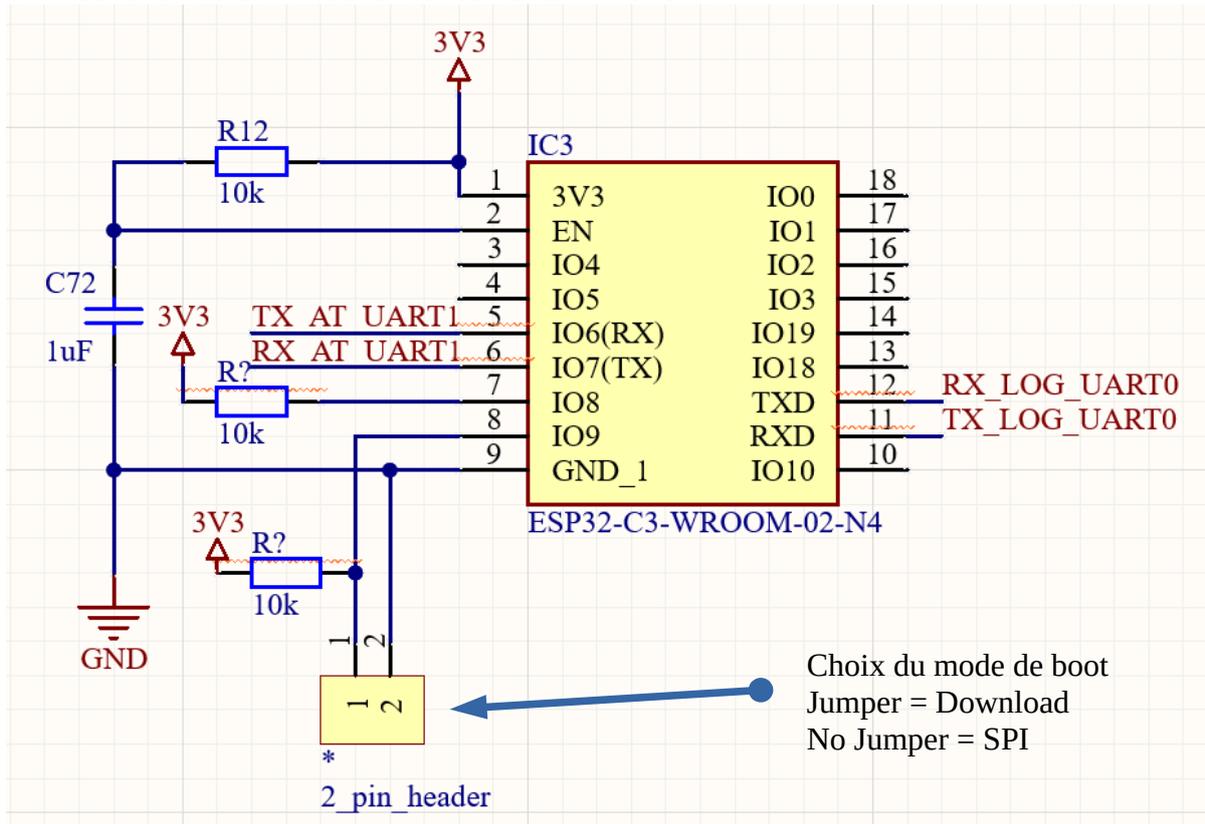


GUIDE ESP32-C3-WROOM-02

Guide réalisé pour cet ESP32 : 356-ESP32C3WROOM02N4

Voici un schéma des connexions à effectuer sur l'ESP32 :



L'UART 0 sert à flasher l'ESP32 et à récupérer les LOG, alors que l'UART 1 permet de communiquer en utilisant les commandes AT une fois que le programme est flashé. Ces informations viennent de ce tableau disponible dans le guide de l'ESP32.

Table 6: ESP32-C3 Series Hardware Connection Pinout

Function of Connection	ESP Board Pins	Other Device Pins
Download/Log output ¹	UART0 <ul style="list-style-type: none"> • GPIO20 (RX) • GPIO21 (TX) 	PC <ul style="list-style-type: none"> • TX • RX
AT command/response ²	UART1 <ul style="list-style-type: none"> • GPIO6 (RX) • GPIO7 (TX) • GPIO5 (CTS) • GPIO4 (RTS) 	USB to serial converter <ul style="list-style-type: none"> • TX • RX • RTS • CTS

Ensuite l'ESP32 a plusieurs modes de démarrage :

Booting Mode ¹			
Pin	Default	SPI Boot	Download Boot
IO8	N/A	Any value	1
IO9	Pull-up	1	0

Il y a le mode Download qui permet de flasher l'ESP32 avec son firmware, et le mode SPI boot qui permet de démarrer sur le firmware préalablement flashé.

