IoT Lab With Micropython and Friends

Sev Leonard

Hello!

- Portlandian for 12 years
- Pythonista for 5
- He/Him
- Research Software Engineer at Oregon Health & Science University
- Prior: Consulting, Analog Design Engineer at Intel

Acknowledgments

Joe Fitzpatrick, Securing Hardware: @SecurelyFitz

My partner Gibbs - care and feeding of a tutorial presenter, wire stripper extraordinaire

Agenda

- Project & Hardware overview
- MicroPython introduction
- Communicating with MQTT
- Temperature scavenger hunt
- IoT security
- Extra: multi sensor networks

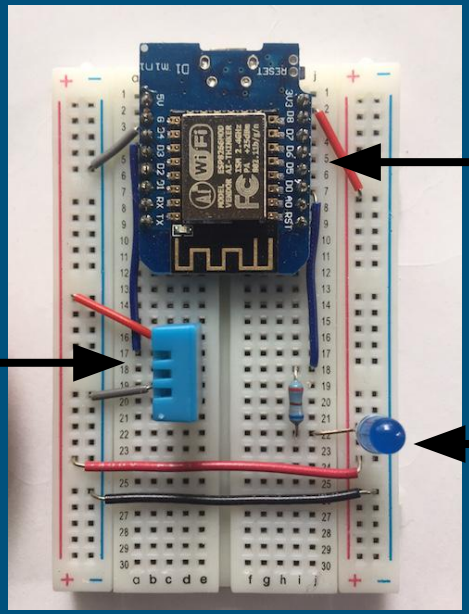
Pinot Noir

IoT and wineries

http://www.myomegasys.com/index.php/services/tracovino-vineyard

Willamette Valley Vineyards http://www.wvv.com

Vine sensor



WeMos WiFi microcontroller ESP8266

Temp & Humidity Sensor DHT11

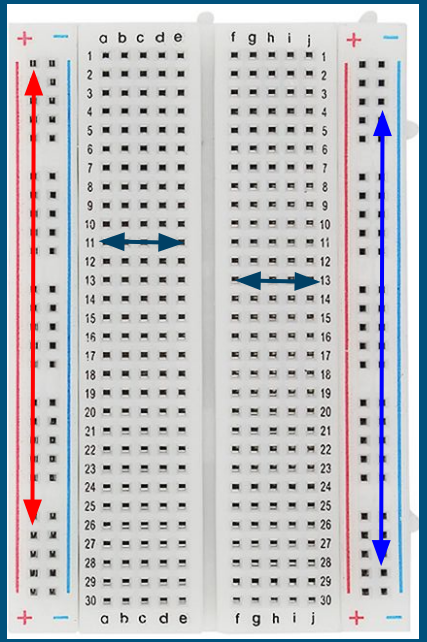
LED

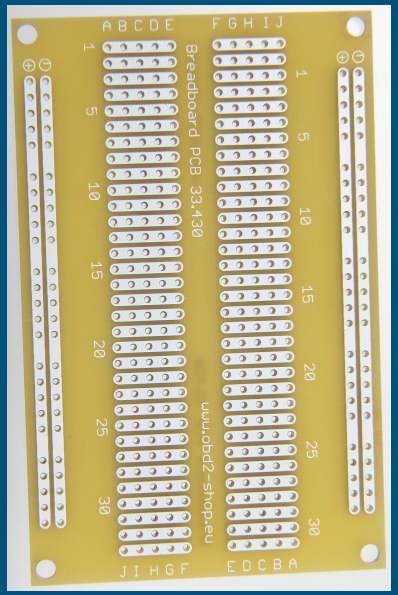
Breadboard?

In the early days of radio, amateurs nailed bare copper wires or terminal strips to a wooden board (often literally a board to slice bread on) and soldered electronic components to them. - Wikipedia

Supply lines (+/-) are VERTICAL

Signal lines (a - j) are HORIZONTAL

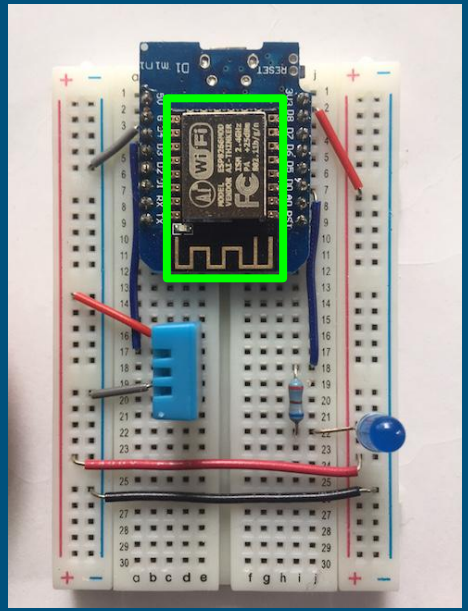




https://en.wikipedia.org/wiki/Breadboard

ESP8266

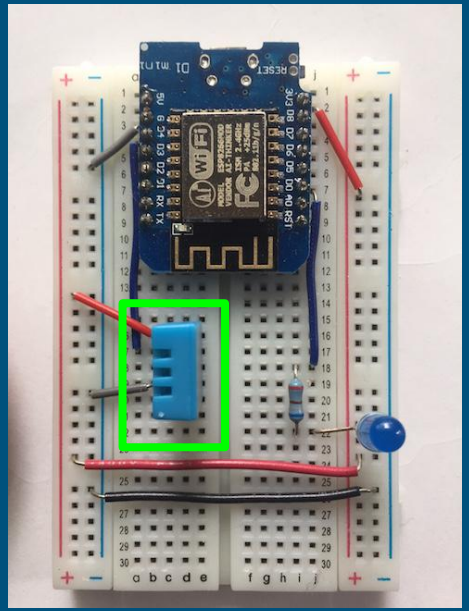
- WiFi enabled
- 3.3V supply



DHT11 - Digital Temp and Humidity Sensor

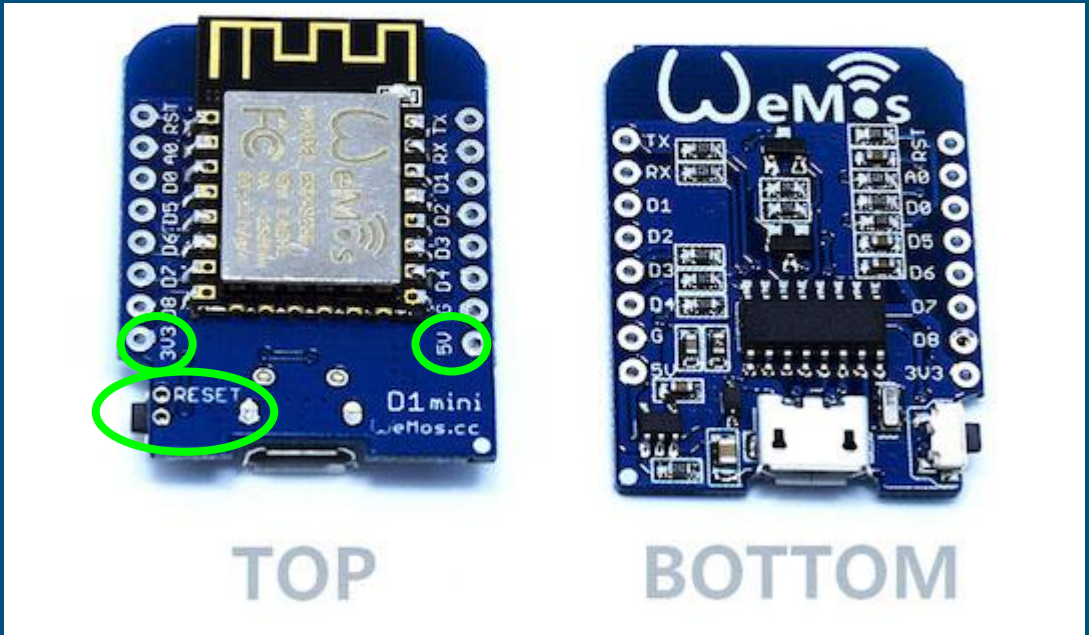
Blue light on WeMos will flash when a measurement is taken

Max sampling rate: 1s



WeMos D1 Mini - mini wifi board

- Supply Voltage
- Reset button



https://www.wemos.cc/product/d1-mini.html

Image source: alexnld.com

What is Micropython?

A streamlined version of Python for use on microcontrollers

Based on Python 3.4

Funded via kickstarter

Super neato!

micropython.org

About HARDware

- Its hard!
- Its not you!
- There is no StackOverflow for microfractures
- Its very rewarding when it works!!
- Deep breaths

pdxhackerspace.org

Agenda

- Project & Hardware overview
- MicroPython introduction
- Communicating with MQTT
- Temperature scavenger hunt
- IoT security
- Extra: multi sensor networks

Micropython Introduction

- Working with webREPL
- Writing and reading files
- Garbage collection
- Programming the LED and DHT
- Defining functions
- File transfer

Driver connection (optional)

screen /dev/tty.wchusbserial1410 115200

Lets get started!

