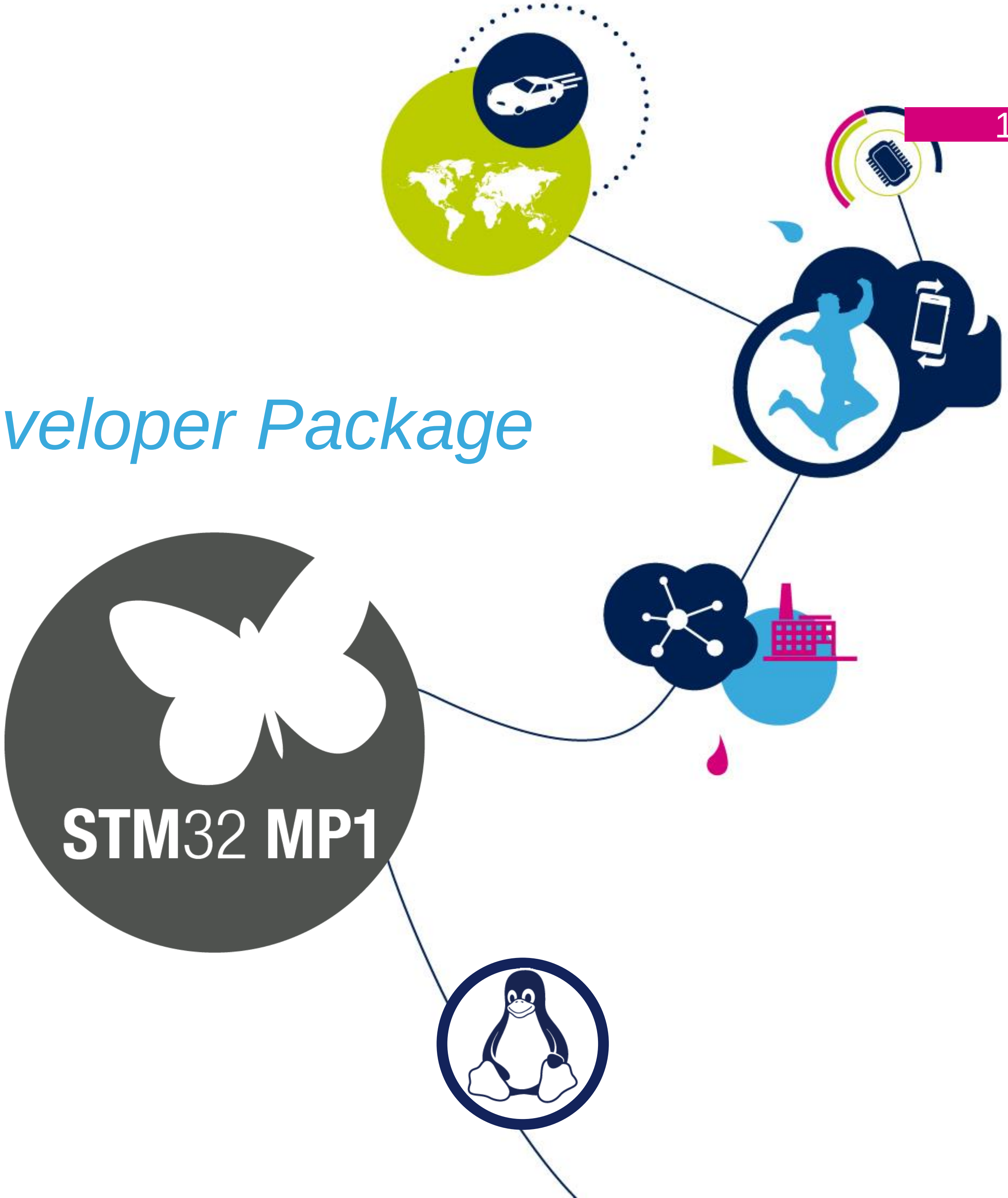
Simple Linux application with Developer Package



Lab *Developer Package*

Lab Objective

- Reboot the board and observe a kernel boot log
- Use the already installed SDK
- Compile in Linux a simple hello world application
- Run the hello world application on the target

Lab *Developer Package*

Lab Objective

- Reboot the board and observe a kernel boot log
- Use the already installed SDK
- Compile in Linux a simple hello world application
- Run the hello world application on the target

Linux command on the host

```
root@stm32mp1:/#
```

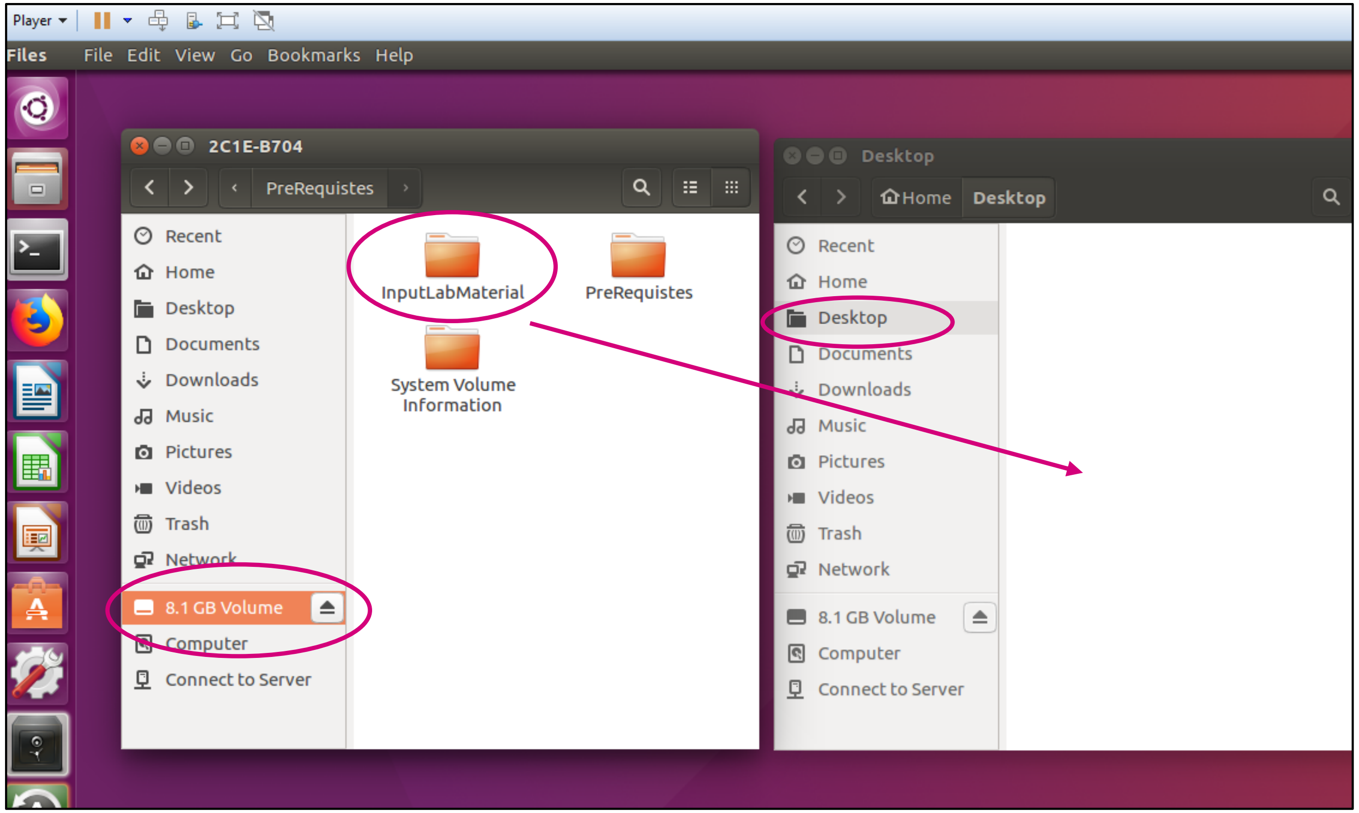
Linux command on the target

theory | practice

Prerequisite for all the workshop labs

1) Start Linux Host
2) Copy the workshop material to Linux Host desktop from a USB stick

- If the USB stick does not auto attach, select it via the VM “Player/Removable Devices”
- Drag and Drop *InputLabMaterial* and *WorkshopSlides* folders to *$HOME/Desktop/*



Tip: open workshop slides in Linux Host for copy/pasting

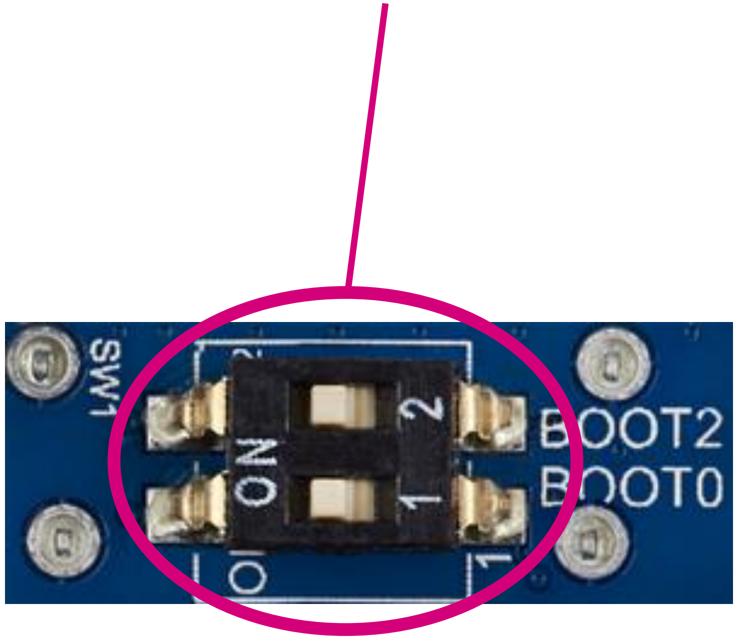
Lab *Developer Package*

Hardware setup

- In the next steps, we will setup the hardware
- Contents:
  - 1 x STM32MP157C-DK2 development board
  - 1 x microSD card
  - 1 x micro USB cable
  - 1 x USB-C cable

Prerequisites - boot switch

- Make sure the boot switch is in the position ’11’ (SD card selected as boot device)

