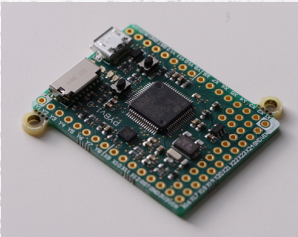


MicroPython: REPLs everywhere

Damien P. George

DAMTP & The Cavendish,
University of Cambridge

Inria (Lille), 2nd September 2015



Part I: Shrinking Python down to run on a microcontroller



Motivation for MicroPython

Electronics circuits now pack an enormous amount of functionality in a tiny package.

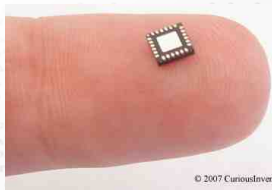
Need a way to control all these sophisticated devices.

Scripting languages enable rapid development.

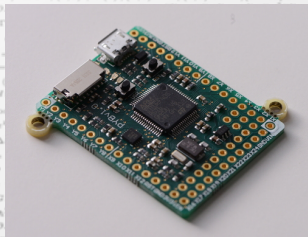
Is it possible to put Python on a microcontroller?

Why is it hard?

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



© 2007 CuriousInventor.com



Why Python?

- ▶ High-level language with powerful features (classes, list comprehension, generators, exceptions, ...).
- ▶ 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 (this may sound surprising, but 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 → 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) → 36 bytes RAM!

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

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`.

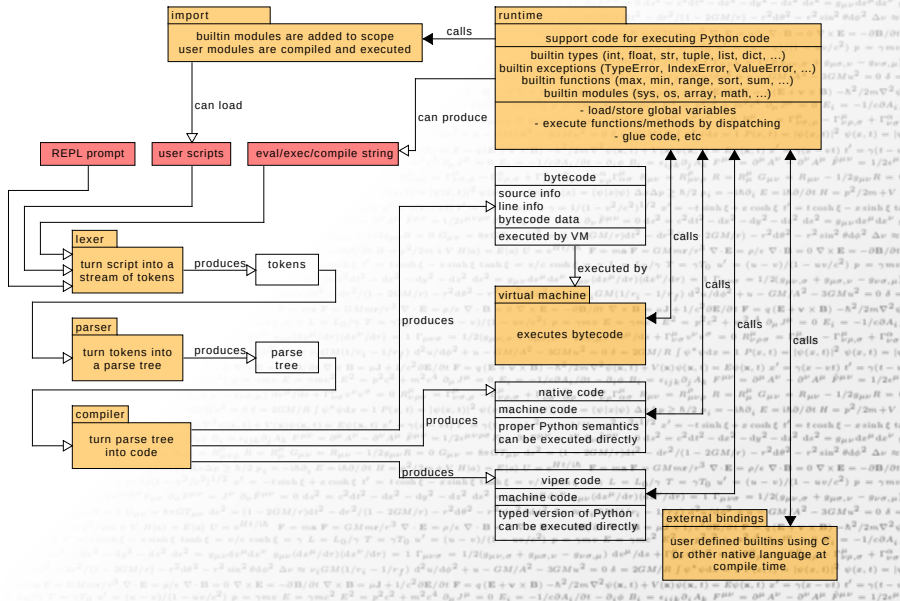
PC demo

Internals

- ▶ Lexical analyser: simple tokeniser that interns all identifiers and (most) strings.
- ▶ Parser: grammar represented in table form, evaluated by a single, non-recursive function.
- ▶ Compiler: passes over the parse tree 3-4 times.
- ▶ Code emitter: emits bytecode, or machine code.
- ▶ Virtual machine: interprets bytecode.
- ▶ Runtime: many helper functions, implementing functionality for the objects.

A given port (eg Linux, bare metal) provides specific RAM and I/O hooks, a REPL, and custom modules.

Internals



Object representation

A MicroPython object is a machine word, and has 3 different forms.

Integers:

- ▶ xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxx1
- ▶ Transparent transition to arbitrary precision integers.

Strings:

- ▶ xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxx10
- ▶ Certain strings are not interned.

Objects:

- ▶ xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxx00
- ▶ A pointer to a structure.
- ▶ First element is a pointer to a type object.
- ▶ ROMable (type, tuple, dictionary, function, module ...).

Note: still room for packing truncated floats.

