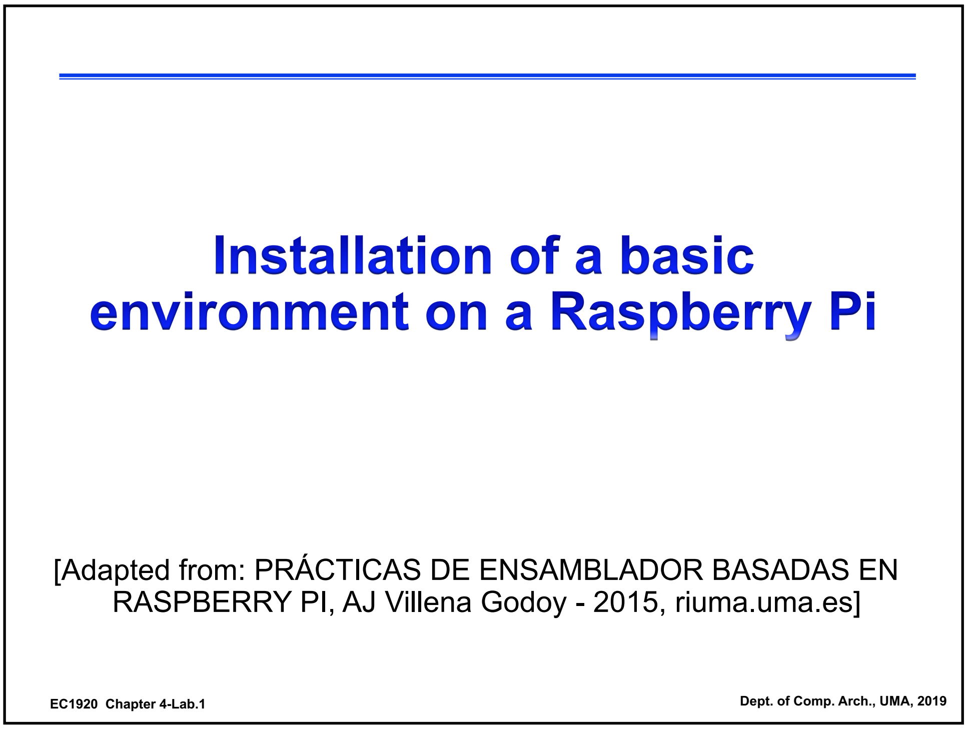


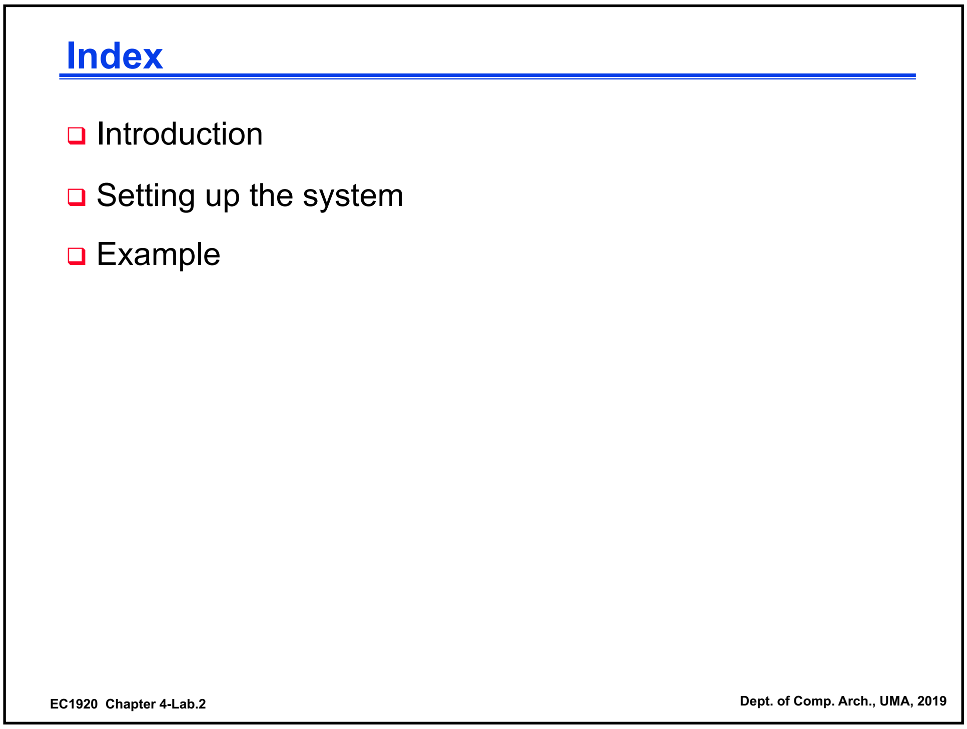
Installation of a basic environment on a Raspberry Pi

[Adapted from: PRÁCTICAS DE ENSAMBLADOR BASADAS EN RASPBERRY PI, AJ Villena Godoy - 2015, riuma.uma.es]

