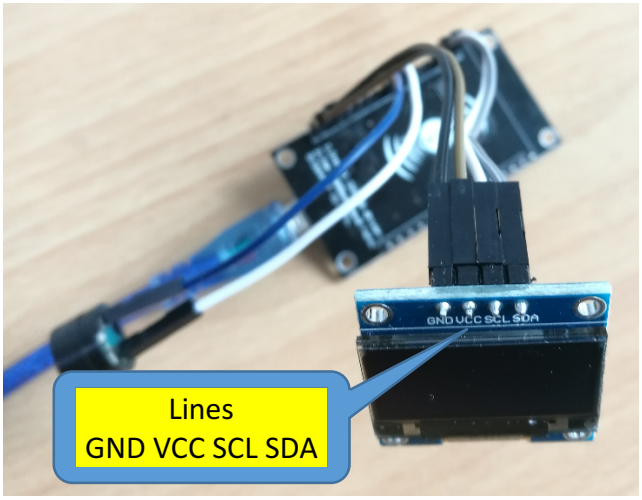
2.3 OLED Display on Device: Text

In this section, you will write a Python program that will display a string on the display panel connected to the ESP8266. First, you need to connect OLED Display and the Beeper to ESP8266. The connections are defined as follows:

```
OLED    esp8266                 Beeper       esp8266
SDA <--> D2                       +     <-->   D5
SCL <--> D1                       Hx    <-->   G
VCC <--> 3V
GND <--> G
```

The following figures show the connection. Do not mixed the wires!





The following code convert text to image and display.

The Python code is given as follows. Copy the code into a separate file and name the file main.py, and then follow the process in Section 2.2 to send it to ESP8266.

```
"""
display demo

Before running the code, connect OLED and ESP8266 as follows:
    OLED    esp8266
    SDA <--> D2
    SCL <--> D1
    VCC <--> 3V
    GND <--> G
"""
import ssd1306
import machine
from machine import I2C, Pin

# init display
i2c = I2C(sda=Pin(4), scl=Pin(5))
display = ssd1306.SSD1306_I2C(128, 64, i2c)
display.fill(0)

# show welcome
display.text('Hello world', 10, 10)
display.show()
```

2.4 OLED Display on Device: Image

In this section, you will write a Python program that will display an image on the display panel connected to the ESP8266.

The Python code is given as follows. Copy the code into a separate file and name the file main.py, and then follow the process in Section 2.2 to send it to ESP8266.

```
"""
display an image on OLED
"""

import ssd1306
import machine
import sys
import time
from machine import I2C, Pin

# draw a picture function
def draw(pic,offset_x,offset_y):
    for y, row in enumerate(pic):
        for x, col in enumerate(row):
            if col == "1":
                display.pixel(x+offset_x, y+offset_y, 1)

# init display
i2c = I2C(sda=Pin(4), scl=Pin(5))
```

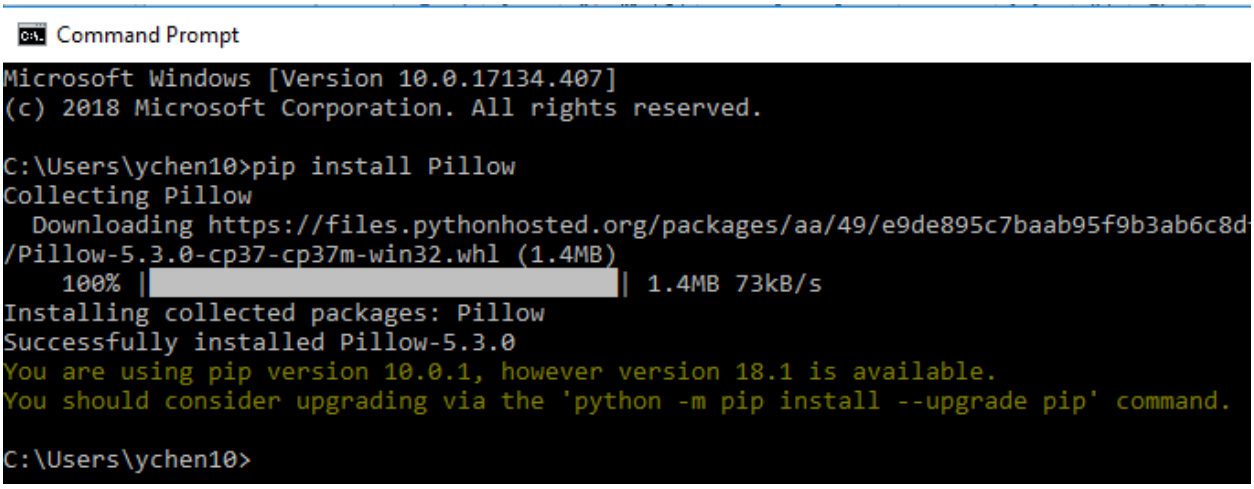
```
display = ssd1306.SSD1306_I2C(128, 64, i2c)
display.fill(0)

# image data
testMap = [
'1111111111100000',
'1000010000100000',
'0100010001000000',
'0010010010000000',
'0001010100000001',
'0000111000000101',
'0000010000010101',
'0000010001010101',
'0000010101010101',
'0000010101010101',]

# draw image
draw(testMap,105,16)
display.show()
```