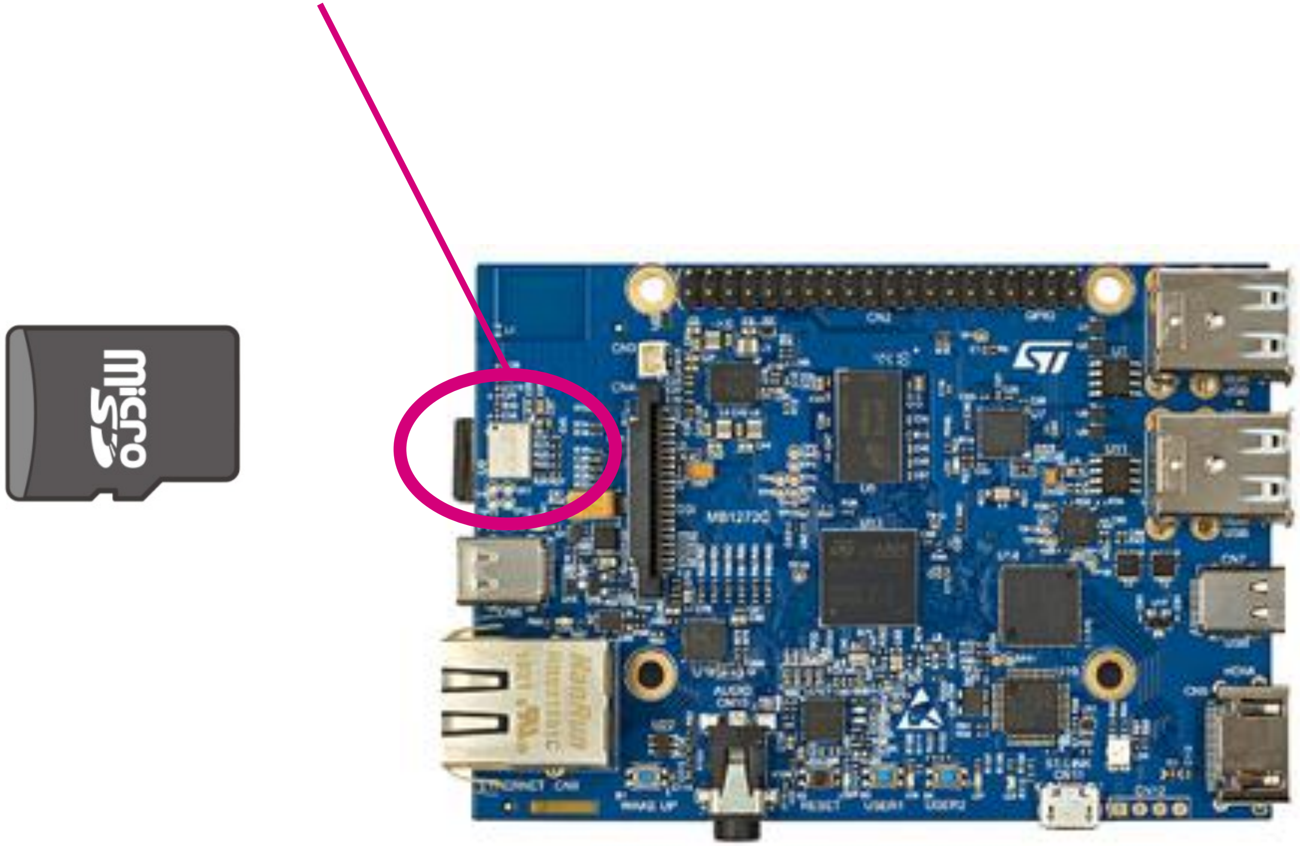


Note: the boot switch is located on the ***bottom side*** *of the board*

Lab *Developer Package*

Prerequisites - SD card

- Make sure the SD card is plugged in

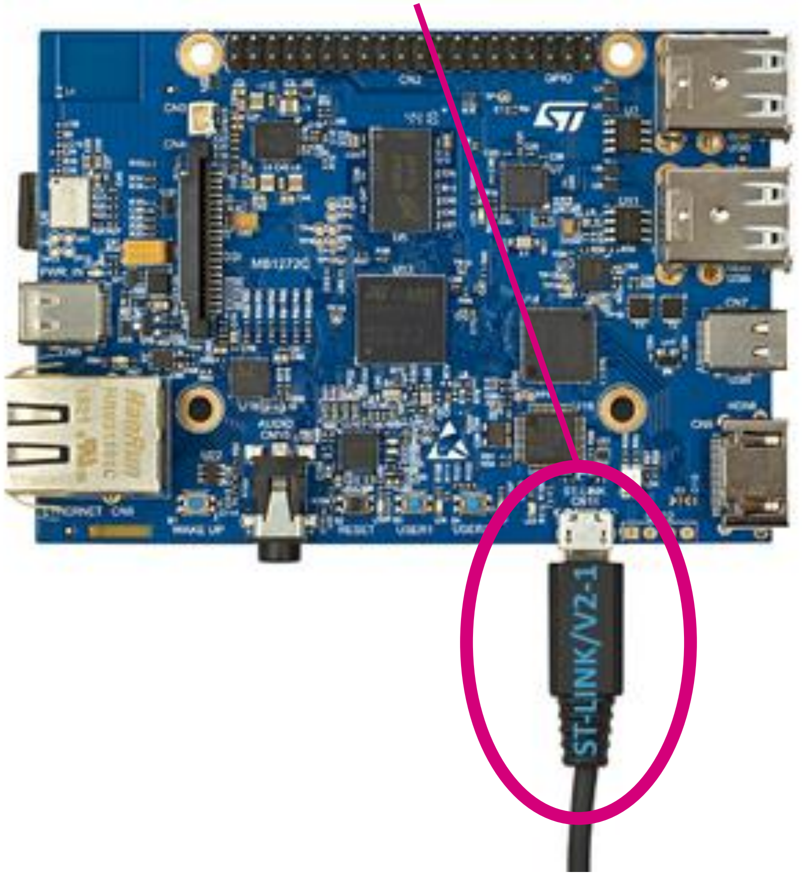


Note: the top view with the display removed

Lab *Developer Package*

Prerequisites - ST-LINK

- Connect the host PC to the on-board ST-LINK/V2 using a micro USB cable

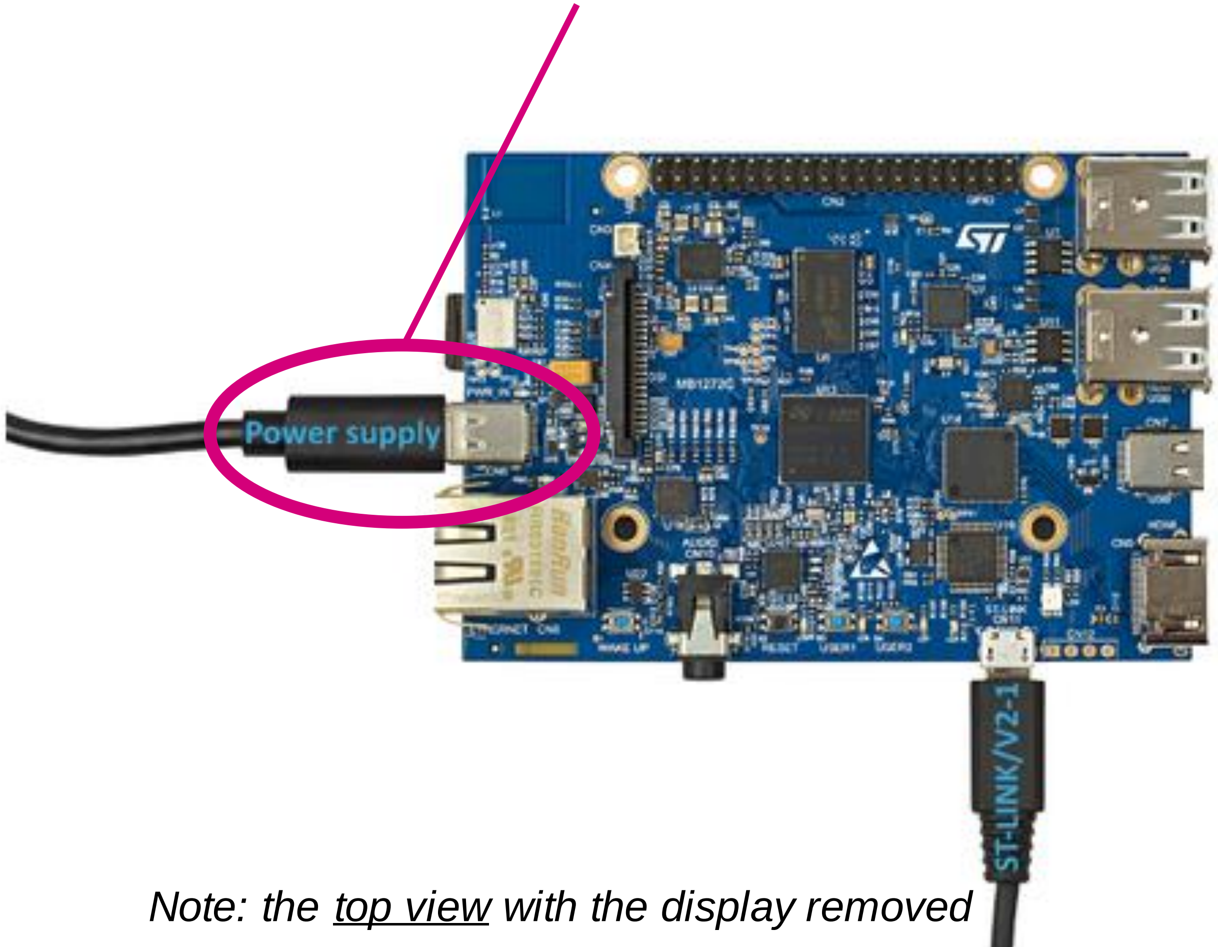


Note: the top view with the display removed

Lab *Developer Package*

Prerequisites - USB-C power supply

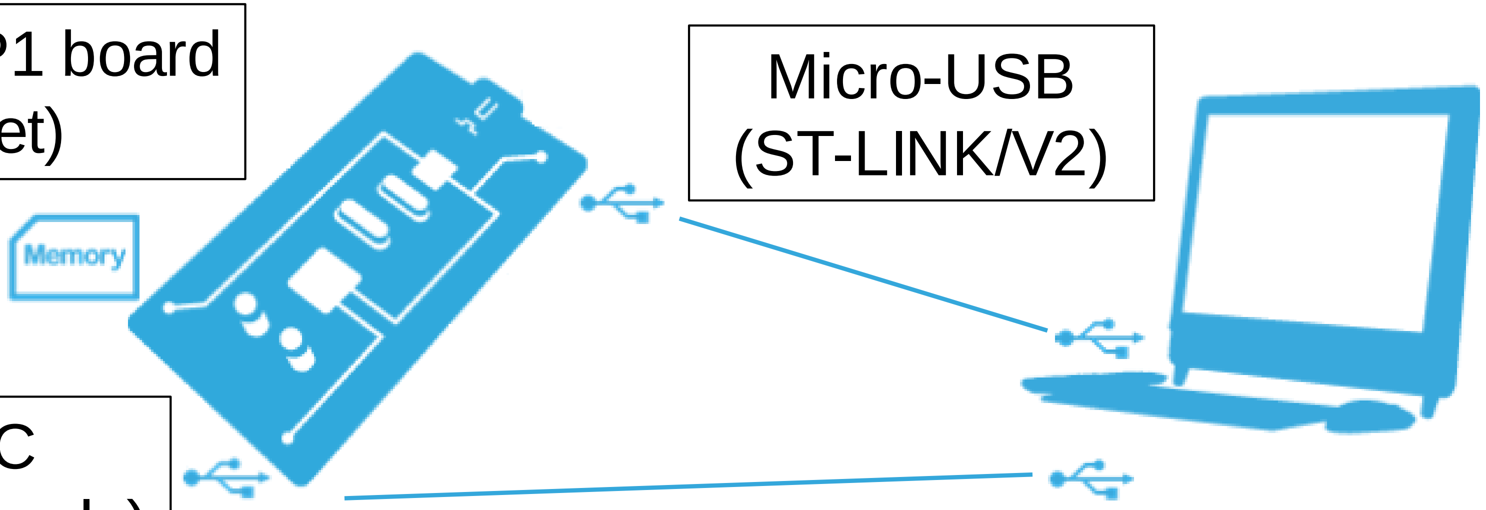
- Connect USB-C cable to power supply the development board



Note: the top view with the display removed

Prerequisites - summary

- Make sure the boot switch is in the position ’11’
  (SD card selected as boot device)
- Make sure the SD card is plugged in
- Connect the host PC to the on-board ST-LINK/V2 using a micro USB cable
- Connect USB-C cable to power supply the development board



STM32MP1 board (target)

microSD

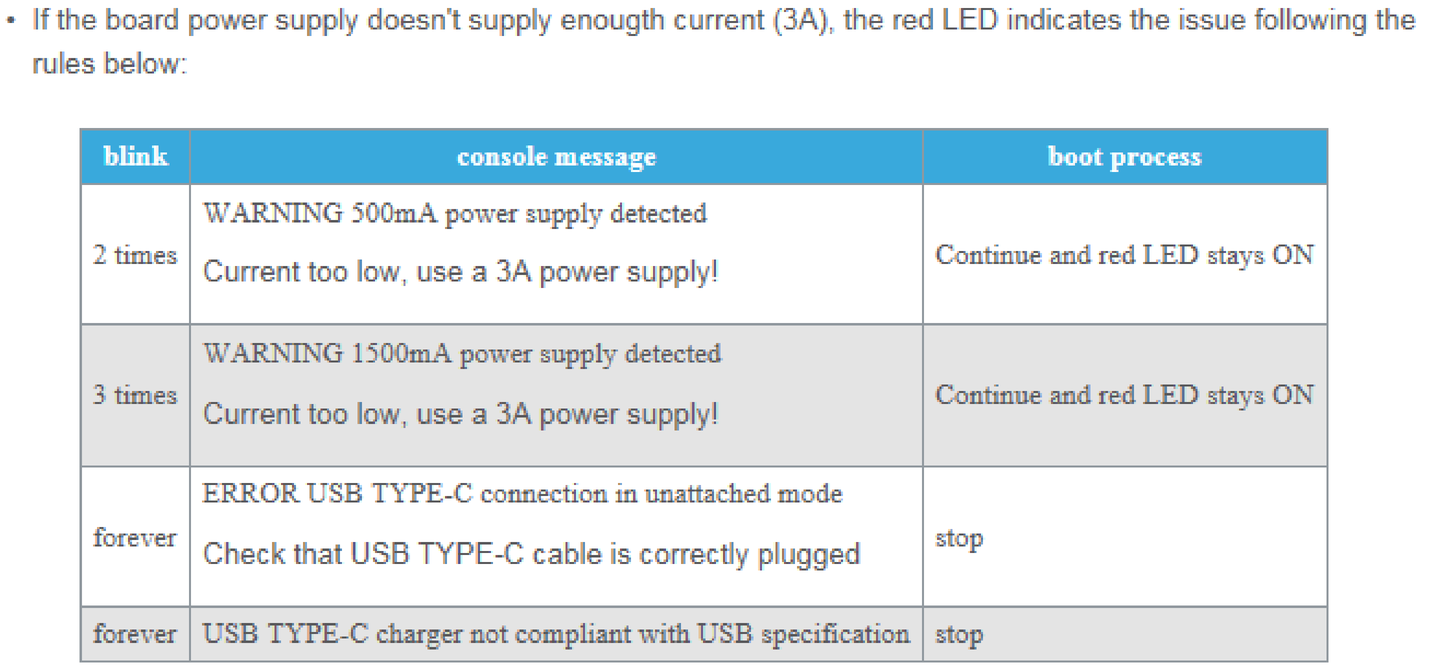