EC1920 Chapter 4-Lab.1

Dept. of Comp. Arch., UMA, 2019

1



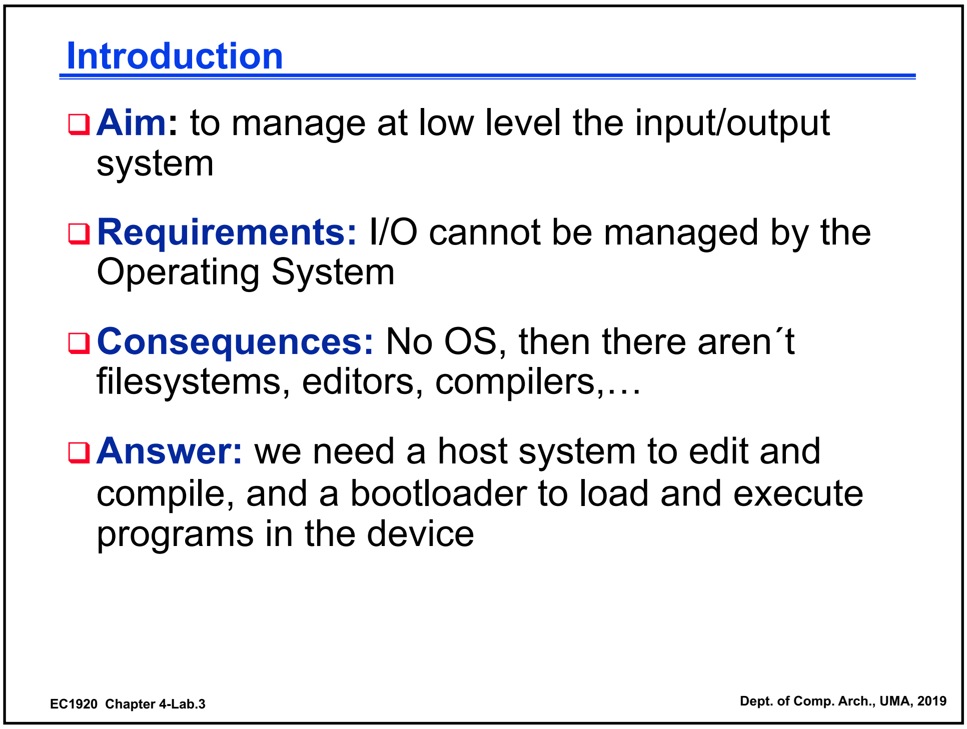
Index

- ❑ Introduction
- ❑ Setting up the system
- ❑ Example

EC1920 Chapter 4-Lab.2

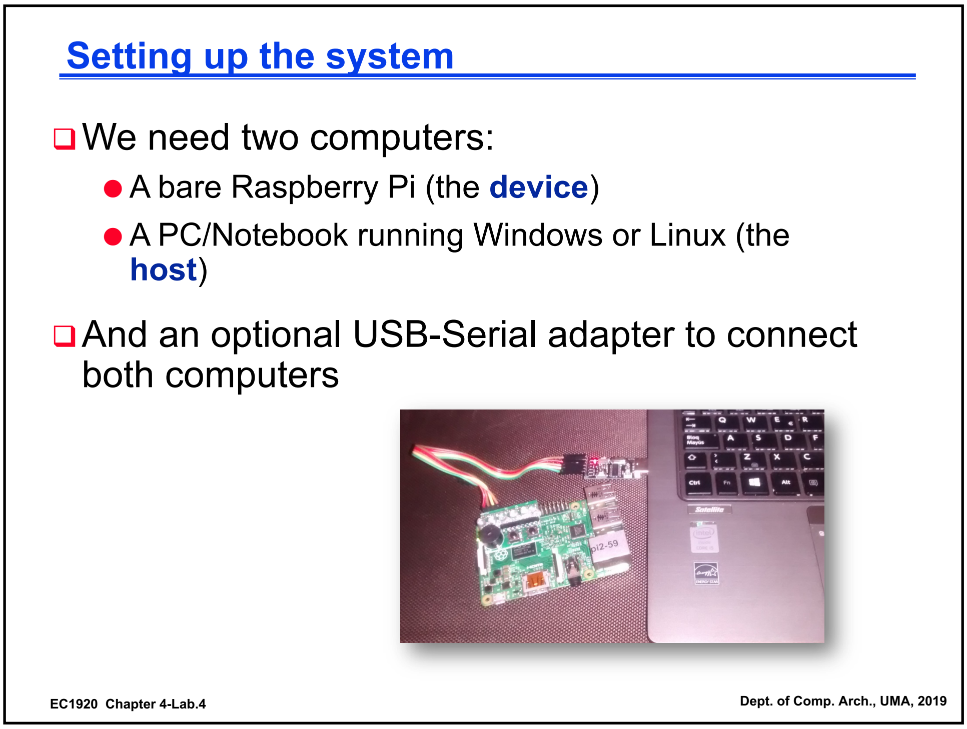
Dept. of Comp. Arch., UMA, 2019

2



Introduction

- **Aim:** to manage at low level the input/output system
- **Requirements:** I/O cannot be managed by the Operating System
- **Consequences:** No OS, then there aren´t filesystems, editors, compilers,…
- **Answer:** we need a host system to edit and compile, and a bootloader to load and execute programs in the device