Emitters: bytecode

@micropython.bytecode

```
def add(x, y):  
    return x + y
```

Compiles to:

```
00:  b0  LOAD_FAST_0  
01:  b1  LOAD_FAST_1  
02:  db  BINARY_OP_ADD  
03:  5b  RETURN_VALUE
```

Emitters: native

@micropython.native

def add(x, y):

return x + y

```
00:  e92d41fe  push    {r1, r2, r3, r4, r5, r6, r7, r8, lr}
04:  e24dd028  sub     sp, sp, #40          ; 0x28
08:  e59f7000  ldr     r7, [pc]            ; 0x10
0c:  ea000000  b       0x14
10:  080794e0  .word   0x080794e0
14:  e1a04003  mov     r4, r3
18:  e1a03002  mov     r3, r2
1c:  e1a02001  mov     r2, r1
20:  e1a01000  mov     r1, r0
24:  e3a00074  mov     r0, #116           ; 0x74
28:  e58d0000  str     r0, [sp]
2c:  e3a00080  mov     r0, #128           ; 0x80
30:  e58d0004  str     r0, [sp, #4]
34:  e3a00004  mov     r0, #4
38:  e58d0014  str     r0, [sp, #20]
3c:  e28d0000  add     r0, sp, #0
40:  e92d0010  stmfd   sp!, {r4}
44:  e1a0e00f  mov     lr, pc
48:  e597f0a0  ldr     pc, [r7, #160]     ; 0xa0
4c:  e8bd0001  ldmfdd sp!, {r0}
50:  e59d4024  ldr     r4, [sp, #36]     ; 0x24
54:  e59d5020  ldr     r5, [sp, #32]
58:  e1a02005  mov     r2, r5
5c:  e1a01004  mov     r1, r4
60:  e3a00005  mov     r0, #5
64:  e1a0e00f  mov     lr, pc
68:  e597f034  ldr     pc, [r7, #52]     ; 0x34
6c:  e28dd028  add     sp, sp, #40       ; 0x28
70:  e8bd81fe  pop     {r1, r2, r3, r4, r5, r6, r7, r8, pc}
```

Emitters: viper

```
@micropython.viper
```

```
def add(x:int, y:int) -> int:
    return x + y
```

Compiles to:

```
00:  e92d41fe    push    {r1, r2, r3, r4, r5, r6, r7, r8, lr}
04:  e59f7000    ldr     r7, [pc]; 0xc
08:  ea000000    b       0x10
0c:  080794e0    .word   0x080794e0
10:  e1a04000    mov     r4, r0
14:  e1a05001    mov     r5, r1
18:  e1a01004    mov     r1, r4
1c:  e0811005    add     r1, r1, r5
20:  e1a00001    mov     r0, r1
24:  e8bd81fe    pop     {r1, r2, r3, r4, r5, r6, r7, r8, pc}
```

Emitters: inline assembler

```
@micropython.asm_thumb
```

```
def sum_bytes(r0, r1):
```

```
    mov(r2, 0)
```

```
    b(loop_entry)
```

```
    label(loop1)
```

```
    ldrb(r3, [r1, 0])
```

```
    add(r2, r2, r3)
```

```
    add(r1, r1, 1)
```

```
    sub(r0, r0, 1)
```

```
    label(loop_entry)
```

```
    cmp(r0, 0)
```

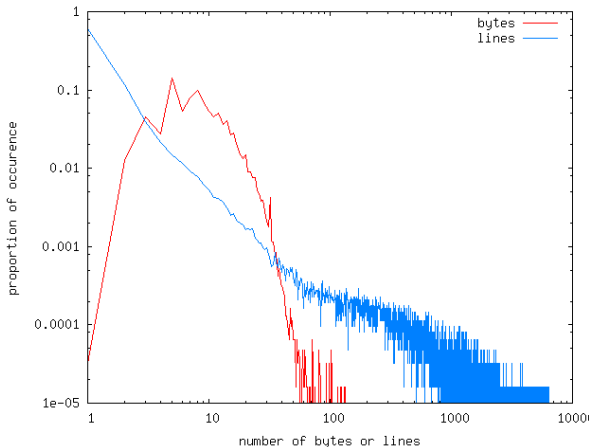
```
    bgt(loop1)
```

```
    mov(r0, r2)
```

```
Call as normal: print(sum_bytes(4, b'abcd'))
```