Micro-USB (ST-LINK/V2)

Linux host PC

USB-C (power supply)

Power supply information

- STM32MP157C-DK2 development board can be powered using USB Type-C or USB Type-A from laptop for demo purpose
- Be very careful this mode is limited in terms of power
  It can NOT work for high power use cases
- Red LED may indicate not enough power supply
- For nominal behavior of dk2 board, it is recommended to connect the USB Type-C™ to USB Type-C™ cable to a 5V/3A power supply

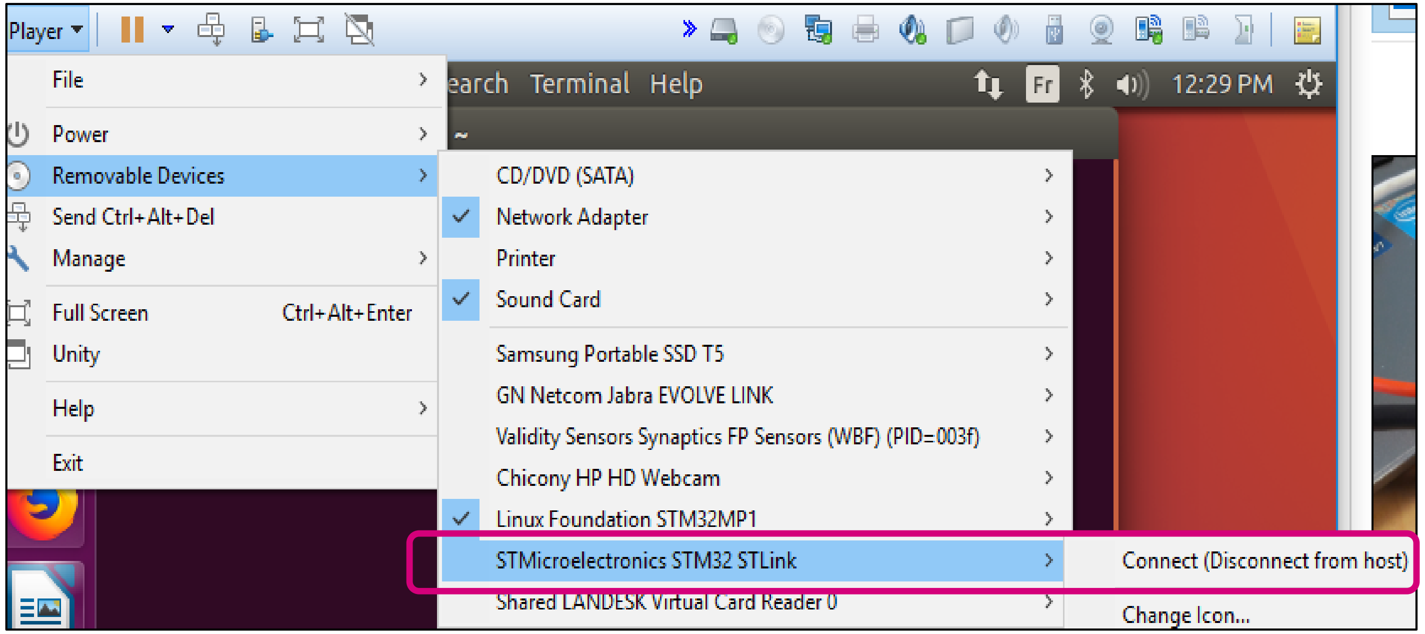


- If the board power supply doesn't supply enougth current (3A), the red LED indicates the issue following the rules below:

blink	console message	boot process
2 times	WARNING 500mA power supply detected Current too low, use a 3A power supply!	Continue and red LED stays ON
3 times	WARNING 1500mA power supply detected Current too low, use a 3A power supply!	Continue and red LED stays ON
forever	ERROR USB TYPE-C connection in unattached mode Check that USB TYPE-C cable is correctly plugged	stop
forever	USB TYPE-C charger not compliant with USB specification	stop

