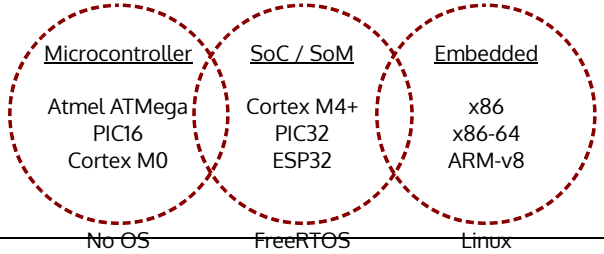
Easy IoT
with MicroPython
on ESP SoCs

Nick Moore
@mnemote
nick@mnemote.com

ESP8266 / ESP32 SoCs

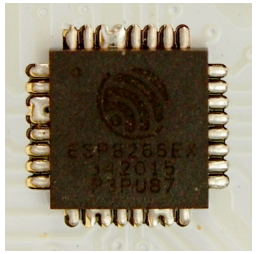
What is a SoC

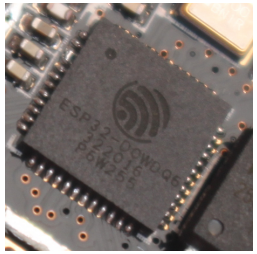
System on a Chip, or System on a Module



ESP8266 / ESP32

ESPRESSIF



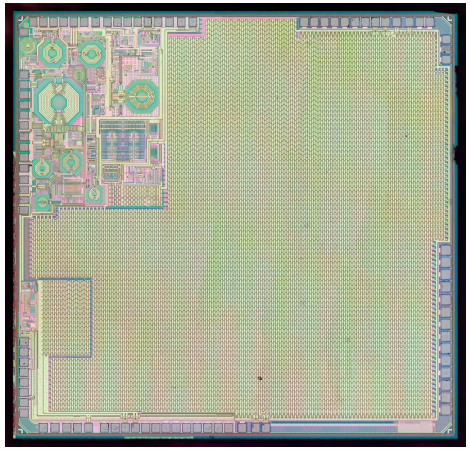


ESP8266 CPU

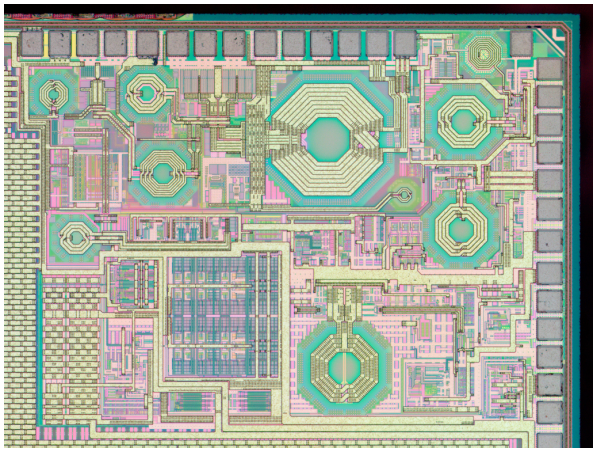
Arduino	ESP8266
AVR ATMega328P	**Tensilica Xtensa LX106**
8 bit	**32 bit**
1 core	**1 core**
20 MHz	**80/160 MHz**
2 KB RAM	**160 KB RAM**
32 KB Flash	**4 MB Flash**

