



NOSERENGINEERING
WE KNOW HOW

Embedded-Prototyping mit MicroPython



Tobias Badertscher, Christian Müller | Noser Bern | November 2019

Eine LED blinken lassen und einen Sensor über UART ansprechen – das kann doch nicht so schwierig sein...

Java-Entwickler, 2019

Komm, wir machen schnell IoT mit WLAN und so...

Projektleiter, ca. 1850

Was ist Embedded?

Server / PC



Raspberry Pi o.Ä.



Embedded OS



Bare Metal



Viel Leistung

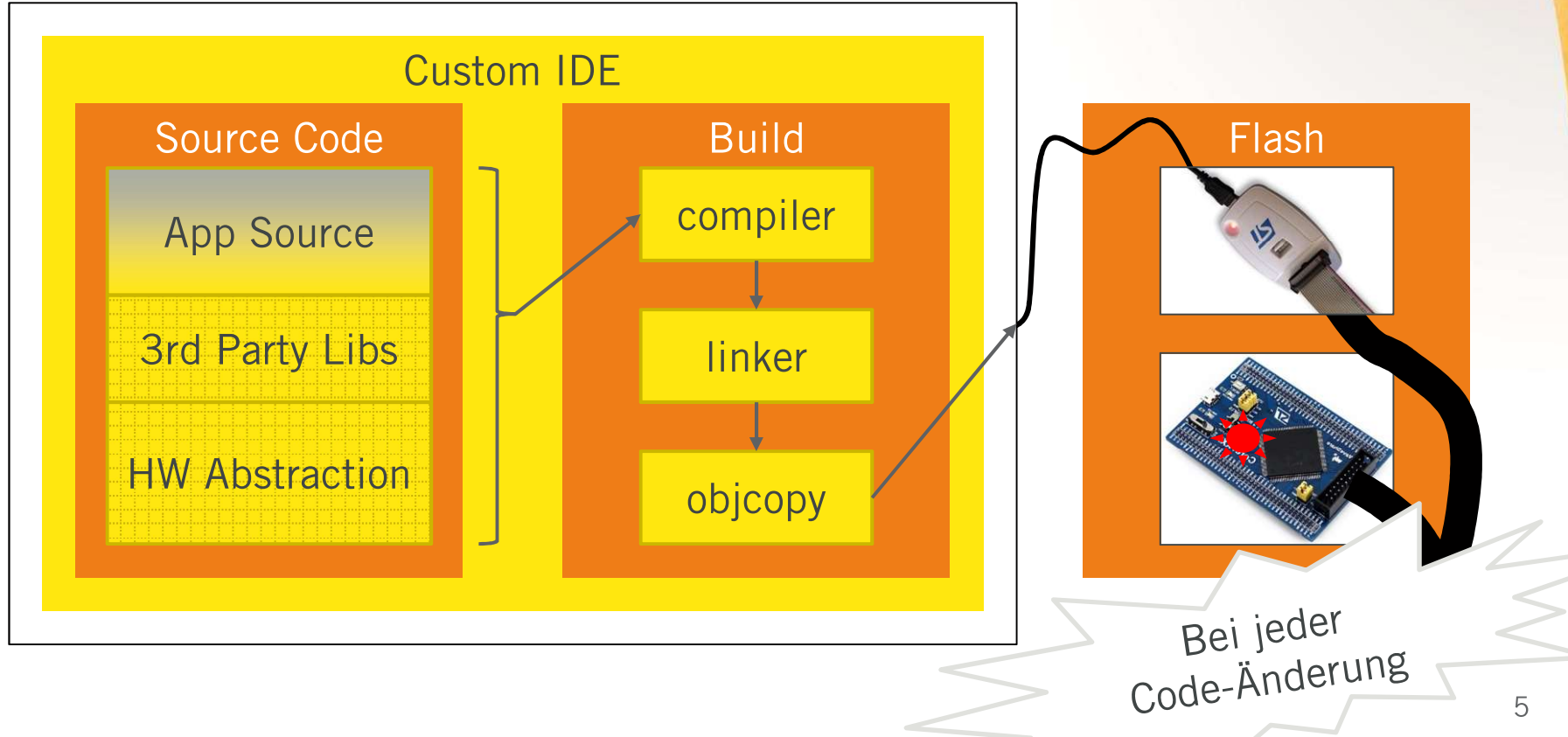
High Level Sprachen / Skripts

OS Features (Tasks, Filesystem...)