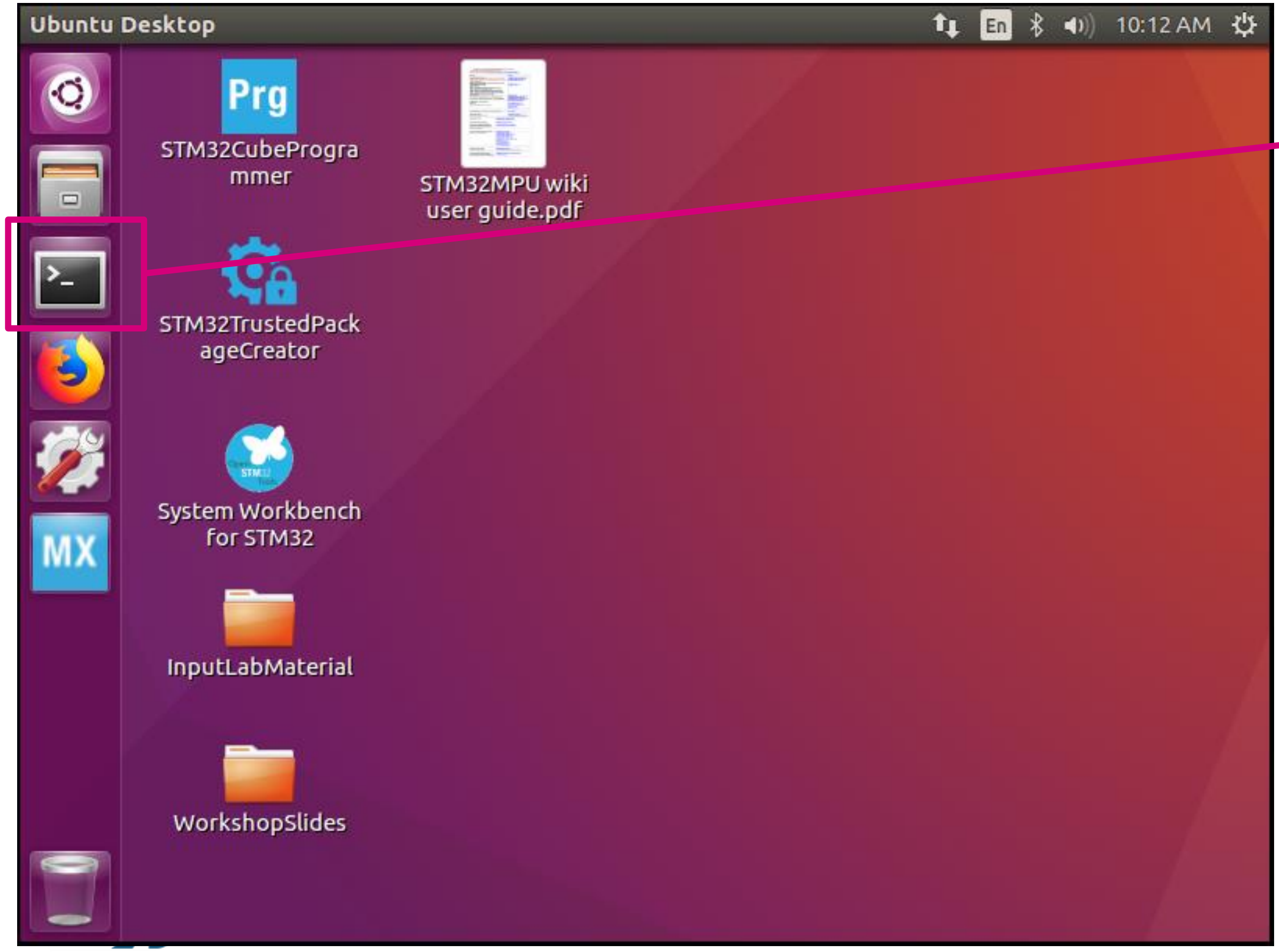
Enable USB ST-Link probe on Linux host

- Connect with the ST-Link debugger/programmer to the Linux host


Player
earch Terminal Help
Fr
12:29 PM
File
Power
Removable Devices
Send Ctrl+Alt+Del
Manage
Full Screen Ctrl+Alt+Enter
Unity
Help
Exit
CD/DVD (SATA)
✓ Network Adapter
Printer
✓ Sound Card
Samsung Portable SSD T5
GN Netcom Jabra EVOLVE LINK
Validity Sensors Synaptics FP Sensors (WBF) (PID=003f)
Chicony HP HD Webcam
✓ Linux Foundation STM32MP1
STMicroelectronics STM32 STLink
Shared LANDESK Virtual Card Reader 0
Connect (Disconnect from host)
Change Icon...

Lab *Developer Package*

Launch a new terminal


Ubuntu Desktop
En
10:12 AM
Prg
STM32CubeProgrammer
STM32MPU wiki user guide.pdf
STM32TrustedPackageCreator
System Workbench for STM32
MX
InputLabMaterial
WorkshopSlides

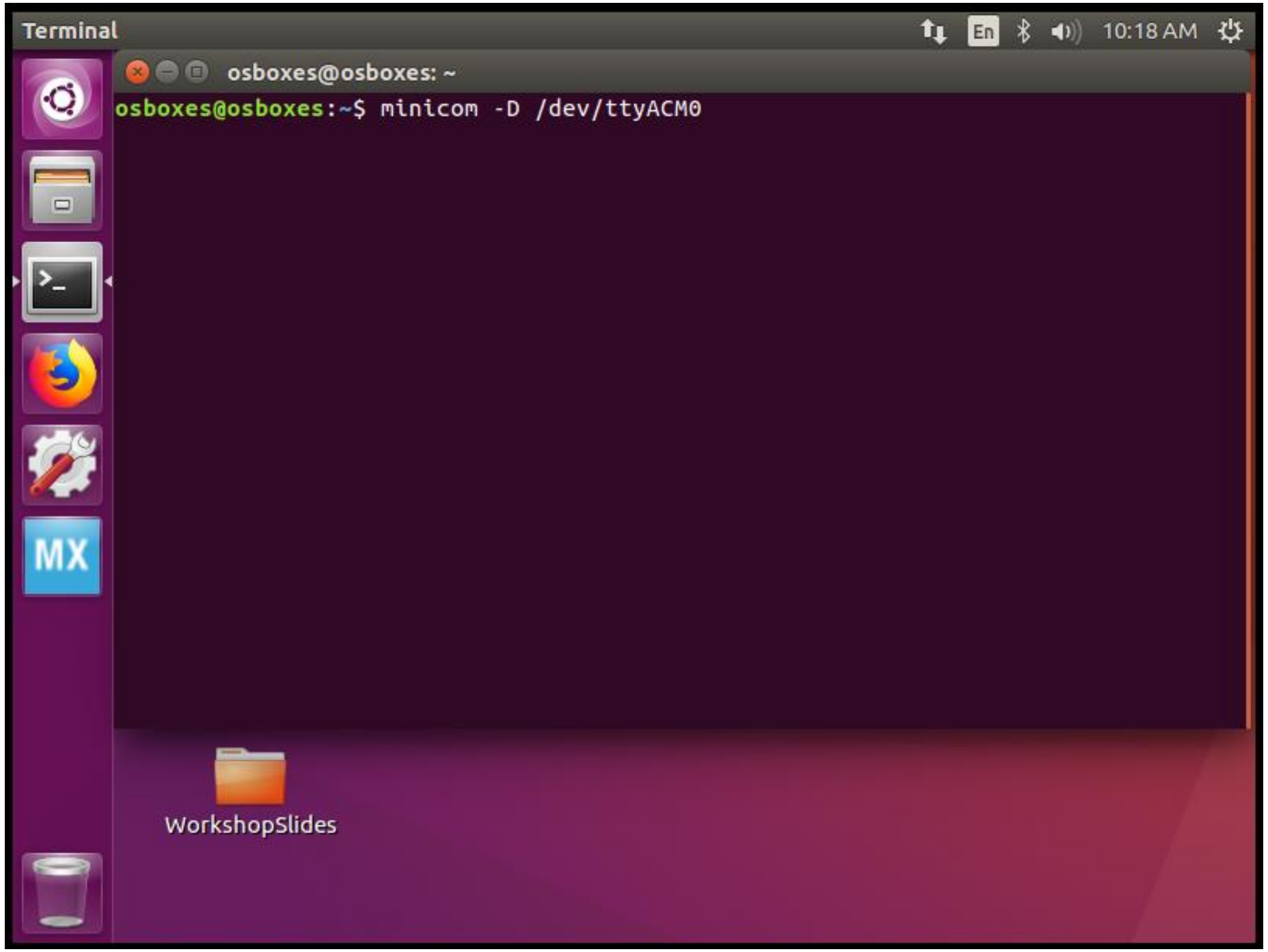
- Launch a new terminal on Linux host

Hint:

You can use the keyboard shortcut

<CTRL><ALT>T

Start minicom



```
osboxes@osboxes:~$ minicom -D /dev/ttyACM0
```

- Start a minicom terminal in order to establish a serial connection between Linux host and the target

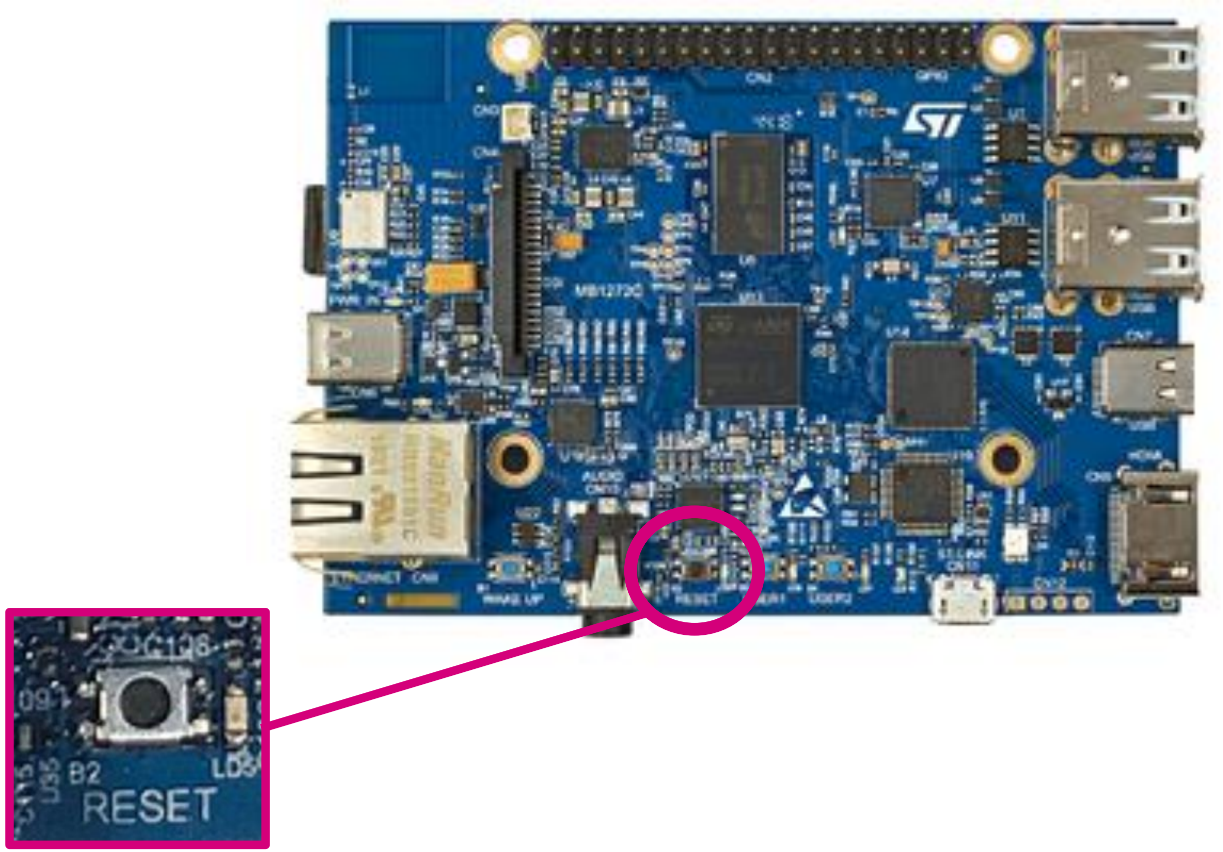
Enter on Host

```
minicom -D /dev/ttyACM0
```

