

# Embinux

---

ENRIQUE CALDERÓN

LUIS UGARTECHEA

*Universidad Nacional Autónoma de México*

*Facultad de Ingeniería*

April 28, 2025



# ¿Qué es Linux?

- ▶ Kernel creado por Linus Torvalds en 1991
- ▶ En unión con GNU (Richard Stallman, 1983) se vuelve un sistema operativo completo
- ▶ FOSS (Free and Open Source Software) basado en UNIX
- ▶ GNU/Linux es el nombre correcto, aunque comúnmente se le llama Linux

# ¿Qué es un sistema embebido?

- ▶ Sistema informático con hardware y software diseñado para función específica
- ▶ Controla funciones dentro de un sistema multifunción más grande
- ▶ Características:
  - ▶ Bajo consumo de energía
  - ▶ Bajo costo
  - ▶ Tamaño reducido
  - ▶ Fácil adaptación para otros dispositivos

# Componentes de sistemas embebidos

- ▶ **Microprocesador:** Núcleo del sistema, ejecuta instrucciones, controla operaciones
- ▶ **Memoria:**
  - ▶ RAM (datos temporales)
  - ▶ Flash (datos permanentes: firmware, SO, configuración)
- ▶ **Software:** La parte “inteligente”, contiene instrucciones que ejecuta el sistema
- ▶ **Periféricos:** Permiten interacción con el exterior (sensores, actuadores, pantallas)

# Linux en sistemas embebidos

- ▶ Opción popular por flexibilidad, escalabilidad, estabilidad y ser open source
- ▶ Arquitectura modular permite adaptación a necesidades específicas
- ▶ Se pueden eliminar componentes innecesarios para ahorrar recursos
- ▶ Acceso a gran comunidad, herramientas y bibliotecas

# Ventajas de Linux embebido

- ▶ Acceso a comunidad, herramientas y bibliotecas
- ▶ Versiones específicas para recursos limitados:
  - ▶ Yocto
  - ▶ Buildroot
  - ▶ OpenWrt
- ▶ Kernel configurable para hardware específico
- ▶ Reducción de tiempo de arranque y uso de memoria
- ▶ Integración de controladores específicos

# Casos comunes de uso

- ▶ **Routers y dispositivos de red:** Enrutadores, switches, firewalls
- ▶ **Televisores inteligentes y set-top boxes:** Smart TVs, dispositivos streaming
- ▶ **Automatización industrial:** PLCs, sistemas SCADA, paneles HMI
- ▶ **Electrodomésticos inteligentes:** Refrigeradores, aspiradoras robot, etc.
- ▶ **Automóviles:** Unidades de control, sistemas de infoentretenimiento
- ▶ **Dispositivos portátiles y wearables:** Relojes inteligentes, GPS, cámaras

# ¿Qué es buildroot?

- ▶ Herramienta de construcción de sistemas Linux embebidos
- ▶ Crea sistema Linux completo a partir de paquetes y configuraciones
- ▶ Utiliza cross-compilation para generar sistema para arquitectura diferente
- ▶ Incluye propio compilador y herramientas de construcción



# Herramientas necesarias

- ▶ **Compilador:** Buildroot puede generar uno basado en GCC (junto con una C standard library como glibc, uClibc-ng o musl).
- ▶ **Herramientas de construcción:** make, tar, gzip, etc.
- ▶ **Sistema operativo:** Cualquier distribución Linux
- ▶ **Conexión a internet:** Para descargar paquetes y herramientas
- ▶ **Hardware:** Computadora con capacidad suficiente (compilación puede tardar horas)

# Selección de hardware

- ▶ Buildroot permite compilar para cualquier arquitectura
- ▶ Compatible con gran variedad de hardware:
  - ▶ Microcontroladores
  - ▶ SBCs (Single Board Computers)
  - ▶ Computadoras de escritorio