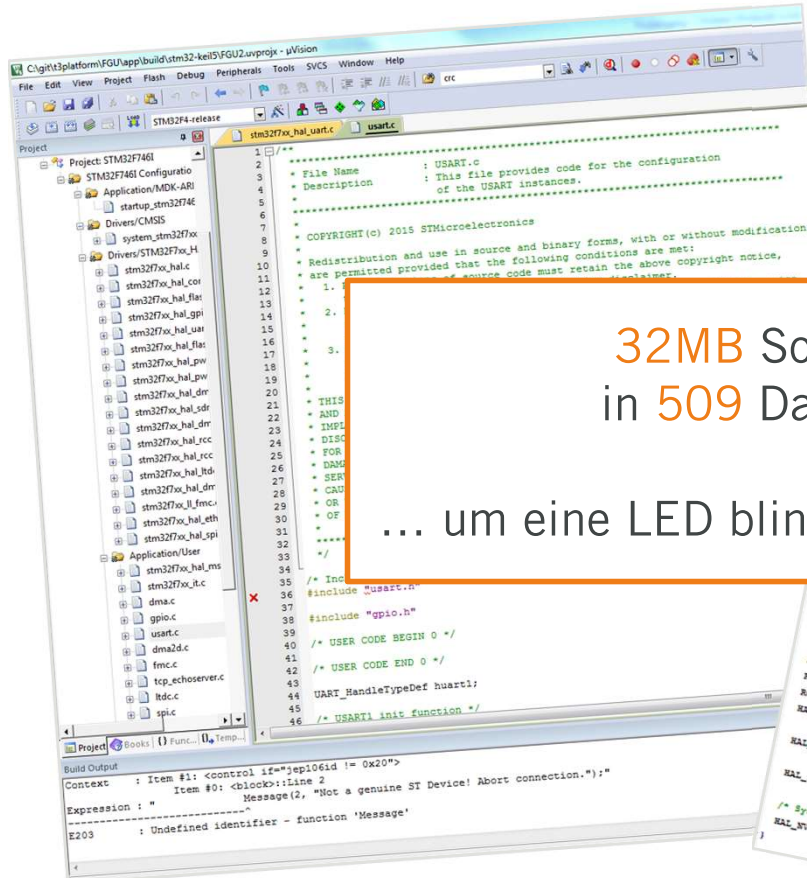
Proprietäre, kompilierte Applikationen

Hardware-Kosten
Ressourcen-Anforderungen

Schnell mal 'ne LED blinken lassen auf einem neuen Target

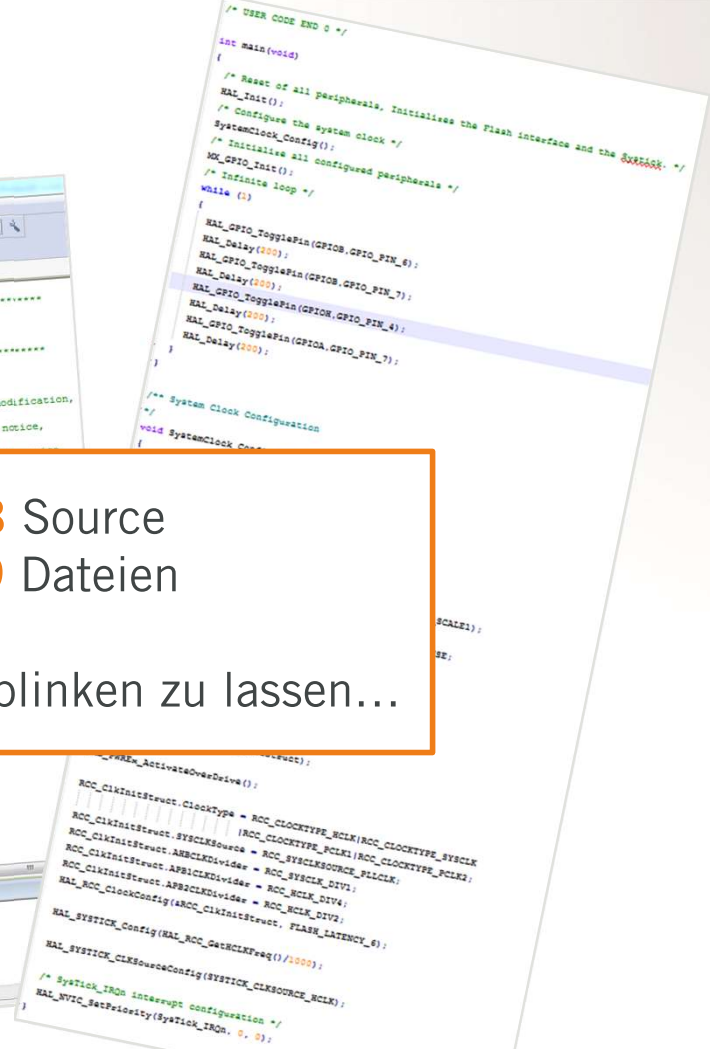


Viel Low-Level-Code

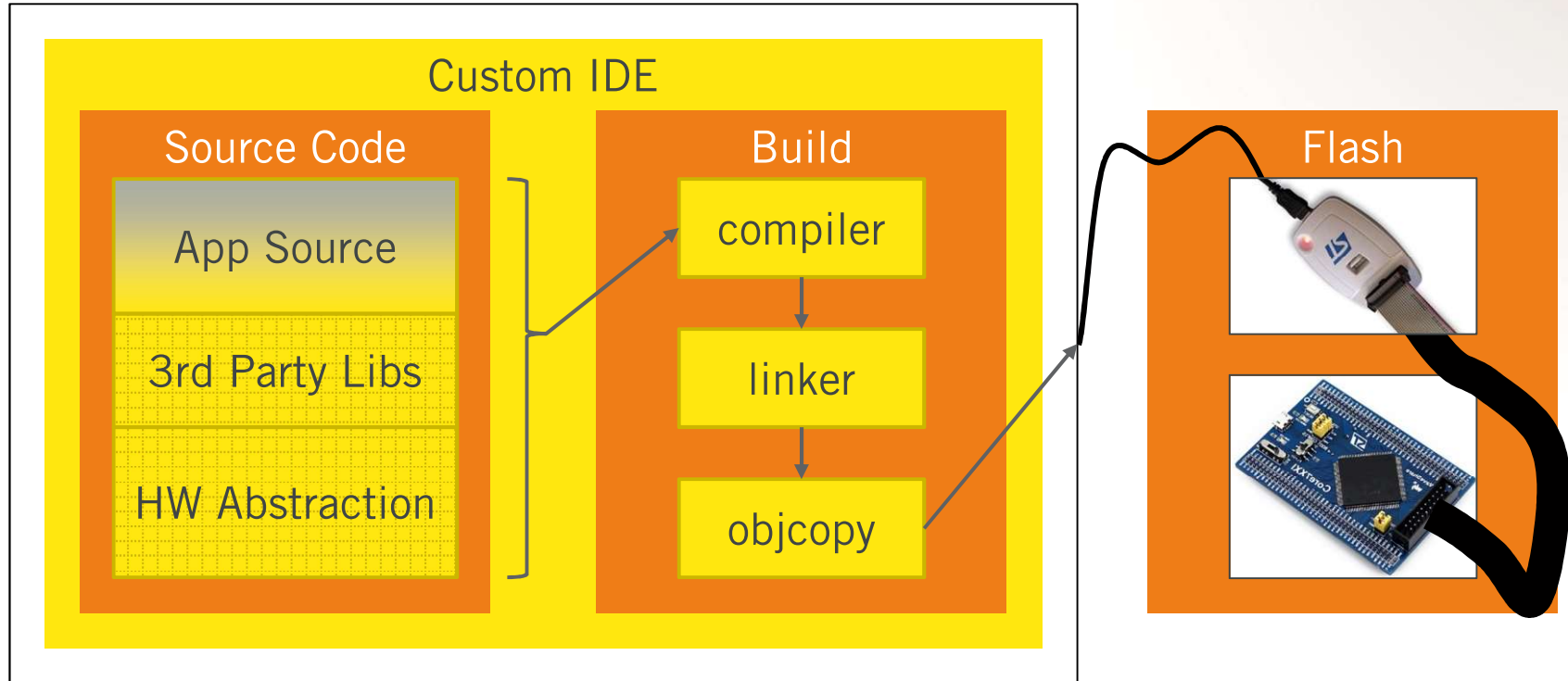


32MB Source
in 509 Dateien

... um eine LED blinken zu lassen...



Schnell mal 'ne LED blinken lassen auf einem neuen Target



Schnell mal 'ne LED blinken lassen auf einem neuen Target mit MicroPython

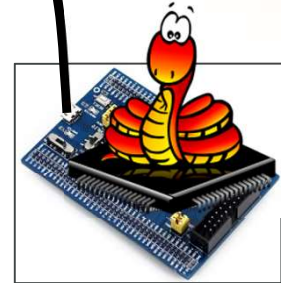
main.py

```
import pyb, time
```


```
while True:  
    pyb.LED(1).toggle()  
    time.sleep(1)
```

Serial Device

Mass Storage



