# Micropython for Mews

Sev Leonard
PyDX 2016

# Acknowledgements


- Joe Fitzpatrick @securelyfitz
- micropython.org
- Countless githubs and blogs
  - github.com/mcauser/MicroPython-ESP8266-Nokia-5110-Conways-Game-of-Life
  - github.com/garybake/upython_wemos_shields

# What is Micropython?

A. Like regular Python, but small and hard to read ✖
B. A version of Python optimized for use on microcontrollers
C. Funded via Kickstarter
D. All of the above
E. Some of the above

E. Some of  the above!

micropython.org

# Some things about hardware

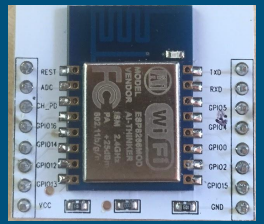
- It’s hard
- Microfractures are a real pain
- Costs $$
- May result in hair loss
- May also result in delighted squeals


pdxhackerspace.org

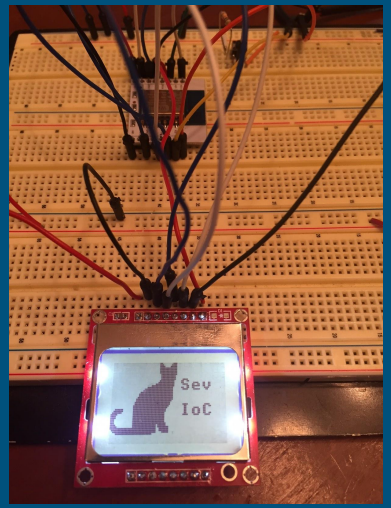
# The Internet of Cats

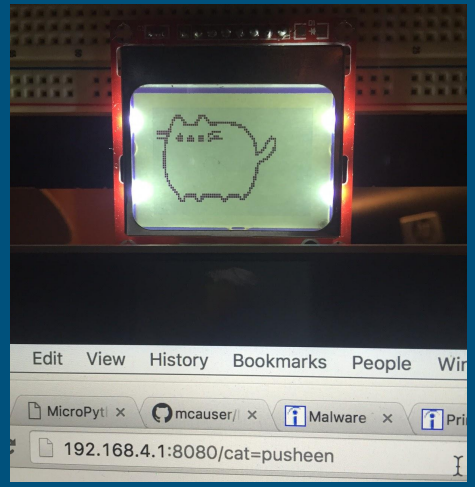
Serving up 1-bit cat pics since 2016

ESP8266



Nokia 5110



# Material costs

- ESP8266: $2 - $6
- Nokia 5110 LCD: $3
- Multimeter: $15 from Sparkfun
- FTDI USB to TTL serial cable: $2 - $10
- Jumper wires: $6

$30 - $40

# The ESP8266

- 3.3V supply
- May require extra ‘juice’ beyond what your USB port can deliver
- Limited to ~25Kb memory
- There are many configurations of the ESP8266!

ESP-12

ESP-01

# What boards can I use?

http://docs.micropython.org/

Overview — MicroPython

docs.micropython.org/en/latest/pyboard/

MicroPython
1.8

Search docs

Quick reference for the pyboard
General information about the pyboard
MicroPython tutorial

Ports and Versions pyboard (latest)

Ports
unix pyboard wipy esp8266
Versions
v1.4.4 v1.4.5 v1.4.6 v1.5 v1.5.1
v1.5.2 v1.6 v1.7 v1.8 v1.8.1
v1.8.2 v1.8.3 v1.8.4 latest
Downloads
PDF

External links
micropython.org GitHub

Docs »

## MicroPython documentation

Welcome! This is the documentation for MicroPython v1.8.4, last updated 30 Sep 2016.

MicroPython runs on a variety of systems and each has their own specific documentation. You are currently viewing the documentation for **the pyboard**.

**Documentation for MicroPython and the pyboard:**

Quick reference for the pyboard
pinout for the pyboard and snippets of useful code

General information about the pyboard
read this first for a quick overview

Tutorials and code examples
start here

Language Reference
information about MicroPython specific language features

The pyboard hardware
schematics, dimensions and component datasheets

Guide for the pyboard on Windows (PDF)
including DFU programming

# How do we teach the board micropython?

micropython.org/download

## MicroPython downloads

For the MicroPython source code, please visit github.com/micropython/micropython.