# Configuración de buildroot

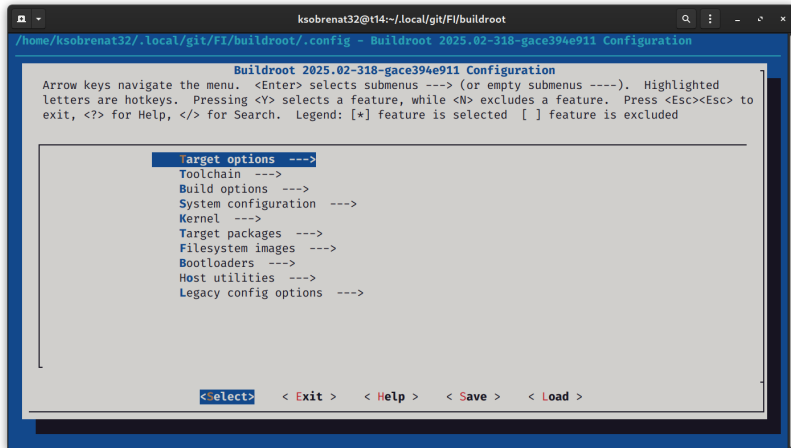
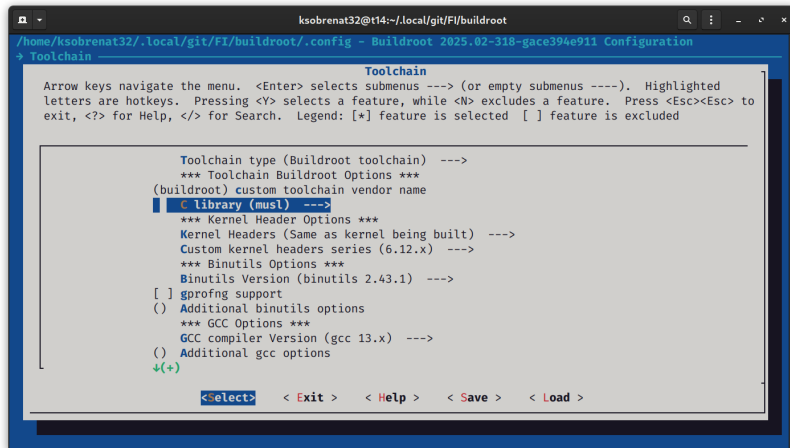


Figure 1: Interfaz de make menuconfig

# Selección de biblioteca C



```
ksobrenat32@t14:~/local/git/FI/buildroot
/home/ksobrenat32/.local/git/FI/buildroot/.config - Buildroot 2025.02-318-gace394e911 Configuration
→ Toolchain

Toolchain
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu ----). Highlighted
letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected [ ] feature is excluded

Toolchain type (Buildroot toolchain) --->
*** Toolchain Buildroot Options ***
(buildroot) custom toolchain vendor name
[*] C library (musl) --->
*** Kernel Header Options ***
Kernel Headers (Same as kernel being built) --->
Custom kernel headers series (6.12.x) --->
*** Binutils Options ***
Binutils Version (binutils 2.43.1) --->
[ ] gprofng support
() Additional binutils options
*** GCC Options ***
GCC compiler Version (gcc 13.x) --->
() Additional gcc options
↓(+)
```

<Select> < Exit > < Help > < Save > < Load >

Figure 2: Configuración de musl

# Compilación

- ▶ Comando `make` inicia la compilación
- ▶ Genera imagen de disco para instalación
- ▶ No requiere compilador externo
- ▶ Detecta automáticamente número de núcleos

# Instalación

- ▶ Imagen de disco en carpeta `output/images`
- ▶ Se instala en el hardware seleccionado
- ▶ Método de instalación depende del hardware

# Ejemplo práctico: Raspberry Pi 4

## Hardware seleccionado

- ▶ Raspberry Pi 4 de 4GB
- ▶ Procesador ARM Cortex-A72
- ▶ Single Board Computer (SBC)
- ▶ Soporte para HDMI, USB y Ethernet
- ▶ Comunicación por conexión serial (adaptador USB a TTL)