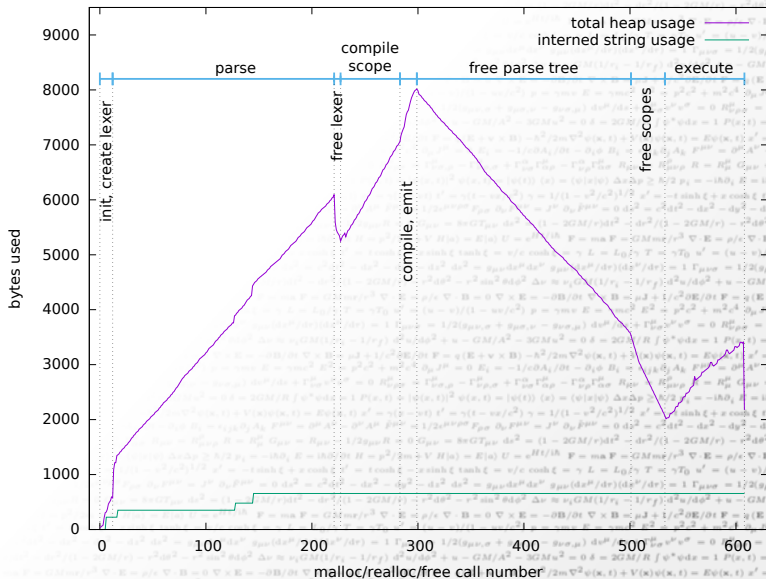
Encoding of source-bytecode map

Histogram showing frequency of having to skip n bytes and n lines.



Encoding optimised for this (very simple Huffman-like).

Example heap usage



Code dashboard

<http://micropython.org/resources/code-dashboard/>



Coding style

MicroPython does *not* follow traditional software engineering practices:

- ▶ optimise first;
- ▶ creative solutions and tricks;
- ▶ sacrifice clarity to get smaller code;
- ▶ sacrifice efficiency to get smaller code (esp. less-used features);
- ▶ use of goto not discouraged;
- ▶ optimise to minimise stack usage;
- ▶ make decisions based on analysis.

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 2250 “stars”, and more than 410 forks.
- ▶ Contributions come from many people, 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.

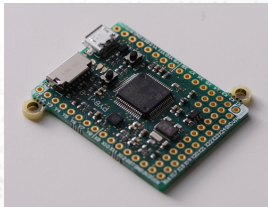
GitHub stars — all 10 million+ projects

1	twbs/bootstrap	CSS	85,841
2	vhf/free-programming-books	None	42,432
3	angular/angular.js	JavaScript	42,102
4	mbostock/d3	JavaScript	41,149
5	nodejs/node-v0.x-archive	JavaScript	38,004
6	jquery/jquery	JavaScript	35,780
7	FortAwesome/Font-Awesome	HTML	35,663
8	h5bp/html5-boilerplate	JavaScript	30,974
9	meteor/meteor	JavaScript	27,843
10	rails/rails	Ruby	27,531
...			
1766	apache/couchdb	JavaScript	2,279
1767	lifesinger/lifesinger.github.com	JavaScript	2,276
1768	googlesamples/android-topeka	Java	2,274
1769	addyosmani/es6-tools	None	2,274
1770	activerecord-hackery/ransack	Ruby	2,274
1771	sciactive/pnotify	CSS	2,273
1772	addyosmani/basket.js	JavaScript	2,273
1773	PostgresApp/PostgresApp	Objective-C	2,273
1774	micropython/micropython	C	2,272
1775	seanpowell/Email-Boilerplate	HTML	2,271
1776	lipka/piecon	JavaScript	2,271
1777	alfajango/jquery-dynatable	JavaScript	2,271
...			

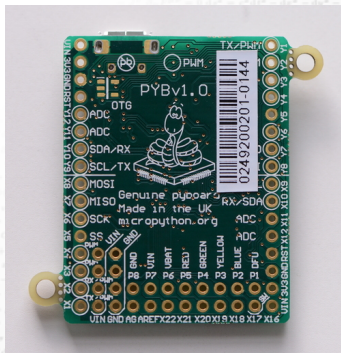
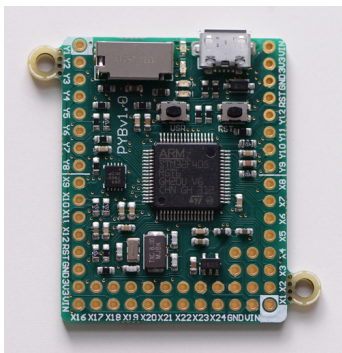
GitHub stars — all C/C++ projects

1	torvalds/linux	C	25,156
2	nwjs/nw.js	C++	24,175
3	atom/electron	C++	15,777
4	ariya/phantomjs	C++	15,084
5	antirez/redis	C	14,701
6	facebook/hhvm	C++	12,497
7	textmate/textmate	C++	10,490
8	git/git	C	10,082
...			
98	jonas/tig	C	2,361
99	swoole/swoole-src	C	2,319
100	raspberrypi/linux	C	2,310
101	SFML/SFML	C++	2,282
102	philipl/pifs	C	2,281
103	micropython/micropython	C	2,272
104	sqlitebrowser/sqlitebrowser	C++	2,269
105	rswier/c4	C	2,255
106	philsquared/Catch	C++	2,228
107	joyent/http-parser	C	2,225
108	nanomsg/nanomsg	C	2,220
109	ivansafirin/Polycode	C++	2,195
110	libuv/libuv	C	2,192
111	mpv-player/mpv	C	2,173
112	arut/nginx-rtmp-module	C	2,158
113	numpy/numpy	C	2,157