2.5 Generate map from png tool

Start CMD console and execute the following commands:

Install Pillow package for reading images:

pip install Pillow



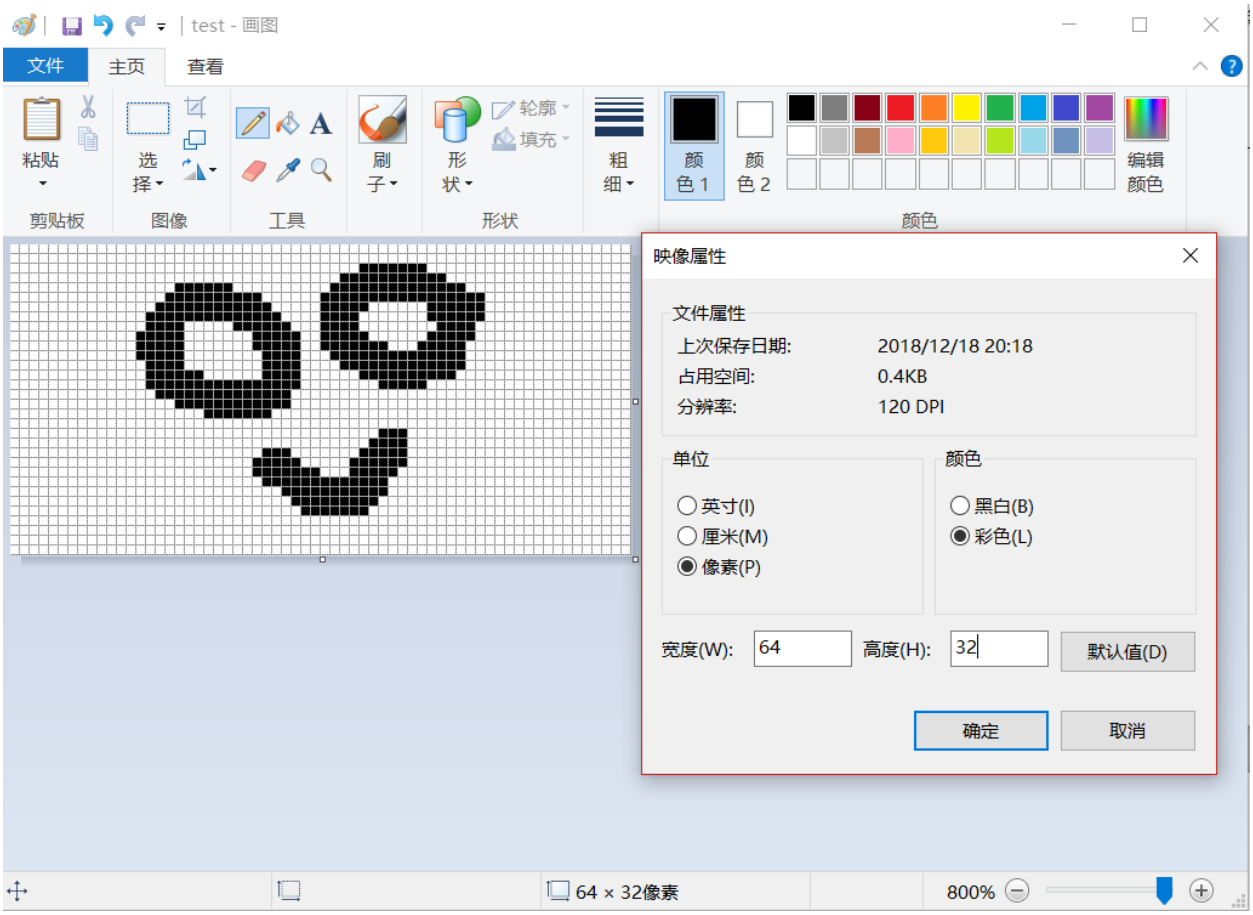
Create a Python program based on the following code:

generate.py

```
import sys
from PIL import Image

argv = sys.argv
if(len(sys.argv)<2):
    print("Please input image file name.")
    print("For example, input face.png, output faceMap.py")
    print("     python3 gengerate.py face")
```

```
    sys.exit(-1)

# open image file
image = Image.open(sys.argv[1]+".png")
width,height = image.size
print('image size',width,height)
pixels = image.load()

# open file
f = open(sys.argv[1]+"Map.py", "w")
f.write(sys.argv[1]+"Map = [\n")

for y in range(height):
    f.write("'")
    for x in range(width):
        # print(x,y,pixels[x,y])
        if(pixels[x,y][0]<=0): # black
            f.write('1')
        else:
            f.write('0')
    f.write("',\n")
f.write(")\n")
```

Use Paint tool to create a png file named test.png



Note, our OLED Display has a resolution of 128 x 64. You can create bigger and higher resolution images.

In CMD console, enter the following command:

python generate.py test

It will generates a file called testMap.py. It replaces testMap in python code of section2.4.

```
testMap = [
'00000000000000000000000000000000000000000000000000000000000000000000000000000000',
'00000000000000000000000000000000000000000000000000000000000000000000000000000000',
'00000000000000000000000000000000000000000000000000000000000000000000000000000000',
'00000000000000000000000000000000000000000000000000000000000000000000000000000000',
'00000000000000000000000000000000000000000000111100000000000000000000000000000000',
'00000000000000000000000000000000000000000011000100000000000000000000000000000000',
'00000000000000001111111000000000000000110000011000000000000000000000000000000000',
'00000000000000011000001000000000000001000000001100000000000000000000000000000000',
'00000000000000110000011000000000000010000000001000000000000000000000000000000000',
'00000000000001000000000100000000000110000000000100000000000000000000000000000000',
'00000000000001000000000100000000000100000000000100000000000000000000000000000000',
'00000000000010000000000100000000001000000000000100000000000000000000000000000000',
'00000000000010000000000100000000001000000000000100000000000000000000000000000000',
'00000000000010000000000100000000001000000000000100000000000000000000000000000000',
'00000000000010000000000100000000001000000000000110000000000000000000000000000000',
'00000000000010000000000000000000001000000000000000000000000000000000000000000000',
'00000000000010000000000000000000001000000000000000000000000000000000000000000000',
'00000000000000000000000000000000001000000000000000000000000000000000000000000000',
'00000000000000000000000000000000000000000000000000000000000000000000000000000000',
'00000000000000000001111111000000000100000000000000000000000000000000000000000000',
'00000000000000001111100000011000001111111000000000000000000000000000000000000000',
'00000000000000110000000000000110000010000011100000000000000000000000000000000000',
'00000000000000100000000000000011000010000000110000000000000000000000000000000000',
'00000000000000010000000000000100000010000000010000000000000000000000000000000000',
'00000000000000010000000000000100000010000000010000000000000000000000000000000000',
'00000000000000010000000000000100000010000000011000000000000000000000000000000000',
'00000000000000010000000000000100000010000000010000000000000000000000000000000000',
'00000000000000010000000011000000001000000010000000000000000000000000000000000000',
'00000000000000001000000100000000001000001000000000000000000000000000000000000000',
'00000000000000001100001000000000001100010000000000000000000000000000000000000000',
'00000000000000000111111000000000000111100000000000000000000000000000000000000000',
'00000000000000000000000000000000000000000000000000000000000000000000000000000000',
]
```

2.6 Beeper

In this section, you will write a Python program that will send squared wave to beeper to play sound. You can change the frequency of the squared wave to generate different sounds to compose music. You can also add delay to change the tempo and tone of your music!

Simple demo

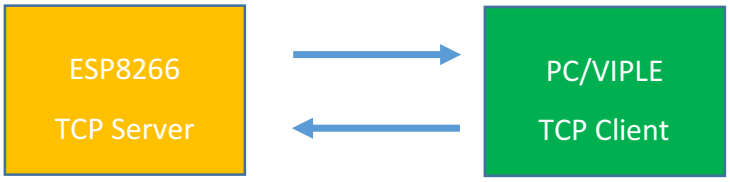
```
'''
Beeper demo
Hardware connection: Please make sure the wire is connected as follows:
    Beeper     esp8266
    +     <-->   D5
    Hx    <-->   G
'''

from machine import Pin, PWM
import time

beeper = PWM(Pin(14, Pin.OUT), freq=440, duty=512)

beeper.freq(264) # note the frequency
time.sleep(1)    # delay

beeper.freq(523) # note the different frequency
time.sleep(1)    # delay

beeper.deinit()
```