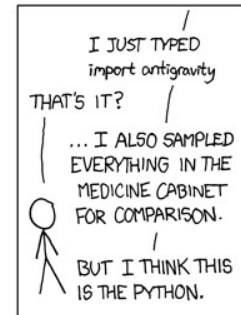
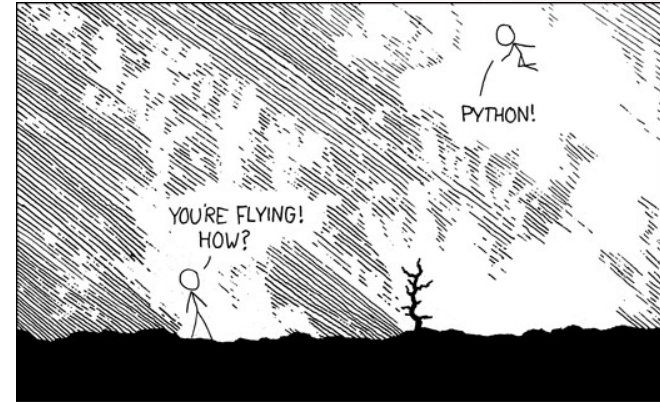
(Micro-)Python



Was macht Python so interessant?

Mit Python kann man schnell und effizient Aufgaben umsetzen

- Interpretiert
- Viele Libraries für alles Mögliche...
- Plattformunabhängig
- Multiparadigmatisch
- Dynamisch typisiert
- Kurz und bündig im Stil
- Automatisierte Speicherverwaltung
- Open-Source, GPL-Kompatibel
- Beliebteste Sprache gem. IEEE



<https://xkcd.com/353/>

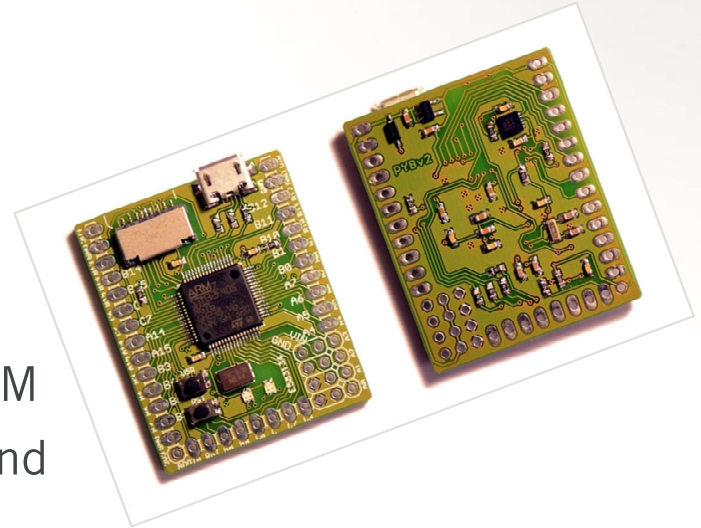
Geschichte von MicroPython

Schlanke Implementation von Python 3, optimiert für Mikrocontroller



Kickstarter-Projekt 2013 lanciert
von Damien George
97.803 £ finanziert
Ziel: 15.000 £