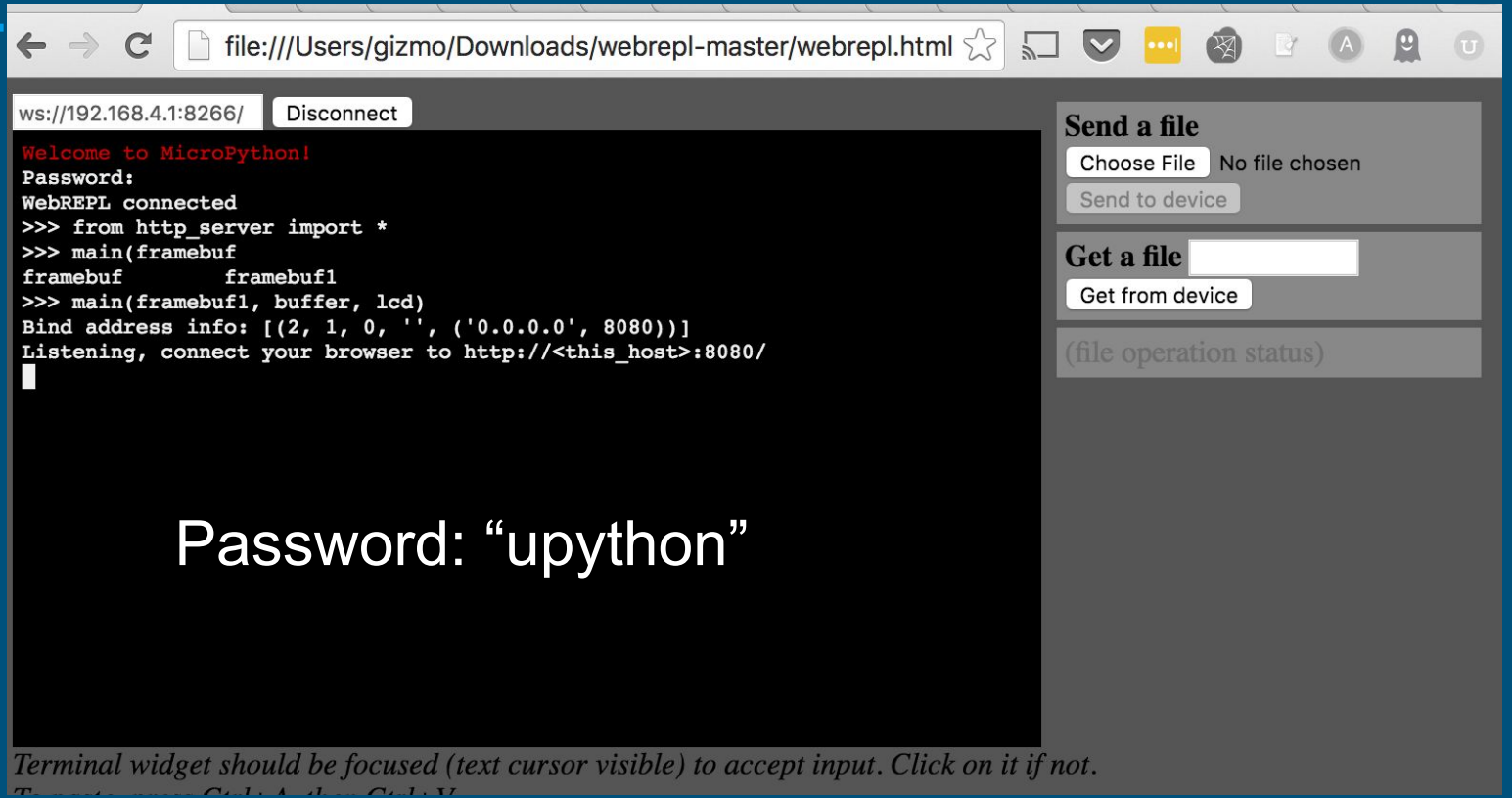
Plug in your vine sensor. You should see the blue LED turn on

Look on the back of the board for the access point SSID, connect to this ssid with the password “micropythoN”

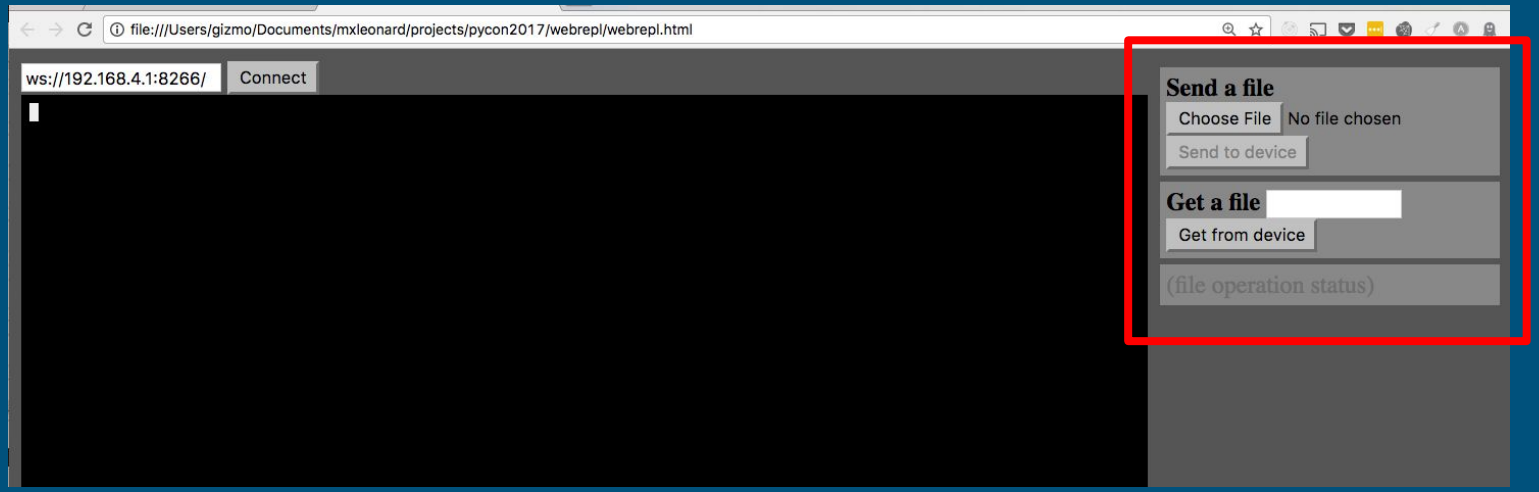
MAKE SURE YOU CONNECT TO THE RIGHT NETWORK!

Open up the webrepl.html file from the project directory in a browser

Web REPL


file:///Users/gizmo/Downloads/webrepl-master/webrepl.html
ws://192.168.4.1:8266/
Disconnect
Welcome to MicroPython!
Password:
WebREPL connected
>>> from http_server import *
>>> main(framebuf
framebuf framebuf1
>>> main(framebuf1, buffer, lcd)
Bind address info: [(2, 1, 0, '', ('0.0.0.0', 8080))]
Listening, connect your browser to http://<this_host>:8080/
Send a file
Choose File
No file chosen
Send to device
Get a file
Get from device
(file operation status)
Password: “upython”
Terminal widget should be focused (text cursor visible) to accept input. Click on it if not.

Sending and Getting files



Tips and Tricks

“Have you tried turning it on and off again?”

- Unplug to power cycle
- Press reset button
- Ctrl + D to soft reset

Wifi & WebREPL

- Check for your ssid with your phone, neighbor
- If WebREPL freezes, won’t disconnect - reload page

When in doubt, wait a bit

Pasting into WebREPL

Pasting code -

Chrome: ctrl+V, ⌘+V

Firefox: use Paste from Edit menu

Edge: ctrl+A ctrl+V

Tips and Quirks

Use CTRL+E to enter paste mode - Be careful about accidentally hitting ‘D’ instead of ‘E’ 💀

Quotes are sacrificed to the Quote God when pasting

Paste buffer is short - you may find that some of the code you copied does not get pasted

WebREPL may freeze. Reload page and reconnect

Command line history, but limited to last ~5 commands

Hello micropython!

```
print(“Hello micropython”)


def greetings(name):

    print(“welcome to micropython “, name, “!”)
```

Reading and writing files

```
with open(“test_file.txt”,‘a’) as f:
    f.write(‘micropython is rad!’)


with open(“test_file.txt”) as f:
    f.read()
```

a - append, will create the file if not exists

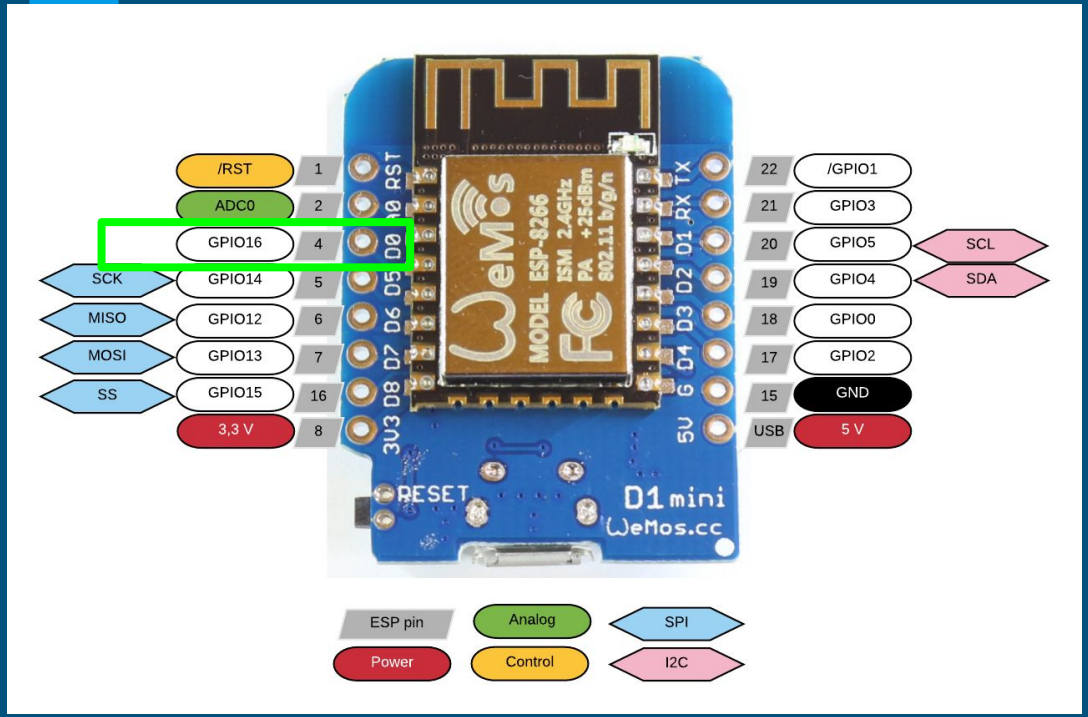
OS lib

```
import os

os.listdir()

os.remove('test_file.txt')

os.remove("main.py") # if present, remove this file
```

Garbage collection

```
import gc

gc.mem_free()

gc.collect()

gc.mem_free()
```

Pin definition




```
led = machine.Pin(16,machine.Pin.OUT)
```

Is your LED wired to D0? If not take a moment to change it.

