Systèmes d'exploitation pour l'embarqué

UV 5.2 - Exécution et Concurrence

Paul Blottière

ENSTA Bretagne

2015 / 2016

https://github.com/pblottiere

Amélioration continue

Contributions

- Dépôt du cours : `https://github.com/pblottiere/embsys`
- Souhaits d'amélioration, erreurs, idées de TP, ... : ouverture d'Issues (avec le bon label!)
- Apports de corrections : Pull Request

Linux : bus et communication

Plan

1. Matériel et filesystem
2. Bus d’extension PCI
3. GPIO
4. Port parallèle
5. Bus CAN
6. Les bus séries

Matériel et filesystem (1)

/dev

Le répertoire /dev (comme *device*) contient des fichiers spéciaux permettant notamment d'accèder aux éléments matériels de la machine.

Matériel et filesystem (1)

/dev

Le répertoire /dev (comme *device*) contient des fichiers spéciaux permettant notamment d'accèder aux éléments matériels de la machine.

```
> ls /dev
...             dvd             ...
cdrw            dvdrw
cdrom           fb0
...             fb1
```

http://www.linux-france.org/article/grl/Guide_Rootard-12.html

Matériel et filesystem (2)

/proc

Système de fichiers virtuels *procfs* qui contient quelques informations sur la configuration matérielle en plus des paramètres logiciels du noyau.

```
> ls /proc/bus
input pci
> ls/proc/bus/pci
00   01   09   0a   10   devices
```

Matériel et filesystem (3)

/sys

Relativement récent dans le kernel Linux : version 2.6.x.

Matériel et filesystem (3)

/sys

Relativement récent dans le kernel Linux : version 2.6.x.

Système de fichiers virtuels *sysfs* géré par le kernel et fournissant des informations basiques sur les périphériques connectés au système.

Matériel et filesystem (3)

/sys

Relativement récent dans le kernel Linux : version 2.6.x.

Système de fichiers virtuels *sysfs* géré par le kernel et fournissant des informations basiques sur les périphériques connectés au système.

A l'époque, /dev était statique (**mknod**). Désormais, des utilitaires comme udev ou mdev scan /sys et peuple le /dev dynamiquement.

Matériel et filesystem (4)

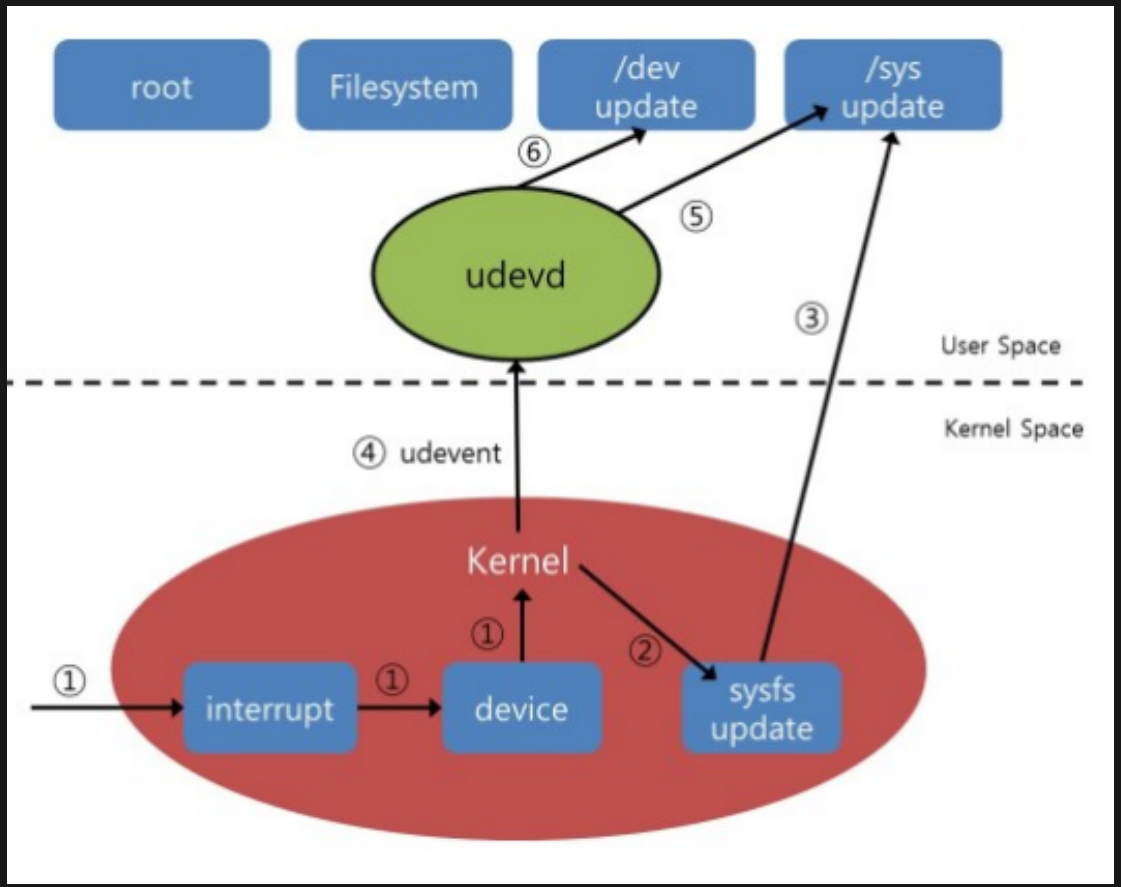
/sys

Le kernel notifie l'espace utilisateur lors de l'insertion grâce au mécanisme de *hotplug*.