- MicroPython Board (MCU:STM32F405RG)
 - Cortex-M4F / 1024KB Flash / 192KB RAM
 - built-in USB zeigt serielle Schnittstelle und Massenspeicher
- Open Source, MIT License

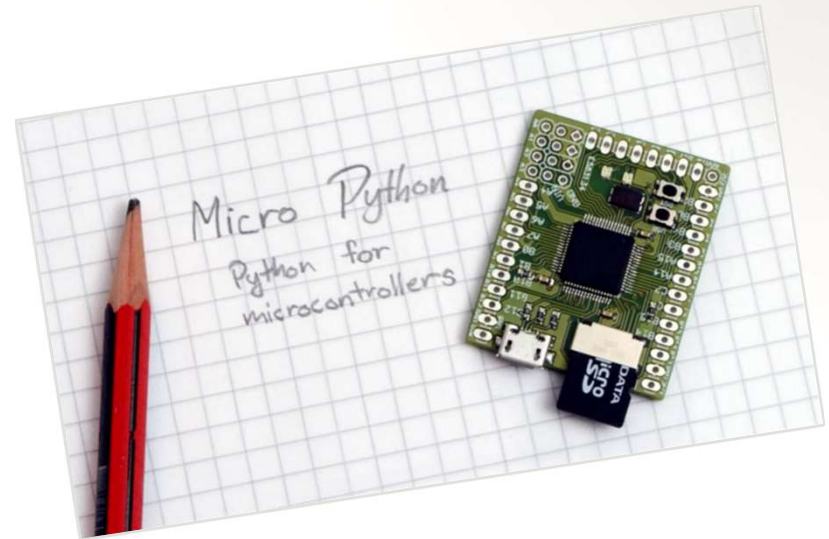


<https://www.kickstarter.com/projects/214379695/micro-python-python-for-microcontrollers/description>
<https://github.com/micropython/micropython>

MicroPython Sprachfeatures

- Objektorientiert, Funktional, Prozedural, ...egal!
- List- / String- / File-Handling
- Interaktive Konsole
- Exception Handling, Error Backtrace
- Garbage Collection
- Multithreading
- ...

... ähnlich wie beim «grossen» Python



Liste von Unterschieden zu Python3: [Github.com/micropython](https://github.com/micropython)

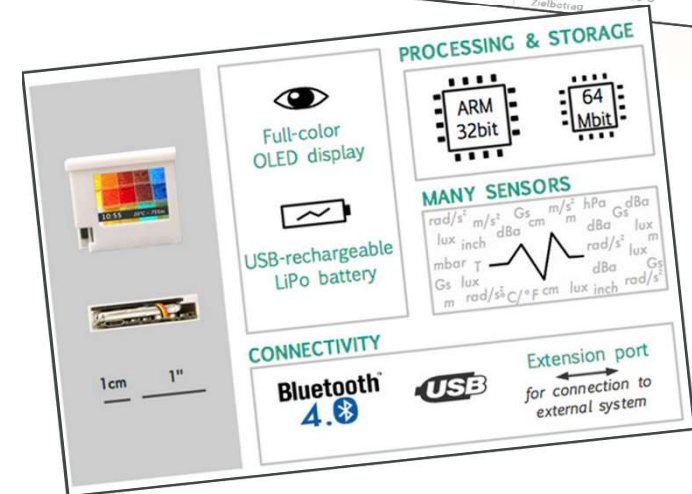
Wie ich MicroPython entdeckt habe

LimiFrog

- Kickstarter Projekt 2015
- MicroPython geplant

Ich war von beiden Projekten begeistert

→ Portiere MicroPython auf STM32L475
(bisher auf STM32F407 verfügbar)



Beispiel-Boards mit MicroPython



Pyboard D-series (ab 45 USD)

- STM32F722 (216MHz M7)
2.5 MB Flash, 256 kB RAM
- WiFi / Bluetooth
- Micro SD Card slot
- 46 GPIOs, 2x I²C, 4x UART,
3x SPI, 1x CAN
- 12-bit ADC & DAC



OpenMV H7 (65.00 USD)

- STM32H743 (480MHz M7)
- SPI, I²C, CAN, Serial, 12-bit
ADC / DAC
- Bildsensor (VGA, 16-bit RGB,
60 fps)

Python Machine-Vision-
Library
für Feature / Eye Detection,
Blob Tracking, Barcode
Decoding, Neural Networks...



