# Embedded System Design Techniques™

# Designing IoT Sensor Nodes using the ESP8266

## Session 4: Connecting the ESP8266 to the internet

July 13th, 2017

Jacob Beningo

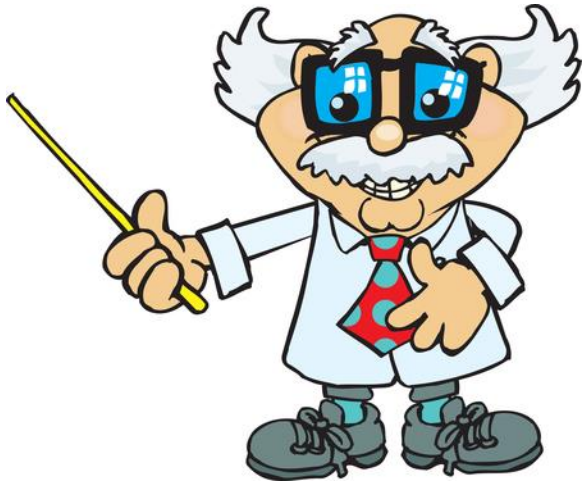
# Course Overview

**Topics:**

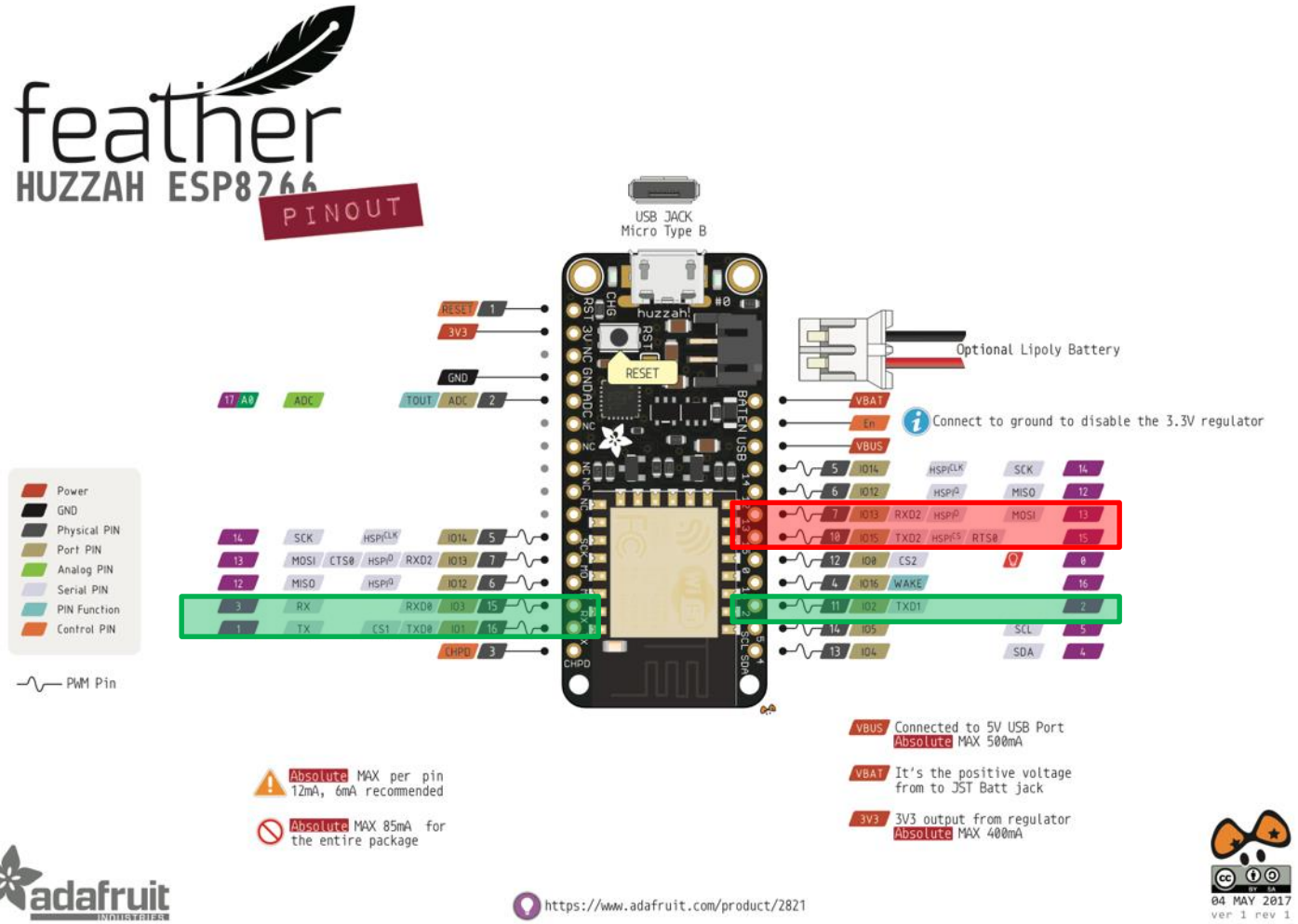
- The IoT Architecture
- Getting Started with the ESP8266
- Interfacing Sensors to the ESP8266
- **Connecting the ESP8266 to the internet**
- Device Management and the Automated Universe

# Session Overview

- The WebREPL
- Network Basics
- Configuration of Wi-Fi
- Sockets



# ESP8266 Feather Board



# Setting Up the WebREPL

- **import** webrepl_setup

```
>>> import webrepl_setup
WebREPL daemon auto-start status: disabled

Would you like to (E)nable or (D)isable it running on boot?
(Empty line to quit)
> 
```

To enable WebREPL, you must set password for it

New password: *******

Confirm password: ********

Changes will be activated after reboot

Would you like to reboot now? (y/n)

# Setting Up the WebREPL

```
WebREPL daemon started on ws://192.168.4.1:8266
Started webrepl in normal mode
OSError: [Errno 2] ENOENT

MicroPython v1.9.1-8-g7213e78d on 2017-06-12; ESP module with ESP8266
Type "help()" for more information.
>>>
```

ESP8266 is now an Access Point!

