Running Python on ESP devices with NuttX

Who am I?

Eren Terzioğlu

Computer Engineer from Yildiz Technical University. Software engineer at Espressif Systems. Maker, Electronics Hobbyist

/erenterzioglu
/eren-terzioglu

/eren-terzioglu

What is NuttX?

NuttX is a real-time operating system (RTOS) with an emphasis on standards compliance and small footprint. Scalable from 8-bit to 64-bit microcontroller environments, the primary governing standards in NuttX are POSIX and ANSI standards.

- Source: Apache NuttX

Operating system	**First commit**	**Governance**	**License**	**Contributors**	**Pulse (jun10/2024)**
Zephyr	2014	community	Apache 2.0	100+	942
NuttX	2007	community	Apache 2.0	100+	135
RT-Thread	2009	community	Apache 2.0	100+	67
RIOT	2010	community	LGPL2.1	100+	71
Tyzen RT	2015	Samsung	Apache 2.0	100+	36
myNewt	2015	Community	Apache 2.0	100+	25
mbed OS	2013	ARM	Apache 2.0 or BSD-3 Clause	100+	7
FreeRTOS	2004	Richard Barry	MIT	100+	6
Contiki-NG	2016	community	BSD-3 Clause	100+	4
CMSIS-5	2016	ARM	Apache 2.0	100+	0
Azure-RTOS	2020	Microsoft	Microsoft Software License	10+	archived

NuttX is the 2nd most popular community-based RTOS (along with Zephyr in the 1st position):

— Table by Alin Jerpelea, presented on NuttX Workshop 2024

Advantages of NuttX

- Apache 2.0 Licensed
- Small footprint
- Variety of architecture support (from Z80 to 64 bit RISC-V)
- Community support
- POSIX compliant
- C, C++, Zig, Rust compiled based languages support
- Lua, BASIC and now, Python interpreted languages support

arm	risc-v
arm64	sim
avr	sparc
ceva	tricore
dummy	x86
hc	x86_64
mips	xtensa
misoc	z16
or1k	z80
renesas	

Supported architectures on NuttX

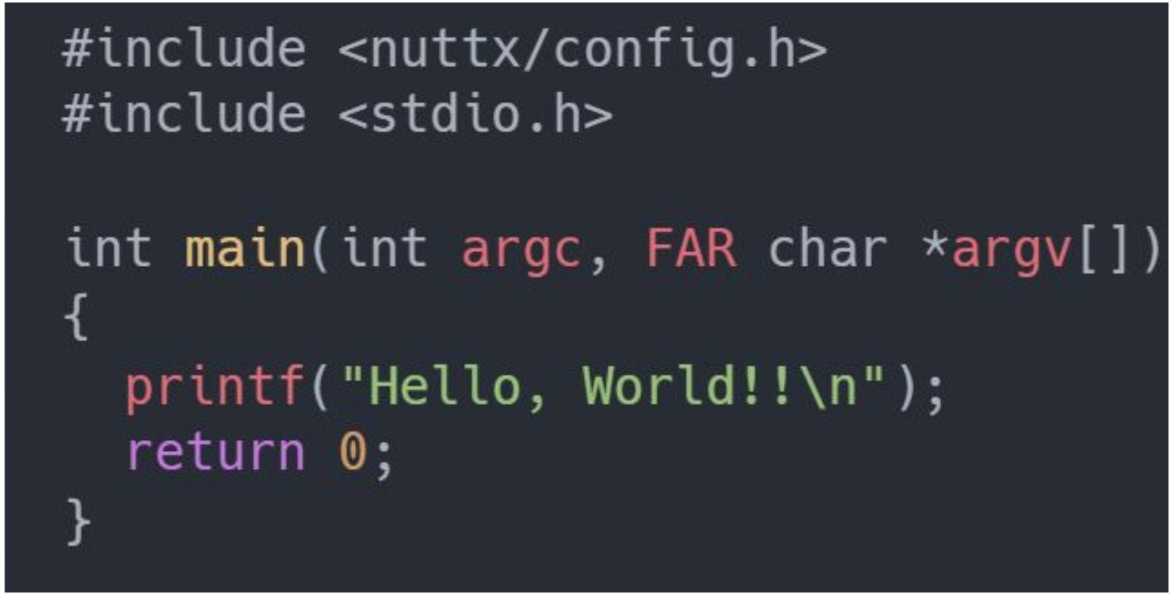
Projects using NuttX

- PX4 autopilot drones.
- Pixhawk an advanced, User-Friendly Autopilot.
- OsmocomBB uses NuttX to develop an operating system for cell phones.
- Haltian's Thingsee IoT gateway devices use the ThingseeOS operating system, which is based on NuttX.
- Motorola Moto Z.
- Sony is using NuttX in their audio processors.
- Sony is using NuttX in the Spresense development board.
- Samsung TizenRT based on NuttX RTOS.
- Xiaomi Vela, an IoT software platform based on NuttX.