Matériel et filesystem (4)

/sys

Le kernel notifie l'espace utilisateur lors de l'insertion grâce au mécanisme de *hotplug*.



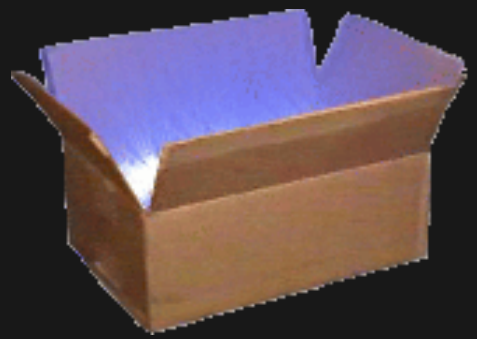
[udevd]

Matériel et filesystem (5)

/sys

Dans les systèmes embarqués :

- mdev (de busybox) est davantage utilisé que udev sur les petits systèmes (démon udevd peut consommer plus d'1 MB de RAM).
- le mécansime de hotplug peut être désactivé.



Bus d’extension PCI (1)

Description

ISA : Industry Standard Architecture / IBM / années 1980

PCI : Peripheral Component Interconnect / Intel / années 1990 (half duplex)

AGP : Accelerated Graphics Port / Intel / années 1990

PCIe : PCIexpress / Intel / années 2000 (full duplex)

Bus d’extension PCI (1)

Description

ISA : Industry Standard Architecture / IBM / années 1980

PCI : Peripheral Component Interconnect / Intel / années 1990 (half duplex)

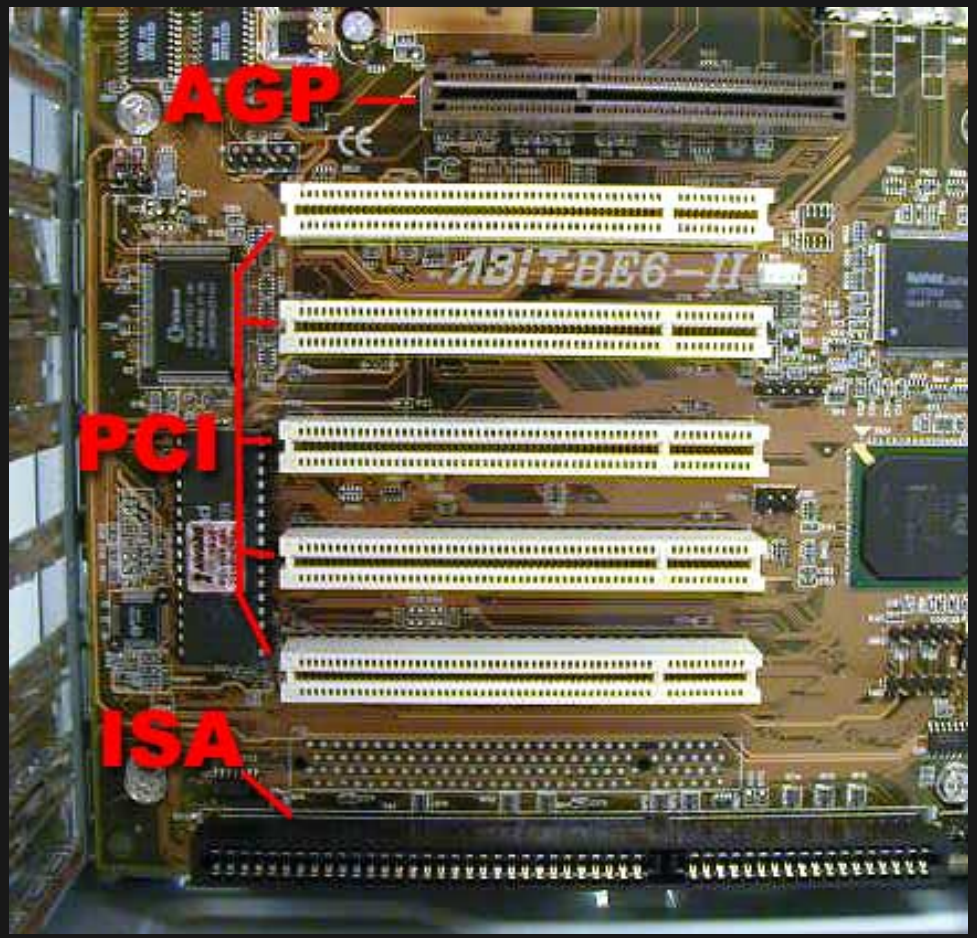
AGP : Accelerated Graphics Port / Intel / années 1990

PCIe : PCIexpress / Intel / années 2000 (full duplex)

=> spécification de bus interne permettant la connexion de cartes d’extension sur la carte mère.