# Setting Up the WebREPL


# Networking the ESP8266

- There are two WiFi interfaces
  - Station Mode (ESP8266 connects to a router)
  - Access Point (Devices connect to the ESP8266).
- Use the REPL to check the interface status as follows:
- You can check if the interfaces are active by:

```
MicroPython v1.9.1-8-g7213e78d on 2017-06-12; ESP module with ESP8266
Type "help()" for more information.
>>> import network
>>> sta_if = network.WLAN(network.STA_IF)
>>> ap_if = network.WLAN(network.AP_IF)
>>> sta_if.active()
False
>>> ap_if.active()
True
>>>
```

# Networking the ESP8266

- Enable station mode and disable the access pt

```
>>> sta_if.active(True)
#5 ets_task(4020ed88, 28, 3fff9648, 10)
>>> ap_if.active(False)
>>> sta_if.active(True)
>>> ap_if.active()
False
>>> sta_if.active()
True
```

- Check the network settings using:

```
>>> ap_if.ifconfig()
('0.0.0.0', '0.0.0.0', '0.0.0.0', '208.67.222.222')
```

- The returned values are: IP address, netmask, gateway, DNS.

# Networking the ESP8266

- Connect to your WiFi network:

>>> sta_if.connect('<your ESSID>', '<your password>')

i.e. >>> sta_if.connect('MADSCIE', '12345678')

- To check if the connection is established use:

```
>>> sta_if.isconnected()
True
```

- Verify the IP address:

```
>>> sta_if.ifconfig()
('10.0.0.253', '255.255.255.0', '10.0.0.2', '10.0.0.2')
```

# WebREPL Revisited

micropython.org/webrepl/

Apps Bookmarks EmbeddedNews Subscriber Downlo... Qqest Time and Att... Other Bookmarks

ws://192.168.4.1:8266/ Connect

**Send a file**

Choose File No file chosen

Send to device

**Get a file**

Get from device

(file operation status)

*Terminal widget should be focused (text cursor visible) to accept input. Click on it if not.*
*To paste, press Ctrl+A, then Ctrl+V*

# Networking the ESP8266

network.py

```
1  import machine
2  import time
3  import network
4
5  pin = machine.Pin(2, machine.Pin.OUT)
6  pin.on()
7
8  network_connect()
9
10 while True:
11
12     pin.off()
13     time.sleep(0.5)
14     pin.on()
15     time.sleep(0.5)
16
17
18 def network_connect():
19
20     sta_if = network.WLAN(network.STA_IF)
21     if not sta_if.isconnected():
22         print('Establishing network connection ...')
23         sta_if.active(True)
24         sta_if.connect('MyNetwork', 'MyPassword')
25         while not sta_if.isconnected():
26             pass
27     print('network config:', sta_if.ifconfig())
```

# Socket Communication

- A socket represents an endpoint on a network device
- Two sockets are connected together to create a communication channel
- Internet protocols are built on top of sockets, such as email (SMTP), the web (HTTP), telnet, ssh, among many others.

# Socket Communication

- Socket module contains the library code for socket communication:

```
>>> import socket
```

- Set the server ip address using getaddrinfo:

```
>>> addr_info =  socket.getaddrinfo("myserver", 23)
```

- Getaddrinfo will return a list of addresses and ports
- Select the desired entry

    i.e addr = addr_info[3][6]

# Socket Communication - Receive

- Creating the socket connection is easy:

```
>>> s = socket.socket()
>>> s.connect(addr)
```

- Receive data using a simple loop similar to the following:

```
>>> while True:
...     data = s.recv(10)
...     print(str(data, 'utf8')'')
```

# Socket Communication - Transmit

```
def http_get(url):
    _, _, host, path = url.split('/', 3)
    addr = socket.getaddrinfo(host, 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (path, host), 'utf8'))
    while True:
        data = s.recv(100)
        if data:
            print(str(data, 'utf8'), end='')
        else:
            break
    s.close()
```

# Socket Communication Test

- Make sure that you import the socket module before running this function. Then you can try:

```
>>> http_get('http://micropython.org/ks/test.html')
```

# Additional Resources

- Download Course Material for
  - Python Doxygen Templates
  - Example source code
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm


From www.beningo.com under

- Blog > CEC – Designing IoT Sensor Nodes using the ESP8266

# The Lecturer – Jacob Beningo

**Jacob Beningo**

Principal Consultant

## Social Media / Contact

- E : jacob@beningo.com
- T : 810-844-1522
- : Jacob_Beningo
- : Beningo Engineering
- : JacobBeningo
- EDN : Embedded Basics
- ARM Connected Community

## Consulting

- Advising
- Coaching
- Content
- Consulting
- Training

BENINGO EMBEDDED GROUP

**www.beningo.com**