Daily dumps of the GitHub repository are available from this server:

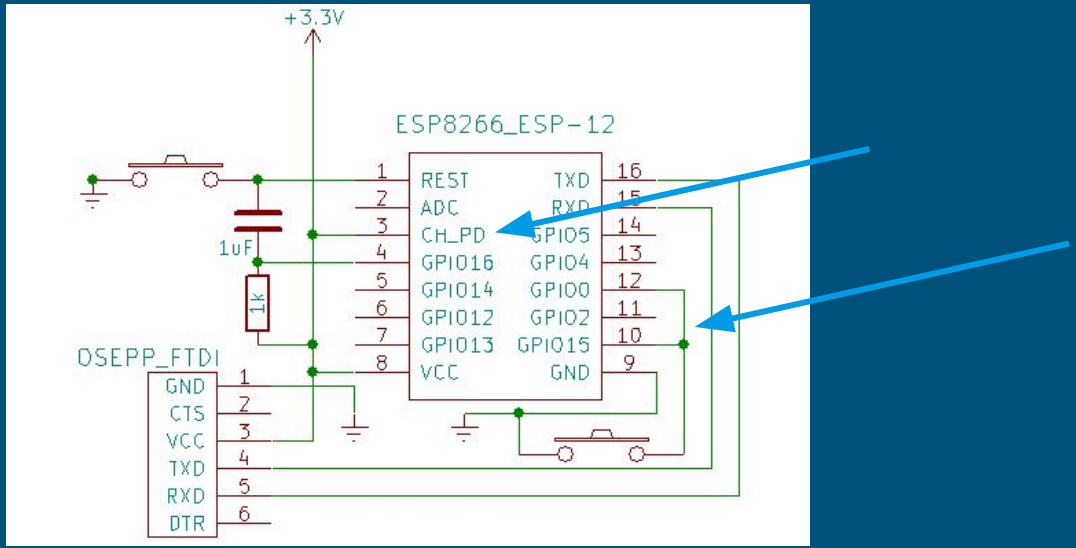
- micropython-master.zip
- pyboard-master.zip

Links to firmware below: pyboard WiPy ESP8266 other

github.com/themadinventor/esptool/

pip install esptool

# Loading the firmware




Source: http://www.agcross.com/2015/09/the-esp8266-wifi-chip-part-3-flashing-custom-firmware/

# 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

```
MicroPython v1.8.3-24-g095e43a on 2016-08-16; ESP module with ESP8266
Type "help()" for more information.
>>> print('Hello world!')
Hello world!
>>>
```

# 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)

*Terminal widget should be focused (text cursor visible) to accept input. Click on it if not.*

# Web REPL tips

- Enable in boot.py if not memory constrained
- Make sure you can see the wifi network!

# Collecting garbage

```
from machine import Pin, HSPI

import gc

import upcd8544

gc.collect()

import framebuf

import math

gc.collect()
```

# Connecting the Nokia 5110 LCD

| WeMos D1 Mini (ESP8266) | Nokia 5110 PCD8544 LCD | Description |
|---|---|---|
| D2 (GPIO4) | 0 RST | Output from ESP to reset display |
| D1 (GPIO5) | 1 CE | Output from ESP to chip select/enable display |
| D6 (GPIO12) | 2 DC | Output from display data/command to ESP |
| D7 (GPIO13) | 3 Din | Output from ESP SPI MOSI to display data input |
| D5 (GPIO14) | 4 Clk | Output from ESP SPI clock |
| 3V3 | 5 Vcc | 3.3V from ESP to display |
| D0 (GPIO16) | 6 BL | 3.3V to turn backlight on, or PWM |
| G | 7 Gnd | Ground |

github.com/mcauser/MicroPython-ESP8266-Nokia-5110-Conways-Game-of-Life

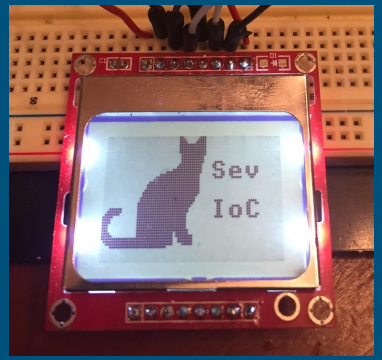
# Setting up the Nokia 5110 LCD - setup_lcd.py

```
spi = HSPI(baudrate=80000000, polarity=0, phase=0)
RST = Pin(4)
CE = Pin(5)
DC = Pin(12)
BL = Pin(16)
lcd = upcd8544.PCD8544(spi, RST, CE, DC, BL)

width = 84
height = 48
pages = height // 8
buffer = bytearray(pages * width)
framebuf1 = framebuf.FrameBuffer1(buffer, width, height)
```

# Writing to the LCD

- Clear the display: framebuf1.fill(0)
- Draw a pixel: framebuf1.pixel(x,y,1)
- Write some text: framebuf1.text(“Hello!”, x, y, col)

  lcd.data(buffer)



# Let’s get some cats!

How do we draw cats on a 48x84 pixel monochrome screen?