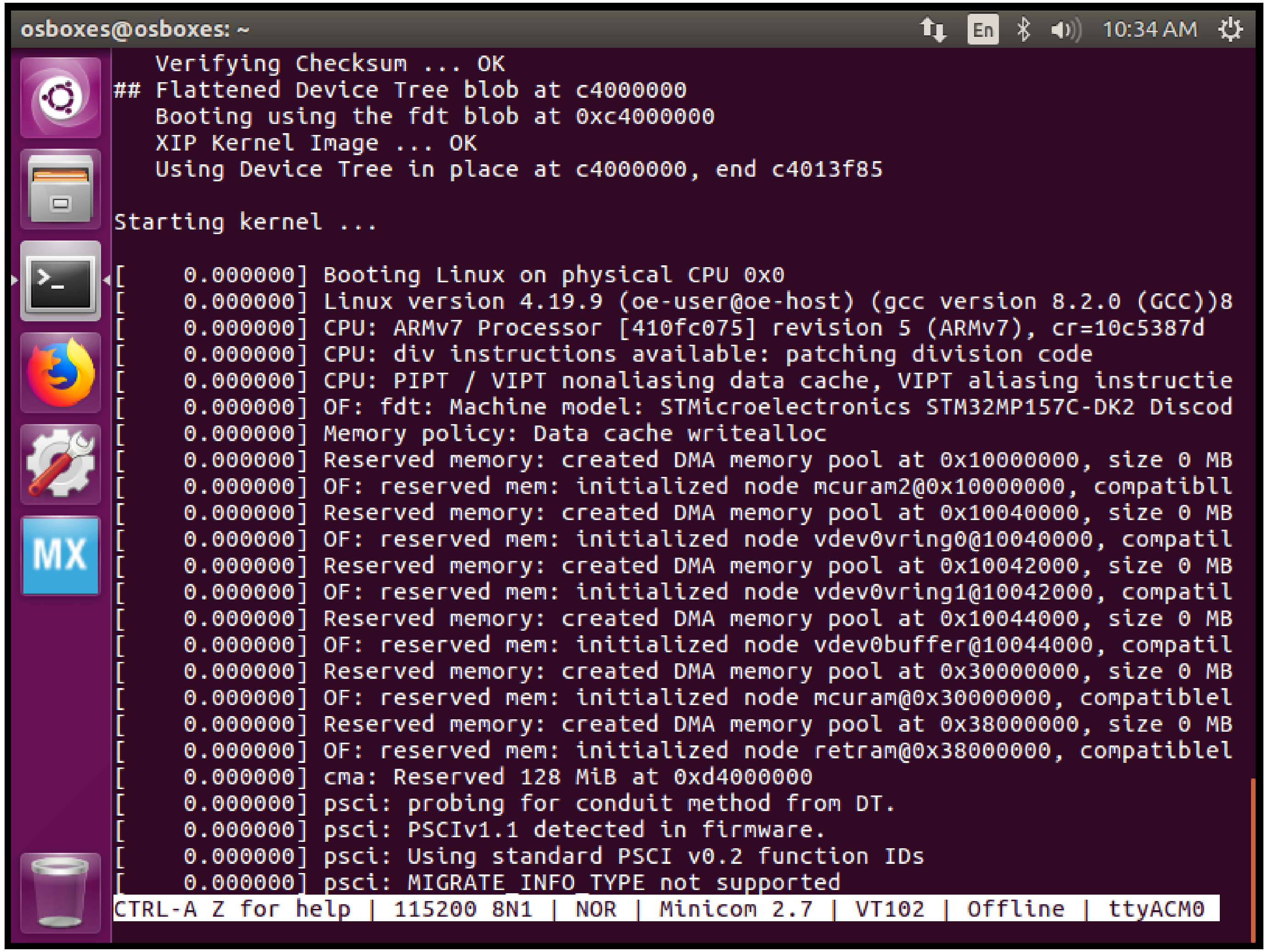
Lab *Developer Package*

Reboot the board

- Press the black reset button to reboot the board and observe the kernel boot log in the minicom terminal



Observe the kernel boot log



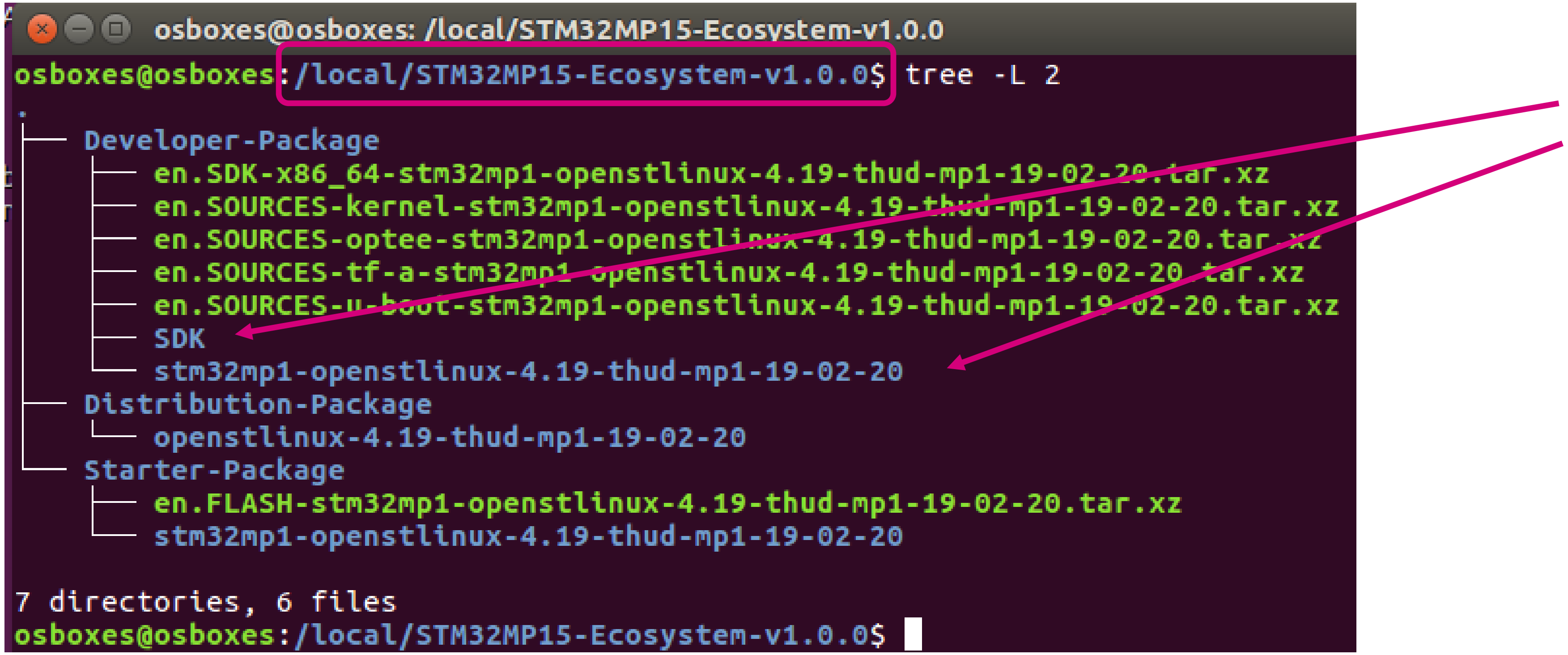
```
   Verifying Checksum ... OK
## Flattened Device Tree blob at c4000000
   Booting using the fdt blob at 0xc4000000
   XIP Kernel Image ... OK
   Using Device Tree in place at c4000000, end c4013f85

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.19.9 (oe-user@oe-host) (gcc version 8.2.0 (GCC))8
[    0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387d
[    0.000000] CPU: div instructions available: patching division code
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instructie
[    0.000000] OF: fdt: Machine model: STMicroelectronics STM32MP157C-DK2 Discod
[    0.000000] Memory policy: Data cache writealloc
[    0.000000] Reserved memory: created DMA memory pool at 0x10000000, size 0 MB
[    0.000000] OF: reserved mem: initialized node mcuram2@0x10000000, compatibll
[    0.000000] Reserved memory: created DMA memory pool at 0x10040000, size 0 MB
[    0.000000] OF: reserved mem: initialized node vdev0vring0@10040000, compatil
[    0.000000] Reserved memory: created DMA memory pool at 0x10042000, size 0 MB
[    0.000000] OF: reserved mem: initialized node vdev0vring1@10042000, compatil
[    0.000000] Reserved memory: created DMA memory pool at 0x10044000, size 0 MB
[    0.000000] OF: reserved mem: initialized node vdev0buffer@10044000, compatil
[    0.000000] Reserved memory: created DMA memory pool at 0x30000000, size 0 MB
[    0.000000] OF: reserved mem: initialized node mcuram@0x30000000, compatiblel
[    0.000000] Reserved memory: created DMA memory pool at 0x38000000, size 0 MB
[    0.000000] OF: reserved mem: initialized node retram@0x38000000, compatiblel
[    0.000000] cma: Reserved 128 MiB at 0xd4000000
[    0.000000] psci: probing for conduit method from DT.
[    0.000000] psci: PSCIv1.1 detected in firmware.
[    0.000000] psci: Using standard PSCI v0.2 function IDs
[    0.000000] psci: MIGRATE_INFO_TYPE not supported
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyACM0
```