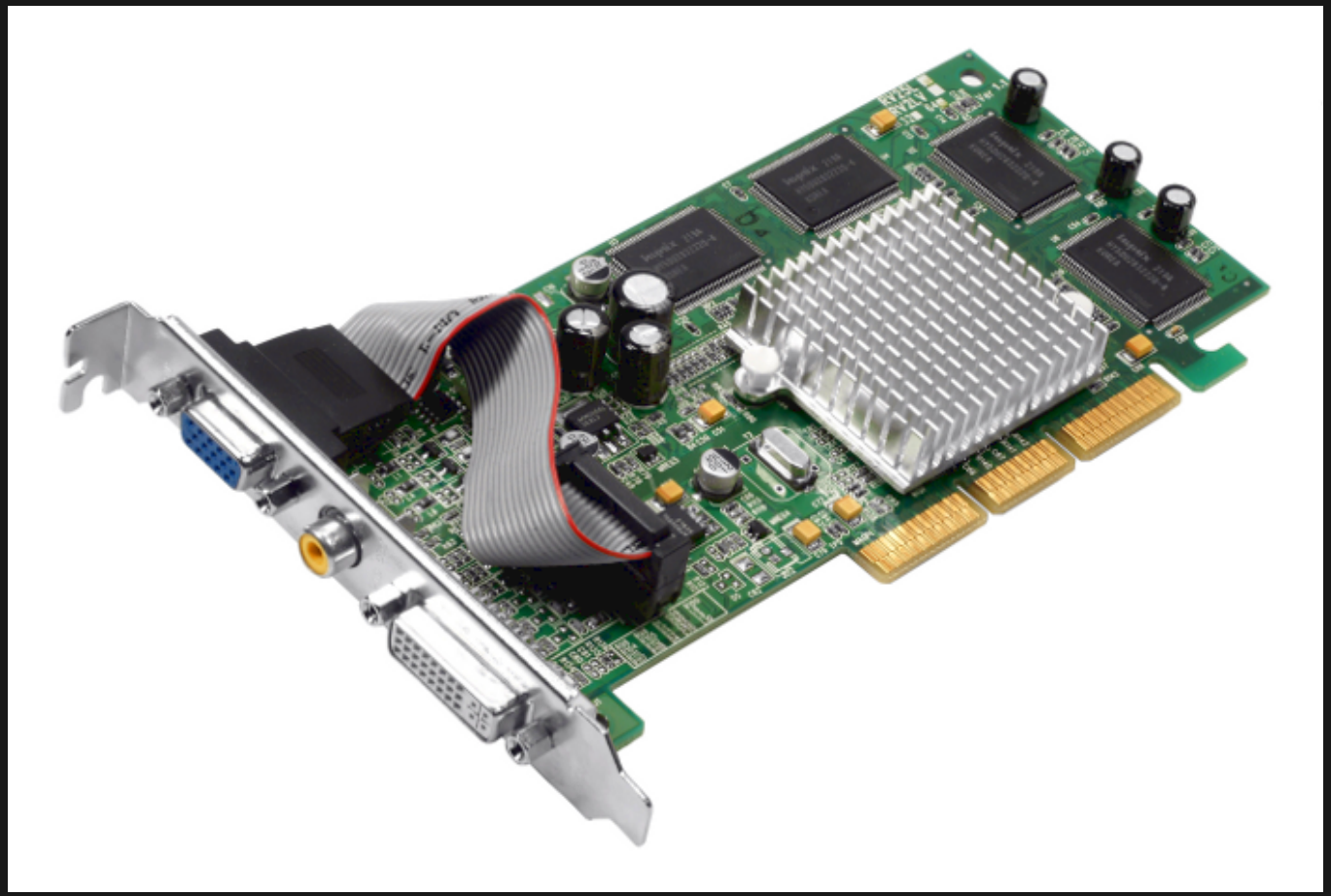
Bus d’extension PCI (2)

Exemples



Bus d’extension PCI (3)

Exemples



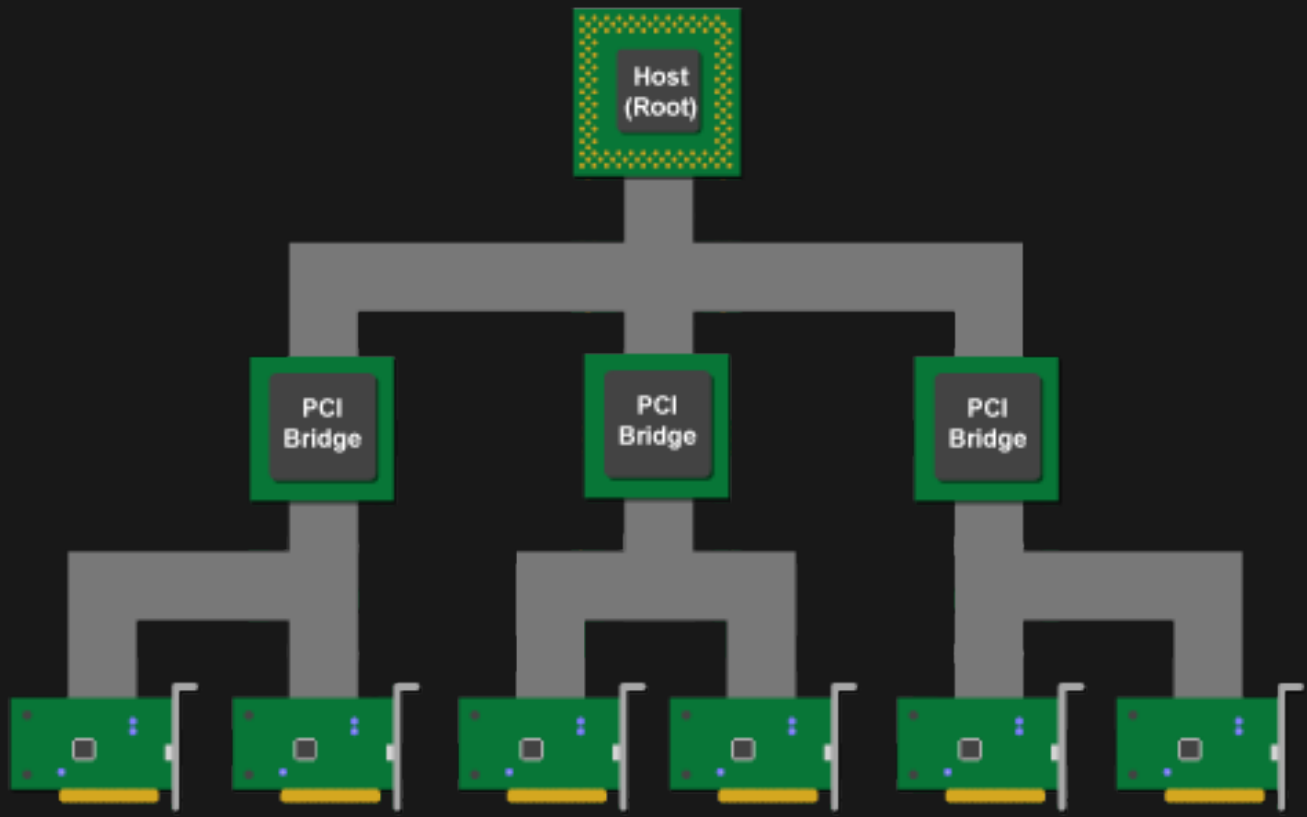
Bus d’extension PCI (4)