1. Find a cat
   - Converts easily to back and white
   - Scales to something that fits in 48x84
   - Not super detailed

# Create a cat image with ImageMagick

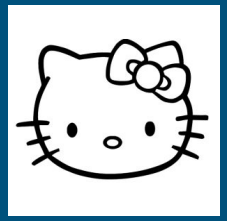
1. Get our cat onto a white background

```
convert hello_kitty.png  -background white -alpha remove
hello_kitty_white.png
```

2. Scale the cat

Original size: 503/503 ~ → 48/48

```
convert hello_kitty_white.png -resize 48x48
small_cat_paws.png
```



# Convert the cat to a bitmap!

Using convert_png.py based on github.com/garybake/upython_wemos_shields

```
python convert_png.py small_cat_paws.png
```

```
111111111111111111111111101111111111111111111111
111111111111111111111111000001111100000111111111
111111110000011111111111001100000000000011111111
111111100000000011111000001111000011111001111111
111111100111100000000000011111100111111001111111
111111101111111000111100111000000111110011111111
111111101111111111111100110000000000100001111111
111111101111111111111100110000111000000000111111
111111100111111111111101111000111100111100111111
111111100011111111111100111101111000111100111111
111111100011111111111000000000111100011100111111
111111100111111111111110001100110000111100111111
111111100111111111111111111110000000111001111111
111111001111111111111111111111111001110001111111
111111001111111111111111111111111000000001111111
111111011111111111111111111111111111001100111111
111111011111111111111111111111111111111100111111
111111001111111111111111111111111111111100111111
111111001111111111111111111111111111110000000011
110000000111111111111111111111111001111100000000111
110000000111100011111111111111111000111111001111111
111111001111100011111111111111111000111111000111111
111111001111100011111111011111111001111111000001111
111111000001111111111000001111111111111000001111
111100000111111111111001001111111111111001111111
111111100111111111111100011111111111100001111111
111111110000111111111111111111111111110000111111
111111100001111111111111111111111111100000011111
111111000100011111111111111111111111100011111111
111111001111000111111111111111111100000111111111
111111111111000000111111111100000000111111111111
111111111111100000000000000000000011111111111111
```

# Image drawing - draw_image_from_text.py

```
for line in f:
    for char in line:
        if char == '0':
            framebuf1.pixel(y,x,1)
            gc.collect()
lcd.data(buffer)
gc.collect()
```

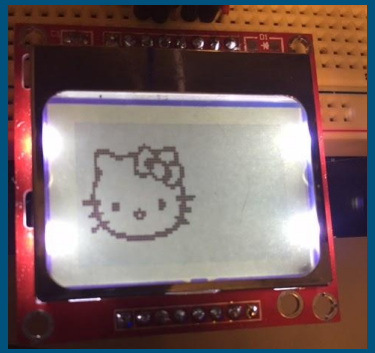
# Framebuf interface reference

github.com/micropython/micropython/blob/master/drivers/display/ssd1306.py

```
def fill(self, col):
    self.framebuf.fill(col)

def pixel(self, x, y, col):
    self.framebuf.pixel(x, y, col)

def scroll(self, dx, dy):
    self.framebuf.scroll(dx, dy)

def text(self, string, x, y, col=1):
    self.framebuf.text(string, x, y, col)
```

# Lets draw a cat!

```
from setup_lcd import *

from draw_image_from_text import *

draw_image('hello_kitty.txt', framebuf1, buffer, lcd)
```



# Enable drawing on boot via boot.py

```
import gc

gc.collect()

from setup_lcd import *

from draw_image_from_file import *

gc.collect()

import webrepl

webrepl.start()

gc.collect()
```