Image source: https://3.bp.blogspot.com

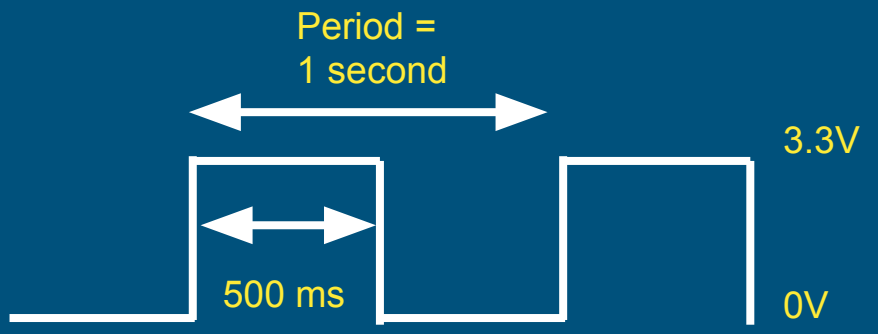
Machine module

```
from machine import Pin

led = machine.Pin(16,machine.Pin.OUT)

led.low()
led.high()

led.value()
```

Pulse Width Modulation

Rewire your led to Pin 14 (D5)

```
from machine import PWM
pwm = PWM(machine.Pin(14))
pwm.freq(1)
pwm.duty(500)
```



1 Hz frequency → 1 second period

500 = 500ms, amount of time the PWM signal stays high

Stop the blinking!

pwm.freq(0) - what happens?

pwm.duty(0)

Digital Humidity Temperature Sensor (DHT)

dht_module.measure()

dht_module.temperature()

dht_module.humidity()

```
>>> dht_module.measure()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "dht.py", line 13, in measure
OSError: [Errno 110] ETIMEDOUT
```

My DHT isn’t working!

Sorry! Try this instead:

```
import urandom
baseTemp = 28
def getTemp():
    return baseTemp + urandom.getrandbits(4)
```

DHT setup

```
import dht

my_dht = dht.DHT11(machine.Pin(2))

my_dht.measure()

my_dht.temperature()

my_dht.humidity()

import time

time.sleep(2)
```

Log DHT data

Create a function that:

1. Measures the temperature & humidity
2. Writes the measurements to separate files, separated by commas (.csv)
3. Prints measurements to the console
4. Polls at regular intervals

```
def measure_temp_humidity(temp_outfile, humidity_outfile, poll_time_s):
```

Local text editor, send via webREPL or paste mode (CTRL +E - beware the Quote God and character limits)

```
import dht
import time
my_dht = dht.DHT11(machine.Pin(2))

def measure_humidity(outfile, poll_time_s):
    while True:
        my_dht.measure()
        humidity = my_dht.humidity()
        with open(outfile, 'a') as f:
            f.write(str(humidity) + ",")
        print("humidity: ", humidity)
        time.sleep(poll_time_s)
```

dht_functions.py

with open(outfile) as f:

f.readlines()

Boot.py and main.py

```
import gc
import webrepl
webrepl.start()
gc.collect()
import dht
import machine
gc.collect()
dht_module = dht.DHT11(machine.Pin(2))
```

Setting up a main.py file

Create a file main.py containing:

- Your temp & humidity
- Required imports (pin, time, machine)

Transfer to the vine sensor

Reboot and reconnect webrepl

Great Job!

- We learned how to use micropython to:
  - Write to files
  - Program pins
    - LED - set values, use PWM
    - DHT - perform measurements, retrieve temp and humidity
- Gained familiarity with webREPL
  - Programming the ESP
  - File transfer

Your vine sensor is now setup to take measurements and log results!

Take a break!

Next up

How can we send real time data from the vine sensor?

- Overview of MQTT: a protocol for sending messages
- Using the mosquitto broker
- Test out pub/sub on our local machines
- Setup mqtt-spy
- Program the vine sensor to send and receive MQTT messages

Agenda

- Project & Hardware overview
- MicroPython introduction
- Communicating with MQTT
- Temperature scavenger hunt
- IoT security
- Extra: multi sensor networks

IoT communication - MQTT

- Our vine sensors are most helpful as a group

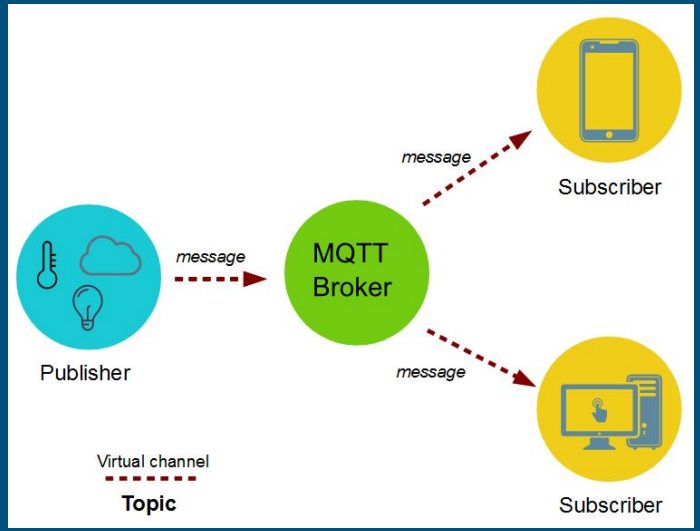


Image source: http://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html

MQTT

MQ Telemetry Transport

Communication is by topic

I.e. we will have topics ‘temperature’ and ‘humidity’

Easy to setup topics and add publishers/subscribers

Supports passwords and TLS/SSL

Mosquitto

Broker for MQTT - facilitates connections between the publishers and subscribers

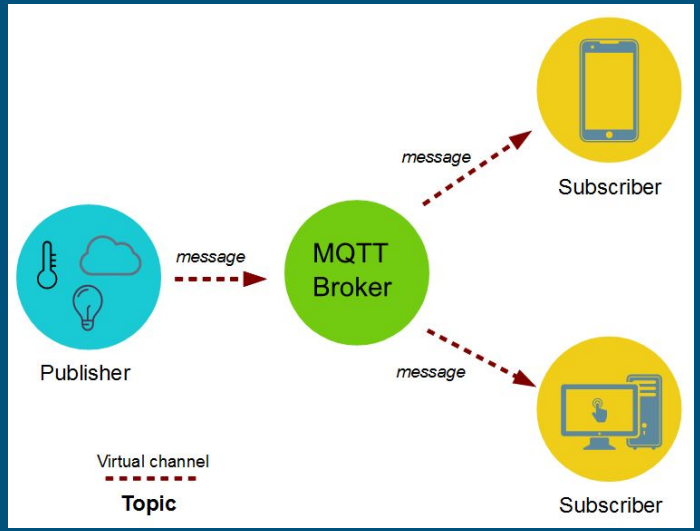


Image source: http://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html

mosquitto.conf

Example conf: esp8266_micropython_lab/mosquitto.conf

MacOS w/Brew: /usr/local/Cellar/mosquitto/<version>/etc/mosquitto/mosquitto.conf

Linux: /etc/mosquitto

Windows: C:\Program Files (x86)\mosquitto

```
# set a log to help debug issues
log_dest file /usr/local/Cellar/mosquitto/1.4.11_2/etc/mosquitto/mosquitto.log
log_dest stdout

# when we set a password, dont allow anonymous connections
#allow_anonymous false
#password_file /usr/local/Cellar/mosquitto/1.4.11_2/etc/mosquitto/passwd

# listeners for local host and when you are logged into the vine sensor
# only one at a time can be active
# replace 192.168.4.2 with your IP when connected to the vine sensor AP
listener 1883 localhost
#listener 1883 192.168.4.2

#TLS/SSL MQTT typically runs on port 8883
#listener 8883

# certificate / key files for TLS/SSL
#cafile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/ca.crt
#certfile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/beaker.local.crt
#keyfile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/beaker.local.key
```

Back to MQTT - Topics

Topics can be anything, as long as the publisher and subscriber are listening to the same one!

I.e: subscribe - topic ‘test’

publish -topic ‘test’ -message ‘is this thing on?’

Topics can have hierarchy

/vineyard/

temp/

vine_sensor/

7

8

….