Part II: The pyboard hardware



The pyboard



- ▶ STM32F405RG: 192k RAM, 1M ROM, 168MHz, Cortex M4F.
- ▶ USB micro connector for device (and host).
- ▶ Micro SD card.
- ▶ 3-axis accelerometer (MMA7660).
- ▶ Real-time clock, 4 LEDs, 2 switches.
- ▶ 30 GPIO: symmetric pin layout, plus extra pins.
- ▶ Internal file system. "/flash" and "/sd".

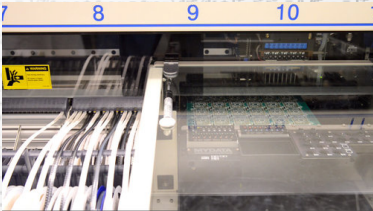
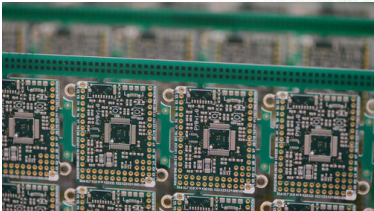
Pyboard usage

- ▶ Standard Python prompt (REPL) over USB serial device, or UART.
- ▶ Raw prompt: reads a Python script until EOF, executes it, then sends back the result.
- ▶ A script running from the flash/SD. Serial connection becomes stdin/stdout.
- ▶ Powered by USB or battery.

pyboard demo

Manufacturing

Jaltek Systems, Luton UK — manufactured 7000+ boards.



Testing and programming



pyboard is running your job!

Current code

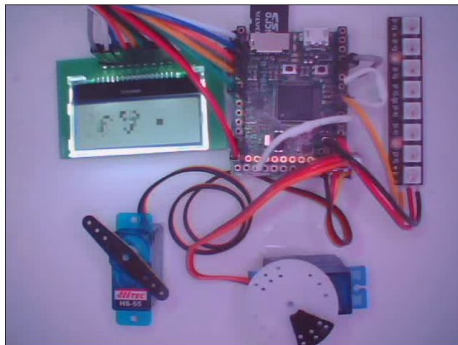
```
# Conway's Game of Life
import pyb

PIXEL_SIZE = const(4)

# we use big pixels so it's easier to see
@micropython.viper
def big_pixel(x:int, y:int, col):
    pixel = lcd.pixel
```

Current output

MicroPython running Conway's Game of Life!



My code

running... clear ← example →

```
1 # Conway's Game of Life
2 import pyb
3
4 PIXEL_SIZE = const(4)
5
6 # we use big pixels so it's easier to see
7 @micropython.viper
8 def big_pixel(x:int, y:int, col):
9     pixel = lcd.pixel
10     for i in range(PIXEL_SIZE):
11         for j in range(PIXEL_SIZE):
12             pixel(x+i, y+j, col)
13
14 # do 1 iteration of Conway's Game of Life
15 @micropython.viper
16 def conway_step(lcd):
17     get = lcd.get
18     for x in range(0, 128, PIXEL_SIZE):
19         for y in range(0, 32, PIXEL_SIZE):
20             self = int(get(x, y))
21
```

My output

MicroPython running Conway's Game of Life!

Other hardware

MicroPython runs on lots of other hardware:

- ▶ STM32F4xx discovery boards,
- ▶ Espruino Pico (STM32F401),
- ▶ CC3200 wi-fi SoC (WiPy),
- ▶ ESP8266 wi-fi SoC,
- ▶ 16-bit dsPIC33F,
- ▶ ...

Part III: The Kickstarter campaign

KICKSTARTER

Crowd funding

Pitch an idea, get the public to fund it, give them something in return.

Started by ArtistShare in 2003.

IndieGoGo in 2008, Kickstarter in 2009, plus many others.

- ▶ Musicians, photographers, writers, video games, hardware, ...
- ▶ Science projects
- ▶ Roll your own

2012: US\$2.7 billion, more than one million individual campaigns

2013: crowdfunding industry grew to over \$5.1 billion

Kickstarter: collected so far over US\$1 billion in funds

The stages

Stage 1: the idea!

- ▶ Prototyping.
- ▶ Fear that someone will beat you.
- ▶ Rush to get it online.
- ▶ Making the campaign, including video.