A noter que les pin IO8 et IO9 ne doivent pas être mise à 0 en même temps.

Pour commencer coupez l'alimentation de votre ESP32. Connectez la pin IO8 à 1, et IO9 à 0 afin de la mettre en mode download (Branchez le Jumper). Connecter votre adaptateur USB-UART sur l'UART0. Pour finir allumer l'ESP32 en l'alimentant.

Vous devriez obtenir ce message : (Ou n'importe quelle message d'erreur, ex invalid_header)

```
ESP-ROM:esp32c3-api1-20210207
Build:Feb  7 2021
rst:0x1 (POWERON),boot:0x4 (DOWNLOAD(USB/UART0/1))
waiting for download
```

Cette étape permet simplement de vérifier que la connexion est fonctionnelle.

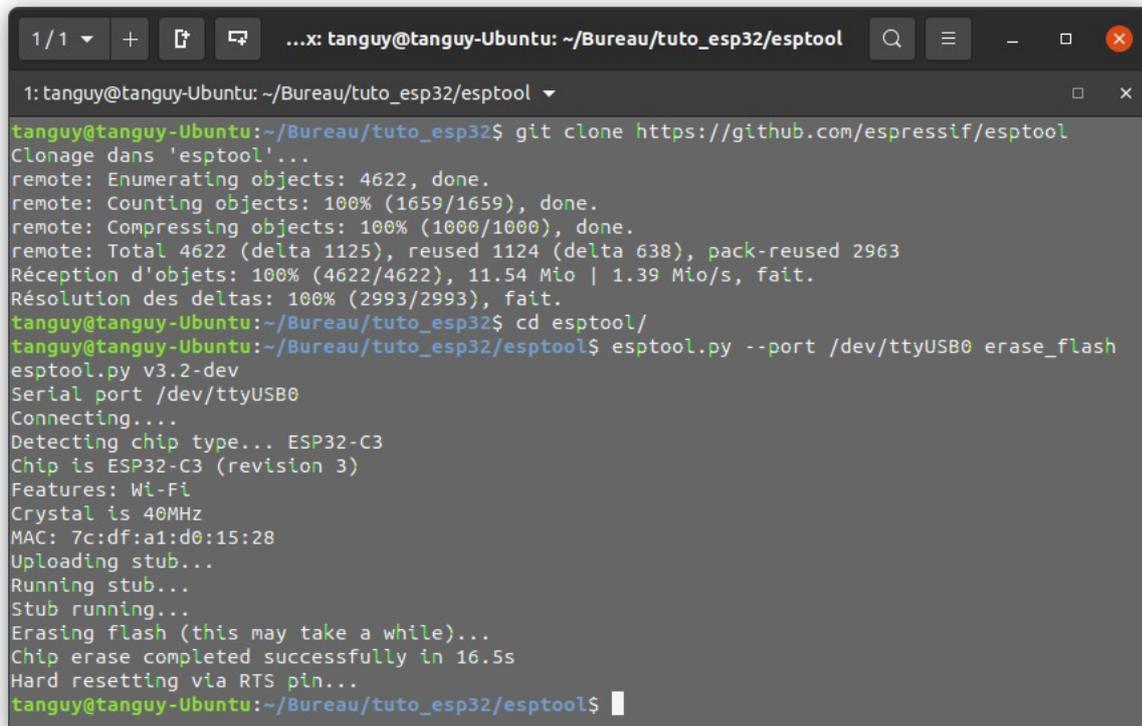
Télécharger l'outil esptool à cette adresse :

<https://github.com/espressif/esptool>

Ensuite ouvrez un terminal dans le dossier, et exécutez la commande suivante :

```
> esptool.py --port /dev/ttyUSB0 erase_flash
```

Voici ce que vous devriez obtenir :



```
1/1 + [ ] ...x: tanguy@tanguy-Ubuntu: ~/Bureau/tuto_esp32/esptool
1: tanguy@tanguy-Ubuntu: ~/Bureau/tuto_esp32/esptool
tanguy@tanguy-Ubuntu:~/Bureau/tuto_esp32$ git clone https://github.com/espressif/esptool
Clonage dans 'esptool'...
remote: Enumerating objects: 4622, done.
remote: Counting objects: 100% (1659/1659), done.
remote: Compressing objects: 100% (1000/1000), done.
remote: Total 4622 (delta 1125), reused 1124 (delta 638), pack-reused 2963
Réception d'objets: 100% (4622/4622), 11.54 Mio | 1.39 Mio/s, fait.
Résolution des deltas: 100% (2993/2993), fait.
tanguy@tanguy-Ubuntu:~/Bureau/tuto_esp32$ cd esptool/
tanguy@tanguy-Ubuntu:~/Bureau/tuto_esp32/esptool$ esptool.py --port /dev/ttyUSB0 erase_flash
esptool.py v3.2-dev
Serial port /dev/ttyUSB0
Connecting...
Detecting chip type... ESP32-C3
Chip is ESP32-C3 (revision 3)
Features: Wi-Fi
Crystal is 40MHz
MAC: 7c:df:a1:d0:15:28
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 16.5s
Hard resetting via RTS pin...
tanguy@tanguy-Ubuntu:~/Bureau/tuto_esp32/esptool$
```