ESP32 CPU

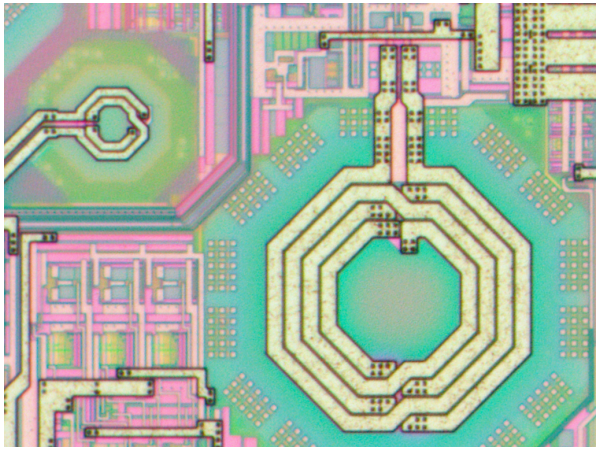
Arduino	ESP8266	ESP32
AVR ATMega328P	Tensilica Xtensa LX106	**Tensilica Xtensa LX6**
8 bit	32 bit	**32 bit**
1 core	1 core	**2 core ***
20 MHz	80/160 MHz	**160/240 MHz**
2 KB RAM	160 KB RAM	**520 KB RAM**
32 KB Flash	1 - 4 MB Flash	**4 - 16 MB Flash**