Stage 2: the campaign.

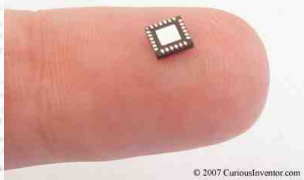
Stage 3: fulfillment.

- ▶ Spend all the money, buy lots of stuff.
- ▶ Finalise your idea and mass produce it.
- ▶ Lots of delays.
- ▶ Work out how to post 1000s of parcels.
- ▶ Packing and shipping.
- ▶ Returns, complaints, support, etc.

Stage 1: the idea

Idea for MicroPython

- ▶ System on a Chip: CPU, RAM, flash memory, timers, USB, Ethernet.
- ▶ Sensors: touch, accelerometer, gyroscope, compass, barometer.
- ▶ Outputs: LEDs, LCDs, DC motors, servos.



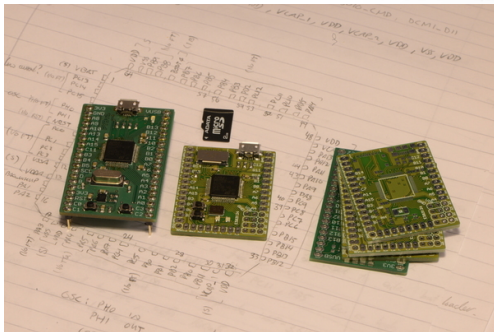
Goal: Make it easy for people to program their electronics projects.

Idea: Shrink Python to run on a microcontroller.

KICKSTARTER

Initial development

- ▶ 30th April 2013: start!
- ▶ 17th September: flashing LED with switch 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.

Officially finished 12 April 2015.

Stage 2: the campaign

The Kickstarter journey

Support

KICKSTARTER

Help

Damien George

Not you?

Guidelines

Basics

Rewards

Story

About you

Account

Review

Launch

Preview

Thank you.

Appreciate you sharing your project with us.

Accepted!

KICKSTARTER

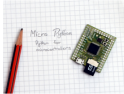
Kickstarter Staff

Tuesday Nov 12 2013, 6:17pm GMT

Thanks for sending us your project for review. It looks like everything falls within our guidelines. Hooray! That means that whenever you're ready, you can hit the green launch button and make your project live. There's no deadline to launch, though, so take your time.

Some last-minute tips before you launch:

- * Check out other projects in your category, and back one to get a feel for the experience.
- * Visit Kickstarter School: <http://www.kickstarter.com/help/school>
- * Use the Preview button to see what your project will look like when it's live, and share it with friends for feedback.
- * Get the Kickstarter app for iPhone (<http://www.kickstarter.com/download>...) to track pledges, send updates, and stay connected with your backers from wherever you are.
- * Add a video, if you haven't already! This is really important. Here's why: http://www.kickstarter.com/help/school/making_your_video.



Micro Python: Python for microcontrollers

by Damien George

Submitted: Nov 11, 2013

Status: Accepted

You'll do great! We can't wait to see your project live.

Back

Exit

See,

D.P. George

MicroPython

37/61

The Kickstarter journey

Support

Guidelines Basics Rewards Story About you Account Review Launch Preview 0

funded.

Post funding

Once your project's funding has ended, it will continue to be public and remain preserved as is. You'll be able to use Project Updates and Project FAQs to provide new information about your project's development post funding.

Your responsibility

If your project is successfully funded, you are required to fulfill all rewards or refund any backer whose reward you do not or cannot fulfill. A failure to do so could result in damage to your reputation or even legal action on behalf of your backers. For more on accountability, see the [FAQ](#).

☒ I have read these important reminders, the [Terms of Use](#), [Privacy Policy](#), and the [Kickstarter Project Guidelines](#).

Launch project now

The Kickstarter journey

KICKSTARTER

Micro Python: Python...

Post update

Edit project

Check dashboard

View backer report

View messages

Send survey

Kickstarter School

Contact us

Creator FAQ

Log out

Micro Python: Python for microcontrollers

by Damien George

Home Updates Backers Comments

Cambridge, United Kingdom Hardware

Your project has launched!

Congratulations, your project is live on Kickstarter!

Drumroll...

Here's your project URL:
<http://www.kickstarter.com/projects/214379695/micro-python-python-for-microcontrollers>

The countdown begins now. You know what to do: tell people about it!

GOOD LUCK!
Kickstarter

0 backers

£0.00

pledged of £15,000 goal

29 days to go

Back This Project

£1 minimum pledge

This project will only be funded if at least £15,000 is pledged by Friday Dec 13, 10:09am GMT.