# Configuración de buildroot

```
git clone https://github.com/buildroot/buildroot.git  
make raspberrypi4_64_defconfig
```

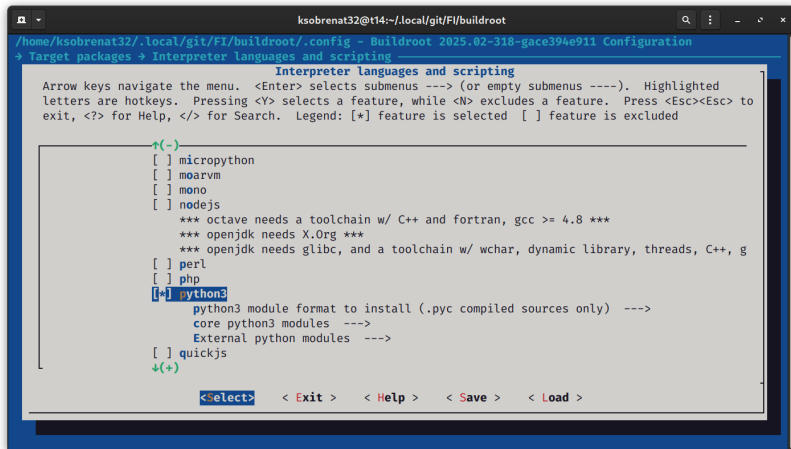
- ▶ Configuración base para Raspberry Pi 4
- ▶ Genera archivo .config con opciones necesarias



# Configuración de herramientas

- ▶ `make menuconfig` para configuración adicional
- ▶ Selección de `musl` en lugar de `glibc` (menor tamaño)
- ▶ Aplicaciones incluidas:
  - ▶ `vim` (editor de texto)
  - ▶ `python3` (intérprete)
  - ▶ `htop` (monitoreo del sistema)

# Configuración de aplicaciones



```
ksobrenat32@t14:~/local/git/FI/buildroot
/home/ksobrenat32/.local/git/FI/buildroot/.config - Buildroot 2025.02-318-gace394e911 Configuration
→ Target packages → Interpreter languages and scripting

Interpreter languages and scripting
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted
letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected [ ] feature is excluded

↑(-)
[ ] micropython
[ ] moarvm
[ ] mono
[ ] nodejs
    *** octave needs a toolchain w/ C++ and fortran, gcc >= 4.8 ***
    *** openjdk needs X.Org ***
    *** openjdk needs glibc, and a toolchain w/ wchar, dynamic library, threads, C++, g
[ ] perl
[ ] php
[*] python3
    python3 module format to install (.pyc compiled sources only) --->
    core python3 modules --->
    External python modules --->
[ ] quickjs
↓(+)
```