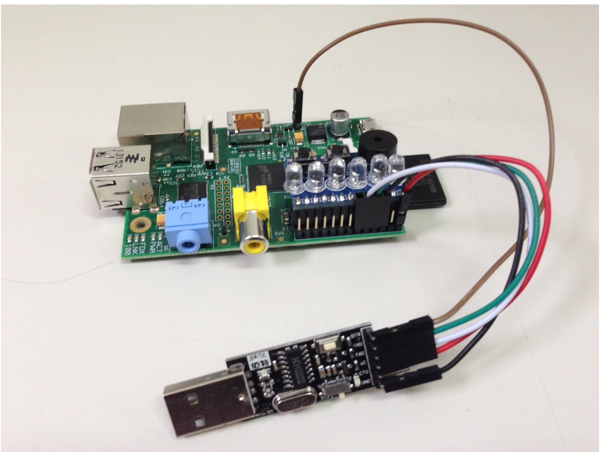


Setting up the system

- We need two computers:
  - A bare Raspberry Pi (the **device**)
  - A PC/Notebook running Windows or Linux (the **host**)
- And an optional USB-Serial adapter to connect both computers

USB-serial adapter

- Pins 2 or 4: 5V, 6: GND, 8: TX and 10: Rx
  - TX and RX are crossed to RXD and TXD in the adapter



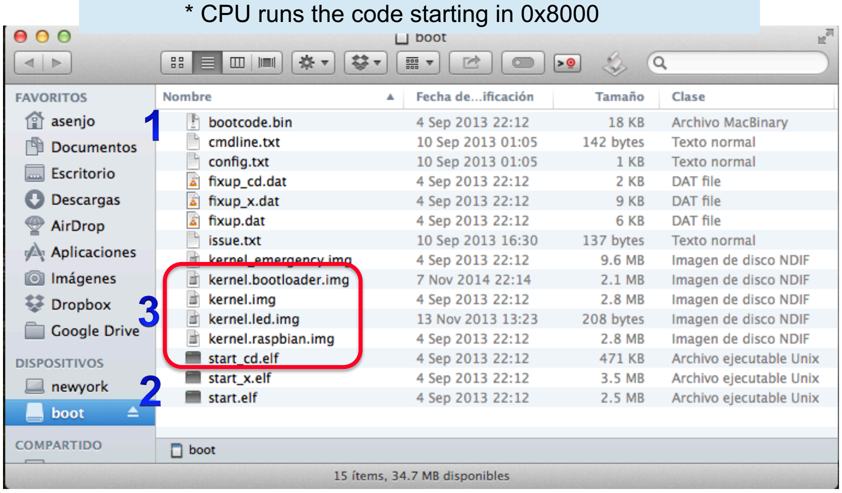
Setting up the device

- We don´t use a OS in the Raspberry
  - During booting [(*)] a `kernel.img` file, containing our executable, is loaded.
- If we have the serial adapter we can use a special `kernel.img` that listens to the serial port to load our `.img`.

[(*)] The Raspberry Pi booting process is explained in detail in the manual.

SD content and booting

1. GPU initiates the booting from an internal ROM.
    * It loads *bootcode.bin* from SD and runs it
2. GPU activates SDRAM, loads *start.elf* and runs it
3. GPU loads *kernel.img* en 0x8000 and wakes up CPU
    * CPU runs the code starting in 0x8000



Nombre	Fecha de...ificación	Tamaño	Clase
bootcode.bin	4 Sep 2013 22:12	18 KB	Archivo MacBinary
cmdline.txt	10 Sep 2013 01:05	142 bytes	Texto normal
config.txt	10 Sep 2013 01:05	1 KB	Texto normal
fixup_cd.dat	4 Sep 2013 22:12	2 KB	DAT file
fixup_x.dat	4 Sep 2013 22:12	9 KB	DAT file
fixup.dat	4 Sep 2013 22:12	6 KB	DAT file
issue.txt	10 Sep 2013 16:30	137 bytes	Texto normal
kernel_emergency.img	4 Sep 2013 22:12	9.6 MB	Imagen de disco NDIF
kernel.bootloader.img	7 Nov 2014 22:14	2.1 MB	Imagen de disco NDIF
kernel.img	4 Sep 2013 22:12	2.8 MB	Imagen de disco NDIF
kernel.led.img	13 Nov 2013 13:23	208 bytes	Imagen de disco NDIF
kernel.raspbian.img	4 Sep 2013 22:12	2.8 MB	Imagen de disco NDIF
start_cd.elf	4 Sep 2013 22:12	471 KB	Archivo ejecutable Unix
start_x.elf	4 Sep 2013 22:12	3.5 MB	Archivo ejecutable Unix
start.elf	4 Sep 2013 22:12	2.5 MB	Archivo ejecutable Unix