Funding period

Nov 13, 2013 - Dec 13, 2013 (30 days)



Project by
Damien George
Cambridge, United Kingdom

The Python language made lean and fast to run on microcontrollers. For beginners and experts, control your electronic project with ease

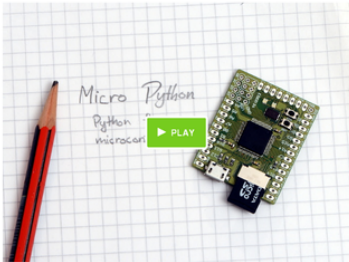
The Kickstarter journey

KICKSTARTER What is Kickstarter? Discover great projects Start a project Search projects Help Sign up Log in

Micro Python: Python for microcontrollers

by Damien George

Home Updates 2 Backers 328 Comments 11 Cambridge, United Kingdom Hardware



328 backers

£15,005

pledged of £15,000 goal

27

days to go

Back This Project

£1 minimum pledge

This project will be funded on Friday Dec 13, 5:09pm EST.

Funding period

Nov 13, 2013 - Dec 13, 2013 (30 days)

Project by

Damien George

Cambridge, United Kingdom

Contact me

First created 2 backed

[Share](#) [556](#) [Tweet](#) [Embed](#) [Remind me](#)

The Python language made lean and fast to run on microcontrollers. For beginners and experts, control your electronic project with ease

What is Micro Python?

The Kickstarter journey

KICKSTARTER

Micro Python: Python for microcontrollers

Post update Last 12/12/2013

Edit project

Check dashboard

View backer report

View messages

Send survey

Kickstarter School

Contact us

Creator FAQ

Log out

Micro Python: Python for microcontrollers

by Damien George

Home

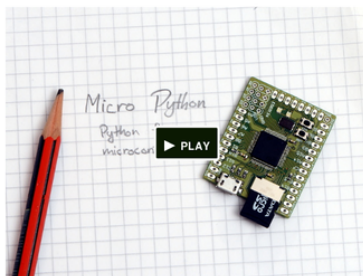
Updates 21

Backers 1,930

Comments 359

Cambridge, United Kingdom

Hardware



1,930

backers

£97,749

pledged of £15,000 goal

1

second to go

Back This Project

£1 minimum pledge

This project will be funded on Friday
Dec 13, 10:09am GMT.

Share 1,898 Tweet Embed

The Python language made lean and fast to run on microcontrollers. For beginners and experts, control



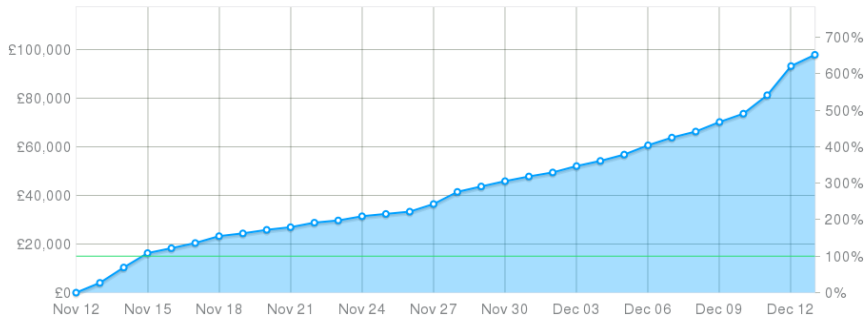
Kickstarter

Your project has been successfully funded! - Kickstarter Congratulations! Your project was successfully funded. Micro Python: Python

10:09 am

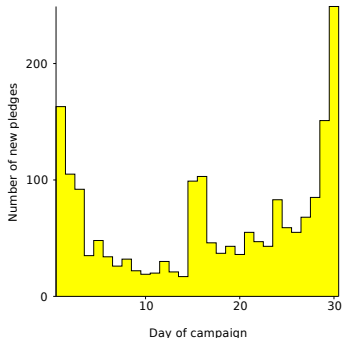
Campaign timeline

Funding progress

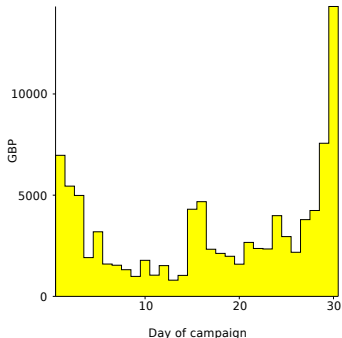


Campaign timeline

Pledges per day



GBP per day

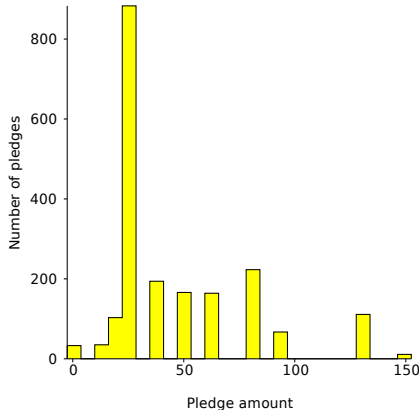


Total backers: 1,931

Total raised: £97,803

Pledge levels

Pledge categories



- ▶ No reward, just donation.
- ▶ A single board, £20 and £24.
- ▶ Multiple boards at £50 and £90.
- ▶ Kits at £40, £60, £80 and £130.
- ▶ Special items at £150, £300 and £1,000.