<Select>   <Exit>   <Help>   <Save>   <Load>

Figure 3: Instalación de Python3

# Configuración de aplicaciones

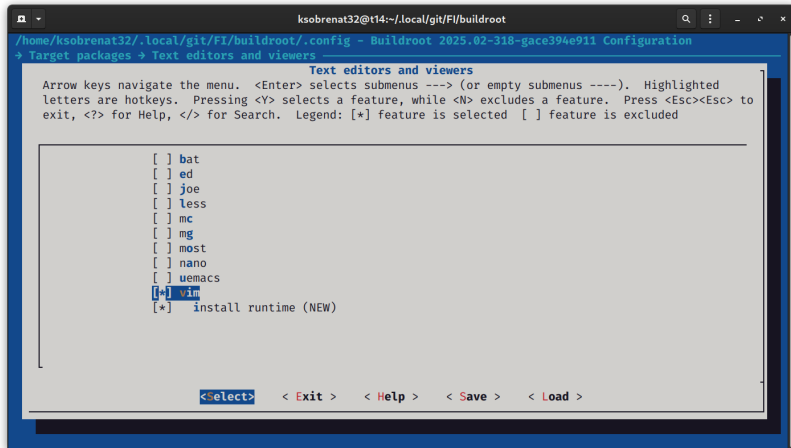


Figure 4: Instalación de Vim

# Configuración de aplicaciones

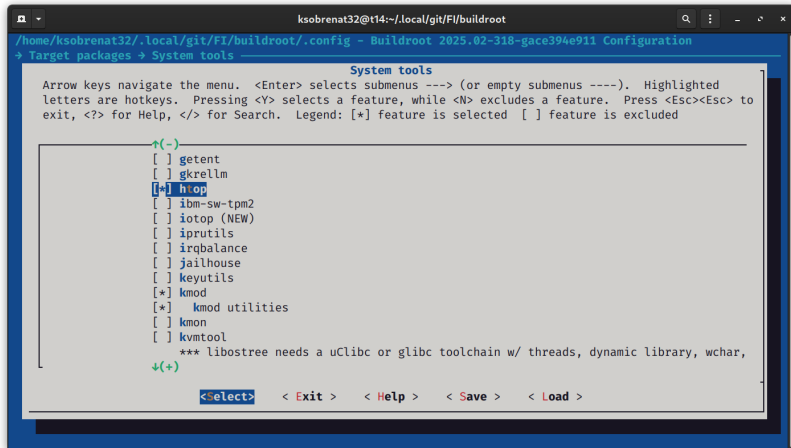


Figure 5: Instalación de Htop

# Compilación

```
ksobrenat32@t14:~/local/git/FI/buildroot
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/bcm2711-rpi-cm4.dtb' ':' (stderr):
INFO: vfat(boot.vfat): adding file 'bcm2711-rpi-cm4s.dtb' as 'bcm2711-rpi-cm4s.dtb' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/bcm2711-rpi-cm4s.dtb' ':'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/cmdline.txt' as 'rpi-firmware/cmdline.txt' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/rpi-firmware/cmdline.txt' ':'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/config.txt' as 'rpi-firmware/config.txt' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/rpi-firmware/config.txt' ':'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/fixup4.dat' as 'rpi-firmware/fixup4.dat' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/rpi-firmware/fixup4.dat' ':'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/overlays' as 'rpi-firmware/overlays' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/rpi-firmware/overlays' ':'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/start4.elf' as 'rpi-firmware/start4.elf' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/rpi-firmware/start4.elf' ':'" (stderr):
INFO: vfat(boot.vfat): adding file 'Image' as 'Image' ...
INFO: vfat(boot.vfat): cmd: "MTTOOLS_SKIP_CHECK=1 mcopy -sp -i '/home/ksobrenat32/.local/git/FI/buildroot/output/im
ages/boot.vfat' '/home/ksobrenat32/.local/git/FI/buildroot/output/images/Image' ':'" (stderr):
INFO: hdiimage(sdcard.img): adding primary partition 'boot' (in MBR) from 'boot.vfat' ...
INFO: hdiimage(sdcard.img): adding primary partition 'rootfs' (in MBR) from 'rootfs.ext4' ...
INFO: hdiimage(sdcard.img): adding primary partition '[MBR]' ...
INFO: hdiimage(sdcard.img): writing MBR
INFO: cmd: "rm -rf '/home/ksobrenat32/.local/git/FI/buildroot/output/build/genimage.tmp'" (stderr):
[ksobrenat32@t14]:buildroot $
```

Figure 6: Compilación exitosa

# Instalación

```
sudo dd if=output/images/sdcard.img \
of=/dev/sdX bs=4M status=progress
sync
```

- ▶ Copiar imagen a tarjeta SD
- ▶ Insertar en Raspberry Pi 4
- ▶ Encender y comprobar funcionamiento