# Potential boot.py weirdness

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "http_server.py", line 27, in main
```

# And now.. The INTERNET!

Setup an HTTP server to serve up our cats, enabling control of the LCD from THE INTERNET*

* As long as you are logged into the ESP8266

** Or if the ESP8266 itself is connected to the Internet

github.com/micropython/micropython/blob/master/examples/network/http_server_simplistic_commented.py

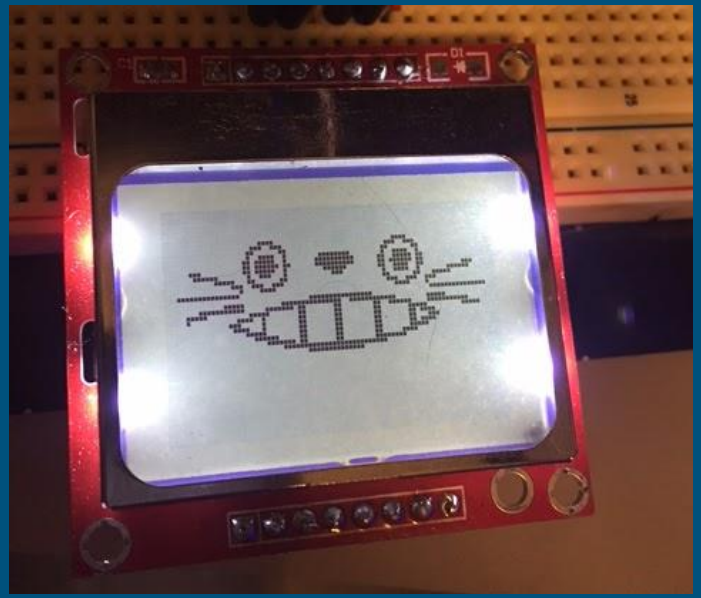
# Processing the cat request

```
req = client_stream.readline()

req = str(req)

print(req)

if req.find('GET /cat=sitting') > 0:

   lcd_cat = 'sitting_cat.txt'

draw_image(lcd_cat, framebuf1, buffer, lcd)
```

# I can haz demo?

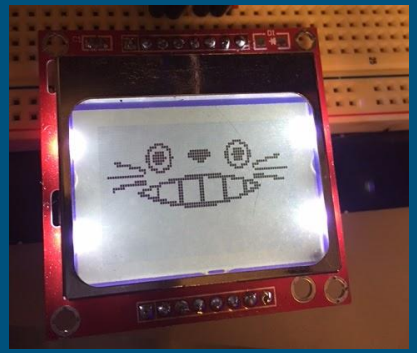
# If not...

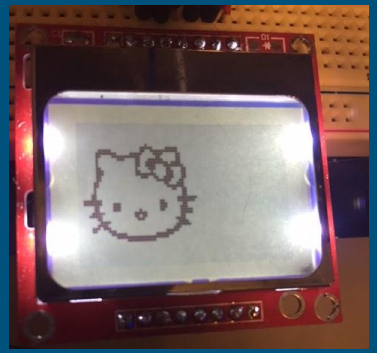
http://192.168.4.1:8080/cat=totoro

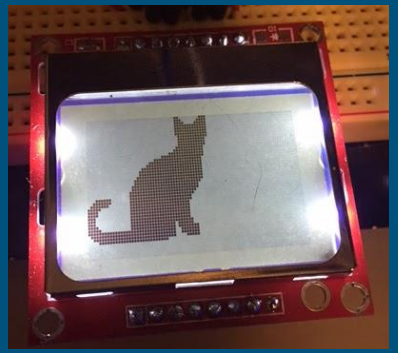


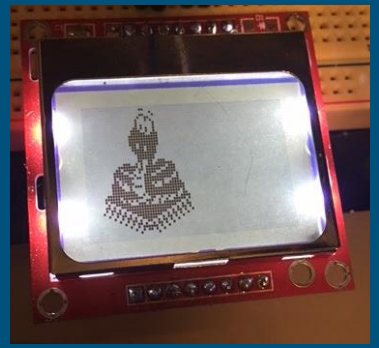
* technically not a cat

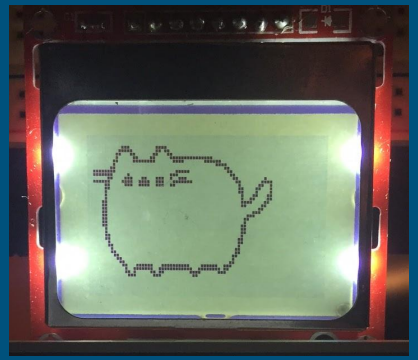
# Moar cats and friends of cats!











# 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?

# Thanks!

github.com/gizm00/pydx_upython

^ watch this space! ^

@gizm0_0

sev@thedatascout.com