A few weeks later ...

Recent transactions for:



[Back to My accounts](#)

Balance

£ 87742.11 C

Available Balance

£ 87742.11 C

As At

31 Dec 2013 12:27

Date▼	Type▼	Description▼	Paid out▼	Paid in▼	Balance (£)
31 Dec	TFR	INTERNET TRANSFER		87742.11	87742.11

[Download transactions](#) [Print](#)

Internet coverage

KICKSTARTER What is Kickstarter? Discover great projects Start a project

Search projects

Help

Discover Technology

Staff Picks

Micro Python: Python for microcontrollers
by Chris G. Clark

The Python language made here and there to run on microcontrollers. Packages and projects, control your electronic project with ease.

Cambridge, United Kingdom

13% \$1,800 28

FINISHED 15,000,000 10/15/10

NVIDIA SPHERE: Next Generation Control of Your Environment
by Greg Eichen

Next generation control of your environment with accurate, extreme location data and a generic control interface.

Riverton, Australia

76% \$68,000 59

FINISHED 10,000,000 10/15/10

Liquid Ambience: A Boutique Ambience-generating guitar effect
by Paul Olsson / Michael Hooten

Combines polyphonic voice generation and lush reverb, delivering a beautiful atmospheric guitar pedal sounds in each your.

Chesham-on-Thames, Great Britain

100% \$7,200 29

FINISHED 10,000,000 10/15/10

[See more staff picks](#)

Search or type a command Explore Git Blog Help

Explore GitHub

All Showcases Trending Stars

Trending repositories

Find what repositories the GitHub community is most excited about today.

Repositories Developers Trending today All languages

intrepid / http-api-design

HTTP API design guide extracted from work on the Heroku Platform API

335 17 built by

Star

micropython / micropython

The Micro Python project

272 17 built by

Star

PrePit looking for recently updated repositories? Try this search

Hacker News new | comments | ask | jobs | submit

1. **Micro Python - a lean and efficient implementation of Python 3** (python.org)
245 points by mazerath 3 hours ago | 41 comments
2. **HTML5 Video in Safari on OS X Yosemite** (netflix.com)
29 points by step 31 minutes ago | 28 comments

HACK A DAY

Too much electronics? Here is a kitten.

HOME OUR VIDEOS SUBMIT A TIP FORUMS STAFF

Interview with [Damien George], Creator of the Micro Python project
November 27, 2013 by Matthew Stephan · 1 Comment

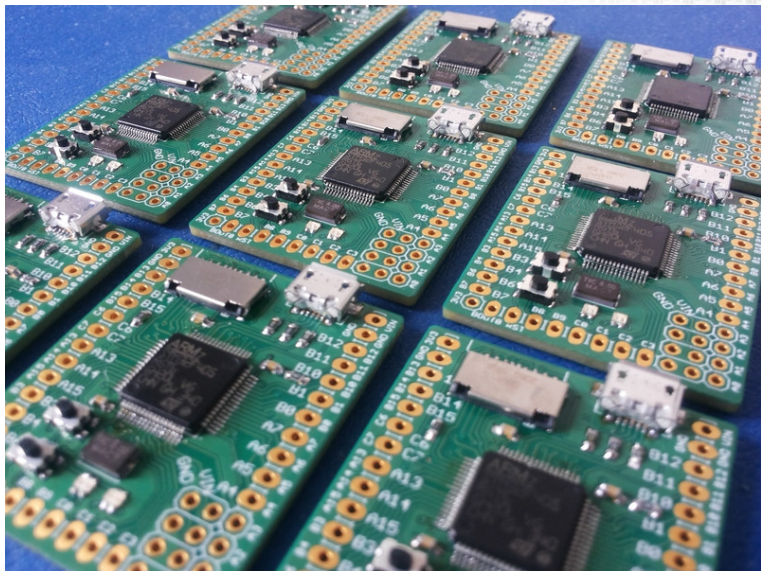
[Damien George] just created **Micro Python** (Kickstarter alert!), a lean and fast implementation of the Python scripting language that is optimized to run on a microcontroller. It includes a complete parser, compiler, virtual machine, runtime system, garbage collector and was written from scratch. Micro Python currently supports 32-bit ARM processors like the STM32F405 (100MHz Cortex-M4, 1MB flash, 128KB ram) shown in the picture above and will be open source once the already successful campaign finishes. Running your python program is as simple as copying your file to the platform (detected as a mass storage device) and rebooting it. The official micro python board includes a micro SD card slot, 4 LEDs, a switch, a real-time clock, an accelerometer and has plenty of I/O pins to interface many peripherals. A nice video can be found on the campaign page and an interview with the project creator is embedded after the break.