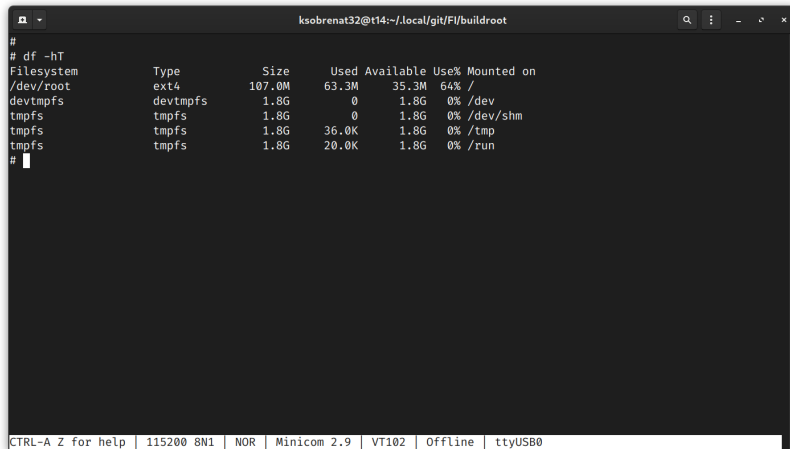
# Pruebas realizadas

```
ksobrenat32@t14:~/local/git/Fl/buildroot
[ 2.870505] usb 1-1: New USB device found, idVendor=2109, idProduct=3431, bcdDevice= 4.21
[ 2.878720] usb 1-1: New USB device strings: Mfr=0, Product=1, SerialNumber=0
[ 2.885865] usb 1-1: Product: USB2.0 Hub
[ 2.891102] hub 1-1:1.0: USB hub found
[ 2.895096] hub 1-1:1.0: 4 ports detected
[ 2.931439] EXT4-fs (mmcblk0p2): re-mounted c3fb2e13-c045-425c-9d59-b0ead3ae572a r/w. Quota mode: none.
Saving 256 bits of creditable seed for next boot
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: [ 3.012189] bcmgenet fd580000.ethernet: configuring instance for external RGMII (RX delay)
[ 3.021339] bcmgenet fd580000.ethernet eth0: Link is Down
udhcpd: started, v1.37.0
udhcpd: broadcasting discover
udhcpd: no lease, forking to background
OK
Starting crond: OK

Welcome to Buildroot
buildroot login: root
# cat /etc/os-release
NAME=Buildroot
VERSION=2025.02-318-gace394e911
ID=buildroot
VERSION_ID=2025.05-git
PRETTY_NAME="Buildroot 2025.05-git"
#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.9 | VT102 | Offline | ttyUSB0
```

Figure 7: Arranque del sistema

# Pruebas realizadas



A terminal window titled 'ksobrenat32@t14:~/local/git/FI/buildroot' displays the output of the 'df -hT' command. The output is a table showing disk space usage for various filesystems. The table has columns for Filesystem, Type, Size, Used, Available, Use%, and Mounted on. The data shows that the root filesystem (/dev/root) is ext4 and is 107.0M in size, with 63.3M used and 35.3M available, representing 64% usage. Other filesystems like devtmpfs, tmpfs, and /dev/shm are all 1.8G in size and show 0% usage.