- Source NuttX - Wikipedia

Sample Application on NuttX

nuttx-apps/examples/hello/hello_main.c



```c
#include <nuttx/config.h>
#include <stdio.h>

int main(int argc, FAR char *argv[])
{
  printf("Hello, World!!\n");
  return 0;
}
```

Why Python on NuttX

- Developers outside traditional embedded programming gain access to a familiar ecosystem for building embedded applications, supported by Python’s vast library ecosystem and open-source tools.
- On the other hand, NuttX provides a POSIX-compatible standardized interface which makes possible to manipulate the actual hardware supported by NuttX. Buses and other peripherals can be accessed directly by the Python applications.

- Python wasn’t designed to run on resource-constrained devices?
  - Recent changes to the Python project, especially targeting WebAssembly, made Python more “friendly” regarding memory usage and other system requirements, making it more suitable for resource-constrained devices.

How?

- The Python interpreter and its internal libraries are provided by the CPython project which is written in C and use POSIX interfaces to access system-provided resources. For that reason being a POSIX-compatible RTOS, could be a target system for building Python!

- Python needs to be cross-compiled for our target hardware that will run the NuttX RTOS. For that reason WASI build is best matching option to choose

https://tmedicci.github.io/articles/2025/01/08/python_on_nuttx.html

How to Use Python on NuttX

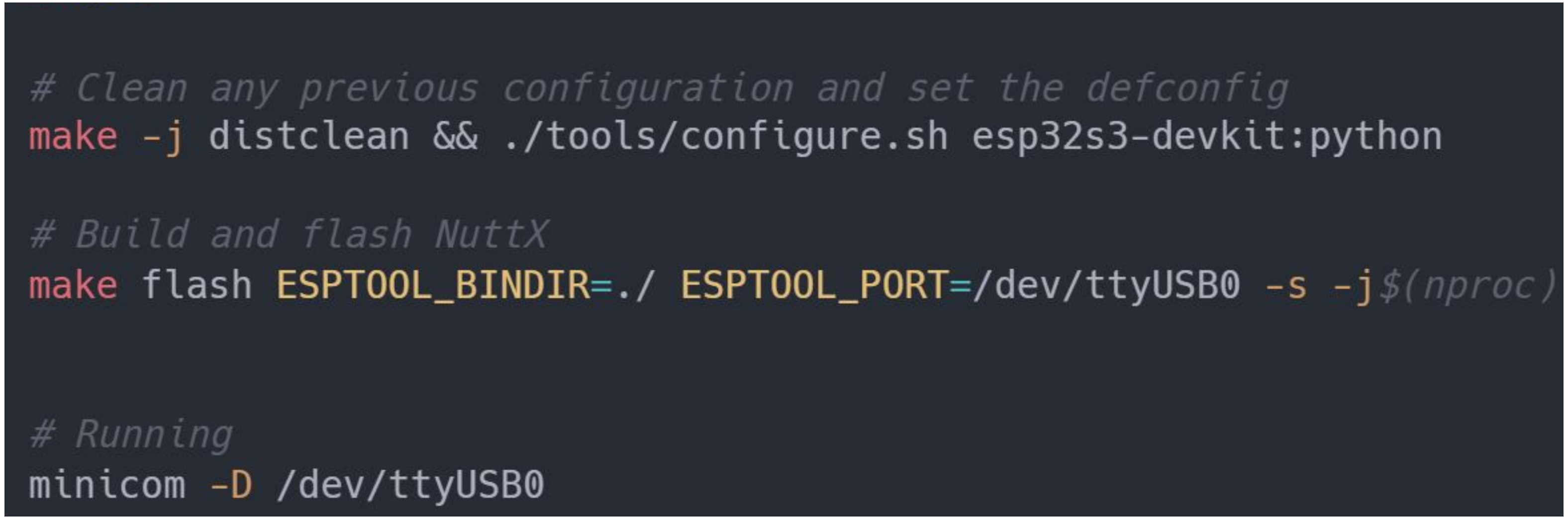
Requirements:

- ESP32-S3 board with at least 16MiB of flash and an external PSRAM of 8MB or more is required to run Python (ESP32-S3-WROOM-2-N32R8V can/will be used).
- NuttX development environment.

https://developer.espressif.com/blog/nuttx-getting-started/

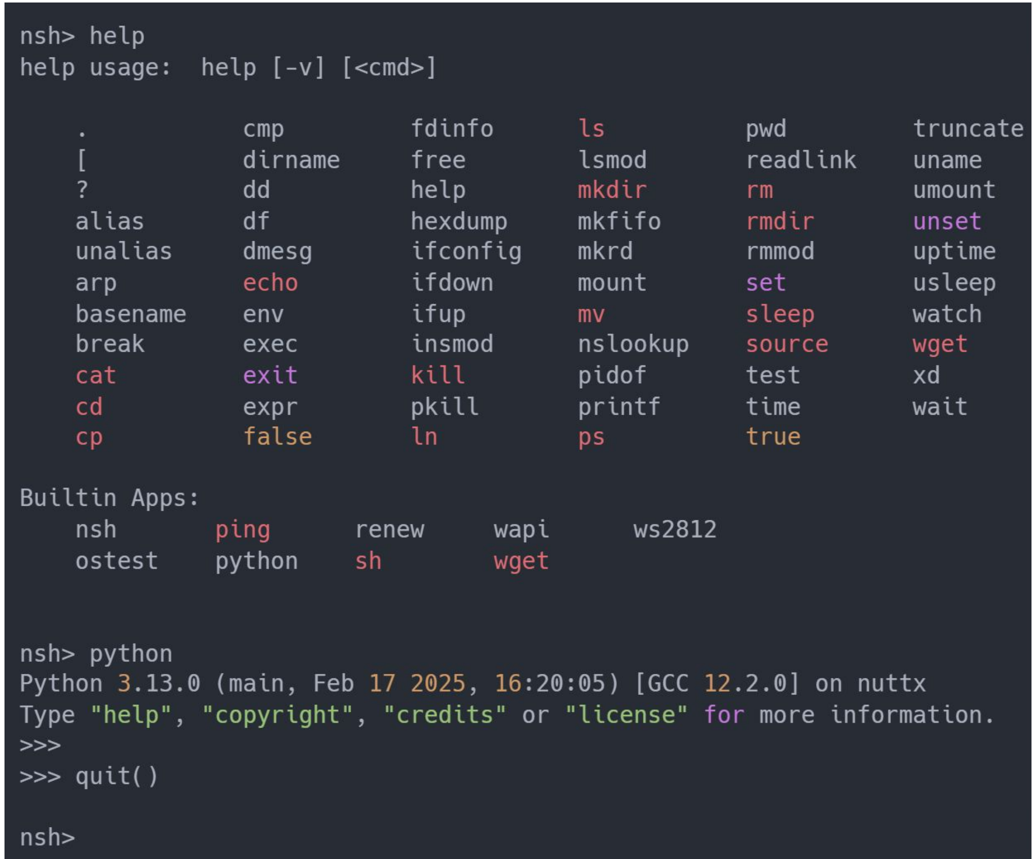
Note: Currently, ESP32-S3 and RISC-V based QEMU targets supported to use Python on NuttX.