15 ítems, 34.7 MB disponibles

EC1920 Chapter 4-Lab.7 Dept. of Comp. Arch., UMA, 2019

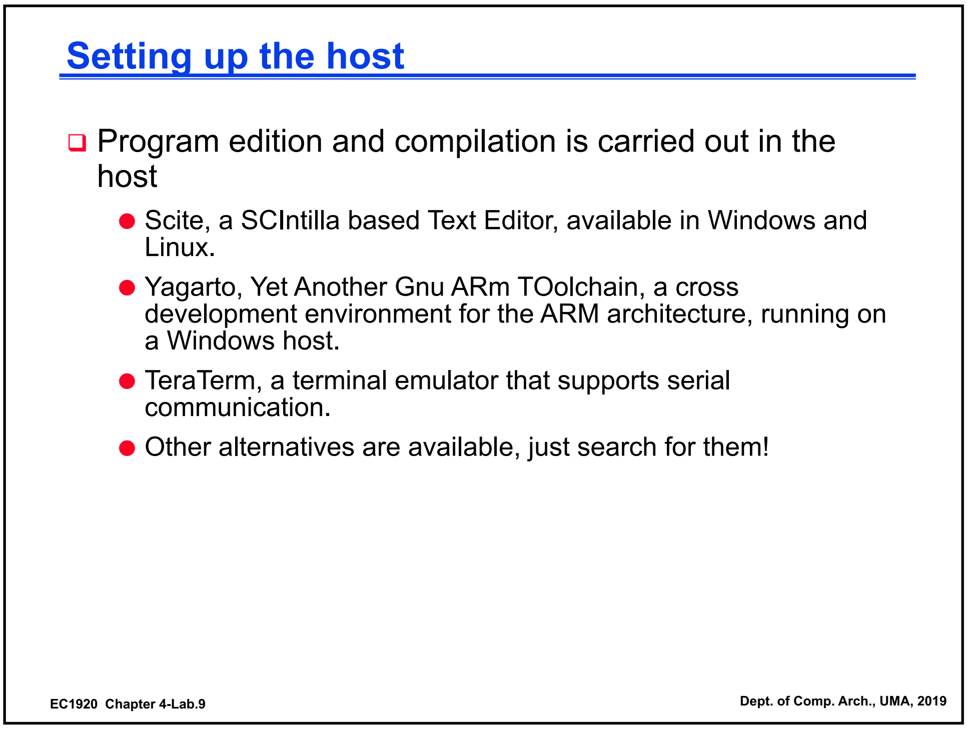
7

Setting up the device

- ❑ Download required files from the course web page.
  - *bootcode.bin* and *start.elf* (and *kernel.img* for USB-serial adapter)
- ❑ Use some program to format the SD card
  - Use FAT filesytem!
- ❑ In case of not using USB-serial adapter
  - Rename your *.img* as *kernel.img* and write it in the SD card
- ❑ Insert SD card in Raspberry

EC1920 Chapter 4-Lab.8 Dept. of Comp. Arch., UMA, 2019

8



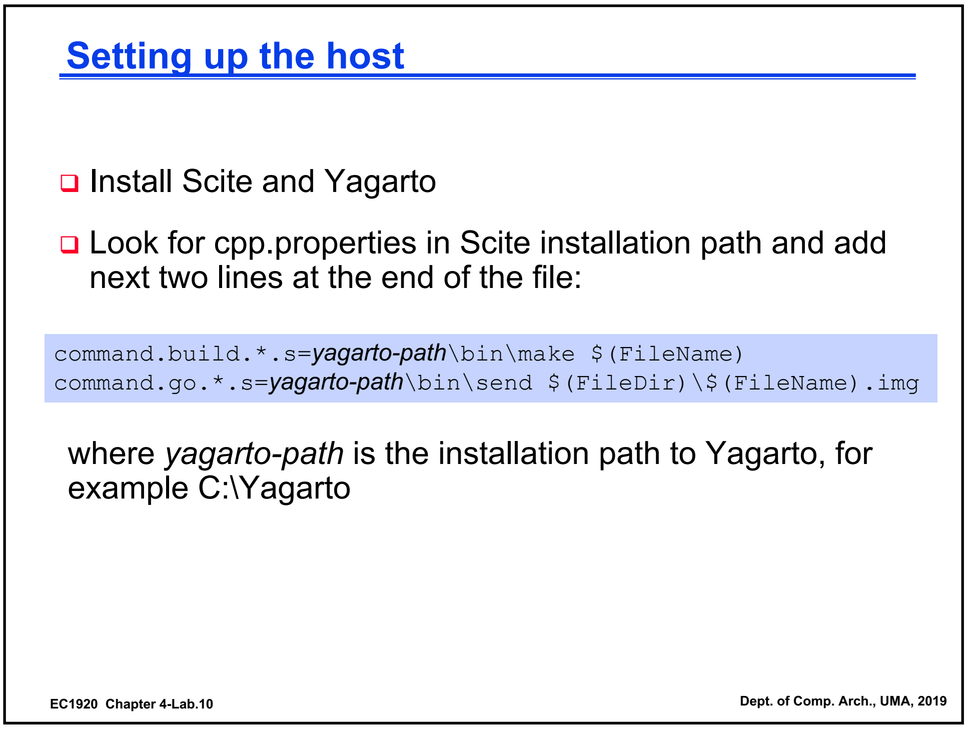
Setting up the host

- ❑ Program edition and compilation is carried out in the host
  - ● Scite, a SCIntilla based Text Editor, available in Windows and Linux.
  - ● Yagarto, Yet Another Gnu ARm TOolchain, a cross development environment for the ARM architecture, running on a Windows host.
  - ● TeraTerm, a terminal emulator that supports serial communication.
  - ● Other alternatives are available, just search for them!

