NodeMcu & Micropython - PCC Class Curriculum

What is this?
We'll be programming the ESP8266 chip on the NodeMcu prototyping platform that has been flashed with a normalish version of the micropython firmware. https://github.com/hydronics2/esp8266-micropython-PCC

The firmware along with a rough guide of the tutorial we'll be working through can be found here:
https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/index.html
And here: https://learn.adafruit.com/micropython-basics-loading-modules/overview

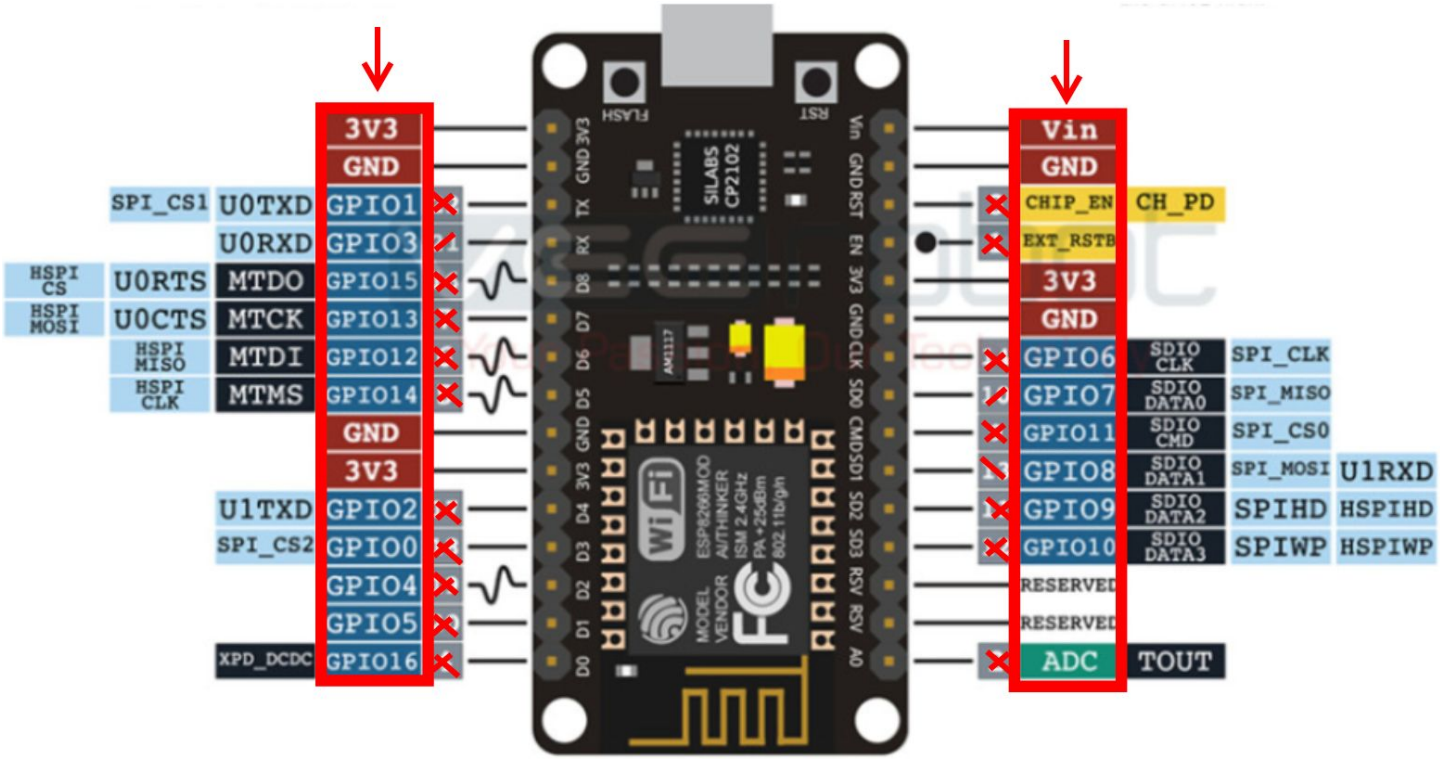
Getting Set Up
Plug the board in via USB and connect to it via WIFI (ssid printed on board)
Wifi Password: **micropythoN**

Next access the webrepl.html page in the Webrepl folder.