Exemples

Bus d’extension PCI (5)

Bridge

Certaines cartes d’extension doivent pouvoir communiquer entre elles. Cela est réalisé par l’intermédiaire d’un *bridge*.



Bus d'extension PCI (6)

Adressage

Un bus PCI est défini sous Linux par quatre numéros :

- domaine : 16 bits
- bus : 8 bits
- port : 5 bits
- fonction : 3 bits

Bus d'extension PCI (6)

Adressage

Un bus PCI est défini sous Linux par quatre numéros :

- domaine : 16 bits
- bus : 8 bits
- port : 5 bits
- fonction : 3 bits

La commande **lspci** peut être utilisée pour lister le matériel PCI : carte d'extension, bridge ou controller.

Bus d'extension PCI (6)

Adressage

Un bus PCI est défini sous Linux par quatre numéros :

- domaine : 16 bits
- bus : 8 bits
- port : 5 bits
- fonction : 3 bits

La commande **lspci** peut être utilisée pour lister le matériel PCI : carte d'extension, bridge ou controller.

[man] : *Some parts of the output, especially in the highly verbose modes, are probably intelligible only to experienced PCI hackers. For exact definitions of the fields, please consult either the PCI specifications or the header.h and /usr/include/linux/pci.h include files.*

Bus d'extension PCI (7)

lspci

```
> lspci
00:00.0 Host bridge: Intel Corporation 2nd Generation Core
        Processor Family DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd
        Generation Core Processor Family PCI Express Root
        Port (rev 09)
00:1a.0 USB controller: Intel Corporation 6 Series/C200
        Series Chipset Family USB Enhanced Host Controller
00:1b.0 Audio device: Intel Corporation 6 Series/C200 Series
        Chipset Family High Definition Audio Controller (rev 05)
00:1f.0 ISA bridge: Intel Corporation HM65 Express Chipset
        Family LPC Controller (rev 05)
01:00.0 VGA compatible controller: Advanced Micro Devices,
        Inc. [AMD/ATI] Seymour [Radeon HD 6400M/7400M Series]
09:00.0 Network controller: Broadcom Corporation BCM4313
        802.11bgn Wireless Network Adapter (rev 01)
10:00.0 Ethernet controller: Qualcomm Atheros AR8151 v2.0
        Gigabit Ethernet (rev c0)
```

Bus d'extension PCI (8)

udevadm

```
root@debian:~# cd /sys/bus/pci/devices/0000\:09\:00.0/
root@debian:/sys/bus/pci/devices/0000:09:00.0# ls
broken_parity_status      ieee80211      remove
class                     index          rescan
config                    irq            reset
consistent_dma_mask_bits  label          resource
d3cold_allowed            local_cpulist  resource0
device                    local_cpus     subsystem
dma_mask_bits             modalias       subsystem_device
driver                    msi_bus        subsystem_vendor
driver_override           net            uevent
enable                    numa_node      vendor
firmware_node             power
root@debian:/sys/bus/pci/devices/0000:09:00.0# udevadm monito
r &
[1] 3023
root@debian:/sys/bus/pci/devices/0000:09:00.0# monitor will p
rint the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent


root@debian:/sys/bus/pci/devices/0000:09:00.0# echo "add" > u
event
KERNEL[28369.927919] add      /devices/pci0000:00/0000:00:1c.
0/0000:09:00.0 (pci)
root@debian:/sys/bus/pci/devices/0000:09:00.0# UDEV  [28369.9
30998] add      /devices/pci0000:00/0000:00:1c.0/0000:09:00.0
 (pci)
```

GPIO (1)

Principe

GPIO : General Purpose Input/Output (1980)

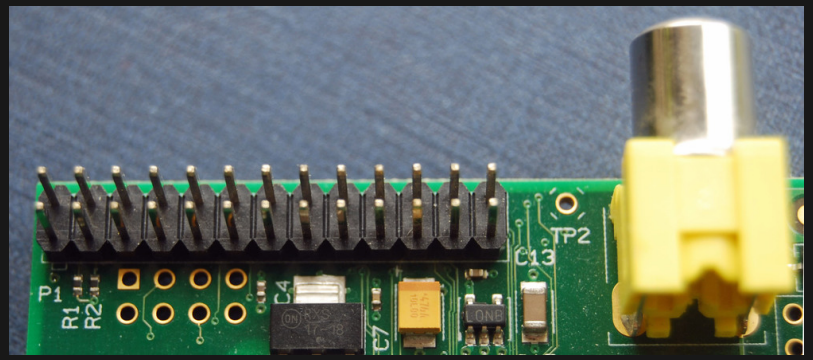
Purement numérique (1 ou 0).

