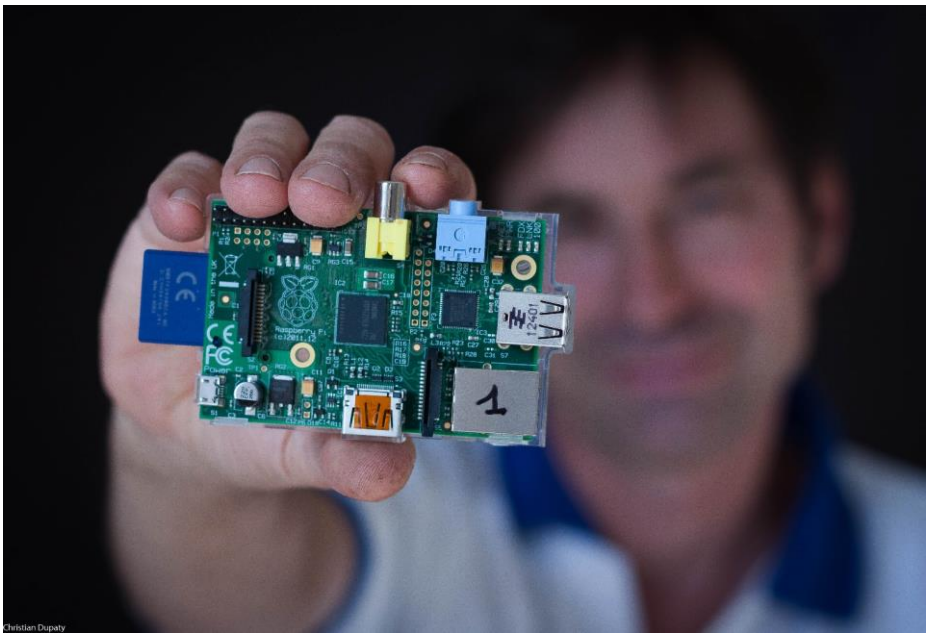
RASPBERRY PI

INSTALLATION-CONFIGURATION

INTERFACES DE COMMUNICATIONS



UART

Christian Dupaty

BTS Systèmes Numériques

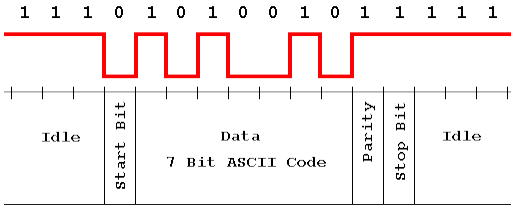
Lycée Fourcade - Gardanne

Académie d'Aix-Marseille

1) TP : UART :

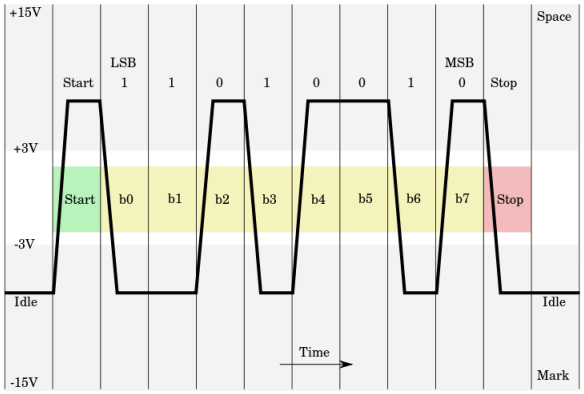
Les communications asynchrones (sans horloge) nécessitent une resynchronisation systématique lors de l'émission d'un octet. L'octet est encadré d'un bit de start et d'un bit de stop. Le bit de start indique le début de la transmission (il peut arriver n'importe quand puisqu'elle est asynchrone) et permet au récepteur de se resynchroniser. Ce protocole associé à un code NRZ (non return to zero) a permis de développer les normes EIA-232 (souvent appelée *RS-232*), EIA-485, EIA-422.
Les communications asynchrones sont utilisées pour les longues distances (Ethernet, téléphonie) mais également en inter-circuit. Le logiciel « terminal ASCII » assure très simplement les échanges avec un circuit intégré ou un autre ordinateur en ASCII à travers une communication asynchrone.

Trame asynchrone (le bit de parité n'est généralement pas employé):



© jeromeabel.net

Trame RS232 (NRZ): le niveau logique 1 est une tension négative, le niveau logique 0 une tension positive.



© wikimedia.org

Quelques modules nécessitant l'utilisation de l'UART du Raspberry Pi.