/vineyard/temp/vine_sensor/7

/vineyard/temp/vine_sensor/8

Or :

/vineyard/temp/vine_sensor/#

Mosquitto.conf - setup for local use

Update your conf file to include:

listener 1883 localhost

Alternately use 127.0.0.1 for localhost

Restart mosquitto

brew services restart mosquitto

Or execute ‘mosquitto’ on Windows or Linux

Mosquitto client (MacOS)

1. Start mosquitto:

```
brew services start mosquitto
```

2. In a terminal start listening to test_topic on localhost

```
mosquitto_sub -h 127.0.0.1 -t test_topic
```

3. In another terminal, send some messages to test_topic

```
mosquitto_pub -h 127.0.0.1 -t test_topic -m 'Hello World'
```

Mosquitto Client (Windows & Linux)

Open the directory where you have mosquitto installed and open 3 command prompts

1. In one command window, start mosquitto

mosquitto (may need -c /path/to/mosquitto.conf)

2. In another command window, start listening to test_topic on localhost

mosquitto_sub -h 127.0.0.1 -t test_topic

3. In the 3rd command window, send some messages to test_topic

mosquitto_pub -h 127.0.0.1 -t test_topic -m 'Hello World'

Pub/Sub with the vine sensor

Publish from vine sensor to your computer

- Reporting sensor data to the vineyard server

Publish from your computer to the vine sensor

- Update polling frequency, for example

Setting up mosquitto on WiFi

Sign into your vine sensor’s wireless network - ssid: vineN

Get your IP address:

```
ifconfig | grep inet
```

Pick the address that starts with 192.168…. i.e. 192.168.4.3

Mosquitto conf updates

/usr/local/Cellar/mosquitto/1.4.11_2/etc/mosquitto/mosquitto.conf

At the end of the conf file:

```
# listener 1883 localhost
listener 1883 192.168.4.2  ← replace with your IP address
```

Restart mosquitto

```
brew services restart mosquitto
```

OR ctrl+c mosquitto and restart

Subscribe to VineSensor

In a terminal window subscribe to a topic:

```
mosquitto_sub -h 192.168.4.3 -t vine_sensor/message
```

Log into the vine AP and log into WEBREPL

```
from umqtt.simple import MQTTClient
c = MQTTClient("umqtt_client",’192.168.4.3’)
c.connect()
>>> 0
```

Send message from the vine sensor

```
c.publish(b"vine_sensor/message",b"hi from vine7!")
```

Check your terminal window subscriber

Try it with the DHT

Create topics: /vine_sensor/temp and /vine_sensor/humidity

Publish data from the DHT on this topic

Remember you have my_dht defined by your main.py file

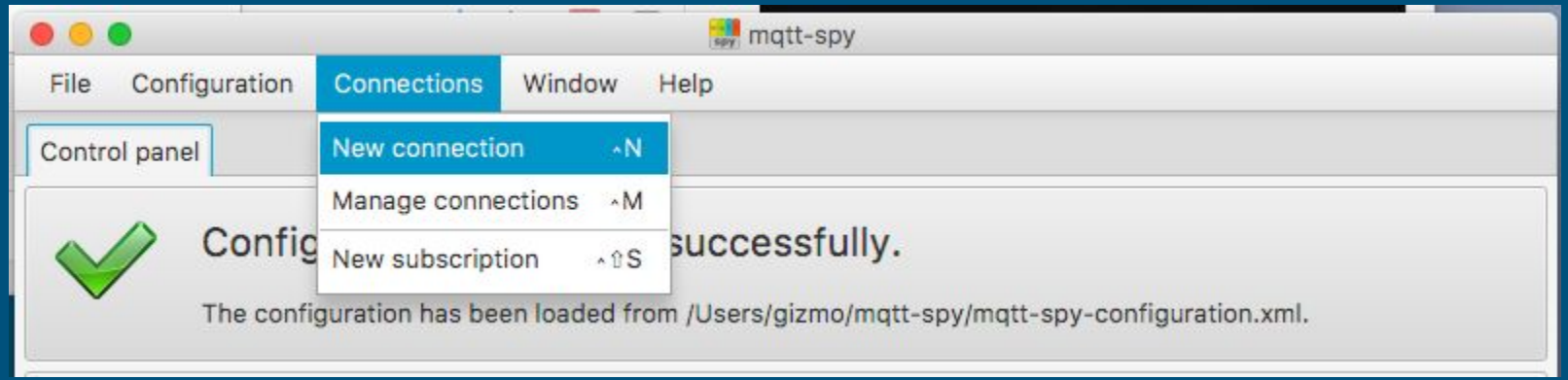
Send temperature

```
mosquitto_sub -h 192.168.4.3 -t vine_sensor/temp
```

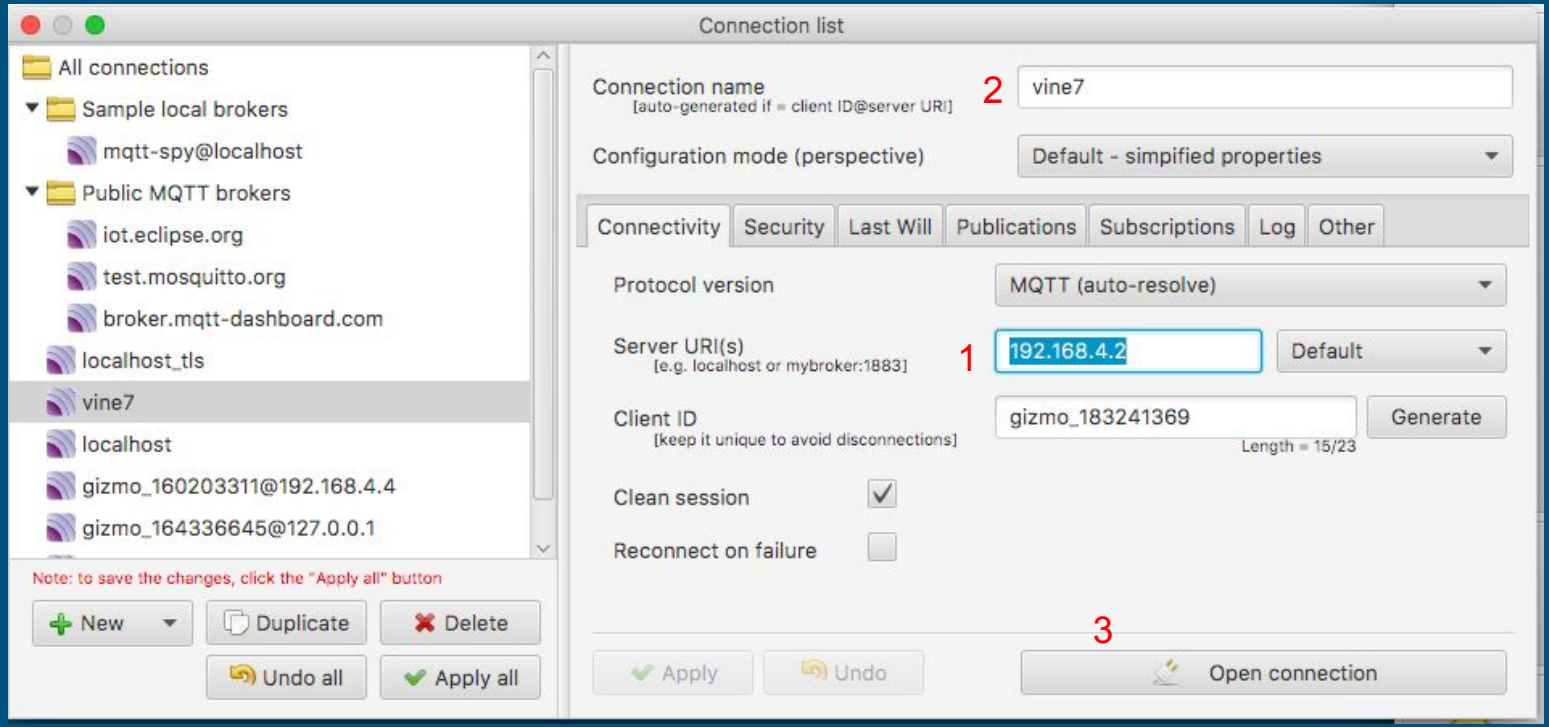
```
>>> my_dht.measure()
>>> c.publish(b"/vine_sensor/temp",str(my_dht.temperature()))
>>> c.publish(b"/vine_sensor/humidity",str(my_dht.humidity()))
```

