Scripting the Internet of Things

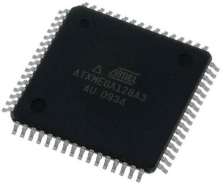
Damien P. George

George Robotics Limited,
Cambridge, UK

PyCon AU, Melbourne, 12th August 2016

The Internet of Things?

IoT = Microcontrollers + wireless communications



- lighting: homes and office buildings
- heating and cooling houses
- traffic monitoring: sensors in/above roads
- farming: water levels, livestock tracking
- logistics: real-time tracking of shipping containers

End Nodes

The end nodes are the fingertips: sensors and actuators.

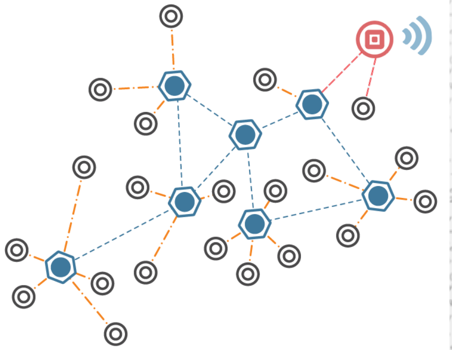


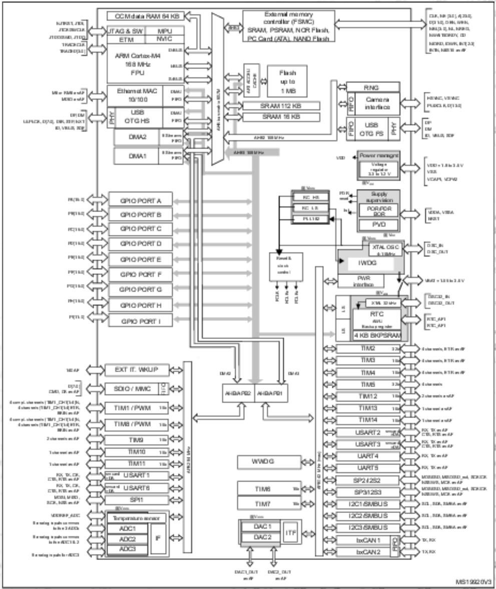
Image source: http://www.mtechlog.com/2015/10/iot-routing-solution-for-wireless.html

Image source: http://www.omega.com/pptst/UWIR-2.html

Complexity of hardware!

High-level scripting languages allow:

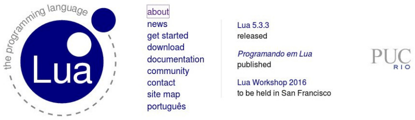
- easier to read/write code
- abstraction of HW
- rapid prototyping
- more portable code
- library reuse



Source: STM32F405 manual

Lua and eLua

www.lua.org www.eluaproject.net




```
function factorial(n)
    local x = 1
    for i = 2, n do
        x = x * i
    end
    return x
end

gpio.mode(1, gpio.OUTPUT)
gpio.write(1, gpio.HIGH)
```

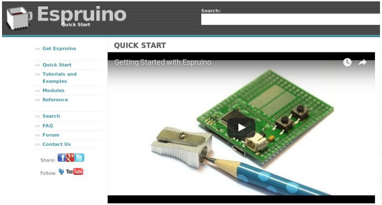
Pros: simple language, light-weight, fast

Cons: simple language, no native bitwise ops, no integers (recently fixed!, eLua yet to catch up)

Uses in IoT: NodeMCU ESP8266 board

JavaScript

www.espruino.org, jerryscript.net, www.tessel.io, duktape.org



JerryScript: JavaScript engine for the Internet of Things


```
setInterval(function() {
    digitalWrite(LED1, Math.random()>0.5);
}, 20);
```

Pros: very popular language, large community, simple but powerful

Cons: some crazy semantics, callback-based, all numbers are floats

Uses in IoT: Espruino boards, ESP8266, Tessel boards, ...

Ruby

www.ruby-lang.org   mruby.org

Ruby
A PROGRAMMER'S BEST FRIEND

mruby is the lightweight implementation of the Ruby language complying with part of the ISO standard.
mruby can be linked and embedded within your application.

```
class Integer
    def factorial
        f = 1; for i in 1..self; f *= i; end; f
    end
end
```

Pros: popular language, lots of features and libraries

Cons: no proper support for microcontrollers

Uses in IoT: none yet?

python™

Is it possible to put Python on a microcontroller?

Why is it hard?

- Very little memory (RAM, ROM) on a microcontroller.

Motivation for using Python:

- High-level language with powerful features (classes, list comprehension, generators, exceptions, . . .) and libraries.
- Large existing community.
- Very easy to learn, powerful for advanced users: shallow but long learning curve.
- Ideal for microcontrollers: native bitwise operations, procedural code, distinction between int and float, robust exceptions.
- Lots of opportunities for optimisation (Python is *compiled*).