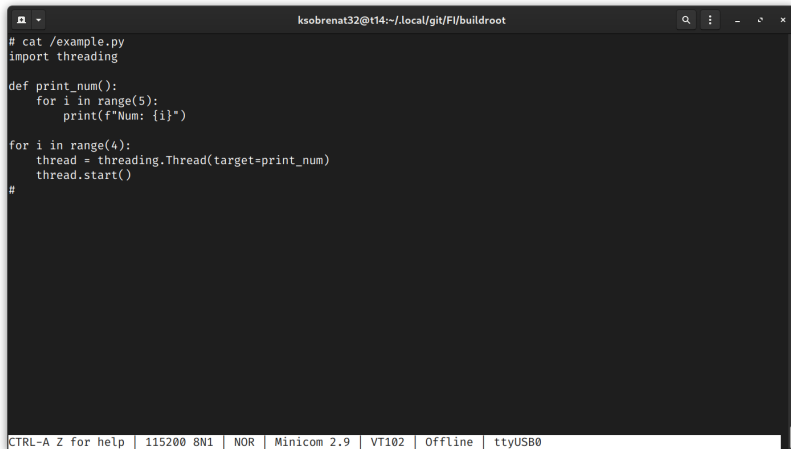
```
# df -hT
Filesystem      Type      Size      Used Available Use% Mounted on
/dev/root       ext4      107.0M    63.3M    35.3M    64% /
devtmpfs        devtmpfs  1.8G      0        1.8G     0% /dev
tmpfs           tmpfs     1.8G      0        1.8G     0% /dev/shm
tmpfs           tmpfs     1.8G     36.0K    1.8G     0% /tmp
tmpfs           tmpfs     1.8G     20.0K    1.8G     0% /run
#
```

At the bottom of the terminal window, a status bar shows: CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.9 | VT102 | Offline | ttyUSB0

Figure 8: Uso del sistema de archivos



# Funcionamiento de Python



A terminal window with a dark background and light gray text. The window title bar shows the user 'ksobrenat32' at 't14' in the directory '~/local/git/Fl/buildroot'. The code displayed is a Python script that uses the 'threading' module to create four threads, each printing numbers 0 to 4. The status bar at the bottom of the terminal displays various system information: 'CTRL-A Z for help', '115200 8N1', 'NOR', 'Minicom 2.9', 'VT102', 'Offline', and 'ttyUSB0'.

```
ksobrenat32@t14:~/local/git/Fl/buildroot
# cat /example.py
import threading

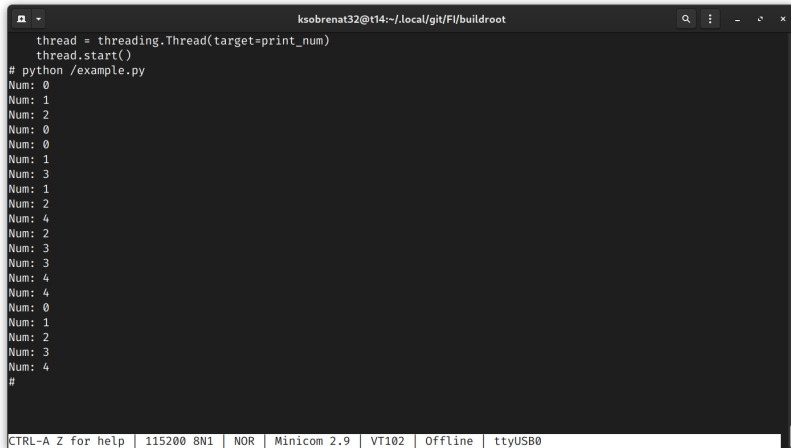
def print_num():
    for i in range(5):
        print(f"Num: {i}")

for i in range(4):
    thread = threading.Thread(target=print_num)
    thread.start()
#
```

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.9 | VT102 | Offline | ttyUSB0