Setup MQTT-SPY

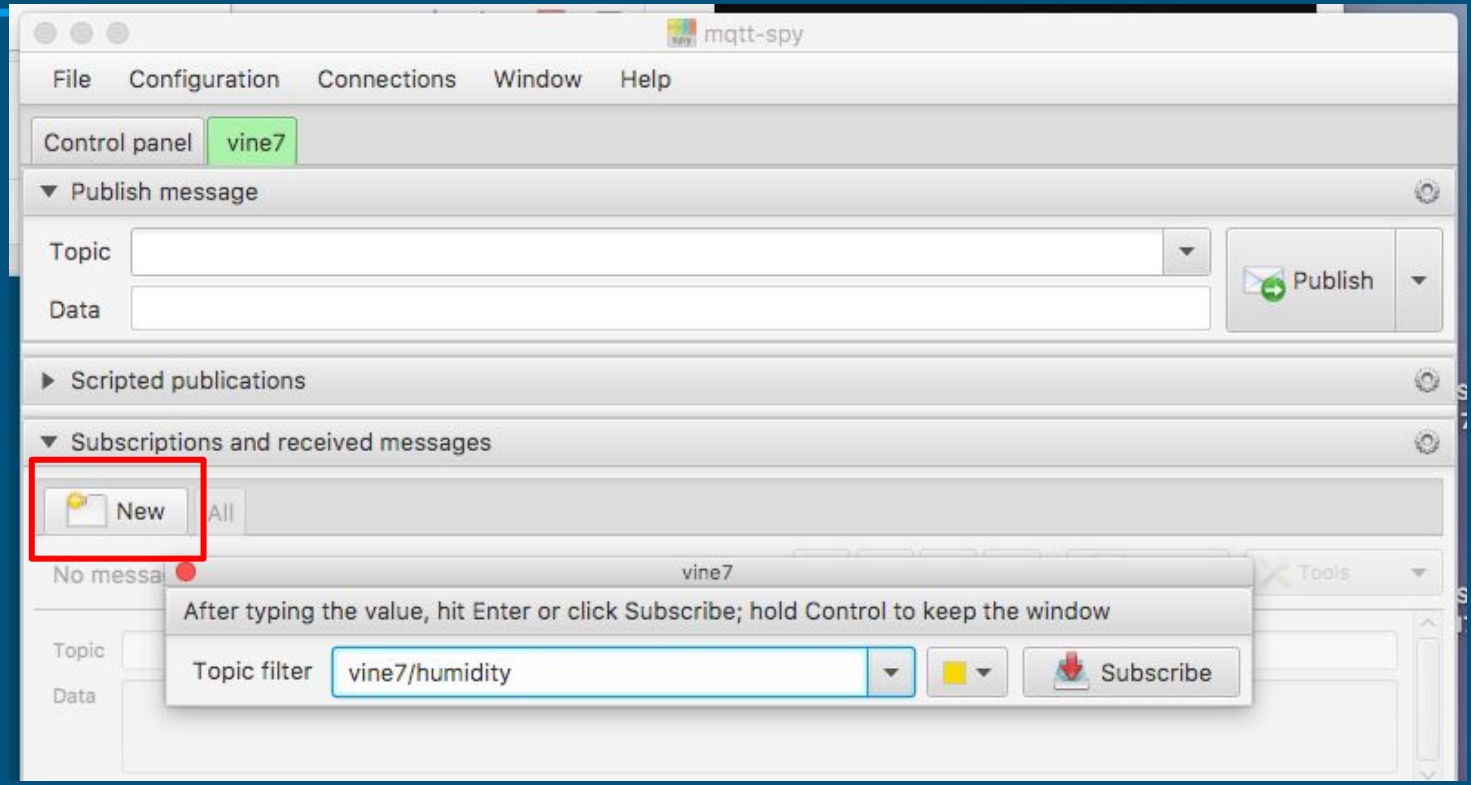
Double click the jar file to run



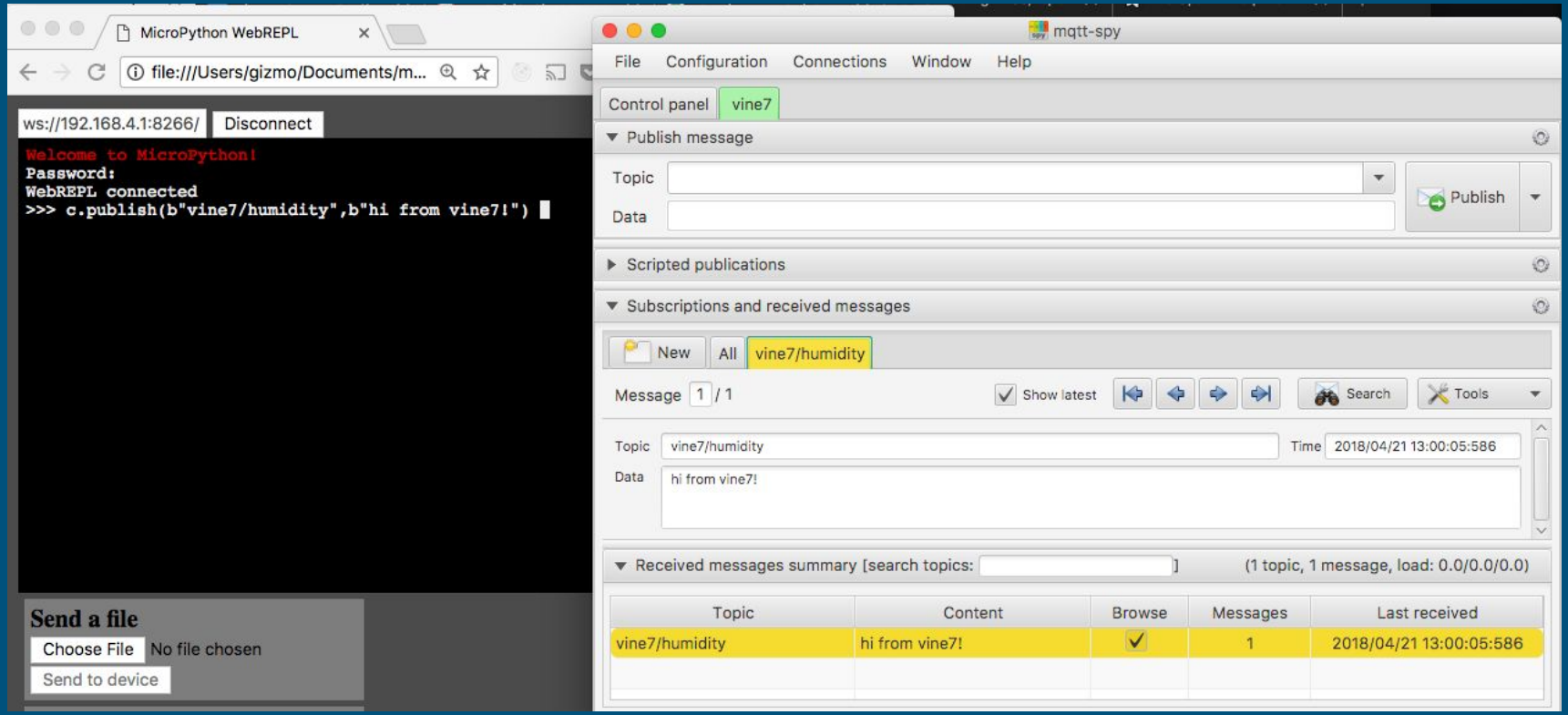
MQTT Spy - Set Connection



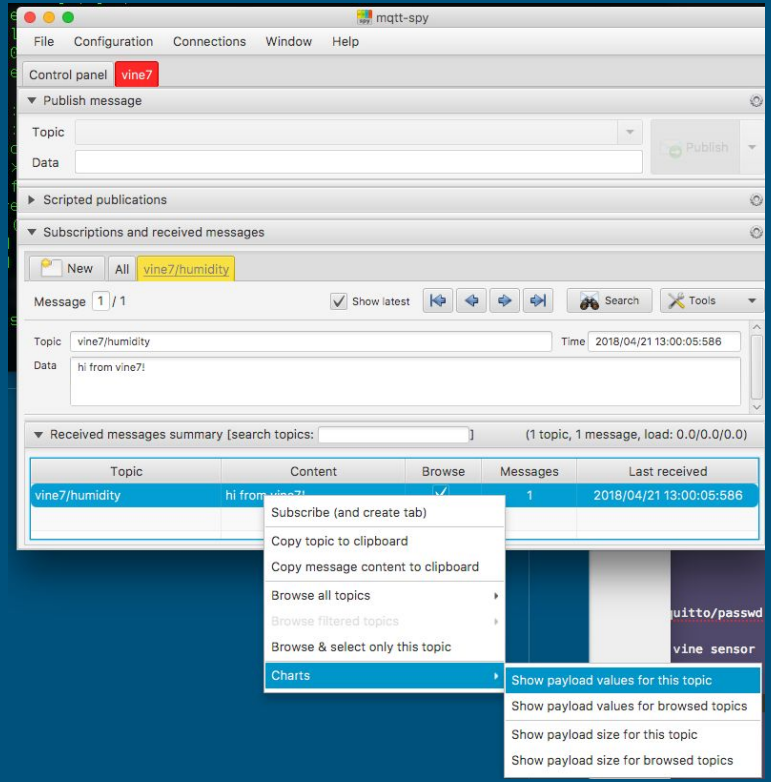
MQTT Spy - Subscribe



MQTT Spy - Receive messages

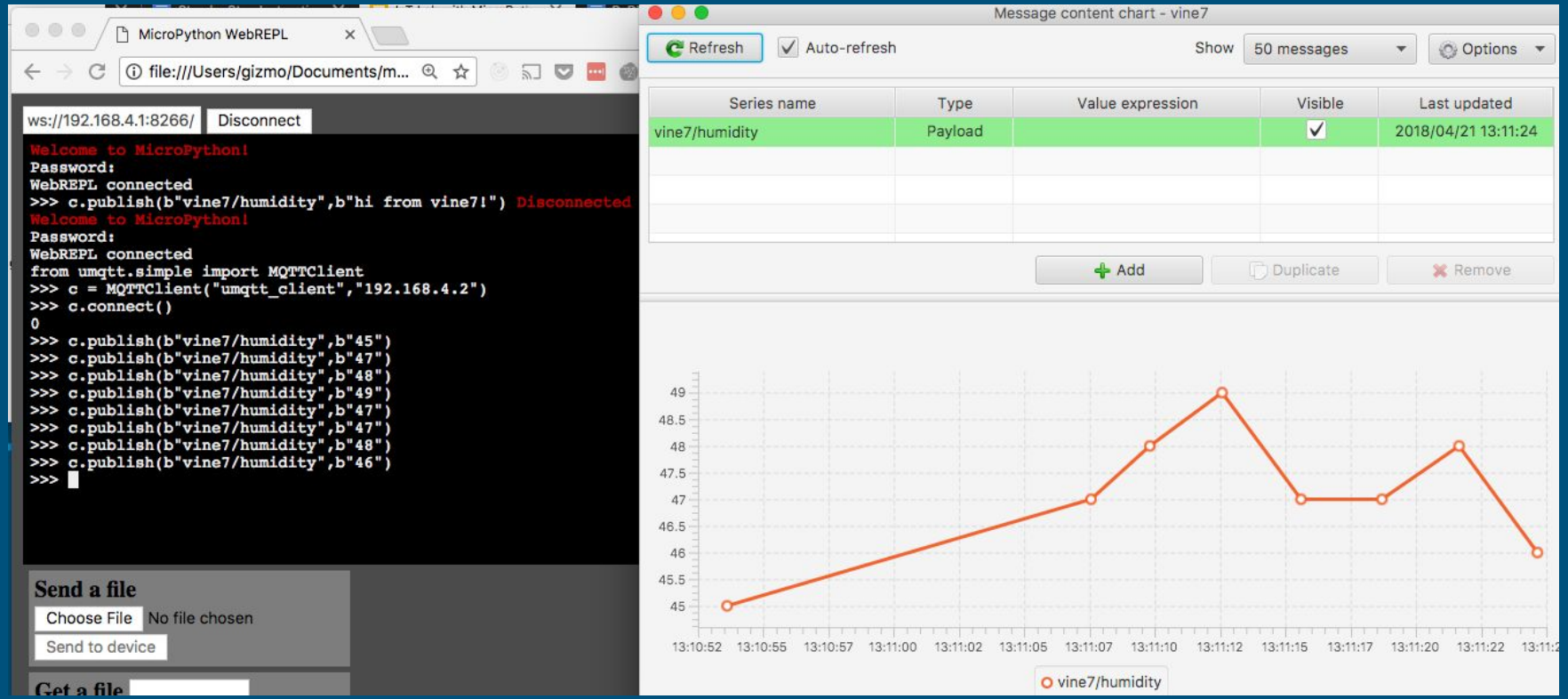


MQTT Spy - Chart messages



Right click on your topic

MQTT Spy - Subscription chart over time



Turn it around - publish to the vine sensor

Sign into WEBREPL and create a callback function:

```
def cb(topic, message):
...     print(topic + ": " + message)
```