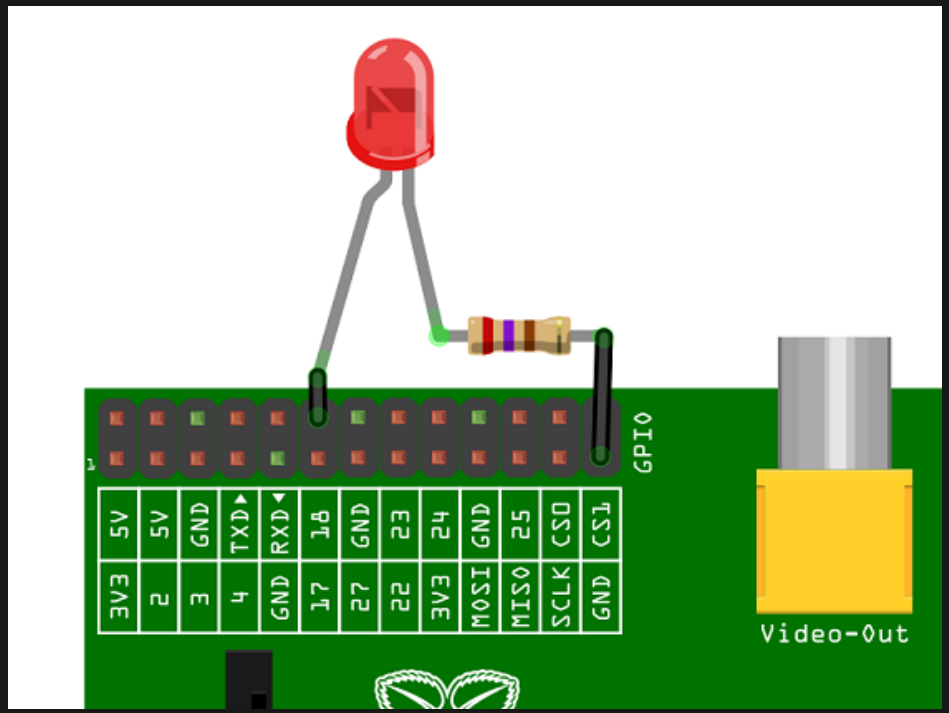
Souvent de tension 3v3 et de faible courant (de 3mA à 50mA)

Très utilisé : trigger, interruption, led, ...

GPIO (2)

Raspberry Pi





GPIO (3)

De l’espace utilisateur

Dans l’espace utilisateur, les GPIOs sont accessibles à travers /sys :

GPIO (3)

De l'espace utilisateur

Dans l'espace utilisateur, les GPIOs sont accessibles à travers /sys :

```
> cd /sys/class/gpio
> ls
export     unexport
> echo 18 > export
export     gpio18     unexport
> cd gpio18
> ls
active_low  direction   edge     subsystem     uevent
value
> cat direction
in
> echo out > direction
> cat value
0
> echo 1 > value
```

GPIO (4)

De l’espace utilisateur

Pour avoir accès aux GPIOs à travers /sys, il faut que le kernel soit configuré pour.

GPIO (4)

De l'espace utilisateur

Pour avoir accès aux GPIOs à travers /sys, il faut que le kernel soit configuré pour.

Example de kernel non configuré :

```
> cat /boot/config-4.2.0-1-amd64 | grep GPIO_SYSFS -B 13 -A 1
#
# Pin controllers
#
# CONFIG_DEBUG_PINCTRL is not set
# CONFIG_PINCTRL_AMD is not set
# CONFIG_PINCTRL_BAYTRAIL is not set
# CONFIG_PINCTRL_CHERRYVIEW is not set
# CONFIG_PINCTRL_SUNRISEPOINT is not set
CONFIG_ARCH_WANT_OPTIONAL_GPIOLIB=y
CONFIG_GPIOLIB=y
CONFIG_GPIO_DEVRES=y
CONFIG_GPIO_ACPI=y
# CONFIG_DEBUG_GPIO is not set
# CONFIG_GPIO_SYSFS is not set
```

GPIO (5)

De l'espace kernel

Depuis l'espace kernel, un module peut avoir accès aux broches en lecture/écriture grâce aux appels :

- gpio_get_value
- gpio_set_value
- gpio_request
- gpio_direction_input
- gpio_direction_output
- ...

Port parallèle (1)

Principe

Port parallèle unidirectionnel : 1980

Port parallèle bidirectionnel : 1990

Communication octet par octet (transmission simultanée de 8 bits).

Notamment utilisé par les périphériques type imprimante (/dev/lpX).

Port parallèle (2)

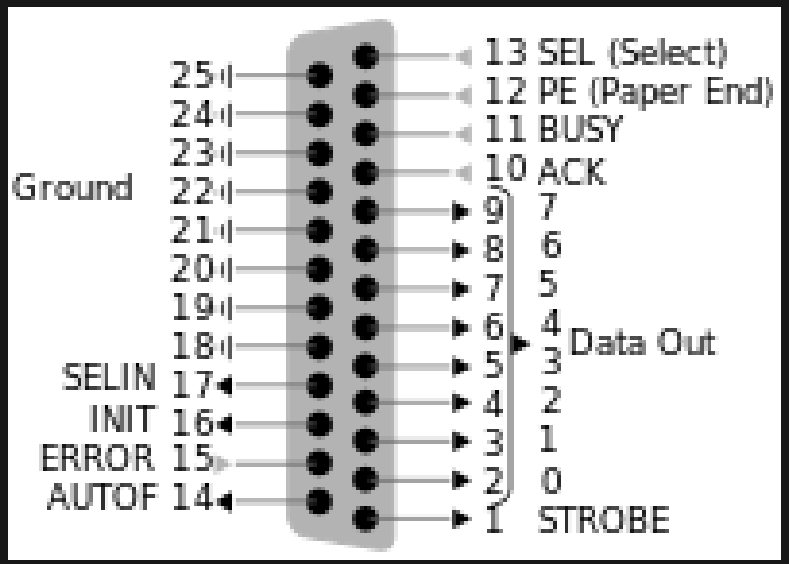
Connecteur DB25

Côté connecteur, un port parallèle est représenté par un DB25 :