Why can't we use CPython? (or PyPy?)

- Integer operations:

  Integer object (max 30 bits): 4 words (16 bytes)

  Preallocates 257+5=262 ints $\longrightarrow$ 4k RAM!

  Could ROM them, but that's still 4k ROM.

  And each integer outside the preallocated ones would be another 16 bytes.

- Method calls:

  led.on(): creates a bound-method object, 5 words (20 bytes)

  led.intensity(1000) $\longrightarrow$ 36 bytes RAM!

- For loops: require heap to allocate a range iterator.

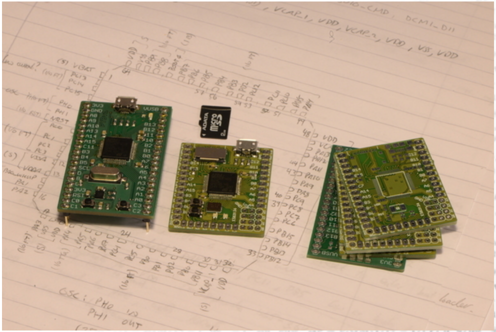
MicroPython: Python for microcontrollers

(and embedded systems, constrained environments, IoT, ...)

Crowdfunding via Kickstarter

Kickstarter is a good way to see if your idea has traction, or not.

- 30th April 2013: start!
- 17th September: flashing LED with button in bytecode Python.
- 21st October: REPL, filesystem, USB VCP and MSD on PYBv2.



1 weekend to make the video.

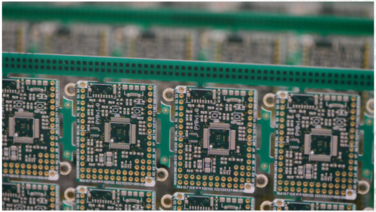
Kickstarter launched on 13 November 2013, ran for 30 days.

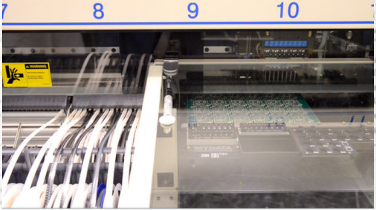
Total backers: 1,931
Total raised: £97,803 ($180k)

Officially finished 12 April 2015.

Manufacturing

Jaltek Systems, Luton UK — manufactured 13,000+ boards.





It's all about the RAM

If you ask me **'why is it done that way?'**,
I will most likely answer: **'to minimise RAM usage'**.

- Interned strings, most already in ROM.
- Small integers stuffed in a pointer.
- Optimised method calls (thanks PyPy!).
- Range object is optimised (if possible).
- Python stack frames live on the C stack.
- ROM absolutely everything that can be ROMed!
- Garbage collection only (no reference counts).
- Exceptions implemented with custom `setjmp/longjmp`.

GitHub and the open-source community

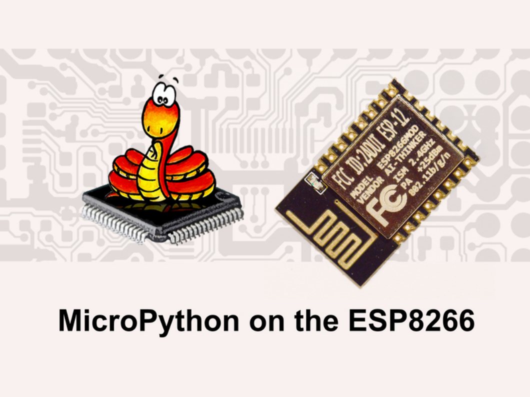
`https://github.com/micropython`

MicroPython is a *public* project on GitHub.

- A global coding conversation.
- Anyone can clone the code, make a fork, submit issues, make pull requests.
- MicroPython has over 3500 "stars" (top 0.02%), and more than 740 forks.
- Contributions come from many people (120+), with many different systems.
- Leads to: more robust code and build system, more features, more supported hardware.
- Hard to balance inviting atmosphere with strict code control.

A big project needs many contributors, and open-source allows such projects to exist.

And then went back for more...



Kickstarter #2 was a *pure software* campaign.

Finished on 2nd March 2016 with 1384 backers, £28,334 ($50k).

Live Demo!

MicroPython brings Python to resource-limited systems.

It allows rapid development of IoT applications.

Future development:

- continued development of ESP8266 port
- support for IoT: sensors, umqtt
- improved (micro)asyncio support
- optimise multithreading
- more features for the micro:bit, further ESA work

micropython.org

forum.micropython.org

github.com/micropython