original image: zeptobars.com



original image: zeptobars.com



original image: zeptobars.com

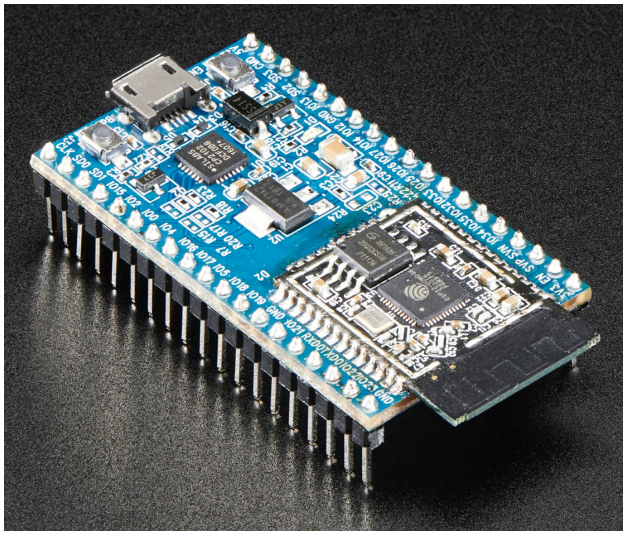
ESP32 Modules

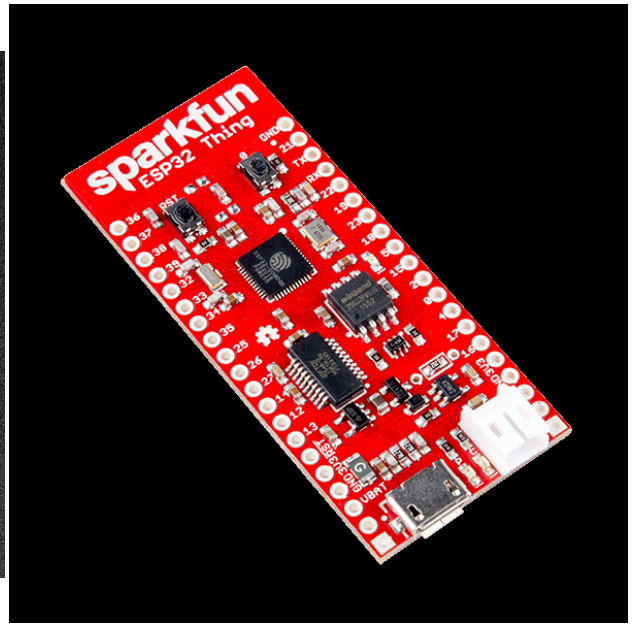
WROOM-32 or ESP-32S

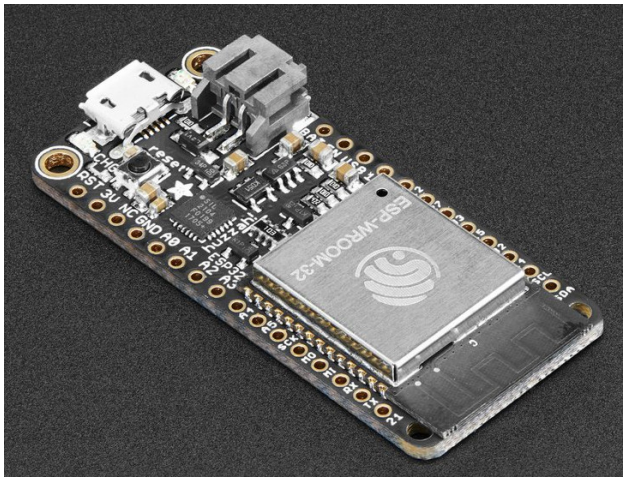
original image: espressif.com

ESP32 Boards

ESP32-DevKitC / Sparkfun ESP32 Thing / AdaFruit HUZZAH32







original images: espressif.com and sparkfun.com and adafruit.com

ESP32 Networking

- WiFi 802.11 b/g/n
- Bluetooth
- Bluetooth LE
- Ethernet MAC
- UART / I2C / I2S / SPI

ESP32 I/O

- Digital I/O
- ADC inputs
- DAC outputs
- PWM outputs
- Capacitive Touch Inputs
- Hall Effect Sensor
- RMT "Remote Control" module

ESP IDF

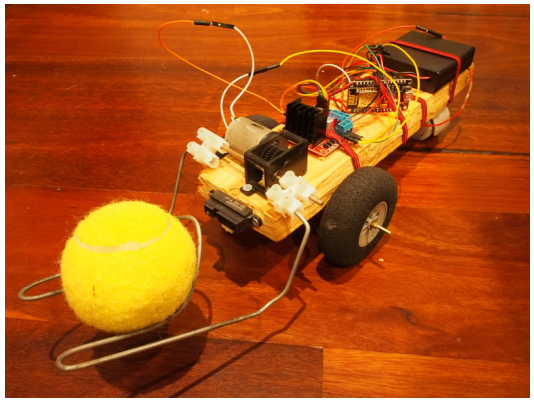
IoT Development Framework

C SDK for building programs to run on ESP32

Contains LWIP, examples, instructions for building toolchain

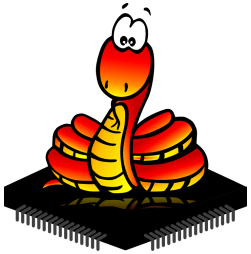
- github: espressif/esp-idf

The Internet of Toys — 1




BuzzConf

MicroPython



micropython.org

MicroPython

- Python 3.4
- C99
- MIT-licensed

MicroPython Platforms

- PyBoard (ARM Cortex)
- 16-bit PIC
- ESP8266
- **ESP32**

For more background, see
Damien's talk about the Kickstarter campaign
from PyConAU 2016

MicroPython for ESP32

github: micropython/micropython-esp32

```
$ git clone --recursive https://github.com/micropython/micropython-esp32.git
$ more micropython-esp32/ports/esp32/README.md
```

OR

http://micropython.org/downloads#esp32

```
$ pip install esptool
$ esptool.py write_flash --flash_mode dio 0x1000 esp32-20170919-v1.9.2-272-g0d183d7f.bin
```

Running MicroPython — 1

```
$ miniterm.py /dev/ttyUSB0 115200 --raw
MicroPython v1.9.2-272-g0d183d7f on 2017-09-19;
ESP32 module with ESP32
Type "help()" for more information.
>>>
```

Running MicroPython — 2

```
$ miniterm.py /dev/ttyUSB0 115200 --raw
MicroPython v1.9.2-272-g0d183d7f on 2017-09-19;
ESP32 module with ESP32
Type "help()" for more information.
>>> help()
Welcome to MicroPython on the ESP32!
```

(... etc ...)