µGame

- Atmel SAMD21
(48MHz M0+)
- 32kB RAM, 2MB Flash
- 1.44" TFT display
- 6 Buttons + Speaker

Unterstützte Plattformen / Boards

- bare-arm
- cc3200
 - LAUNCHXL
 - WIPY
- esp32
- esp8266
- javascript
- nrf
 - arduino_p
 - blueio_tag
 - dvk_bl652
 - evk_nina_b
 - feather52
 - ibk_blyst_na
 - idk_blyst_na
 - microbit
 - particle_xenon
 - pca10000
 - pca10001

- pca10028
- pca10031
- pca10040
- pca10056
- wt51822_s4at

- HYDRABUS
- LIMIFROG
- MIKROE_CLICKER

- PYBD_SF3
- PYBD_SF6
- PYBD_SF10

- ARM Cortex M4, M0, M0+, ESP86/ESP32, PIC16, (auch für Windows, Linux)
- Minimal 256kB Flash & 16kB RAM
- Latenz: Startup 150µs, Interrupt 9 µs

Siehe auch ElektronikNet.de

- STM32L475E_IOT01A
- CERB40
- ESPRUIINO_PICO

- NUCLEO_L432KC
- NUCLEO_L452RE
- NUCLEO_L476RG
- NUCLEO_WB55
- OLIMEX_E407
- PYBD_SF2

- teensy
- unix
- windows
- zephyr

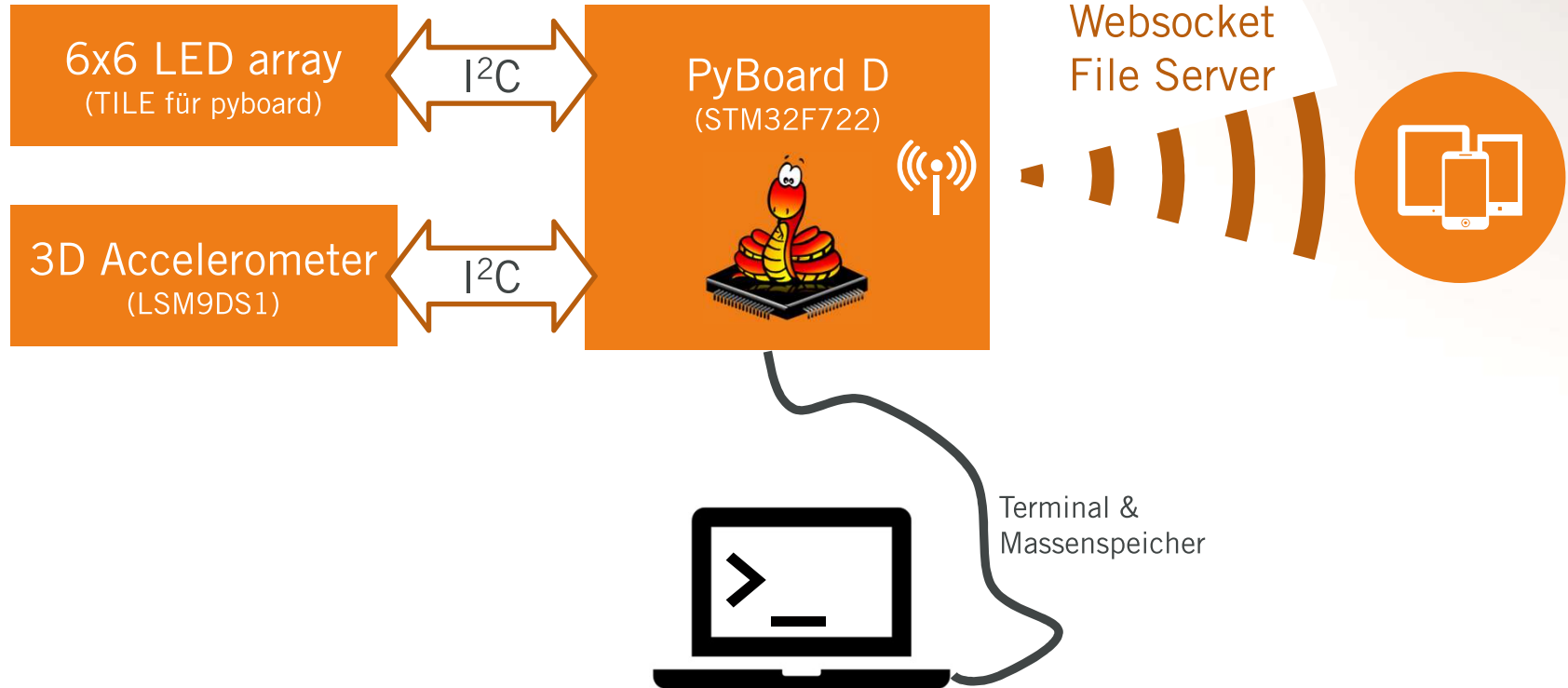
A person wearing a white long-sleeved shirt is seated at a wooden desk, writing on a notepad with a red pen. The notepad contains handwritten notes and diagrams. To the left, a laptop is open, and a tablet is visible in the background. The scene is brightly lit, suggesting an office or classroom environment. A semi-transparent white box with the text "Live-Demo" is overlaid on the left side of the image.

Live-Demo

Hardware-Übersicht



Live-Demo: Setup



Peripherie über I²C ansteuern (Led36)

Plug & Play des Devices (USB Drive, Virtual Com Port)

→ Konsole auf Com-Port unterstützt Auto-Completion!

LED36:

- Kleiner Aufsteckprint mit 36 LED die von einem eigenen Prozessor kontrolliert werden
- Kommunikation über Embedded Bus (I²C) mit definierter Adresse (0x60)
- Ein Mikrokontroller hat meist mehrere I²C Buse: 1, 2, ...