For reference: web repl client: https://github.com/micropython/webrepl/archive/master.zip
https://micropython.org/webrepl/

The password for the webrepl.html client page is: **python**

ESP8266 on the NodeMcu Pins



Reference for above picture: https://github.com/lvidarte/esp8266/wiki/MicroPython:-GPIO-Pins

Connect your board to the external world through the GPIO pins (General Purpose Input Output). Not all pins are available to use, in most cases only pins 0, 2, 4, 5, 12, 13, 14, 15, and 16 can be used.

GPIO Pins, associated Peripheral and Python Sketch

Peripheral	**type**	**pins**	**Example Sketch**
onBoard Leds	Output LED	Pin 16 (inverted signal)	
onBoard Led	Output, LED	Pin 2	blinkDemo.py
Button	Input	Pin 14	buttonDemo.py
WS2812 LED Array	Data, OUTPUT	Pin 0	ledDemo.py
Ultrasonic Sensor	Trigger, OUTPUT	Pin 16	ultrasonicDemo.py oledUltrasonicDemo.py
	Echo, Input	Pin 12	
0.96” i2c OLED	SCL, OUTPUT	Pin 4	oledPrintDemo.py oledUltrasonicDemo.py
	SDA, OUTPUT	Pin 5	
wifi			simpleHttpServer.py startKickAssServer.py

I’ve preloaded scripts and libraries onto your NodeMcu uController.

Python Scripts

- blinkDemo.py
- buttonDemo.py
- ledDemo.py
- ultrasonicDemo.py
- oledPrintDemo.py
- oledUltrasonicDemo.py
- simpleHttpServer.py
- startKickAssServer.py