- Observe the kernel boot log

Lab *Developer Package*

Developer Package Prerequisite

- The Developer Package and SDK are already installed



```
osboxes@osboxes:/local/STM32MP15-Ecosystem-v1.0.0$ tree -L 2
.
├── Developer-Package
│   ├── en.SDK-x86_64-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
│   ├── en.SOURCES-kernel-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
│   ├── en.SOURCES-optee-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
│   ├── en.SOURCES-tf-a-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
│   ├── en.SOURCES-u-boot-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
│   ├── SDK
│   └── stm32mp1-openstlinux-4.19-thud-mp1-19-02-20
├── Distribution-Package
│   └── openstlinux-4.19-thud-mp1-19-02-20
└── Starter-Package
    ├── en.FLASH-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
    └── stm32mp1-openstlinux-4.19-thud-mp1-19-02-20

7 directories, 6 files
osboxes@osboxes:/local/STM32MP15-Ecosystem-v1.0.0$
```

SDK

Kernel, TF-A, Uboot
source code

Note: in terminal, root of the developer package is already defined in $SDK_ROOT as /local/STM32MP15-Ecosystem-v1.0.0/Developer-Package

Build environment setup

1) Open a new terminal window

Lab *Developer Package*

Build environment setup

1) Open a new terminal window
2) Setup build environment

```
source $SDK_ROOT/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-openstlinux_weston-linux-gnueabi
```

Lab *Developer Package*

Build environment setup

1) Open a new terminal window
2) Setup build environment

```
source $SDK_ROOT/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-openstlinux_weston-linux-gnueabi
```

3) Check the environment

```
$CC --version
```

- Expected return value:

```
arm-openstlinux_weston-linux-gnueabi-gcc (GCC) 8.2.0
```

Lab *Developer Package*

Copy lab material

1) Copy lab material from the desktop directory to the SDK

```
cp -a $HOME/Desktop/InputLabMaterial/Lab-DeveloperPackage $SDK_ROOT/
```

Lab *Developer Package*

Copy lab material

1) Copy lab material from the desktop directory to the SDK

```
cp -a $HOME/Desktop/InputLabMaterial/Lab-DeveloperPackage $SDK_ROOT/
```

2) Move to the hello world example directory in the SDK

```
cd $SDK_ROOT/Lab-DeveloperPackage/gtk_hello_world_example/
```

Lab *Developer Package*

Copy lab material

1) Copy lab material from the desktop directory to the SDK

```
cp -a $HOME/Desktop/InputLabMaterial/Lab-DeveloperPackage $SDK_ROOT/
```

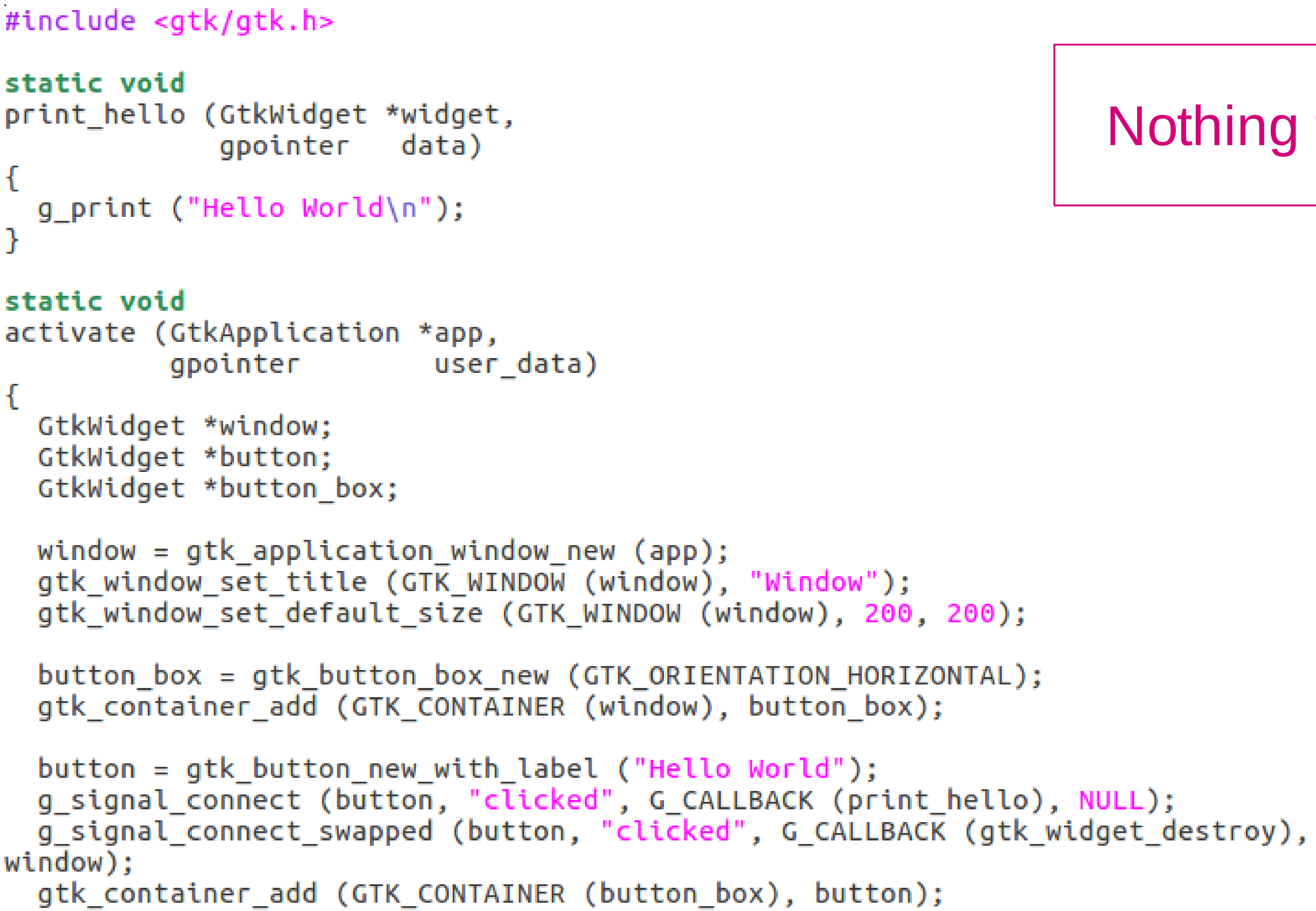
2) Move to the hello world example directory in the SDK

```
cd $SDK_ROOT/Lab-DeveloperPackage/gtk_hello_world_example/
```

3) Open the source file using the gedit text editor

```
gedit gtk_hello_world.c &
```

Review source code of the application

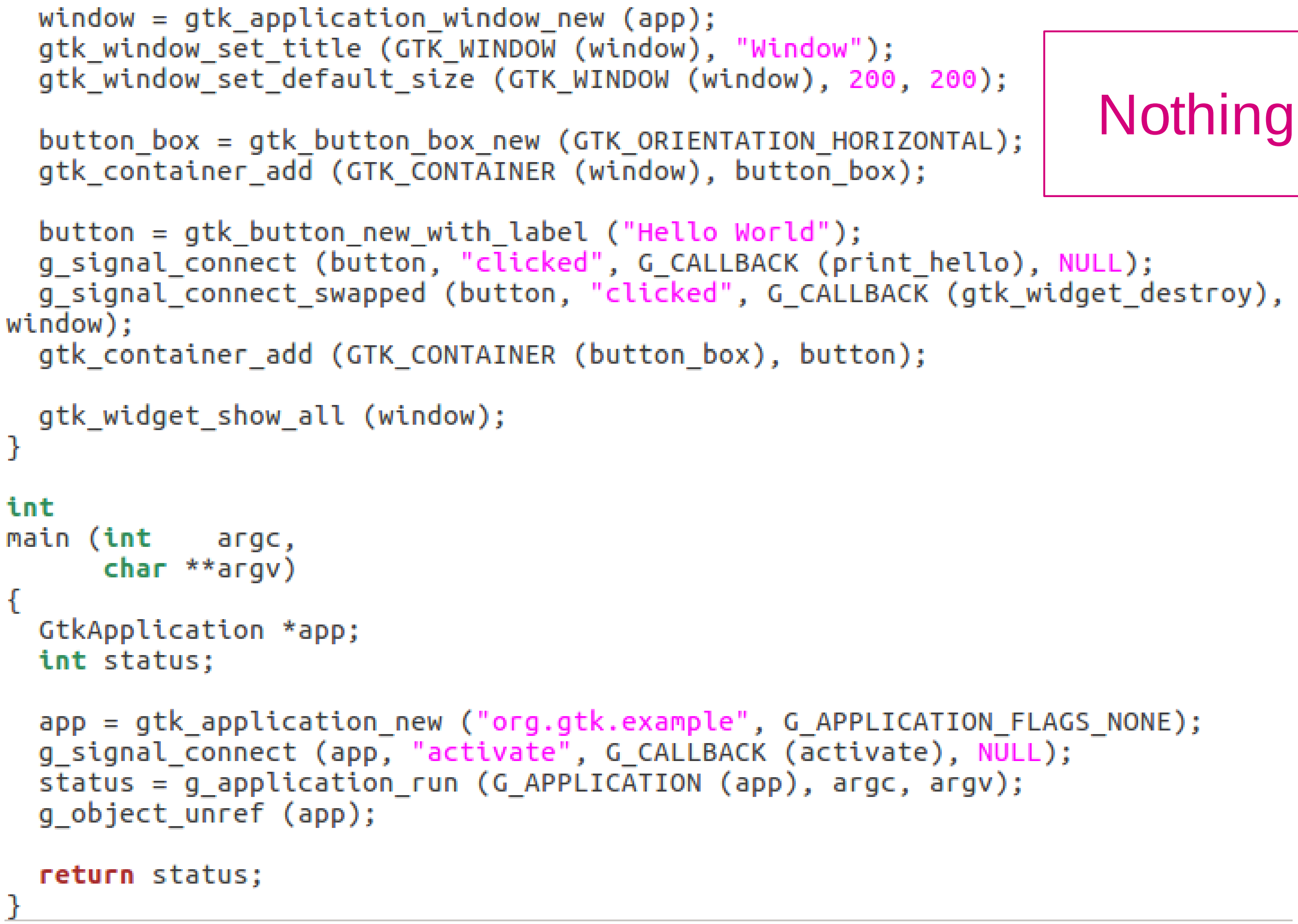


```
#include <gtk/gtk.h>

static void
print_hello (GtkWidget *widget,
             gpointer   data)
{
  g_print ("Hello World\n");
}

static void
activate (GtkApplication *app,
          gpointer        user_data)
{
  GtkWidget *window;
  GtkWidget *button;
  GtkWidget *button_box;

  window = gtk_application_window_new (app);
  gtk_window_set_title (GTK_WINDOW (window), "Window");
  gtk_window_set_default_size (GTK_WINDOW (window), 200, 200);

  button_box = gtk_button_box_new (GTK_ORIENTATION_HORIZONTAL);
  gtk_container_add (GTK_CONTAINER (window), button_box);

  button = gtk_button_new_with_label ("Hello World");
  g_signal_connect (button, "clicked", G_CALLBACK (print_hello), NULL);
  g_signal_connect_swapped (button, "clicked", G_CALLBACK (gtk_widget_destroy),
window);
  gtk_container_add (GTK_CONTAINER (button_box), button);
```

Nothing to change here!