Running MicroPython — 3

```
>>> help('modules')
__main__          flashbdev         random            uos
_boot             framebuf          re                upip
_onewire          gc                select            upip_utarfile
_thread           hashlib           socket            urandom
apa106            heapq             ssl               ure
array             inisetup          struct            uselect
binascii          io                sys               usocket
btree             json              time              ussl
builtins          machine           ubinascii         ustruct
cmath             math              ucollections      utime
collections       micropython       uctypes           utimeq
dht               neopixel          uerrno            uzlib
ds18x20           network           uheapq            zlib
errno             onewire           uio
esp               os                ujson
```

Running MicroPython — 4

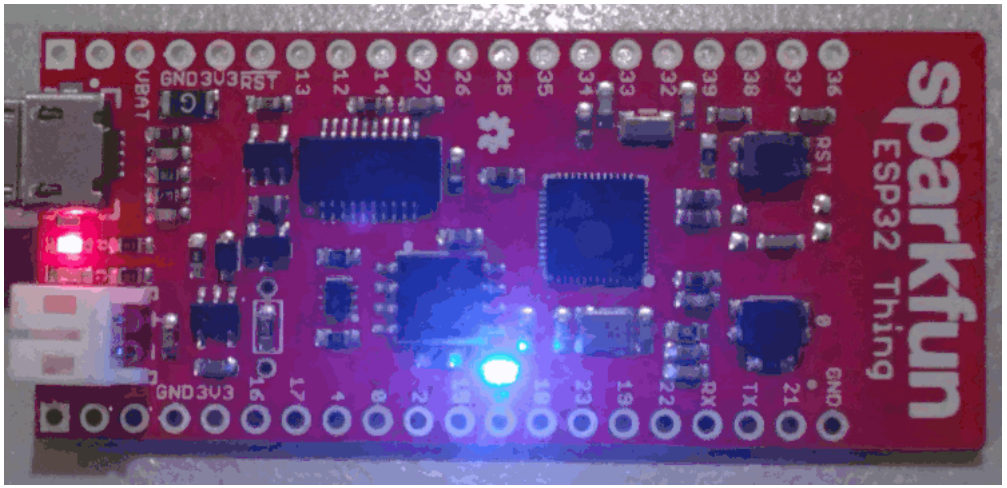
```
$ miniterm.py /dev/ttyUSB0 115200 --raw
MicroPython v1.9.2-272-g0d183d7f on 2017-09-19;
ESP32 module with ESP32
Type "help()" for more information.
>>> import machine
>>> pin = machine.Pin(5, machine.Pin.OUT)
>>> pin.value(True)
```

Running MicroPython — 5

```
>>> import machine
>>> import time
>>> pin = machine.Pin(5, machine.Pin.OUT)
>>> while True:
...     pin.value(True)
...     time.sleep(1)
...     pin.value(False)
...     time.sleep(1)
```

Running MicroPython — 6

```
>>> import machine
>>> import time
>>> pin = machine.Pin(5, machine.Pin.OUT)
>>> while True:
...     pin.value(True)
...     time.sleep(1)
...     pin.value(False)
...     time.sleep(1)
```



Running MicroPython — 7

With thanks to apple //jse

Programming MicroPython — mpy-utils

github: nickzoic/mpy-utils

- File transfer
- FUSE mount
- Diff

Using REPL as a file transfer protocol: slow and very alpha

Programming MicroPython — modules/ and scripts/

- `modules/` : pre-compiled and frozen into the runtime
- `scripts/` : source frozen into the runtime

Also **upip**, a micropython package manager

Programming MicroPython — Filesystem

```
>>> import os
>>> os.listdir()
['boot.py']
>>> f = open('hello.world', "w")
>>> f.write("Hello, World!")
13
>>> f.close()
>>> os.listdir()
['boot.py', 'hello.world']
```