EC1920 Chapter 4-Lab.9

Dept. of Comp. Arch., UMA, 2019

9



Setting up the host

- ❑ Install Scite and Yagarto
- ❑ Look for cpp.properties in Scite installation path and add next two lines at the end of the file:

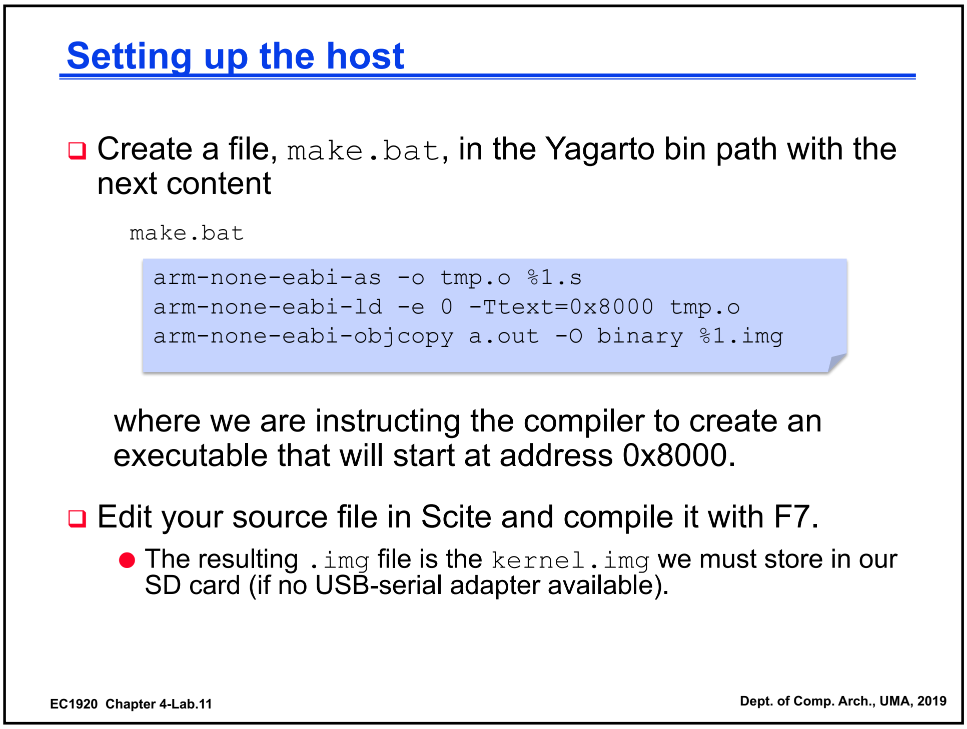
```
command.build.*.s=yagarto-path\bin\make $(FileName)
command.go.*.s=yagarto-path\bin\send $(FileDir)\$(FileName).img
```

where *yagarto-path* is the installation path to Yagarto, for example C:\Yagarto

EC1920 Chapter 4-Lab.10

Dept. of Comp. Arch., UMA, 2019

10



Setting up the host

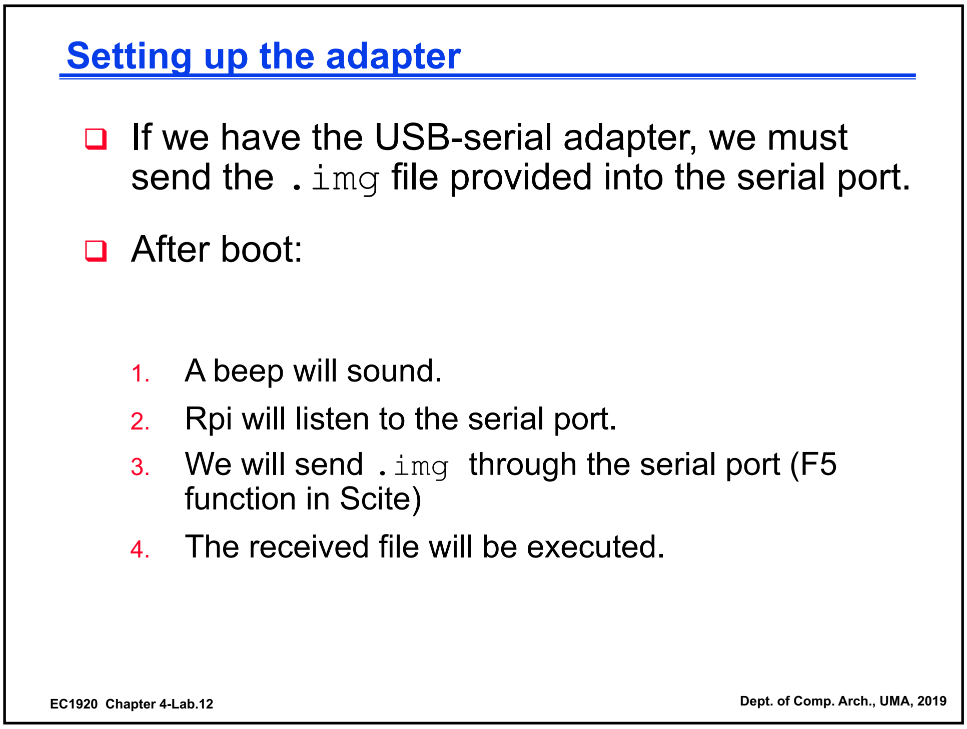
- ❑ Create a file, `make.bat`, in the Yagarto bin path with the next content