Module WIFI Microchip RN171XV

Module XBEE Maxstream XBP24-AWI-001

Télémètre ultrason MS-EZ1

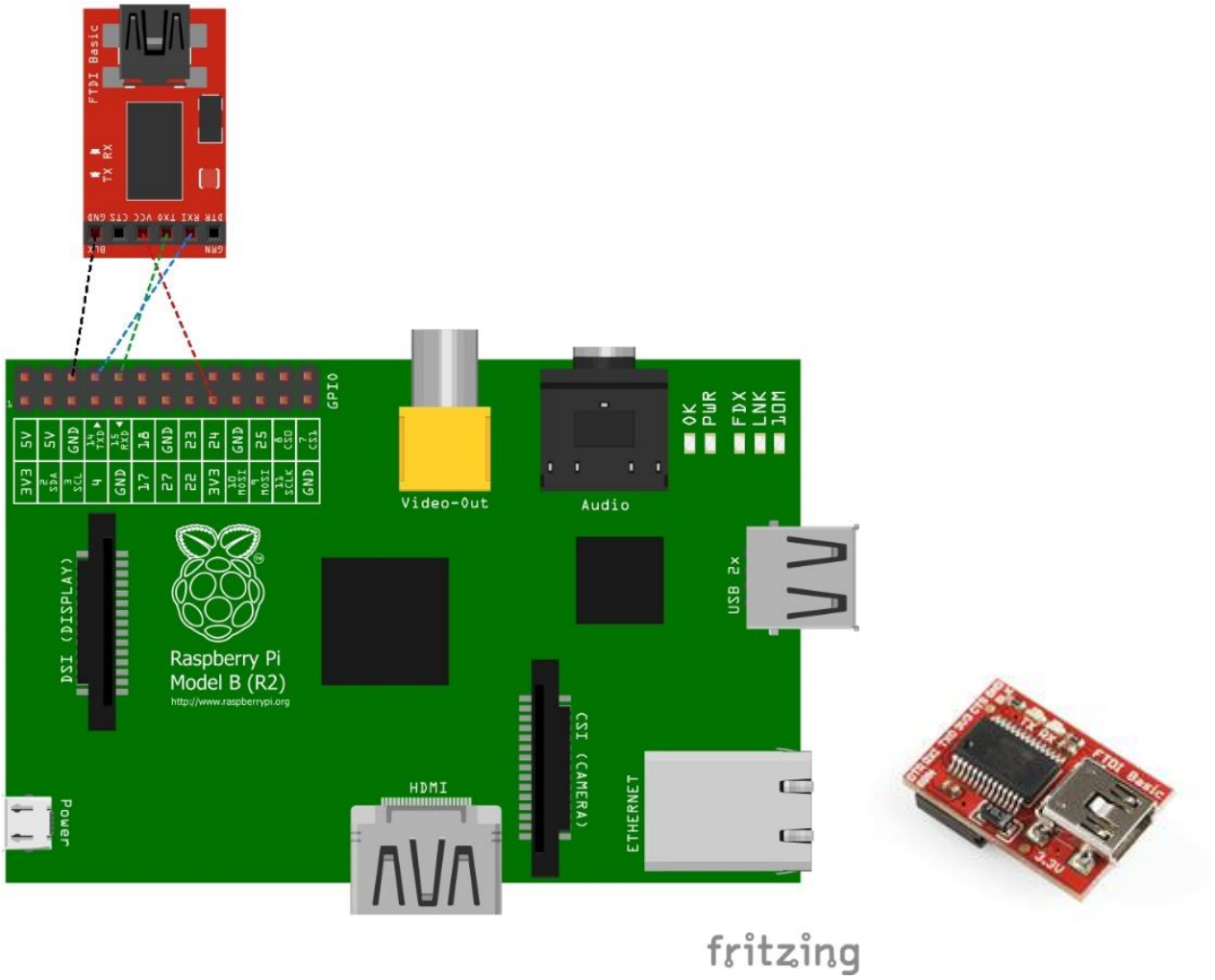
Les UART (Universal Asynchronous Receiver Transmitter) assurent l'émission et la réception asynchrone.

Sur le Raspberry Pi, les niveaux sont 0v, 3.3v
TXD : Port GPIO14 (broche 8)
RXD : Port GPIO15 (broche 10)

Le TP met en œuvre rapidement et simplement une communication filaire asynchrone entre un PC et la Raspberry Pi. Les ordinateurs modernes n'étant plus équipés de ports RS232, il est nécessaire d'utiliser un petit convertisseur USB-série asynchrone.

Par exemple une carte FTDI : https://www.sparkfun.com/products/9873
(installer le driver FTDI sur le PC) puis relier :
TXD(RPi) vers RXD(FTDI) (important, les lignes RX-TX doivent être croisées)
RXD(RPi) vers TXD(FTDI)
GND(Pi) vers GND(FTDI)

NE PAS CONNECTER L'ALIMENTATION 3.3v DU FTDI AVEC RASPBERRY PI



Par défaut l'UART du Raspberry Pi sert de port de débug pour Linux.
Pour essayer cette fonctionnalité (avant de la désactiver afin de contrôler l'UART par un programme Python), configurer un terminal ASCII (HyperTerminal ou Terminal19b) 115200 Bauds, 8 bits, 1 stop, No parity, pas de protocole Handshake.
Le logiciel terminal19b est téléchargeable ici https://sites.google.com/site/terminalbpp/

Libérer l'UART du mode debug
Empêcher l'émission de messages du Kernel et l'activation du mode debugging sur l'UART

```
Sudo nano /boot/cmdline.txt
Supprimer : console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
Enregistrer
```

Désactiver la demande de connexion après le boot

```
Sudo nano /etc/inittab
Commenter la derniere ligne en joutant un # :
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
Enregistrer
```

Redémarrer : `sudo reboot now`

`/dev/ttyAMA0` est maintenant un port Linux libre

Test de connexion avec un PC par liaison série

Récupérer la bibliothèque Python

```
sudo apt-get install python-serial
```

Fichier de demo : demoUART.py

```
import serial

port = serial.Serial("/dev/ttyAMA0", baudrate=115200, timeout=3.0)
# le programme redemarre toutes les 3s

while True:
    port.write("\r\nEcrire quelque chose:")
    rcv = port.read(10)      # lit 10 caracteres
    port.write("\r\nVous avez ecrit :" + repr(rcv))
# l'appel de la function repr() n'est pas indispensable
```

Autre exemple

```
import serial

port = serial.Serial("/dev/ttyAMA0", baudrate=115200, timeout=3.0)
# le programme redemarre toutes les 3s

while True:
    port.write("\r\nEcrire quelque chose:")
    rcv = readline() # lit jusqu a CRLF
    port.write("\r\nVous avez ecrit :" + repr(rcv))
# l'appel de la function repr() n'est pas indispensable
```