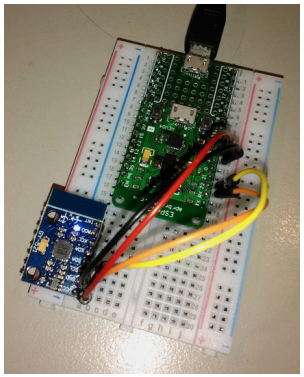
Programming MicroPython — Networking — 1

```
>>> import network
>>> w = network.WLAN(network.STA_IF)
>>> w.active(True)
>>> w.connect("SSID","PASSWORD")
>>> w.ifconfig()
('192.168.1.101', '255.255.255.0', '192.168.1.1',
'192.168.1.1')
```

Programming MicroPython — Networking — 2

```
>>> import socket
>>> s = socket.socket()
>>> s.connect(('example.com', 80))
>>> s.write('GET / HTTP/1.0\r\n')
16
>>> s.write('Host: example.com\r\n\r\n')
21
>>> s.readline()
b'HTTP/1.0 200 OK\r\n'
>>> s.close()
```

Programming MicroPython — I2C — 1



Programming MicroPython — I2C — 2

```
>>> import machine
>>> i2c = machine.I2C(
...   freq=400000,
...   scl=machine.Pin(22),
...   sda=machine.Pin(21)
... )
>>> i2c.scan()
[104]
```

Programming MicroPython — I2C — 3

```
>>> import machine
>>> i2c = machine.I2C(
...   freq=400000,
...   scl=machine.Pin(22),
...   sda=machine.Pin(21)
... )
>>> i2c.readfrom_mem(104, 59, 14)
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

Programming MicroPython — I2C — 4

```
>>> i2c.writeto_mem(104, 107, bytes([0]))
>>> i2c.readfrom_mem(104, 59, 14)
b'\xf3\x90\x00\x0cB\xa0\xf6\x10\xfd\x1d\x00\xa4\xff\x9d'
```

Programming MicroPython — I2C — 5

```
>>> import struct
>>> struct.unpack(
...   ">7h",
...   i2c.readfrom_mem(104, 59, 14)
... )
(-3016, 32, 17024, -2768, -758, 224, -133)
```

Programming MicroPython — I2C — 6

```
class Accelerometer:
    def __init__(self, i2c=None, address=104):
        self.i2c = i2c or machine.I2C(freq=400000,
                scl=machine.Pin(22), sda=machine.Pin(21))
        self.address = address
        self.i2c.writeto_mem(self.address, 107, bytes([0])

    def read_xyz(self):
        return struct.unpack(
            ">3h", self.i2c.readfrom_mem(104, 59, 6)
        )
```

Developing MicroPython

original image: iconsplace.com

How do C modules work? — 1

```
>>> import esp
>>> dir(esp)
['__name__', 'flash_read', 'flash_write', 'flash_erase',
'flash_size', 'flash_user_start', 'gpio_matrix_in',
'gpio_matrix_out', 'neopixel_write', 'dht_readinto']
>>> esp.flash_size()
4194304
```

How do C modules work? — 2

esp-idf/components/spi_flash/include/esp_spi_flash.h

```
/**
 * @brief  Get flash chip size, as set in binary image header
 *
 * @note This value does not necessarily match real flash size.
 *
 * @return size of flash chip, in bytes
 */
size_t spi_flash_get_chip_size();
```

How do C modules work? — 3

ports/esp32/modesp.c

```
#include "esp_spi_flash.h"

STATIC mp_obj_t esp_flash_size(void) {
    return mp_obj_new_int_from_uint(spi_flash_get_chip_size());
}
```

How do C modules work? — 4

ports/esp32/modesp.c

```
#include "esp_spi_flash.h"

STATIC mp_obj_t esp_flash_size(void) {
    return mp_obj_new_int_from_uint(spi_flash_get_chip_size());
}
```

How do C modules work? — 5

ports/esp32/modesp.c

```
#include "esp_spi_flash.h"