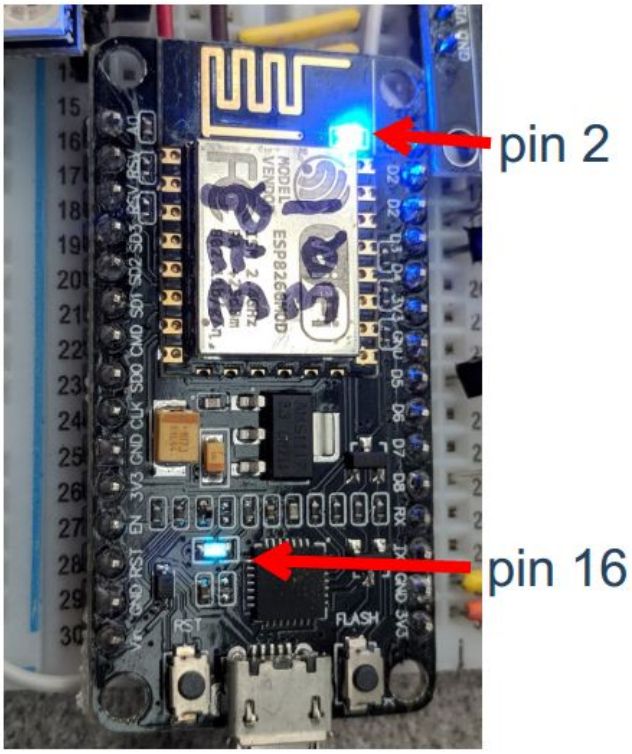
LIbraries

- Ultrasonic Sensor Library
- Oled Display Library

START HERE

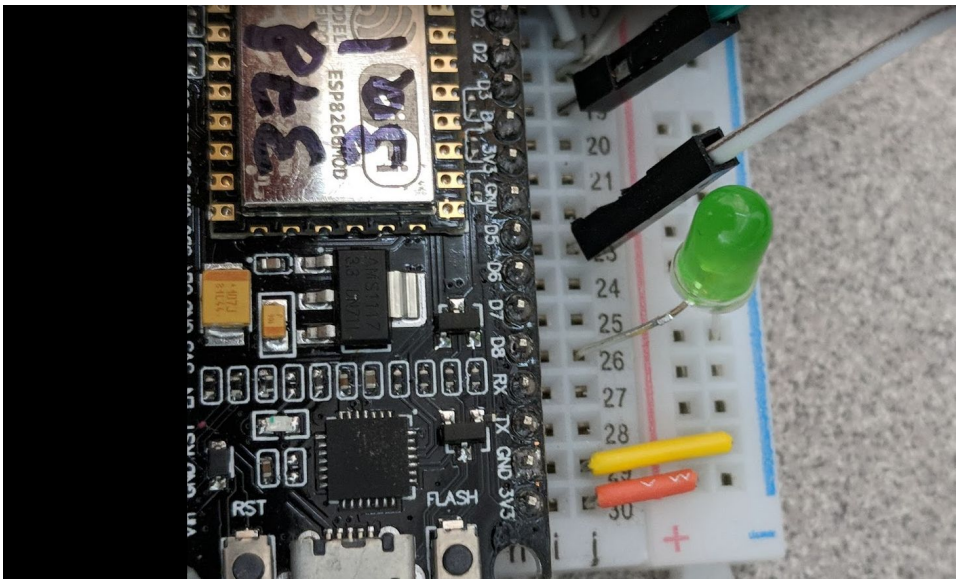
#1 Blink your LEDs

```
>>> import machine
>>> dir(machine)   #shows what options are available
>>> led2 = machine.Pin(2, machine.Pin.OUT)
>>> led2.on()
>>> led2.off()

>>> from time import sleep
>>> while True:
. . .      led2.off()
. . .      sleep(0.5)
. . .       led2.on()
. . .      sleep(0.5)

# Use ctrl C (to exit loop)

#define your own function:
>>>  def led_toggle():
. . .      led2.value(not led2.value())
#...//return return return until you see >>>

#now execute the function
>>> while True:
. . .      led_toggle()
. . .      sleep(0.25)
#...//return return return to start routine

# ctrl C (to exit loop)
```



#2...Try your own LED

```
# select an LED and plug it into
your breadboard….

#use a green or blue LED as it's
3.3Volts without a resistor!



>>> led3 = machine.Pin(15,
machine.Pin.OUT)
>>> led2.on()
>>> led2.off()
```



#3 Digital Inputs with Buttons

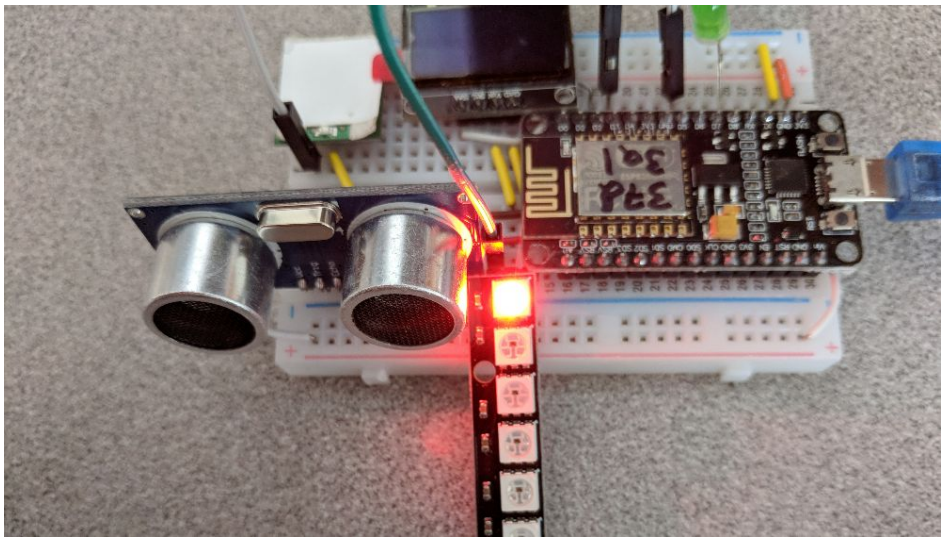
```
>>> import machine
>>> button = machine.Pin(14, machine.Pin.IN)

#now read the button
>>> button.value()   #returns a 1 or 0
#use the white wire to connect pin 14 to ground (blue rail) and read the sensor…>>button.value()
#next try connecting the white wire from 14 to 3V3 or the red rail

#next plug it back into the touch sensor and try that….
```

#4 Why do all this work… lets play with some scripts...

```
>>> import os
>>> os.listdir()

>>> import blinkDemo

>>> import buttonDemo
```