Figure 9: Ejemplo de código Python

# Funcionamiento de Python



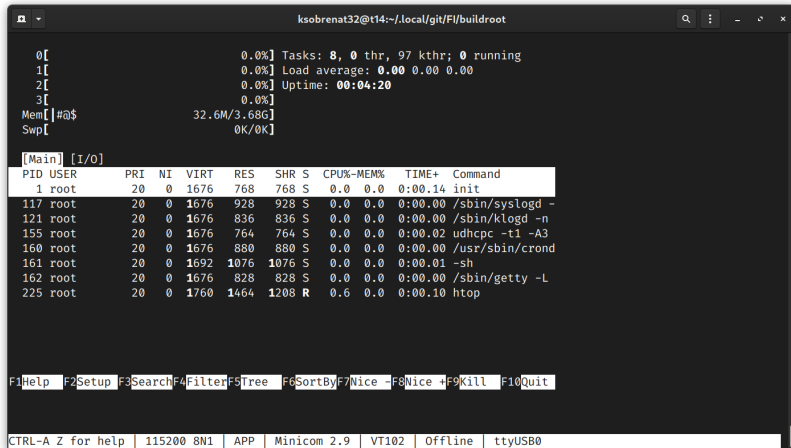
A terminal window with a dark background. The title bar shows the user 'ksobrenat32' at 't14' in the directory '~/local/git/FI/buildroot'. The terminal contains Python code to create and start a thread that prints numbers. Below the code, the output of the script is shown as a series of 'Num:' followed by numbers. The window has standard Linux window controls and a status bar at the bottom.

```
ksobrenat32@t14:~/local/git/FI/buildroot
thread = threading.Thread(target=print_num)
thread.start()
# python /example.py
Num: 0
Num: 1
Num: 2
Num: 0
Num: 0
Num: 1
Num: 3
Num: 1
Num: 2
Num: 4
Num: 2
Num: 3
Num: 3
Num: 4
Num: 4
Num: 0
Num: 1
Num: 2
Num: 3
Num: 4
#
```

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.9 | VT102 | Offline | ttyUSB0

Figure 10: Ejecución de script Python

# Monitoreo del sistema



```
ksobrenat32@t14:~/local/git/FI/buildroot

0[          0.0%] Tasks: 8, 0 thr, 97 kthr; 0 running
1[          0.0%] Load average: 0.00 0.00 0.00
2[          0.0%] Uptime: 00:04:20
3[          0.0%]
Mem[| #@$          32.6M/3.68G]
Swp[          0K/0K]

[Main] [I/O]
PID USER      PRI  NI  VIRT   RES   SHR  S  CPU%-MEM%  TIME+  Command
  1 root         20   0  1676    768    768  S   0.0  0.0   0:00.14  init
117 root         20   0  1676    928    928  S   0.0  0.0   0:00.00  /sbin/syslogd -
121 root         20   0  1676    836    836  S   0.0  0.0   0:00.00  /sbin/klogd -n
155 root         20   0  1676    764    764  S   0.0  0.0   0:00.02  udhcpc -t1 -A3
160 root         20   0  1676    880    880  S   0.0  0.0   0:00.00  /usr/sbin/crond
161 root         20   0  1692   1076   1076  S   0.0  0.0   0:00.01  -sh
162 root         20   0  1676    828    828  S   0.0  0.0   0:00.00  /sbin/getty -L
225 root         20   0  1760   1464   1208  R   0.6  0.0   0:00.10  htop

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

CTRL-A Z for help | 115200 8N1 | APP | Minicom 2.9 | VT102 | Offline | ttyUSB0
```

Figure 11: Uso de htop

# Conclusiones

## Ventajas de sistemas Linux embebidos

- ▶ Menor consumo de recursos
- ▶ Personalización completa
- ▶ Seguridad y estabilidad
- ▶ Amplia comunidad de soporte
- ▶ Disponibilidad de herramientas

## Aplicaciones futuras

- ▶ Internet de las Cosas (IoT)
- ▶ Sistemas industriales
- ▶ Automatización del hogar
- ▶ Robótica
- ▶ Dispositivos médicos

# Referencias

- ▶ Administrador CEUPE. Sistema embebido: Qué es, características y componentes
- ▶ Rebound Electronics. Explicación de los sistemas embebidos
- ▶ Red Hat. Definición de Linux
- ▶ TRBL Services. LINUX Embebido | Qué es, cómo funciona y para qué se usa
- ▶ Buildroot. Buildroot manual
- ▶ Equipo editorial de IONOS. Kernel - El núcleo del sistema operativo