STATIC mp_obj_t esp_flash_size(void) {
    return mp_obj_new_int_from_uint(spi_flash_get_chip_size());
}
STATIC MP_DEFINE_CONST_FUN_OBJ_0(esp_flash_size_obj, esp_flash_size);
```

How do C modules work? — 6

ports/esp32/modesp.c

```
#include "esp_spi_flash.h"

STATIC mp_obj_t esp_flash_size(void) {
    return mp_obj_new_int_from_uint(spi_flash_get_chip_size());
}
STATIC MP_DEFINE_CONST_FUN_OBJ_0(esp_flash_size_obj, esp_flash_size);

STATIC const mp_rom_map_elem_t esp_module_globals_table[] = {
    { MP_ROM_QSTR(MP_QSTR___name__), MP_ROM_QSTR(MP_QSTR_esp) },
    { MP_ROM_QSTR(MP_QSTR_flash_size), MP_ROM_PTR(&esp_flash_size_obj) },
};
```

How do C modules work? — 7

ports/esp32/modesp.c

```
#include "esp_spi_flash.h"

STATIC mp_obj_t esp_flash_size(void) {
    return mp_obj_new_int_from_uint(spi_flash_get_chip_size());
}
STATIC MP_DEFINE_CONST_FUN_OBJ_0(esp_flash_size_obj, esp_flash_size);

STATIC const mp_rom_map_elem_t esp_module_globals_table[] = {
    { MP_ROM_QSTR(MP_QSTR___name__), MP_ROM_QSTR(MP_QSTR_esp) },
    { MP_ROM_QSTR(MP_QSTR_flash_size), MP_ROM_PTR(&esp_flash_size_obj) },
};
STATIC MP_DEFINE_CONST_DICT(esp_module_globals, esp_module_globals_table);

const mp_obj_module_t esp_module = {
    .base = { &mp_type_module },
    .globals = (mp_obj_dict_t*)&esp_module_globals,
};
```

How do C modules work? — 8

ports/esp32/mpconfigport.h

```
extern const struct _mp_obj_module_t esp_module;

#define MICROPY_PORT_BUILTIN_MODULES \
    { MP_OBJ_NEW_QSTR(MP_QSTR_esp), (mp_obj_t)&esp_module }, \
```

esp32/Makefile