Compiling, Flashing and Running



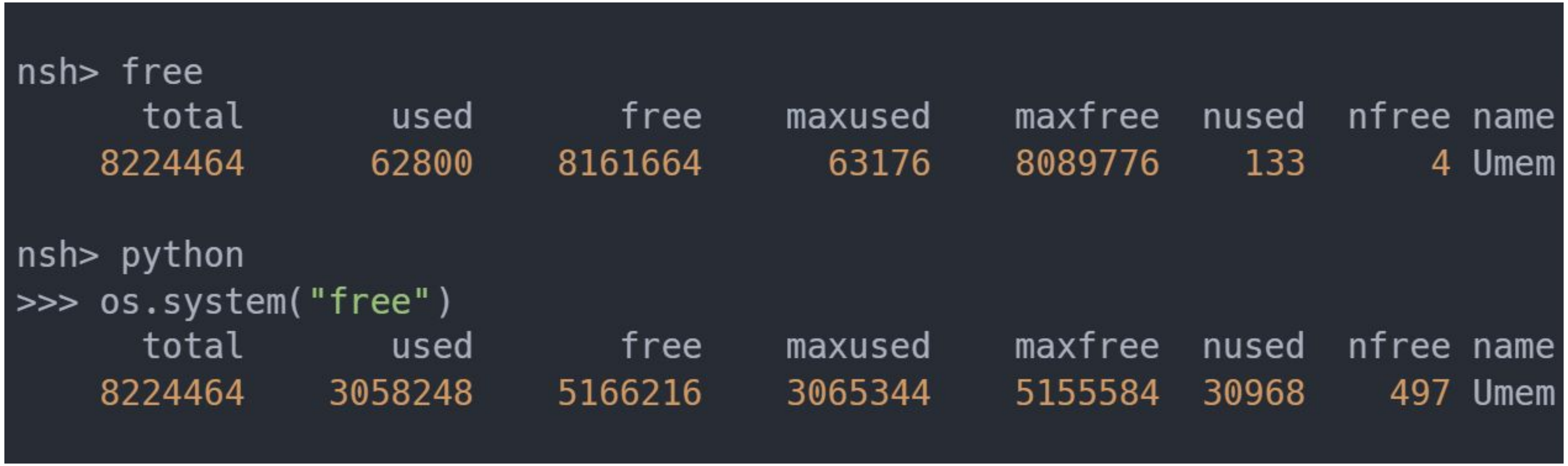
```
# Clean any previous configuration and set the defconfig
make -j distclean && ./tools/configure.sh esp32s3-devkit:python

# Build and flash NuttX
make flash ESPTOOL_BINDIR=./ ESPTOOL_PORT=/dev/ttyUSB0 -s -j$(nproc)


# Running
minicom -D /dev/ttyUSB0
```

Compiling, Flashing and Running



```
nsh> help
help usage:  help [-v] [<cmd>]

    .           cmp         fdinfo      ls          pwd         truncate
    [           dirname     free        lsmod       readlink    uname
    ?           dd          help        mkdir       rm          umount
    alias       df          hexdump     mkfifo      rmdir       unset
    unalias     dmesg       ifconfig    mkrd        rmmod       uptime
    arp         echo        ifdown      mount       set         usleep
    basename    env         ifup        mv          sleep       watch
    break       exec        insmod      nslookup    source      wget
    cat         exit        kill        pidof       test        xd
    cd          expr        pkill       printf      time        wait
    cp          false       ln          ps          true

Builtin Apps:
    nsh         ping        renew       wapi        ws2812
    ostest      python      sh          wget


nsh> python
Python 3.13.0 (main, Feb 17 2025, 16:20:05) [GCC 12.2.0] on nuttx
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> quit()

nsh>
```

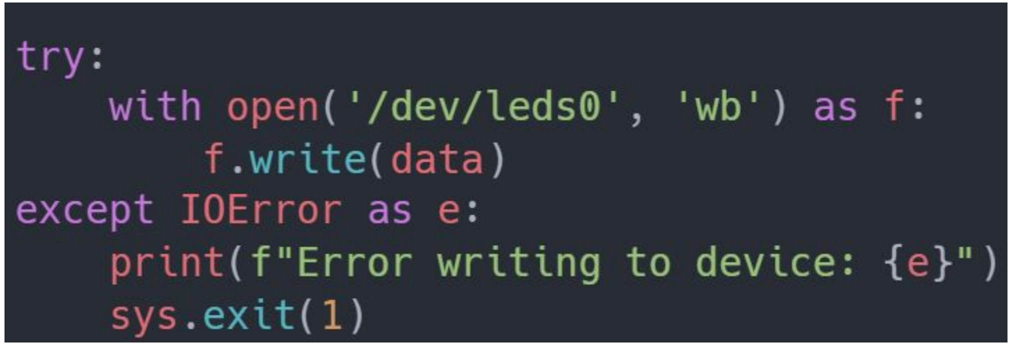
Memory Usage



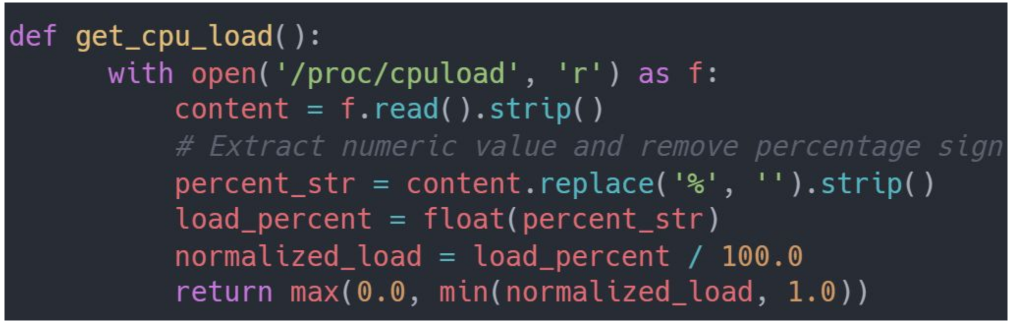
```
nsh> free
      total       used       free    maxused    maxfree  nused  nfree name
    8224464      62800    8161664      63176    8089776    133      4 Umem

nsh> python
>>> os.system("free")
      total       used       free    maxused    maxfree  nused  nfree name
    8224464    3058248    5166216    3065344    5155584  30968    497 Umem
```