make.bat

```
arm-none-eabi-as -o tmp.o %1.s
arm-none-eabi-ld -e 0 -Ttext=0x8000 tmp.o
arm-none-eabi-objcopy a.out -O binary %1.img
```

where we are instructing the compiler to create an executable that will start at address 0x8000.

- ❑ Edit your source file in Scite and compile it with F7.
  - ● The resulting `.img` file is the `kernel.img` we must store in our SD card (if no USB-serial adapter available).



Setting up the adapter

- ❑ If we have the USB-serial adapter, we must send the `.img` file provided into the serial port.
- ❑ After boot:

1. A beep will sound.
2. Rpi will listen to the serial port.
3. We will send `.img` through the serial port (F5 function in Scite)
4. The received file will be executed.

Setting up the adapter

- To be able to send the file, you must install Tera Term, and create a file in Tera Term path, `send.ttl`, with

`send.ttl`

```
connect '/C=3'
setbaud 115200
setdtr 0
xmodemsend param2 1
closett
```

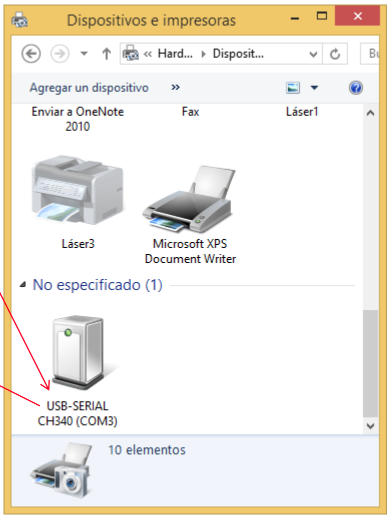
where `'/C=3'` should be substituted by the COMM port of your USB-serial adapter in the host.

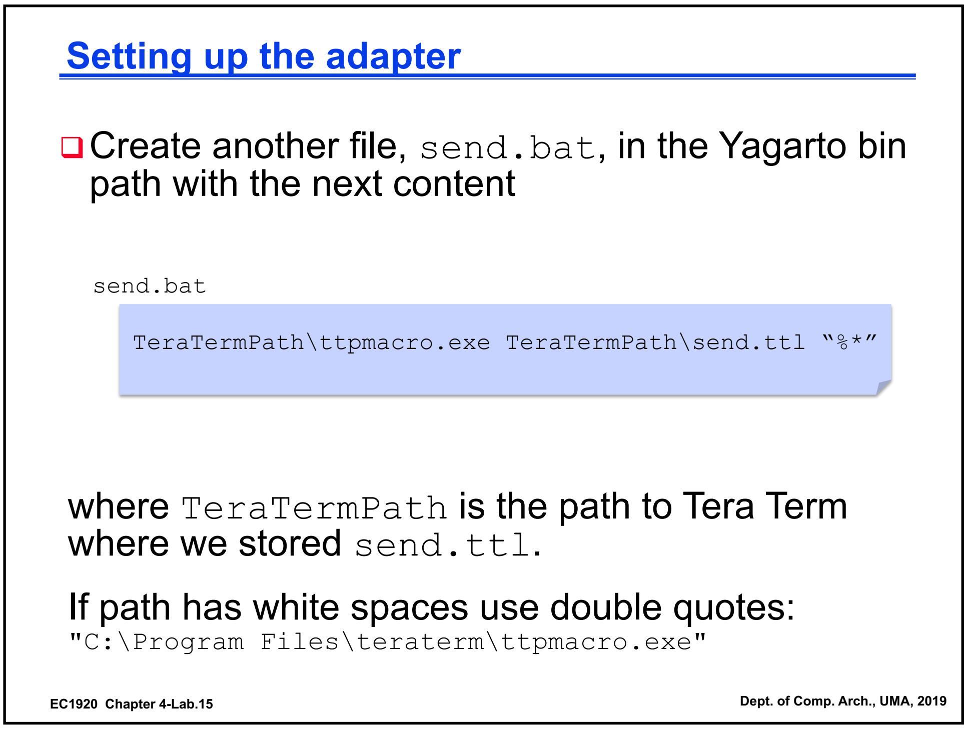
Setting up the adapter

COM3 port

`send.ttl`

```
connect '/C=3'
setbaud 115200
setdtr 0
xmodemsend param2 1
closett
```