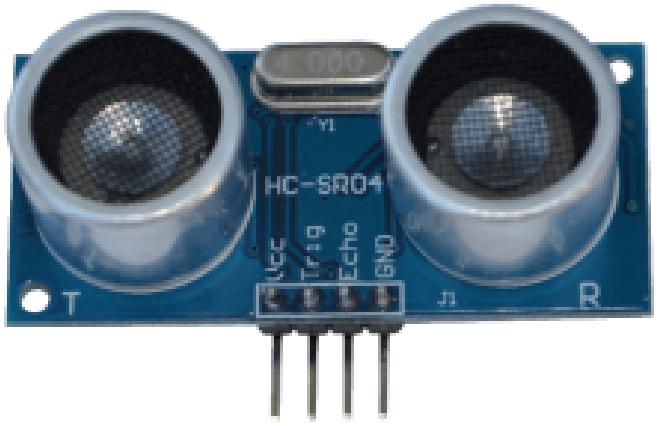
#5 WS2812 LED Demo

```
#Let's light up some addressable WS2812 LEDs

>>> import machine, neopixel
>>> np = neopixel.NeoPixel(machine.Pin(0), 8)  #using pin 0 for data in and 8 total LEDs
>>> np[0] = (255, 0, 0)  #sets the first led to 100% red
>>> np.write()
```



```
#load ledDemo.py script

>>>import ledDemo
```

Ultrasonic Sensor
Connect your ultrasonic sensor to your board
VCC goes to VIN
GND goes to GND
pin 16 goes to trigger
pin 0 goes to echo

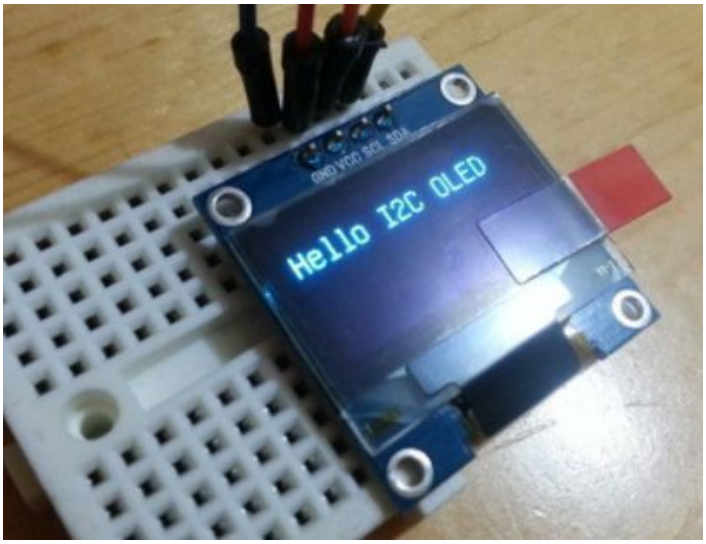


#upload ultrasonic.py to your board (already done)

```
>>> import machine
>>> import time
>>> from time import sleep
>>> import ultrasonic

>>> trig=machine.Pin(16, machine.Pin.OUT)
>>> echo=machine.Pin(12, machine.Pin.IN)
>>> sensor = ultrasonic.Ultrasonic(trig, echo)
>>> while True:
. . . print(sensor.distance())
. . . time.sleep(0.1)
. . .
. . .

>>> import ultrasonicDemo
```

0.96” OLED
#connect gnd, vcc, sda, and scl
library found here: https://github.com/peterhinch/micropython-samples/tree/master/SSD1306

```
>>> import machine, ssd1306
>>> i2c = machine.I2C(scl=machine.Pin(4),
sda=machine.Pin(5))
>>> oled = ssd1306.SSD1306_I2C(128,64, i2c)
>>> oled.text('Happy happy', 0,0)
>>> oled.show()

#run demos
>>>import OledPrintDemo

>>> import oledUltrasonicDemo
```



 # Simple HTML
reads all the pins and reports the values

>>> import simpleHttpServer

#enter into the browser 192.168.4.1:80 to see the current state of the pins (high or low)

 # Better Server
loads index.html with javascript files

>>> import startKickAssServer

#enter into the browser 192.168.4.1:80 to see the current state of the pins (high or low)

General information on micropython: http://docs.micropython.org/en/latest/esp8266/esp8266/general.html

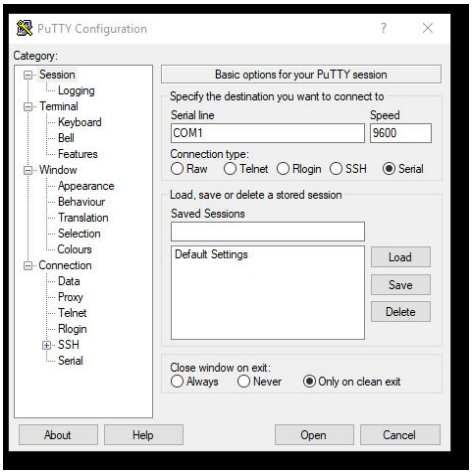
Flashing from Start
If the module stops responding you may have to re-flash the firmware onto the board

https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/intro.html

Serial Connections
Other methods for connecting to your board over serial include Mac OS serial, Putty, BBC Microbit, ampy, or Arduino Serial Monitor.