Creating a Python Script

- Python’s built-in functions like “open” and “write” can be used to open and write to a character driver directly.
- Also, information about the active tasks and other system information are available through the PROCFS filesystem mounted at “/proc/”, which can be read directly with Python’s read function.

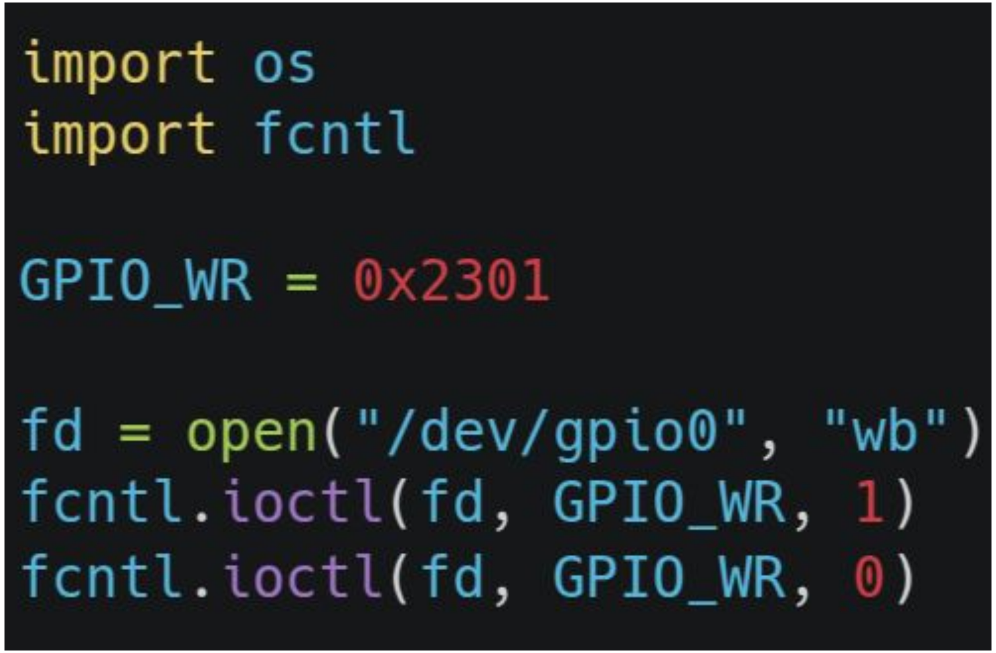


```
try:
    with open('/dev/leds0', 'wb') as f:
        f.write(data)
except IOError as e:
    print(f"Error writing to device: {e}")
    sys.exit(1)
```



```
def get_cpu_load():
      with open('/proc/cpuload', 'r') as f:
          content = f.read().strip()
          # Extract numeric value and remove percentage sign
          percent_str = content.replace('%', '').strip()
          load_percent = float(percent_str)
          normalized_load = load_percent / 100.0
          return max(0.0, min(normalized_load, 1.0))
```

Creating a Python Script



```
import os
import fcntl

GPIO_WR = 0x2301

fd = open("/dev/gpio0", "wb")
fcntl.ioctl(fd, GPIO_WR, 1)
fcntl.ioctl(fd, GPIO_WR, 0)
```

Blink example

Articles and Updates About NuttX?

- NuttX · Developer Portal
- GitHub - apache/nuttx
- GitHub - apache/nuttx-apps
- Lup Yuen LEE's Blog

Questions? Thank You

Sources

- https://nuttx.apache.org/
- https://nuttx.apache.org/docs/latest/
- https://developer.espressif.com/blog/nuttx-getting-started/
- https://tmedicci.github.io/articles/2025/01/08/python_on_nuttx.html
- https://developer.espressif.com/blog/2025/03/nuttx-python-esp32s3
- https://developer.espressif.com/tags/nuttx/
- https://en.wikipedia.org/wiki/Real-time_operating_system
- https://en.wikipedia.org/wiki/POSIX