Setting up the adapter

- ❑ Create another file, `send.bat`, in the Yagarto bin path with the next content

send.bat

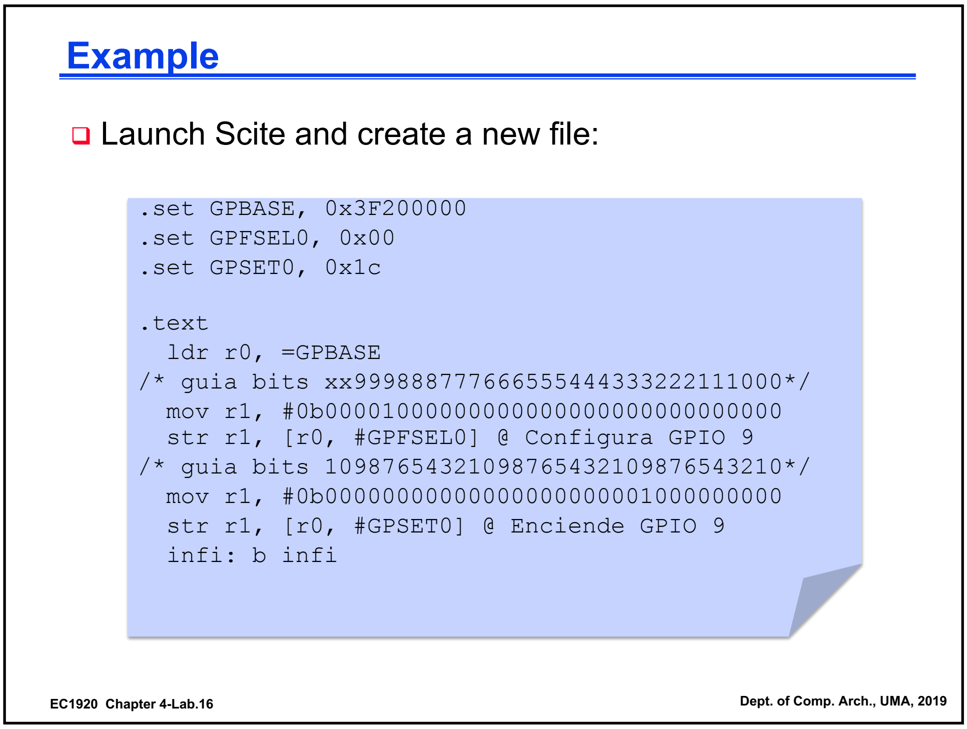
```
TeraTermPath\ttpmacro.exe TeraTermPath\send.ttl "%*"
```

where `TeraTermPath` is the path to Tera Term where we stored `send.ttl`.

If path has white spaces use double quotes:
`"C:\Program Files\teraterm\ttpmacro.exe"`

EC1920 Chapter 4-Lab.15 Dept. of Comp. Arch., UMA, 2019

15



Example

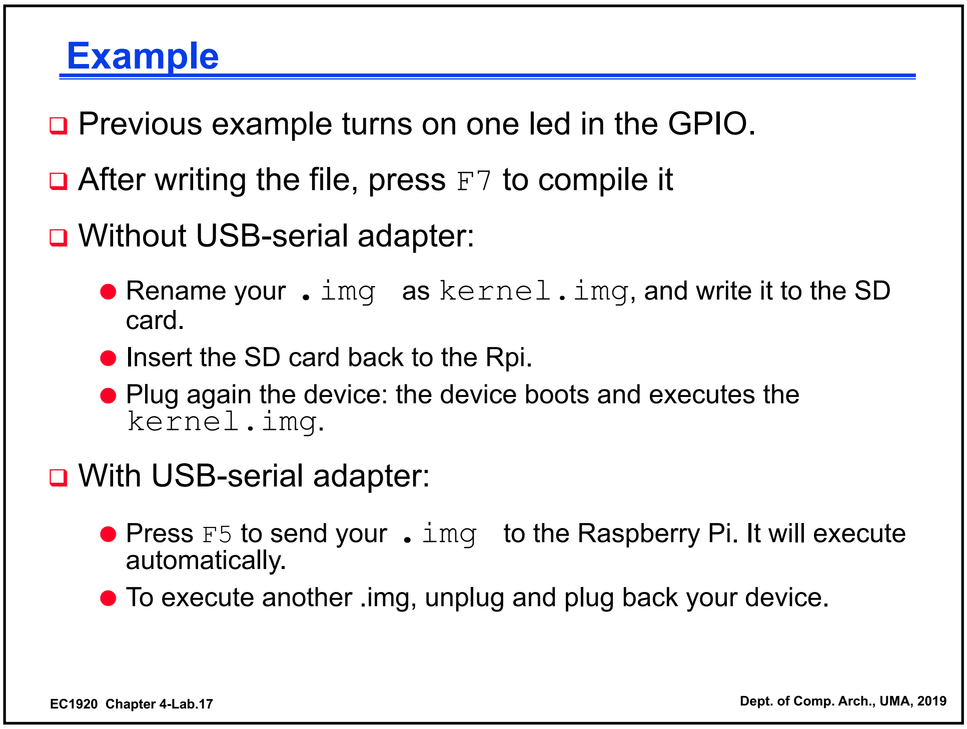
- ❑ Launch Scite and create a new file:

```
.set GPBASE, 0x3F200000
.set GPFSEL0, 0x00
.set GPSET0, 0x1c

.text
  ldr r0, =GPBASE
/* guia bits xx999888777666555444333222111000*/
  mov r1, #0b00001000000000000000000000000000
  str r1, [r0, #GPFSEL0] @ Configura GPIO 9
/* guia bits 10987654321098765432109876543210*/
  mov r1, #0b00000000000000000000001000000000
  str r1, [r0, #GPSET0] @ Enciende GPIO 9
  infi: b infi
```