Commands für LED36

4.1 Basic commands

Table 1: Basic Commands

Type	Cmd	nParam	Description	Notes	Result
1	-	-	trigger text buffer roll		
2	0x11	1	set scroll text		
2	0x14	1	set matrix rotation	0..3	
2	0x16	1	set brightness	0..255 0..255%	
2	0x1a	0	enter bootloader		
2	0x2e	0	roll textbuf		
2	A	3	set pixel	R G B	
2	X	2	pos	P1..P2: <x><y>	
2	Y	4	sysres	P1 0xdeadbeef:int32	
2	c	6	set_text_color	fgred fg_green fg_blue bg_red bg_green bg_blue	
2	f	1	save to flash	P1 n: saves NVRAM to flash, N: factory reset	
2	g	1	get status, P1: ['1','2',-]	'1': get SW version, '2': get serial & I2C, -: <w><h><hf>0x4c	31, 20 or 4 bytes
2	i	3	set matrix	R G B	
2	k	?	text_buf colorwheel	P1: is number of bytes to follow, reset text_buf index	
2	l	?	text_buf	P1: is number of bytes to follow, reset text_buf index	
2	m	?	fill matrix	P1: is number of bytes to follow (RGB gamma corrected)	
2	n	?	fill matrix raw	P1: is number of bytes to follow (RGB)	



Peripherie über I²C ansteuern (Led36)

```
import machine, pyb

# Speisung auf Erweiterungsboard aktivieren
pyb.Pin('EN_3V3').on()

i2c1 = machine.I2C(1) # Instanz des Buses 1
i2c1.scan() # Schaue was an diesem Bus angeschlossen ist
# Adresse 60 ist das LED Tile

i2c1.writeto(1, b'\x01') # Tile initialisieren
i2c1.writeto(60, b'\x02\x16\x64') # Helligkeit konfigurieren
i2c1.writeto(60, b'\x02i\xff\x00\x00') # Rot
i2c1.writeto(60, b'\x02i\x00\xff\x00') # Grün
i2c1.writeto(60, b'\x02i\x00\x00\xff') # Blau
```

Eine Klasse für Led36

```
class ledtile:
    def __init__(self):
        self.i2c = machine.I2C(1)
        self.i2c.writeto(1, b'\x01') # init tile
        self.i2c.writeto(60, b'\x02\x16\x64') # brightness
    def set_color(self, r, g, b):
        cmd = bytearray(b'\x02i  ')
        cmd[2]=r
        cmd[3]=g
        cmd[4]=b
        self.i2c.writeto(60, cmd)
```

Beschleunigungssensor über I2C auslesen

Wir haben auf dem Device noch einen weiteren I2C Bus:

```
> i2c2 = machine.I2C(2)
```

Mal schauen was da angehängt ist

```
> i2c2.scan()
```

```
# 28: LSM9DS1 von STM: Magnetfeldsensor
```


```
# 92: LPS22HH von STM: Druck- und Temperatursensor
```

```
# 106: LSM9DS1 von STM: Beschleunigungs-/Rotationssensor
```


Accelerometer: Auszug Spec-Sheet

Table 1. Device summary

- ▶ 1 Pin description
- ▶ 2 Module specifications
- ▶ 3 LSM9DS1 functionality
- ▶ 4 Application hints
- ▶ 5 Digital interfaces
- ▶ 6 Register mapping
- ▼ 7 Accelerometer and gyroscope register description
 - ▶ 7.1 ACT_THS (04h)
 - ▶ 7.2 ACT_DUR (05h)
 - ▶ 7.3 INT_GEN_CFG_XL (06h)
 - ▶ 7.4 INT_GEN_THS_X_XL (07h)
 - ▶ 7.5 INT_GEN_THS_Y_XL (08h)
 - ▶ 7.6 INT_GEN_THS_Z_XL (09h)
 - ▶ 7.7 INT_GEN_DUR_XL (0Ah)
 - ▶ 7.8 REFERENCE_G (0Bh)
 - ▶ 7.9 INT1_CTRL (0Ch)
 - ▶ 7.10 INT2_CTRL (0Dh)
 - ▶ 7.11 WHO_AM_I (0Fh)
 - ▶ 7.12 CTRL_REG1_G (10h)
 - ▶ 7.13 CTRL_REG2_G (11h)
 - ▶ 7.14 CTRL_REG3_G (12h)
 - ▶ 7.15 ORIENT_CFG_G (13h)
 - ▶ 7.16 INT_GEN_SRC_G (14h)
 - ▶ 7.17 OUT_TEMP_L (15h), OUT_TEMP_H (16h)
 - ▶ 7.18 STATUS_REG (17h)
 - ▶ 7.19 OUT_X_G (18h - 19h)

 life.augmented

LSM9DS1

iNEMO inertial module:
3D accelerometer, 3D gyroscope, 3D magnetometer

Datasheet - production data



LGA-24L (3.5x3x1.0 mm)

Features

- 3 acceleration channels, 3 angular rate channels, 3 magnetic field channels
- $\pm 2/\pm 4/\pm 8/\pm 16$ g linear acceleration full scale
- $\pm 4/\pm 8/\pm 12/\pm 16$ gauss magnetic full scale
- $\pm 245/\pm 500/\pm 2000$ dps angular rate full scale
- 16-bit data output
- SPI / I²C serial interfaces
- Analog supply voltage 1.9 V to 3.6 V
- "Always-on" eco power mode down to 1.9 mA
- Programmable interrupt generators
- Embedded temperature sensor
- Embedded FIFO

Applications

- Indoor navigation
- Smart user interfaces
- Advanced gesture recognition
- Gaming and virtual reality input devices
- Display/map orientation and browsing

Description

The LSM9DS1 is a system-in-package featuring a 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor.