Port parallèle (2)

Connecteur DB25

Côté connecteur, un port parallèle est représenté par un DB25 :



Port parallèle (3)

Port à tout faire

Le DB25 est souvent utilisé comme un melting pot d'entrées/sorties sans lien avec la communication parallèle.

Port parallèle (3)

Port à tout faire

Le DB25 est souvent utilisé comme un melting pot d'entrées/sorties sans lien avec la communication parallèle.

Communication série (1)

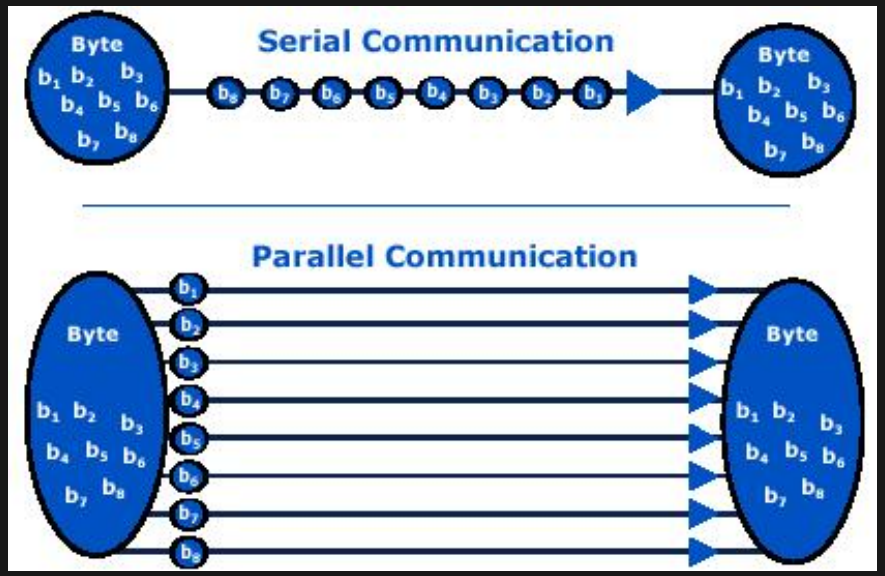
Principe

Contrairement à la communication parallèle, le protocole série envoie des informations bit par bit.

Communication série (1)

Principe

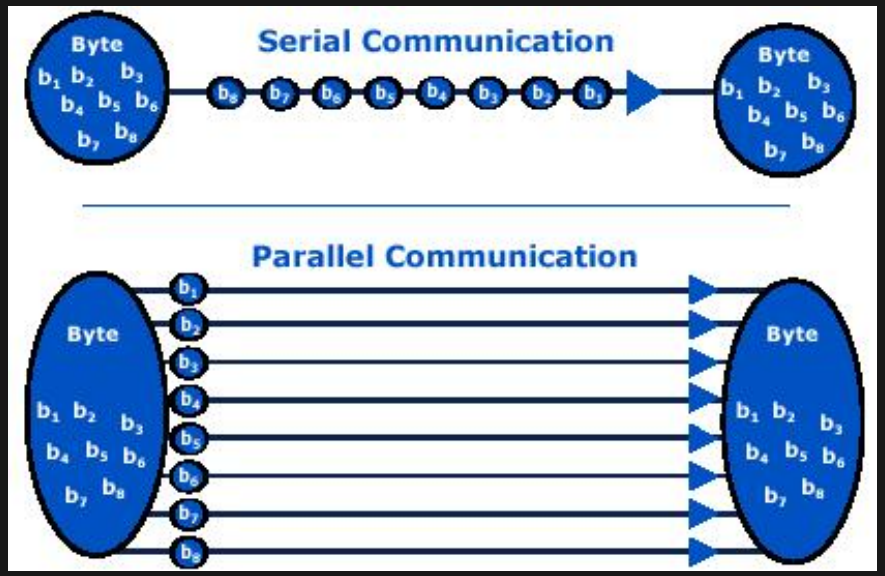
Contrairement à la communication parallèle, le protocole série envoie des informations bit par bit.



Communication série (1)

Principe

Contrairement à la communication parallèle, le protocole série envoie des informations bit par bit.



=> on a alors un multiplexage de type TDM (Time Division Multiplexing)

Communication série (2)

Description

Les avantages de la liaison série par rapport à la communication parallèle :

- moins de diaphonie (induction électromagnétique)
- skew moins important (différence de temps de propagation)

Communication série (2)

Description

Les avantages de la liaison série par rapport à la communication parallèle :

- moins de diaphonie (induction électromagnétique)
- skew moins important (différence de temps de propagation)

Il existe de nombreux types de liaison/bus série :

- RS232 / RS422 / RS485 / RS423
- I2C
- USB
- et d'autres encore (SPI, TTL, ...)