EC1920 Chapter 4-Lab.16 Dept. of Comp. Arch., UMA, 2019

16

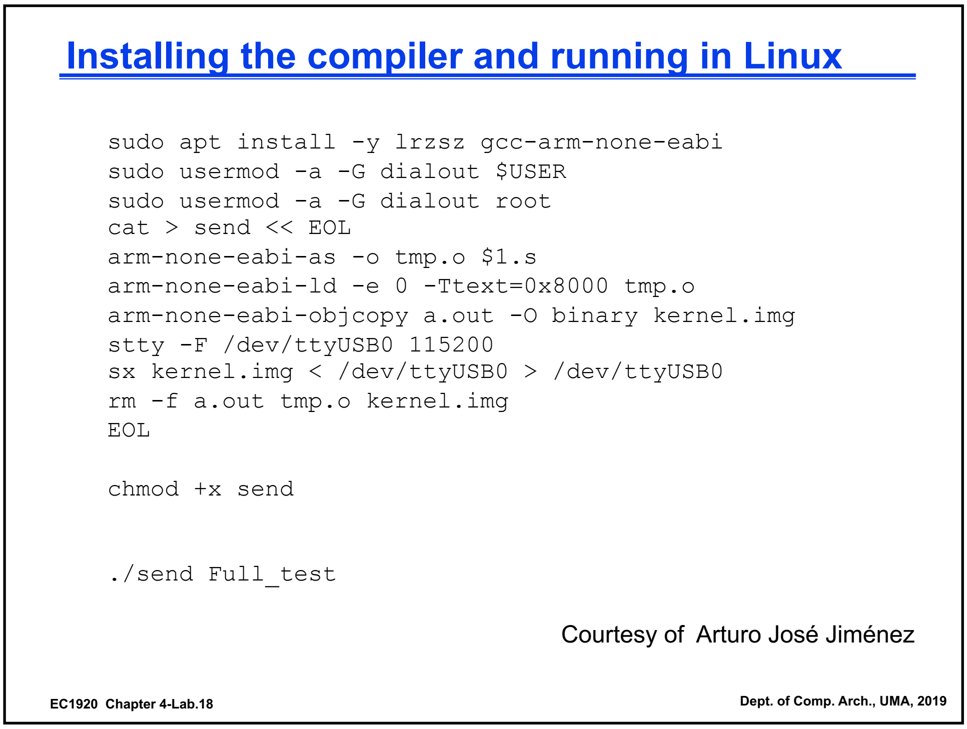


Example

- ❑ Previous example turns on one led in the GPIO.
- ❑ After writing the file, press `F7` to compile it
- ❑ Without USB-serial adapter:
  - ● Rename your `.img` as `kernel.img`, and write it to the SD card.
  - ● Insert the SD card back to the Rpi.
  - ● Plug again the device: the device boots and executes the `kernel.img`.
- ❑ With USB-serial adapter:
  - ● Press `F5` to send your `.img` to the Raspberry Pi. It will execute automatically.
  - ● To execute another .img, unplug and plug back your device.

EC1920 Chapter 4-Lab.17 Dept. of Comp. Arch., UMA, 2019

17



Installing the compiler and running in Linux

```
sudo apt install -y lrzsz gcc-arm-none-eabi
sudo usermod -a -G dialout $USER
sudo usermod -a -G dialout root
cat > send << EOL
arm-none-eabi-as -o tmp.o $1.s
arm-none-eabi-ld -e 0 -Ttext=0x8000 tmp.o
arm-none-eabi-objcopy a.out -O binary kernel.img
stty -F /dev/ttyUSB0 115200
sx kernel.img < /dev/ttyUSB0 > /dev/ttyUSB0
rm -f a.out tmp.o kernel.img
EOL

chmod +x send


./send Full_test
```

Courtesy of Arturo José Jiménez

EC1920 Chapter 4-Lab.18 Dept. of Comp. Arch., UMA, 2019

18