- On Linux or Mac OSX the screen command can be used to connect to the serial port. Run the following command to connect at 115200 baud:
    - `screen /dev/tty.board_name 115200`
- BBC Microbit code editor: https://codewith.mu/
- ...and Putty SSH windows client: https://www.putty.org/
- Ampy can be used to load files onto your device.. More powerful than the webrepl interface: https://learn.adafruit.com/micropython-basics-load-files-and-run-code/file-operations

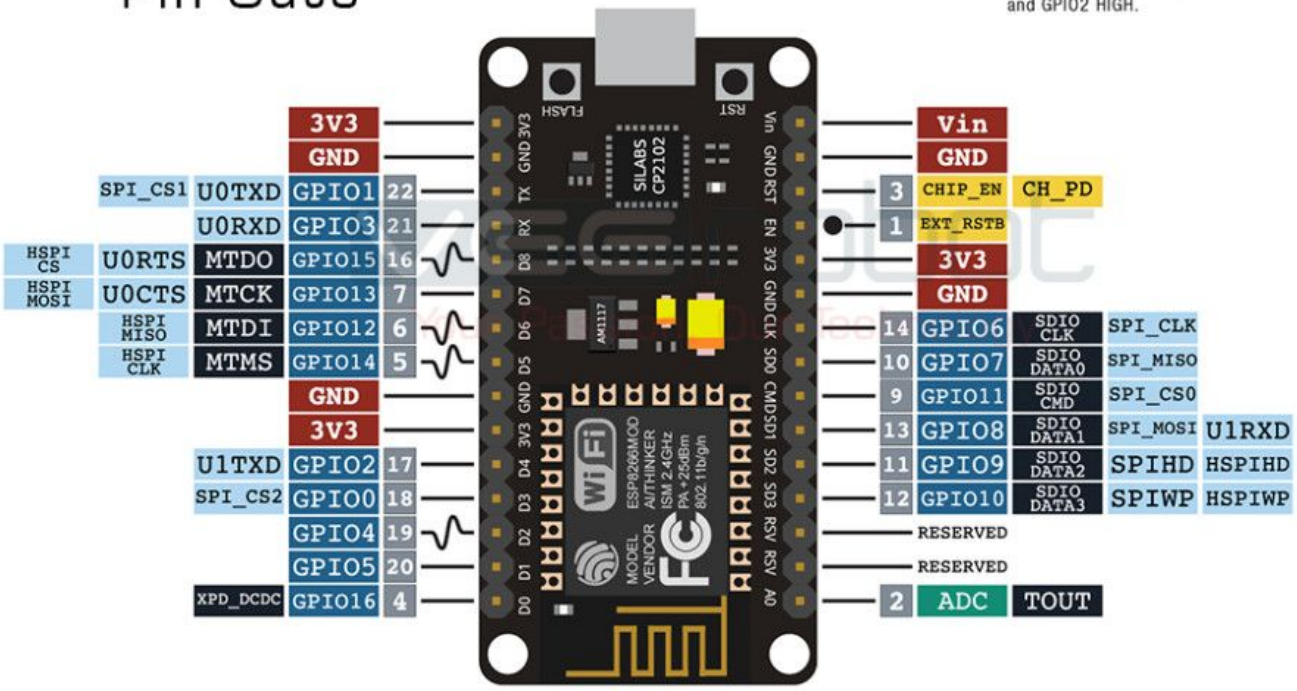
This is the Putty serial monitor interface on Windows.



NODEMcu - ESP-12E Development Board Pin Outs

NOTES:

- ▲ Typ. pin current 6mA (Max. 12mA)
- ▲ For sleep mode, connect GPIO16 and EXT_RSTB. On wakeup, GPIO16 will output LOW for system reset.
- ▲ On boot/reset/wakeup, keep GPIO15 LOW and GPIO2 HIGH.



POWER	SP. FUNCTION(S)
I/O	COMM. INTERFACE
ADC	PIN NUMBER
CONTROL	PWM
N/C	