The LSM9DS1 has a linear acceleration full scale of $\pm 2g/\pm 4g/\pm 8/\pm 16$ g, a magnetic field full scale of $\pm 4/\pm 8/\pm 12/\pm 16$ gauss and an angular rate of $\pm 245/\pm 500/\pm 2000$ dps.

The LSM9DS1 includes an I²C serial bus interface supporting standard and fast mode (100 kHz and 400 kHz) and an SPI serial standard interface.

Magnetic, accelerometer and gyroscope sensing can be enabled or set in power-down mode separately for smart power management.

Accelerometer: Auszug Spec-Sheet

Accelerometer and gyroscope register description

LSM9DS1

7.31 OUT_X_XL (28h - 29h)

Linear acceleration sensor X-axis output register. The value is expressed as a 16-bit word in two's complement.

7.32 OUT_Y_XL (2Ah - 2Bh)

Linear acceleration sensor Y-axis output register. The value is expressed as a 16-bit word in two's complement.

7.33 OUT_Z_XL (2Ch - 2Dh)

Linear acceleration sensor Z-axis output register. The value is expressed as a 16-bit word in two's complement.

Beschleunigungssensor über I2C auslesen

Lineare X-Beschleunigung ist in Device mit Adresse 106 an der Speicher-Adresse 0x28/0x29

```
> val = i2c2.readfrom_mem(106, 0x28, 2) # Lese zwei Bytes
```

Zur Interpretation dieser bytes verwenden wir die struct library

```
> import struct  
> print(struct.unpack("h", val)[0])
```



Wie im grossen Python
So auch File Handling...

Studium des Datenblatt um eine LSM9D1 Klasse zu entwickeln

Webserver Teil1: PyBoard als Accesspoint aufsetzen:

```
import network

wl_ap = network.WLAN(1)
wl_ap.config(essid='Guild42Mp') # set AP SSID
wl_ap.config(password='pybd0123') # set AP password
wl_ap.config(channel=6) # set AP channel
wl_ap.active(1) # enable the AP
```

Webserver Teil 2: Webserver aufsetzen und starten

```
import picoweb  # existing library for web server

site = picoweb.WebApp(__name__)

@site.route("/")
def index(req, resp):
    yield from picoweb.start_response(resp)
    yield from resp.awrite("<h1>Hello Guild</h1>")

site.run(debug=True)
```

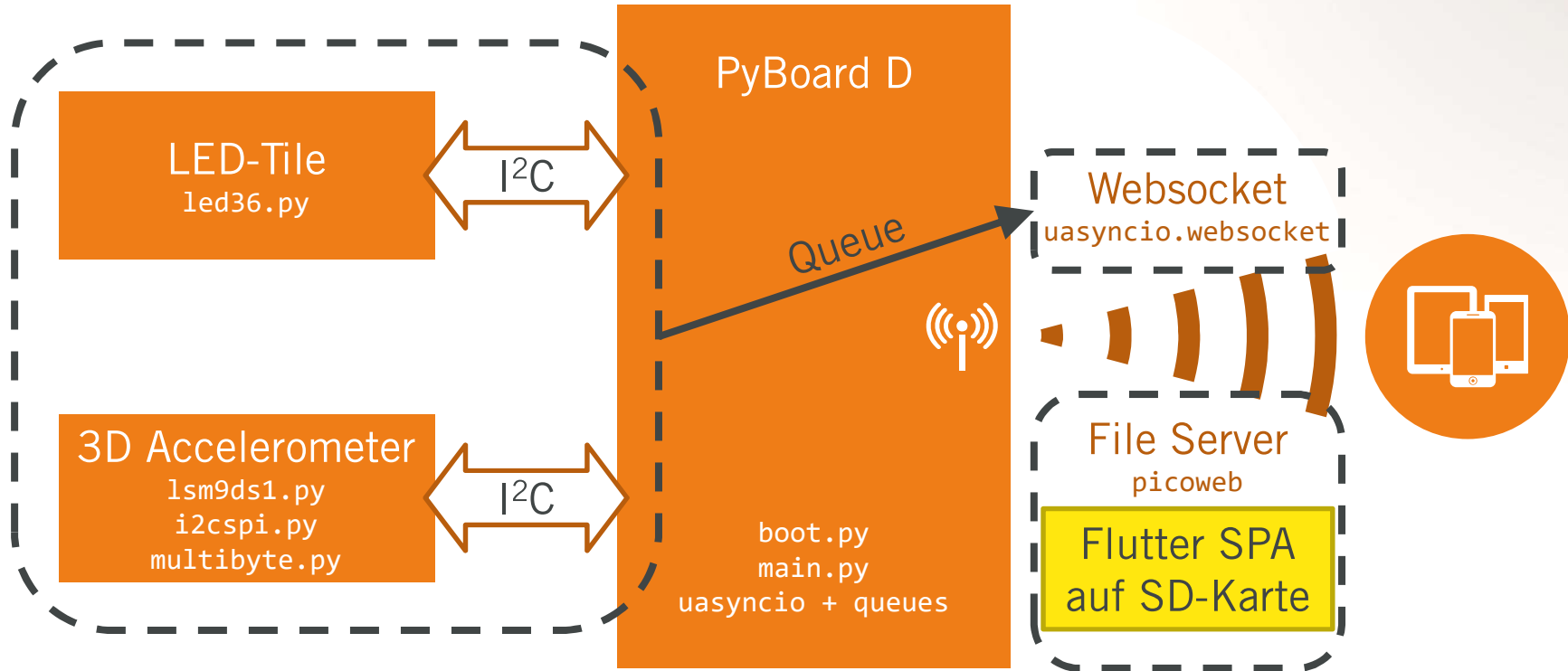
Multitasking mit uasyncio

```
import uasyncio, time

@uasyncio.coroutine
def mytask():
    while True:
        print("Doing something very important")
        time.sleep(1)
        yield

loop = uasyncio.get_event_loop()
loop.create_task(mytask())
loop.run_forever()
```