[Read more...](#)

Filed Under: ARM, Crowdfunding, Interviews, software hacks Tagged With: arm, compiler, python, micro python, python, virtual machine

Stage 3: fulfillment

Hand made boards



Boards, headers, servo motors, ...



Programming and packing



Stamps!



Packing and shipping



D.P. George



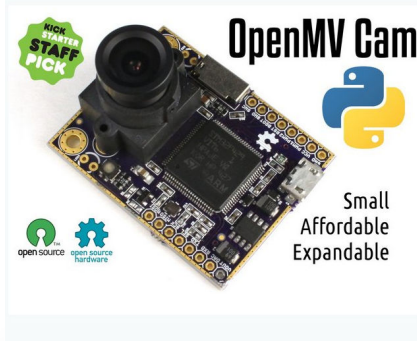
MicroPython

Packing and shipping



Other Kickstarter campaigns

Machine Vision With Python



Add machine vision to your projects by scripting Python. Small, affordable, and expandable with shields.

Follow along!

Created by

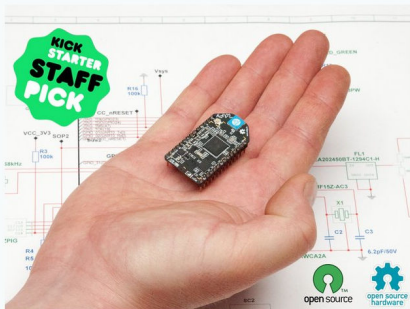
Bot Thoughts LLC



900 backers pledged \$104,498 to help bring this project to life.

Other Kickstarter campaigns

The WiPy: The Internet of Things Taken to the Next Level



The IoT development platform that runs Python in real time, and features the perfect blend of power, friendliness and flexibility.

www.wipy.io

Created by
The WiPy



1,382 backers pledged €75,564 to help bring this project to life.

Lessons and tips

- ▶ Why do you want to do a crowd funding campaign (fun, money, have an idea, start a business)?
- ▶ Work quick to bring it to the public:
 - might be beaten!
 - learn earlier if it's not a good idea
 - spend too long and the idea gets stale
- ▶ Make a video that shows yourself, your idea and your excitement.
- ▶ Plan your rewards carefully and offer a range (people like options) but don't make them too complicated (for your own sake).
- ▶ Don't over-promise; have a core single idea and deliver on that as best you can.
- ▶ Be prepared to respond to a lot of emails!

Part IV: The future



Post Kickstarter

We now have a private limited company.

Continue to sell the pyboard and accessories online.



CERTIFICATE OF INCORPORATION OF A PRIVATE LIMITED COMPANY

Company Number 8861687

The Registrar of Companies for England and Wales, hereby certifies that
GEORGE ROBOTICS LIMITED

is this day incorporated under the Companies Act 2006 as a private company, that the company is limited by shares, and the situation of its registered office is in England and Wales.

Given at Companies House, Cardiff, on 27th January 2014.

Code improvements

- ▶ Option for 64-bit NaN boxing model.
- ▶ Option for 30-bit floats packed into current pointer model.
- ▶ Allow pre-compiled bytecode to be loaded.
- ▶ 100% test coverage (at 94% right now).
- ▶ Define a consistent hardware API for all MCUs / boards.

New hardware

- ▶ Cheaper board.
- ▶ More powerful board (eg Ethernet, SDRAM).
- ▶ More add-on boards (all with a standard form factor).
- ▶ Any new, interesting MCU: make a board out of it!

Future of MicroPython

Continue to improve documentation, write tutorials, support more peripherals, build a community.

Not just for bare metal: works very well on standard systems and anyone can build and run it easily on *nix/Windows/Mac.

Great potential for Internet of Things (IoT): much easier to develop a small internet connected device using Python than C.

Embedding MicroPython in games and mobile apps.

Industrial use — use in space? Determinism is a key point.

micropython.org

forum.micropython.org

github.com/micropython