Note: the application is using GTK (toolkit to creating graphical user interfaces) to create a simple window

Review source code of the application



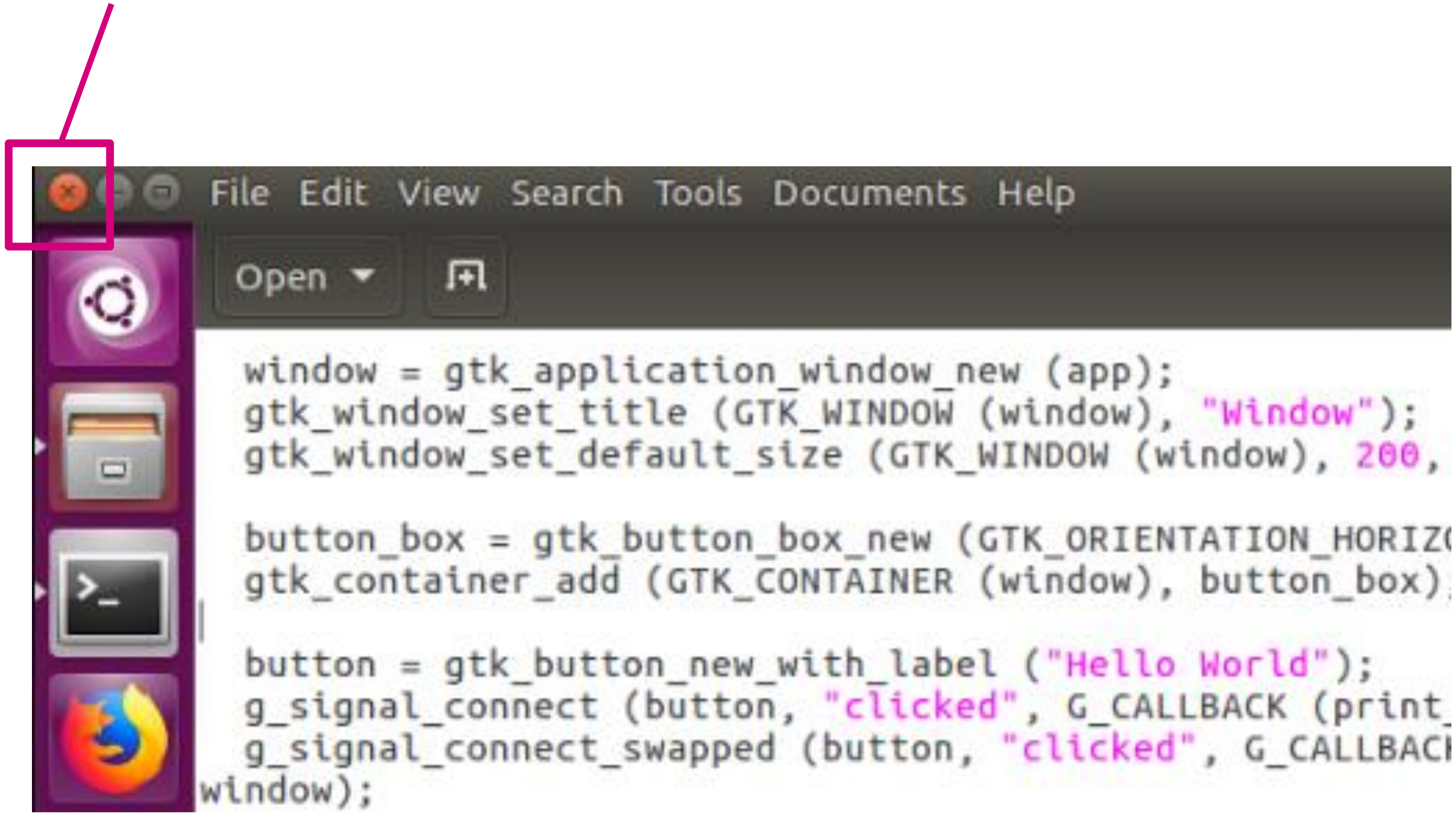
```
  window = gtk_application_window_new (app);
  gtk_window_set_title (GTK_WINDOW (window), "Window");
  gtk_window_set_default_size (GTK_WINDOW (window), 200, 200);

  button_box = gtk_button_box_new (GTK_ORIENTATION_HORIZONTAL);
  gtk_container_add (GTK_CONTAINER (window), button_box);

  button = gtk_button_new_with_label ("Hello World");
  g_signal_connect (button, "clicked", G_CALLBACK (print_hello), NULL);
  g_signal_connect_swapped (button, "clicked", G_CALLBACK (gtk_widget_destroy),
window);
  gtk_container_add (GTK_CONTAINER (button_box), button);

  gtk_widget_show_all (window);
}

int
main (int    argc,
      char **argv)
{
  GtkApplication *app;
  int status;

  app = gtk_application_new ("org.gtk.example", G_APPLICATION_FLAGS_NONE);
  g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
  status = g_application_run (G_APPLICATION (app), argc, argv);
  g_object_unref (app);

  return status;
}
```

Nothing to change here!

Note: the application is using GTK (toolkit to creating graphical user interfaces) to create a simple window

Build simple hello world application

- Close the gedit text editor



```
   window = gtk_application_window_new (app);
   gtk_window_set_title (GTK_WINDOW (window), "Window");
   gtk_window_set_default_size (GTK_WINDOW (window), 200,

   button_box = gtk_button_box_new (GTK_ORIENTATION_HORIZ
   gtk_container_add (GTK_CONTAINER (window), button_box)

   button = gtk_button_new_with_label ("Hello World");
   g_signal_connect (button, "clicked", G_CALLBACK (print_
   g_signal_connect_swapped (button, "clicked", G_CALLBACK
window);
```

Lab *Developer Package*

Compile the hello world example

- Issue make command to compile the example

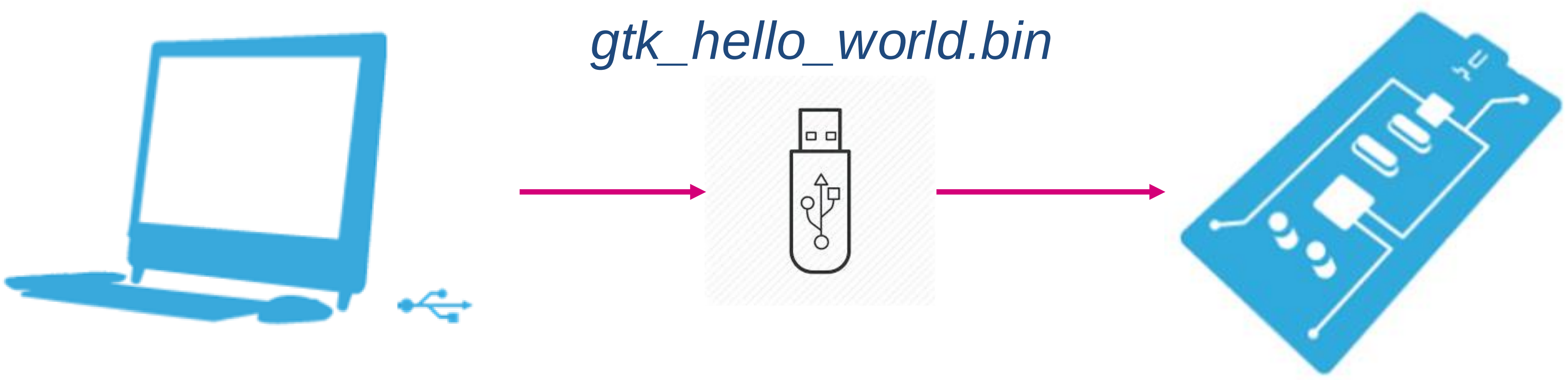
```
make
```

Note: the makefile has been prepared in the same directory

Transferring the binary to the target

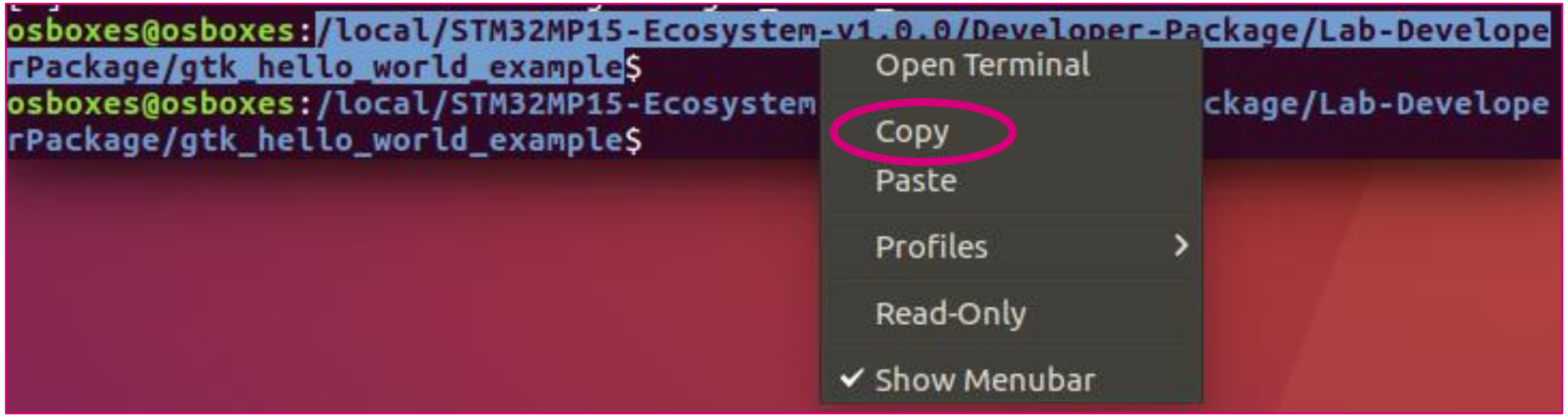
- Now, that we have a binary of the hello world example, the next step will be transferring the binary to the target and executing it there afterwards