```
SRC_C = \
        modesp.c \
```

```
>>> import esp
>>> dir(esp)
['__name__', 'flash_read', 'flash_write', 'flash_erase',
'flash_size', 'flash_user_start', 'gpio_matrix_in',
'gpio_matrix_out', 'neopixel_write', 'dht_readinto']
>>> esp.flash_size()
4194304
```

The State of the IoT

The State of the IoT

- New legal, ethical, etiquette challenges
- Many manufacturers and experimenters breaking new ground
- Market isn't really settled yet

... a bit like mobile phones in the 1990s

... or motorcars in the 1920s

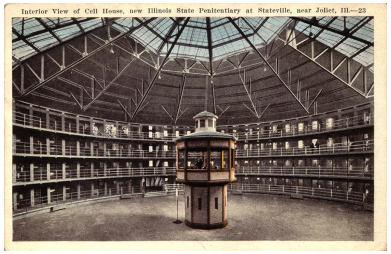
On the one hand
information wants to be expensive,
because it's so valuable.

The right information in the right place
just changes your life.

On the other hand,
information wants to be free,
because the cost of getting it out
is getting lower and lower all the time.

Stewart Brand, 1st Hackers Conference 1984

Panacea or Panopticon?



images: revistamundonatural.com / focault.info

The State of the IoT

- **Analysis** of the ways IoT is currently bad
- **Requirements** for it to be less bad
- **Solutions** which meet those requirements

- @internetofshit
- /r/theinternetofshit

1. If you're not the customer, you're the product.

- The backend services have to get paid for somehow.
- If you're not paying directly, you're paying indirectly:
  ... by accepting advertising
  ... by having your privacy sold.
- This has led to a conflation of IoT and Big Data

2. End Of Life

- Your hi-tech device may suddenly turn into a doorstop if the service provider no longer feels like supporting it.
- Not much cross-brand compatibility

arlogilbert.com

3. Internet Not Found

- Most devices useless if the Internet isn't available.
- Even the most reliable networks sometimes go down.
- Consumer-grade routers fail frequently.

telegraph.co.uk

4. Cryptography

- Crypto support on IoT devices is often very weak.
- Partly because of lack of entropy, CPU, RAM.
- It's tricky to use a protocol like SSL on a tiny CPU.
- (... but getting easier ...)

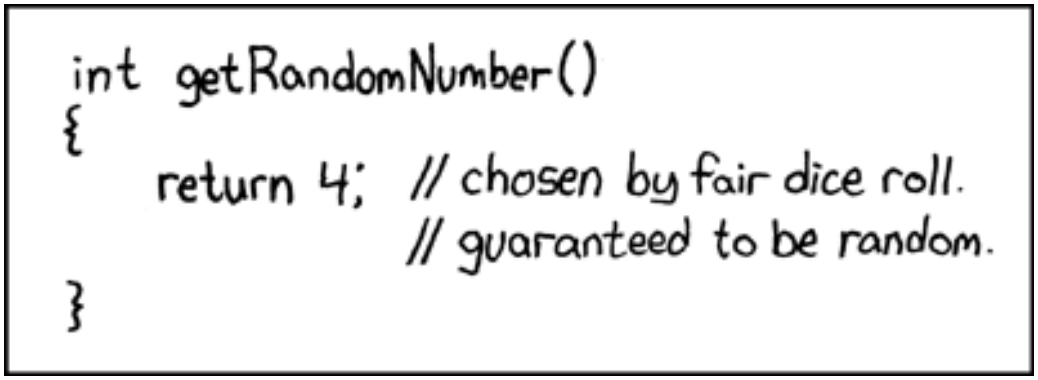


image: xkcd 221

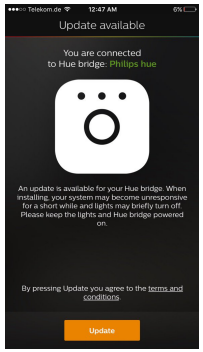
5. Awful Software

- The software for things like lightbulbs is often awful, and open to all kinds of exploitation.
- This shouldn't be too surprising since even the manufacturers of electronic locks have trouble getting this right.

- Matthew Garrett on iRainbow Lightbulbs
- TrendMicro on HID remote door unlocks
- Mirai DDoS

6. Update Cycle

- It's hard enough to get people to change their smoke alarm batteries or update Internet Explorer.
- No-one ever is going to reflash their thermostat.
- While devices are reloading, they are generally unavailable.



Complaints → Requirements

- ~~Internet Mediation~~ Internet Independence
- ~~Vendor Lockin~~ Generic Interfaces
- ~~SSL~~ Simplified Cryptography
- ~~C / Linux~~ Easier development

Requirements → Potential Solutions

- Client/Server → Peer to Peer
- Standards-based Hardware Description Language
- Shared Secrets, initialized by QR code
- Simpler development with **MicroPython!**

MicroPython Meetup

Wednesday, September 27

Connected Community HackerSpace (CCHS)

https://www.meetup.com/MicroPython-Meetup/

http://hackmelbourne.org

BuzzConf Nights

Thursday, September 28

Loop Bar

https://www.meetup.com/BuzzConf/

LinuxConf 2018

Tutorial:

"Getting Started with MicroPython"

https://linux.conf.au/

Questions / Comments

Nick Moore
Mnemote Pty Ltd

- nick@mnemote.com
- http://mnemote.com/
- @mnemote

Slides:

- http://nick.zoic.org/talk/yow1/