Communication série (3)

RS232

Au minimum 3 fils sont nécessaires pour ce type de communication : c'est une liaison point à point non différentielle full-duplex (RX / TX / GND).

Communication série (3)

RS232

Au minimum 3 fils sont nécessaires pour ce type de communication : c'est une liaison point à point non différentielle full-duplex (RX / TX / GND).

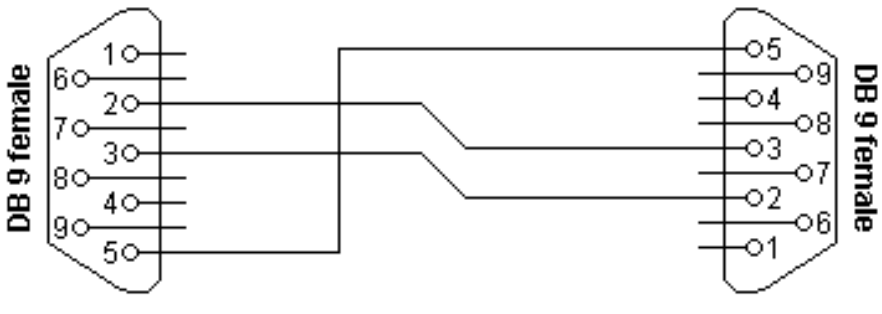
Il existe plusieurs types de brochage permettant d'exploiter ou non la totalité des fonctionnalités du protocole comme le handshaking ou le contrôle de flux.

Communication série (3)

RS232

Au minimum 3 fils sont nécessaires pour ce type de communication : c'est une liaison point à point non différentielle full-duplex (RX / TX / GND).

Il existe plusieurs types de brochage permettant d'exploiter ou non la totalité des fonctionnalités du protocole comme le handshaking ou le contrôle de flux.



Connector 1	Connector 2	Function
2	3	Rx ← Tx
3	2	Tx → Rx
5	5	Signal ground

Communication série (4)

RS232

Pour réaliser une communication série, un protocole doit être établi :

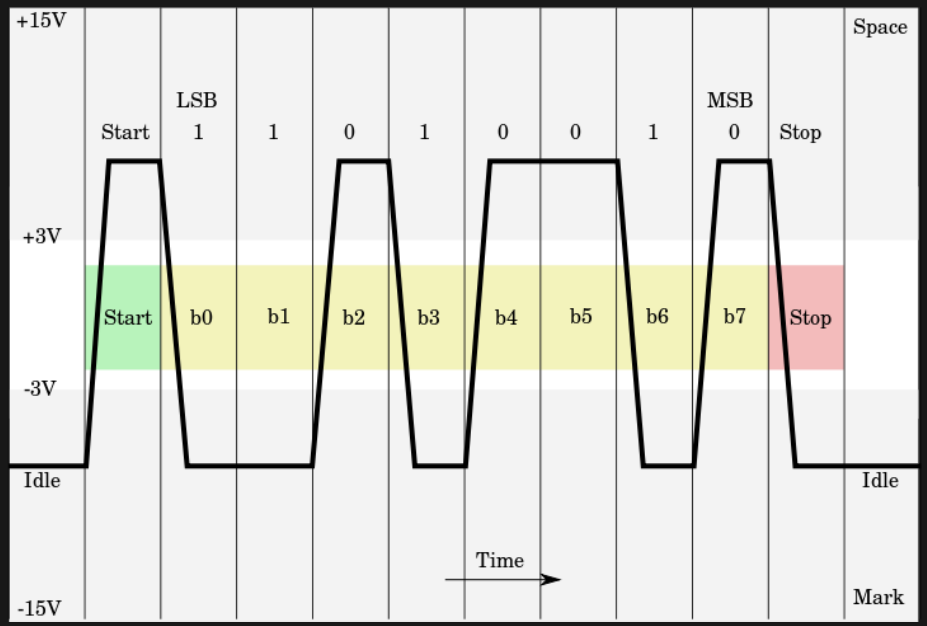
- 1 bit de départ
- 7 ou 8 bits de données
- 1 bit de parité optionnel
- 1 ou plusieurs bits d'arrêt

Communication série (4)

RS232

Pour réaliser une communication série, un protocole doit être établi :

- 1 bit de départ
- 7 ou 8 bits de données
- 1 bit de parité optionnel
- 1 ou plusieurs bits d'arrêt



[K]

Communication série (5)

RS232

Sous Linux, l'API **termios** est utilisée pour gérer les interfaces série à travers **/dev/ttySx** dans l'espace utilisateur.

Communication série (5)

RS232

Sous Linux, l'API **termios** est utilisée pour gérer les interfaces série à travers **/dev/ttySx** dans l'espace utilisateur.

La programmation d'une interface série se fait via :

- la structure **termios** : configuration générale (parité, contrôle de flux, ...)
- **cfsetospeed** / **cfsetispeed** : le débit
- **tcsetattr** : mise en place effective de la configuration
- **fcntl** : lecture bloquante ou non
- **open** / **read** / **clos**e : gestion du file descriptor

Communication série (5)

RS232

Sous Linux, l'API **termios** est utilisée pour gérer les interfaces série à travers **/dev/ttySx** dans l'espace utilisateur.

La programmation d'une interface série se fait via :

- la structure **termios** : configuration générale (parité, contrôle de flux, ...)
- **cfsetospeed** / **cfsetispeed** : le débit
- **tcsetattr** : mise en place effective de la configuration
- **fcntl** : lecture bloquante ou non
- **open** / **read** / **clos**e : gestion du file descriptor

Référence : `http://tldp.org/HOWTO/Serial-Programming-HOWTO/x115.html`

Communication série (6)

I2C

Inter Integrated Circuit : bus série half-duplex.