Publish to the vine sensor continued

```
c.connect()

0

c.set_callback(cb)

c.subscribe(b"vine_sensor/#")

while 1:

...     c.wait_msg()
```

Publish to the vine sensor - computer side

In a terminal window send a message:

```
mosquitto_pub -h 192.168.4.3 -t vine_sensor/temp -m '15'
mosquitto_pub -h 192.168.4.3 -t vine_sensor/humidity -m '50'
```

Check the WEBREPL window & MQTT Spy to confirm you received the message

main.py updates

- DHT logging done
- Add MQTT publishing
  - Like mesaure_temp_humidity but without files. Keep the polling time
- Define the MQTTClient
  - c = MQTTClient("umqtt_client",'192.168.4.2')
- If your DHT isnt working:

```
import urandom
baseTemp = 28
def getTemp():
    return baseTemp + urandom.getrandbits(4)
```

MQTT client functions

```
1  import dht
2  import time
3  from umqtt.simple import MQTTClient
4
5  c = MQTTClient("umqtt_client",'192.168.4.2')
6  my_dht = dht.DHT11(machine.Pin(2))
7
8  def connect_mqtt():
9      try:
10         res = c.connect()
11         if res == 0:
12             return True
13     except Exception as ex:
14         print('unable to connect mqtt:', ex.message)
15         return False
16
17
18 def temp_humidity_mqtt(poll_time_s):
19     if connect_mqtt():
20         while True:
21             my_dht.measure()
22             temp = my_dht.temperature()
23             humidity = my_dht.humidity()
24             try:
25                 c.publish(b"/vine_sensor/temp",str(temp))
26                 c.publish(b"/vine_sensor/humidity",str(humidity))
27             except Exception as ex:
28                 print("unable to publish to MQTT:", ex.message)
29                 return
30             print("temp: ", temp)
31             print("humidity: ", humidity)
32             time.sleep(poll_time_s)
```

Transfer new main.py, reboot

Test it out, do things work as you expect?

Probably time for another break

Agenda

- Project & Hardware overview
- MicroPython introduction
- Communicating with MQTT
- Temperature scavenger hunt
- IoT security
- Extra: multi sensor networks

Sensor scavenger hunt!

Now that your vine sensor is setup to measure and record / relay information, go find some!

If you want you can:

- Share your logged data with the class
- Share a graph from MQTT-SPY
- For Android: IoT MQTT Dashboard
- Chrome extension: MQTT Lens (might work on phone?)

Agenda

- Project & Hardware overview
- MicroPython introduction
- Communicating with MQTT
- Temperature scavenger hunt
- IoT security
- Extra: multi sensor networks

IoT: the S is for Security

Source: the Internet

Security for MQTT

- Passwords
- Encryption
  - Transport Layer Security (TLS)
  - Certificates, private and public keys

Mosquitto and Micropython support both, but Micropython does not (yet) validate certificates

Mosquitto passwords

sudo mosquitto_passwd -c /path/to/etc/mosquitto/passwd <username>

Open the mosquitto conf:
/usr/local/Cellar/mosquitto/1.4.11_2/etc/mosquitto/mosquitto.conf

Add:

allow_anonymous false

password_file /usr/local/Cellar/mosquitto/1.4.11_2/etc/mosquitto/

Sub/Pub with passwords

> mosquitto_sub -h localhost -t /vine_sensor/#
Connection Refused: not authorised.

> mosquitto_sub -h localhost -t /vine_sensor/# -u <username> -P <password>

> mosquitto_pub -h localhost -t /vine_sensor/message -m “testing” -u <username> -P <password>

Lets add MQTT-SPY

- Try to reconnect to localhost, what happens?
  - Look in the mqtt-spy.log file: esp8266_micropython_lab/tools/mqtt-spy.log

- Enable authentication
- Subscribe to /vine_sensor/#

MQTT passwords in Micropython

```
class MQTTClient:

    def __init__(self, client_id, server, port=0, user=None, password=None, keepalive=0,
                 ssl=False, ssl_params={}):
```

Securing the vine sensor client

```
from umqtt.simple import MQTTClient
c_sec = MQTTClient("umqtt_client",'192.168.4.3', user="user",
password="passwd")
c_sec.connect()

c_sec.publish(b"/vine_sensor/temp",b"105")
```

Remember we have temp_humidity_mqtt(poll_time) - set c = c_sec and run

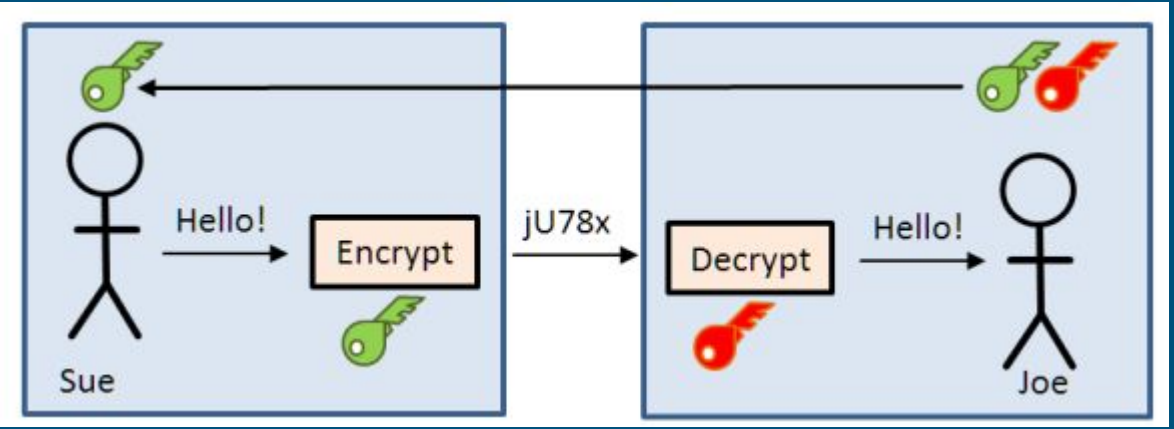
So, we have some passwords

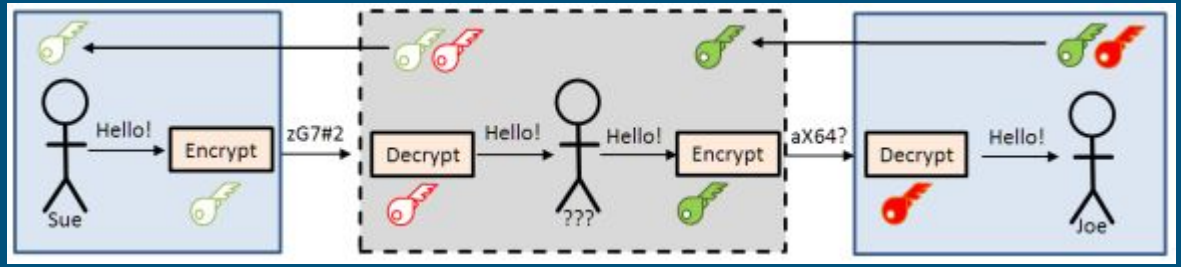
Passwords are helpful, but we are sharing them across an open wifi network

TLS / SSL - encrypting communications

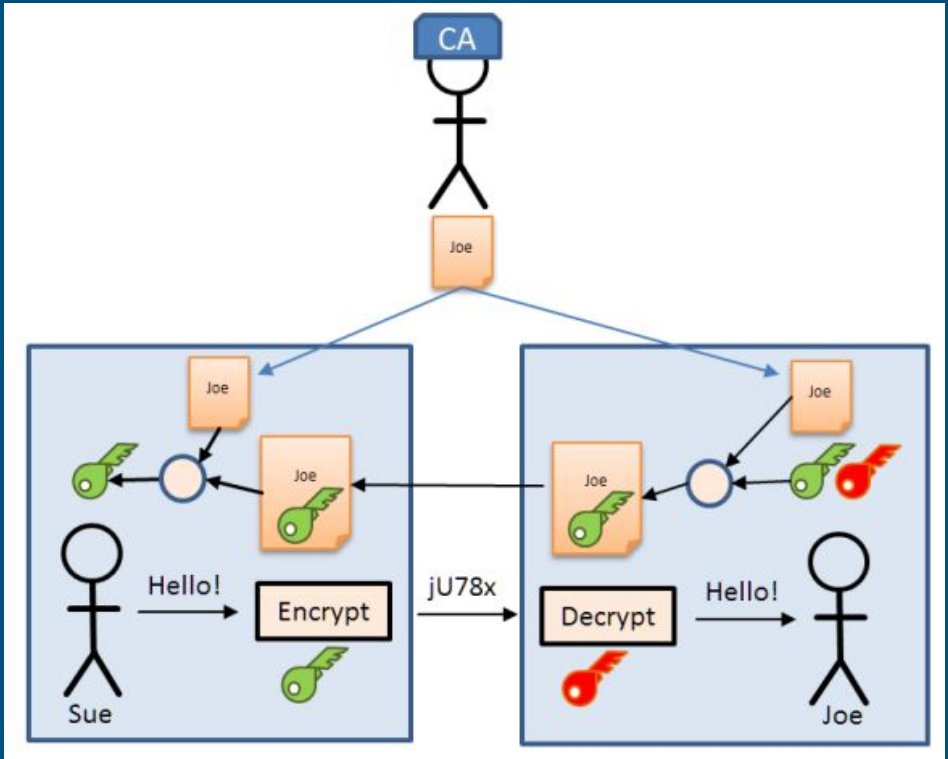
- A protocol for providing secure communications
- Certificates used to verify public key owner
- A Certificate Authority (CA) verifies certificate authenticity

https://mcuoneclipse.com/2017/04/14/introduction-to-security-and-tls-transport-security-layer/





https://mcuoneclipse.com/2017/04/14/introduction-to-security-and-tls-transport-security-layer/



https://mcuoneclipse.com/2017/04/14/introduction-to-security-and-tls-transport-security-layer/