Complex demo: This demo defines tempo, tones to form your rhythm.

```
'''
Beeper demo more complex

Hardware connection: Please make sure the wire is connected as follows:
    Beeper          esp8266
    +         <-->   D5
    Hx        <-->   G
'''

from machine import Pin, PWM
import time
tempo = 5
tones = {
    'c': 262,
    'd': 294,
    'e': 330,
    'f': 349,
    'g': 392,
    'a': 440,
    'b': 494,
    'C': 523,
    ' ': 0,
}
beeper = PWM(Pin(14, Pin.OUT), freq=440, duty=512)
melody = 'cdefgabC'
rhythm = [8, 8, 8, 8, 8, 8, 8, 8]

for tone, length in zip(melody, rhythm):
    beeper.freq(tones[tone])
    time.sleep(tempo/length)
beeper.deinit()
```

2.7 Communication

In this section, you will write a TCP server program, which allows to connect VIPLE (and any TCP client) and ESP8266. Then, you can send and receive strings between them.



TCP Server Program

```
"""
This program creates a TCP server on Esp8266
"""
import socket

# socket server
addr = socket.getaddrinfo('0.0.0.0', 8080)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)

print('ESP8266 is listening on', addr)

# main loop
while True:
    cl, addr = s.accept()
    print('client connected from', addr)
    cl_file = cl.makefile('rwb', 0)
    while True:
        line = cl_file.readline()
        if not line or line == b'\r\n':
            break
        else:
            print(line,len(line))
            cl.send(line)
    cl.close()
    print('client disconnected!', addr)
```

Now, you write a TCP client to communicate with the ESP8266 server:

```
"""

tcp client on PC

"""

import socket


print("Tcp client Demo.")
```

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('192.168.4.1', 8080))
print("connected.")
while True:
    cmd = input("Input: ")
    cmd = cmd + '\r\n'
    sock.send(cmd.encode())

    data = sock.recv(128)
    if(data == b''):
        sock.close()
        print("End of connect.")
        break
    else:
        print("Recevie: " + data.decode())
```

This client program sends a string message to the server. Then, the ESP8266 server program process the message and send a message back to the client.

Section 3: Creating your Own Innovative Application

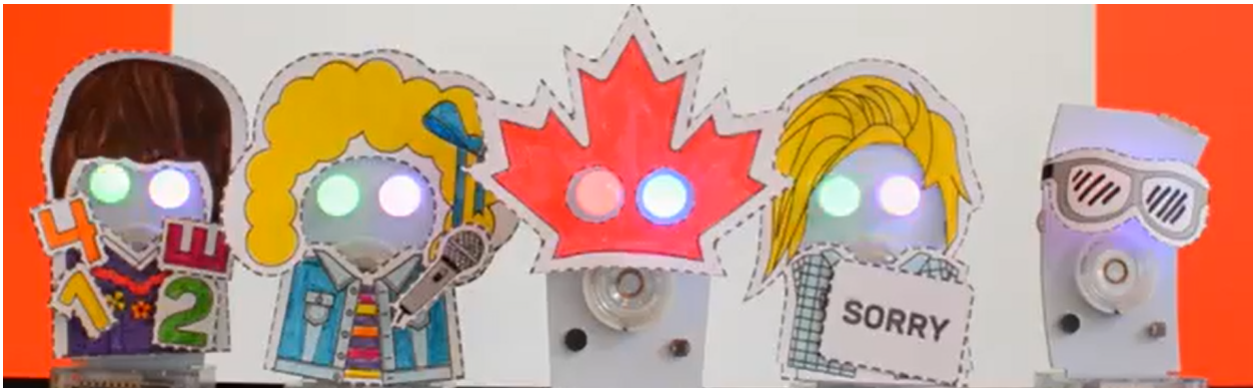
In the previous section, you have a few basic mechanisms that allow you to send string messages, and images the **OLED Display** panel. You can create your own music and play on the **Beeper**. Your ESP8266 can also send message back to your Python program and to VIPLE program. Thus, you can write interactive programs from your PC to your ESP8266 device.

In this section, you will create your **innovative application** using your skills and your imagination.

The following examples show dynamic smiley face expressions that you could create on your OLED Display! The OLED Display has 128 x 64 pixels, and you may create a game one it!



You may also decorate your Smiley face with paper and your painting.



Now, you can exercise your Engineering Design Process! Have a brainstorm section with your team members. Only the sky is the limit of your imagination! Be creative and have fun!