Communication série (6)

I2C

Inter Integrated Circuit : bus série half-duplex.

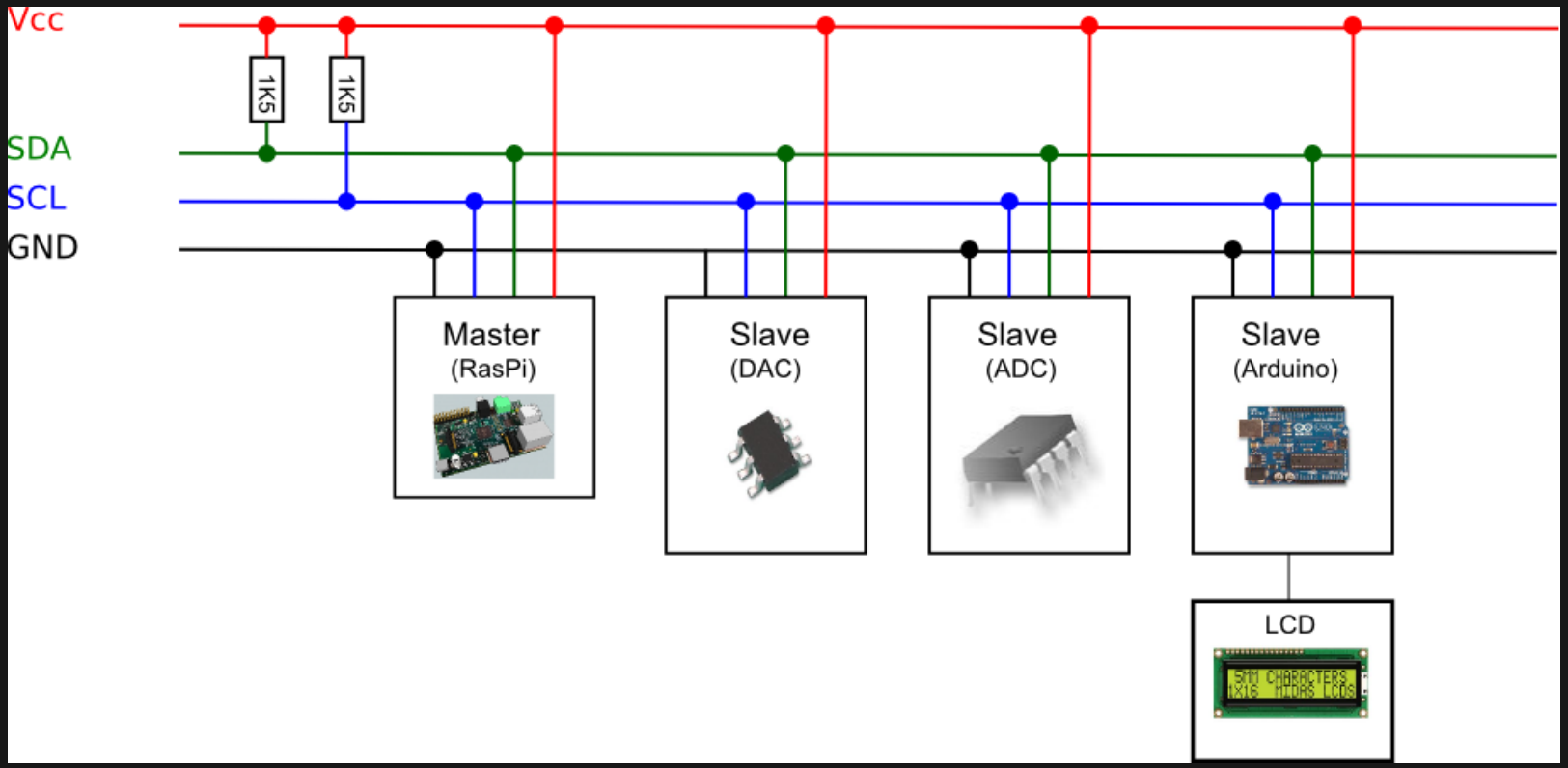
La topologie du bus est de type maître esclave où la connexion est réalisée via 3 fils :

- La masse (GND)
- Serial Data Line (SDA) : un signal de données
- Serial Clock Line (SCL) : un signal de synchronisation

Communication série (7)

I2C

Exemple de montage :



[Montage I2C]

Communication série (8)

I2C

Depuis l'espace utilisateur, le matériel I2C est accessible à travers /sys après chargement d'un module :

```
> sudo modprobe i2c-dev
> cd /sys/class/i2c-dev/
> ls
i2c-0  i2c-1  i2c-10  i2c-11  i2c-12  i2c-13  i2c-14
i2c-2  i2c-3  i2c-4   i2c-5   i2c-6   i2c-7   i2c-8
i2c-9
> cd  i2c-0
> ls
dev  device  name  power  subsystem  uevent
```

Communication série (9)

I2C

Exemple d'ouverture d'un port I2C par flux :

```
int file;
char *filename = "/dev/i2c-2";
if ((file = open(filename, O_RDWR)) < 0) {
    perror("Failed to open the i2c bus");
    exit(1);
}
```

Conclusion

Il existe de très nombreux protocoles de communication et connaître les principaux est indispensable.

Références

- https://github.com/torvalds/linux/tree/master/Documentation
- https://lwn.net/Kernel/LDD3/
- http://tldp.org/HOWTO/Serial-Programming-HOWTO/
- Linux Embarqué - Pierre Ficheux