IRL, do not be your own CA

Get cert from a reputable CA (Certificate Authority)

- https://letsencrypt.org - CA
  - https://certbot.eff.org/ - CA approved cert provider

Generating a certificate

generate-CA.sh

ca.key, ca.crt, ca.srl

<hostname>.key, csr, crt

Copy hostname* and ca.crt to /.../mosquitto/certs

Using our certs and keys with mosquitto

Update mosquitto.conf:

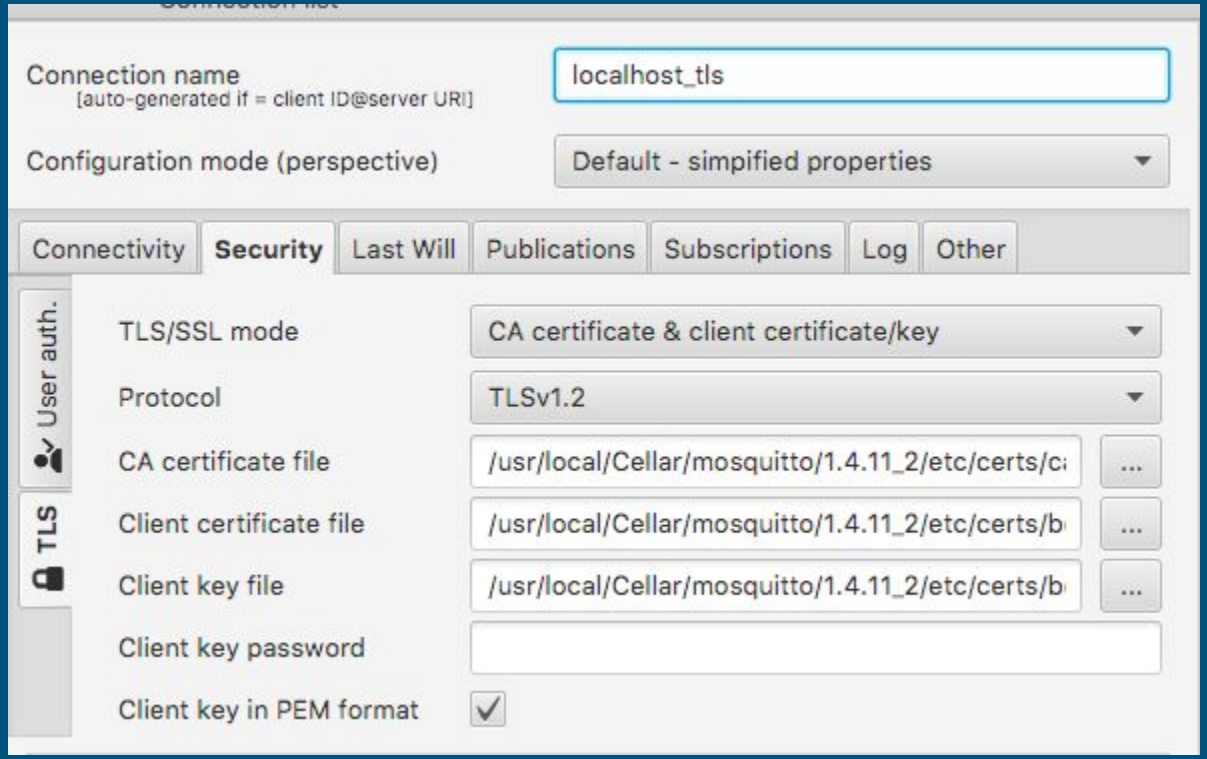
```
#TLS MQTT port
listener 8883 localhost

cafile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/ca.crt
certfile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/beaker.local.crt
keyfile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/beaker.local.key
```

Pass messages on localhost

mosquitto_sub -h localhost -t test_topic -u "user" -P "passwd" --cafile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/ca.crt -p 8883

mosquitto_pub -h 127.0.0.1 -t test_topic -m 'secure message' -u "user" -P "passwd" --cafile /usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/ca.crt -p 8883

MQTT-SPY


Connection name
[auto-generated if = client ID@server URI]
localhost_tls
Configuration mode (perspective)
Default - simpified properties
Connectivity
Security
Last Will
Publications
Subscriptions
Log
Other
User auth.
TLS
TLS/SSL mode
CA certificate & client certificate/key
Protocol
TLSv1.2
CA certificate file
/usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/c
Client certificate file
/usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/b
Client key file
/usr/local/Cellar/mosquitto/1.4.11_2/etc/certs/b
Client key password
Client key in PEM format

IoT with some security

- We’ve learned how to secure MQTT connections with our vine sensor using passwords
- We’ve also learned how to implement TLS encryption, certificate verification TBD for Micropython

Agenda

- Project & Hardware overview
- MicroPython introduction
- Communicating with MQTT
- Temperature scavenger hunt
- IoT security
- Extra: multi sensor networks

Extra curricular...

- Working in pairs or small groups, create local sensor networks

Connect vine sensor to another AP

Example: Vine8 connecting to Vine9 WiFi

Vine8: connect to vine8 wifi and start webrepl. Then join the vine9 network:

```
>>> import network
>>> sta_if = network.WLAN(network.STA_IF)
>>> sta_if.active(True)
```

The next command may/will freeze your webrepl:

```
>>> sta_if.connect('vine9','micropythoN')
```

Vine8: find out your IP on the WiFi

At this point, you may need Vine9 and Vine8 to reboot.

Log back into vine8 network and start webrepl

```
>>> import network

>>> sta_if = network.WLAN(network.STA_IF)

>>> sta_if.ifconfig()

('192.168.4.2', '255.255.255.0', '192.168.4.1', '192.168.4.1')
```

Make note of the above IP address

Vine 8 - turn off AP

THIS WILL DISABLE THE VINE8 WIFI - MAKE SURE YOU KNOW THE IP!

```
>>> ap_if = network.WLAN(network.AP_IF)

>>> ap_if.active(False)
```

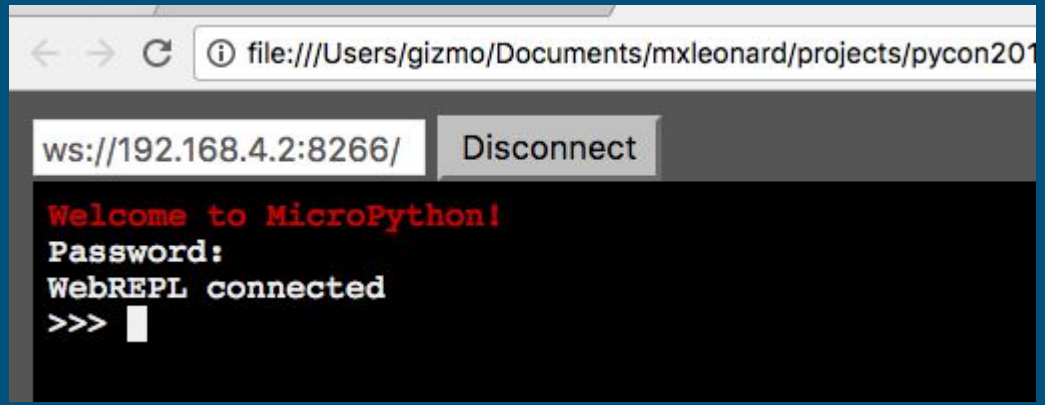
At this point, webrepl will be unresponsive and Vine8 wifi will be off

Getting to the vine8 webrepl from vine9

Disconnect webrepl session on vine8

Log into the vine9 network from your computer

In webrepl, put your vine9 IP in the connection string:



Vine8 sensor accessed via vine9 wifi

Try defining an LED and toggling the value, or taking a measurement from your DHT. Confirm that light blinks on the vine8 sensor, not vine9 :)

To restart the vine8 wifi:

```
>>> ap_if = network.WLAN(network.AP_IF)
```

```
>>> ap_if.active(True)
```

The webrepl will freeze, you will again see vine8 as a WiFi network

So now you are on the same network

With both vines on the same wireless network, you can setup MQTT pub/sub messages

The vine9 user can start mosquitto, try sending messages back and forth

Tutorial summary

WHEW! We covered a lot today, you learned how to:

- Program the ESP8266 using micropython
- Use the MQTT protocol to communicate among IoT networks
- Enable secure communications through TLS
- Hopefully had some fun!

Sources and further reading

http://micropython-on-esp8266-workshop.readthedocs.io

http://micropython-iot-hackathon.readthedocs.io

https://github.com/micropython/micropython-lib/tree/master/umqtt.simple

https://github.com/eclipse/paho.mqtt.python

https://mcuoneclipse.com/2017/04/14/introduction-to-security-and-tls-transport-security-layer/

Thanks!