Live-Demo mit 3 Threads



Einsatz von MicroPython bei Siemens Schweiz AG, Smart Infrastructure

Gerätetests

Hohe Abstraktion von MicroPython

Geringerer Implementationsaufwand

Auch Low-Level-Aufgaben
effizient implementierbar

Steigerung der Testabdeckung

Erhöhung der Produktqualität

Produktinnovation

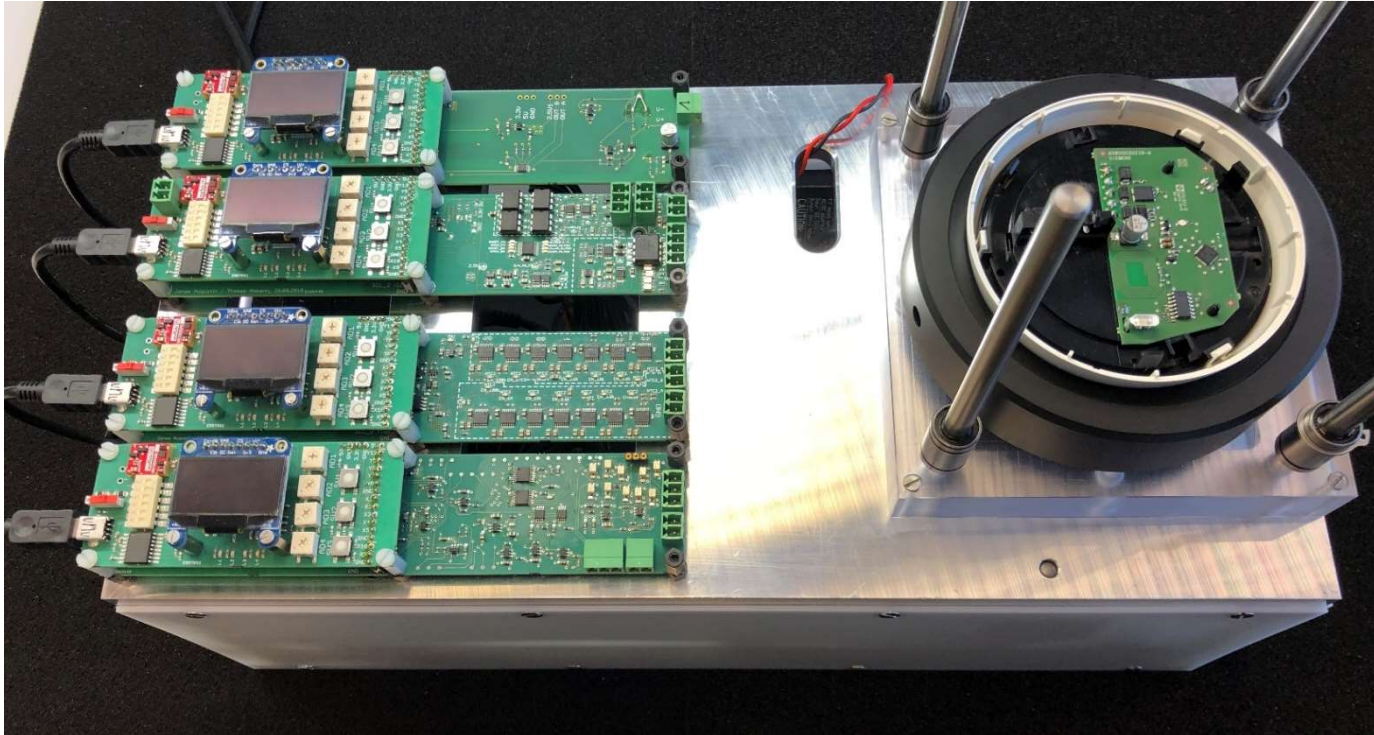
Agiler Entwicklungszyklus

Frühere “Hands-On”-Erfahrung
der Product-Ownerin

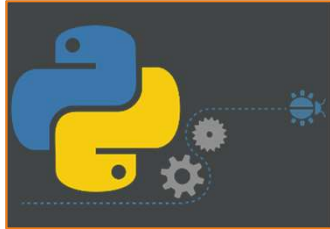
Kürzere Time-to-Market

Rückführung von Inhouse entwickelten Erweiterungen in das offizielle Micropython-Repo zum gegenseitigem Vorteil (Win-Win).

Einsatz von MicroPython bei Siemens Schweiz AG, Smart Infrastructure



Schnellere Entwicklungszyklen



Mächtige High-Level-Features von Python



Kein Kompilieren, flashen, ...



Interaktive Konsole

**Aktive Open Source Community
MIT Lizenz**

Linksammlung

- Blogbeitrag mit allen wichtigen Links
blog.noser.com/embedded-prototyping-mit-micropython/
- Source Code aus Präsentation
github.com/chrismue/pybd_g42_fs
github.com/DanielZ87/MicroPythonDemo_Flutter
- Offizielle Homepage
micropython.org/
github.com/micropython/micropython
github.com/micropython/micropython-lib

Fragen?

**«Gemeinsam treiben
wir weltweit
Innovationen voran –
we know how.»**