- A USB stick will be used to transfer the binary


gtk_hello_world.bin

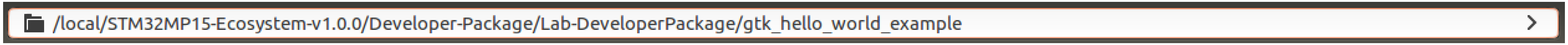
Compile the hello world example

1) Highlight the current path in the terminal and right click on it
2) Select copy



```
osboxes@osboxes:/local/STM32MP15-Ecosystem-v1.0.0/Developer-Package/Lab-Develope
rPackage/gtk_hello_world_example$
osboxes@osboxes:/local/STM32MP15-Ecosystem                    ckage/Lab-Develope
rPackage/gtk_hello_world_example$
```

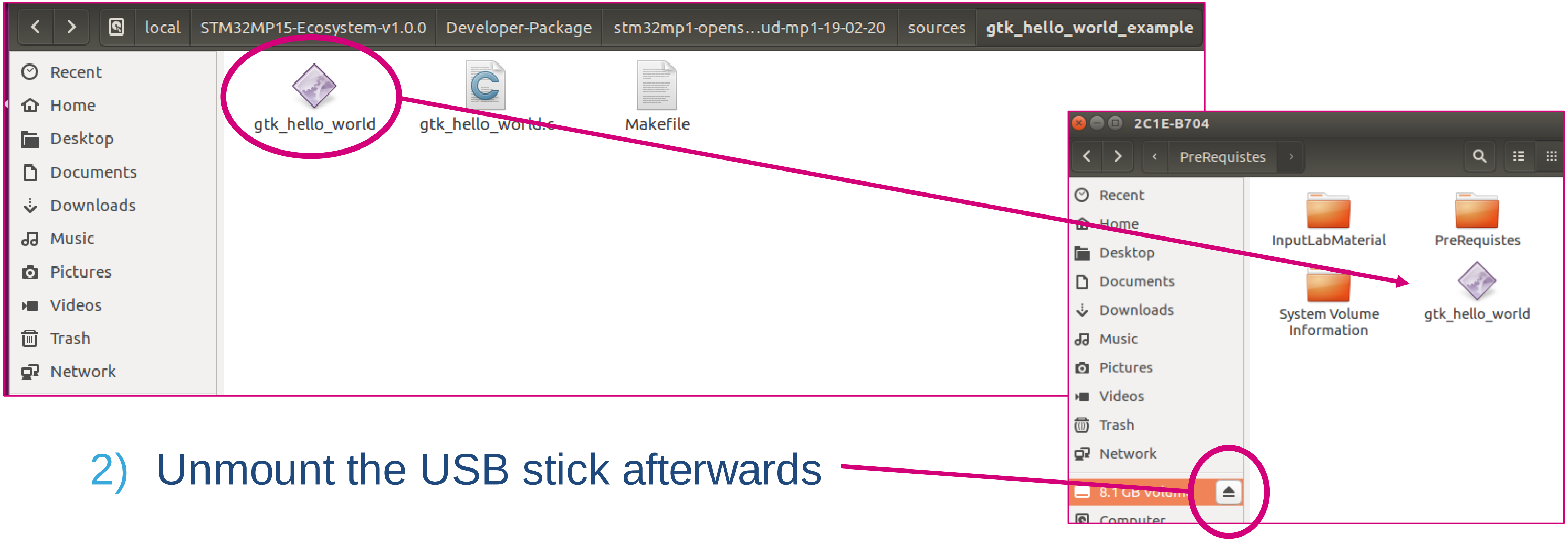
3) Open Files
4) Press CTRL+ L
5) Paste the path



/local/STM32MP15-Ecosystem-v1.0.0/Developer-Package/Lab-DeveloperPackage/gtk_hello_world_example

Compile the hello world example

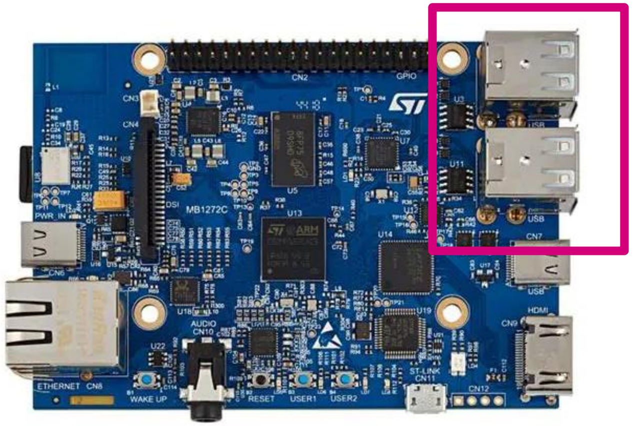
1) Copy *gtk_hello_world* binary to the USB stick (Drag & Drop)

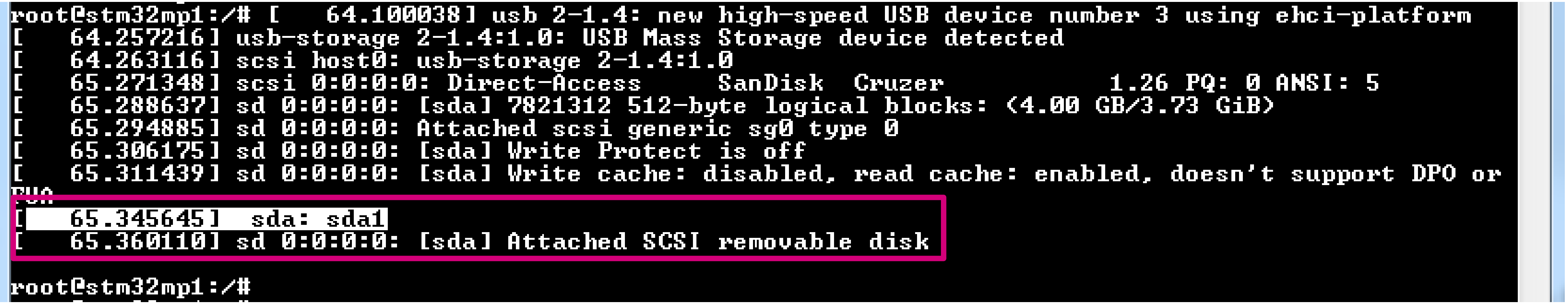


2) Unmount the USB stick afterwards

Plug in the USB stick

1) Plug the USB stick into one of the 4 USB host ports on the discovery board
2) Observe the log in the minicom terminal indicating that the USB stick has been recognized





```
root@stm32mp1:/# [   64.100038] usb 2-1.4: new high-speed USB device number 3 using ehci-platform
[   64.257216] usb-storage 2-1.4:1.0: USB Mass Storage device detected
[   64.263116] scsi host0: usb-storage 2-1.4:1.0
[   65.271348] scsi 0:0:0:0: Direct-Access     SanDisk  Cruzer           1.26 PQ: 0 ANSI: 5
[   65.288637] sd 0:0:0:0: [sda] 7821312 512-byte logical blocks: (4.00 GB/3.73 GiB)
[   65.294885] sd 0:0:0:0: Attached scsi generic sg0 type 0
[   65.306175] sd 0:0:0:0: [sda] Write Protect is off
[   65.311439] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or
FUA
[   65.345645]  sda: sda1
[   65.360110] sd 0:0:0:0: [sda] Attached SCSI removable disk

root@stm32mp1:/#
```

Mount the USB stick on the Linux host

1) Mount the USB stick file system on the Linux host

```
root@stm32mp1:/#
```

```
mount /dev/sda1 /mnt
```

Note:
You can specify the file system type and enable verbose messaging with…

```
mount -t vfat /dev/sda1 /mnt -v
```

Copy the binary file to the local directory

1) Mount the USB stick file system on the Linux host

```
root@stm32mp1:/#
```

```
mount /dev/sda1 /mnt
```

2) Copy the binary to the *usr/local* directory

```
root@stm32mp1:/#
```

```
cp /mnt/gtk_hello_world /usr/local
```

Sync

1) Mount the USB stick file system on the Linux host

```
root@stm32mp1:/#
```

```
mount /dev/sda1 /mnt
```

2) Copy the binary to the *usr/local* directory

```
root@stm32mp1:/#
```

```
cp /mnt/gtk_hello_world /usr/local
```

3) Sync

```
root@stm32mp1:/#
```

```
sync
```

Lab *Developer Package*

Test the binary on the target

1) Move to the directory containing the binary

```
root@stm32mp1:/#
```

```
cd /usr/local
```

Test the binary on the target

1) Move to the directory containing the binary

```
root@stm32mp1:/#
```

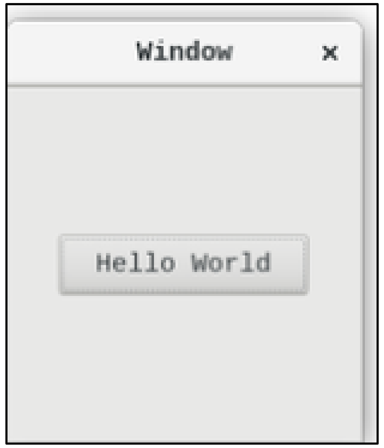
```
cd /usr/local
```

2) Add execution permission and run the binary

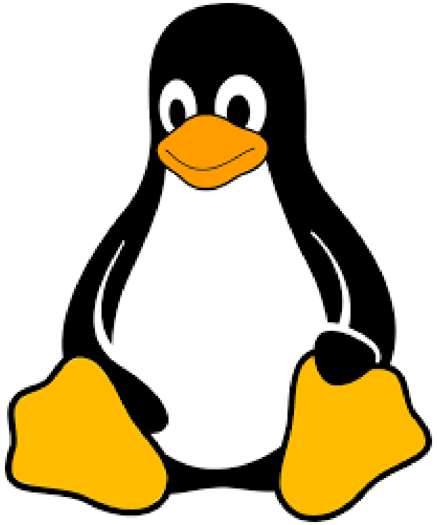
```
root@stm32mp1:/#
```

```
chmod +x gtk_hello_world
./gtk_hello_world
```

3) A window will be displayed on the target



Thank you



life.augmented