If you want to keep your board come see me, $12

$SevLeonard

Tutorial survey: www.surveymonkey.com/r/pycon138

Backup

Copying files to/from the board

Over wifi:

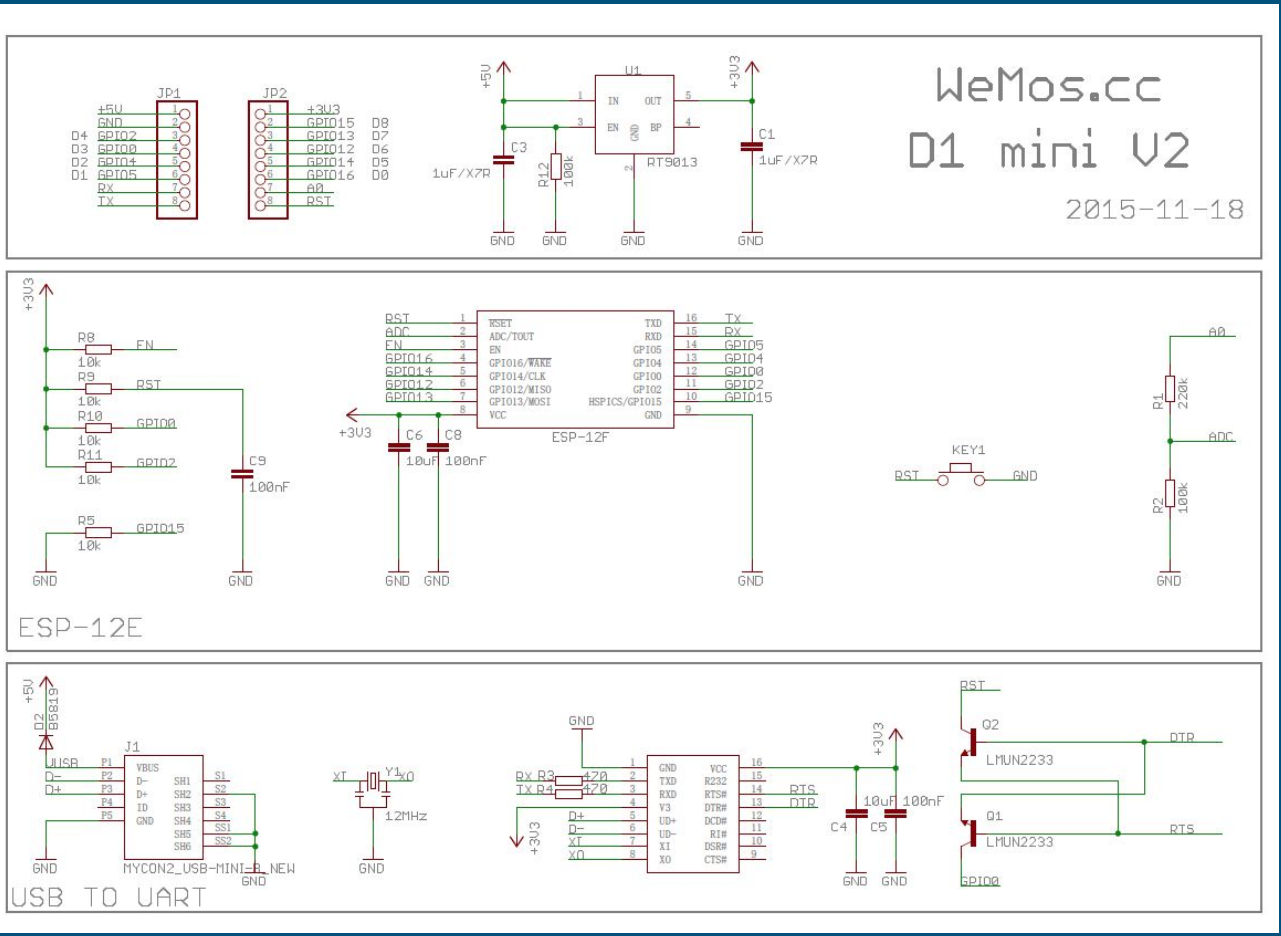
- webrepl_cli.py main.py 192.168.4.1:/main.py
- webrepl_cli.py 192.168.4.1:/main.py main.py

Over USB via AdaFruit ampy:
https://learn.adafruit.com/micropython-basics-load-files-and-run-code/file-operations

- ampy --port /dev/tty/wchusbserial00… --baud 115200 -put main.py
- ampy --port /dev/tty/wchusbserial00… --baud 115200 -get main.py

Getting networks & webrepl setup

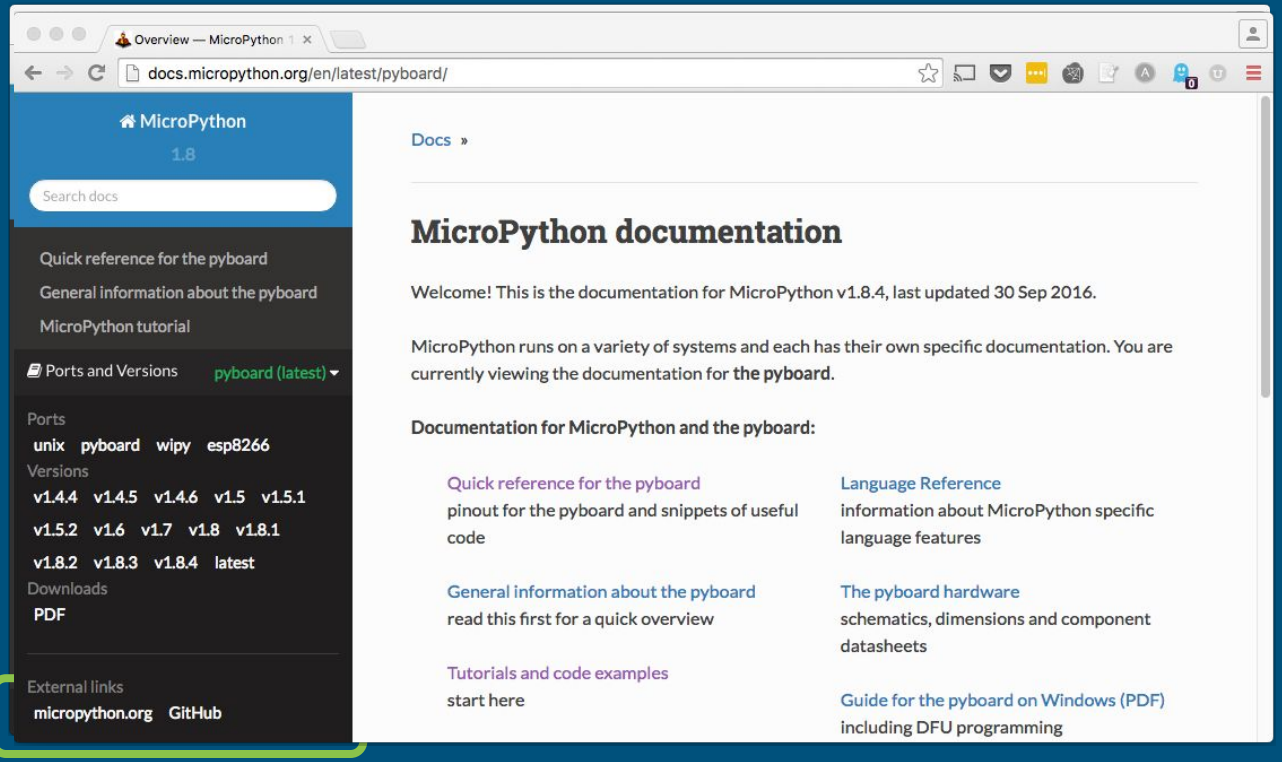
```
import network
ap_if = network.WLAN(network.AP_IF)

ap_if.config(essid='vine50')
ap_if.active(True)

import webrepl
webrepl.start()
```



https://www.wemos.cc/sites/default/files/2016-09/mini_new_V2.pdf

What boards can I use?

http://docs.micropython.org/



How do we teach the board micropython?

micropython.org/download

github.com/themadinventor/esptool/

pip install esptool

Flashing the board

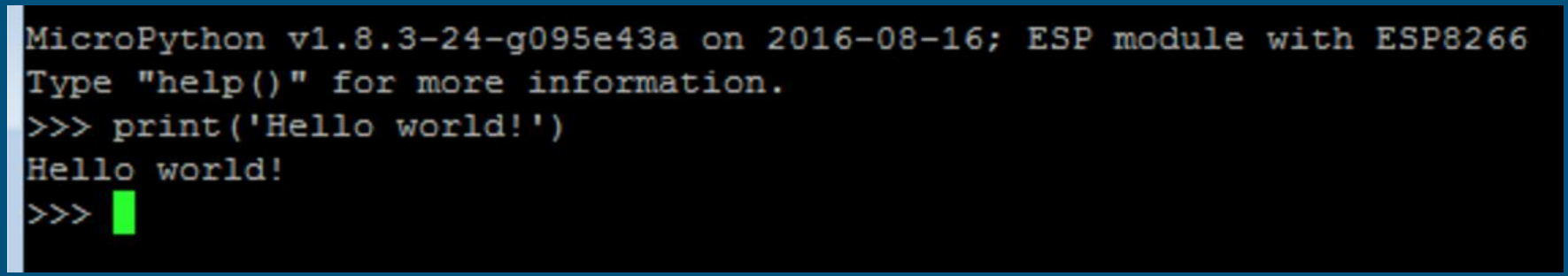
```
esptool.py --port /dev/ttyUSB0 erase_flash
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_size=8m 0
esp8266-2016-05-03-v1.8.bin
 "A fatal error occurred: Failed to connect to ESP8266"
```

Unplugging/replugging in the ESP8266 seemed to fix the problem

REPL time!

REPL - read, evaluate, print loop. In other words; a command line shell

> screen /dev/ttyUSB0 115200



Notable Mentions

- Check your wires!!!!!! A multimeter is your friend
- Steal liberally, but attribute!
- Try a development board, like the pyboard or the Adafruit Feather HUZZAH ESP8266
- Have you tried turning it on and off again?