Maintenant on va télécharger le firmware de l'esp32, rendez vous à cette adresse :

https://github.com/espressif/esp-at/releases/tag/v2.3.0.0_esp32c3

ESP32-C3 AT Release v2.3.0.0

Documentation for Release v2.3.0.0 is available at https://docs.espressif.com/projects/esp-at/en/release-v2.3.0.0_esp32c3/

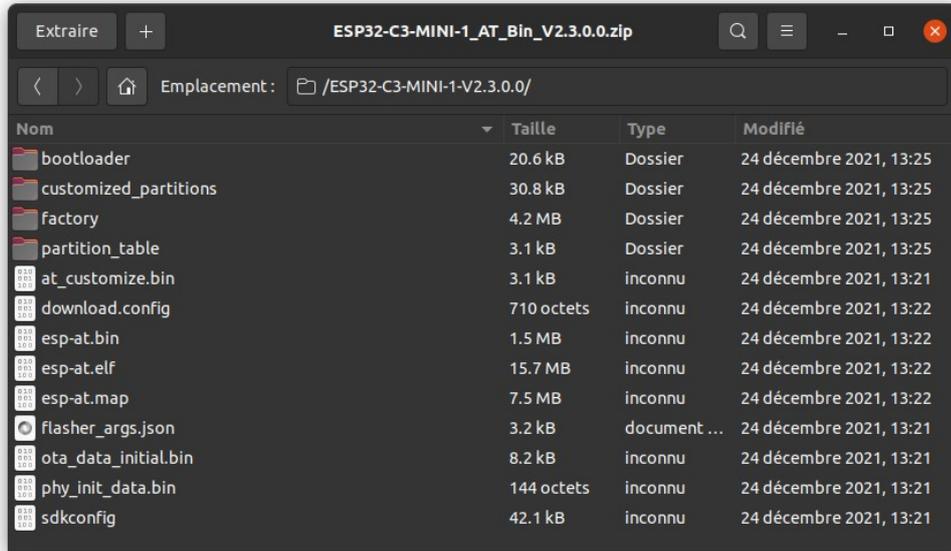
ESP32-C3 AT v2.3.0.0 is a major update for ESP32-C3 AT v2.2.0.0.

The firmwares:

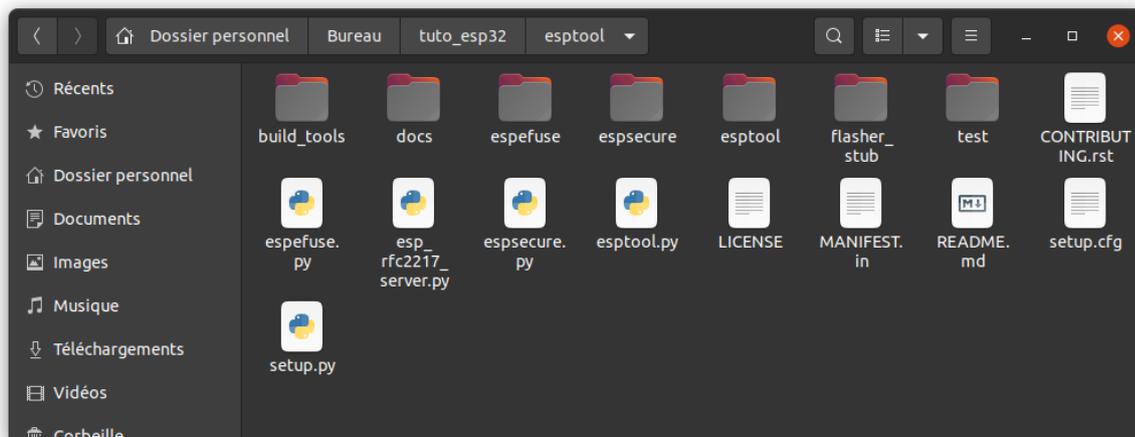
[ESP32-C3-MINI-1_AT_Bin_V2.3.0.0.zip](#)

Cliquez sur [ESP32-C3-MINI-1_AT_Bin_V2.3.0.0.zip](#)

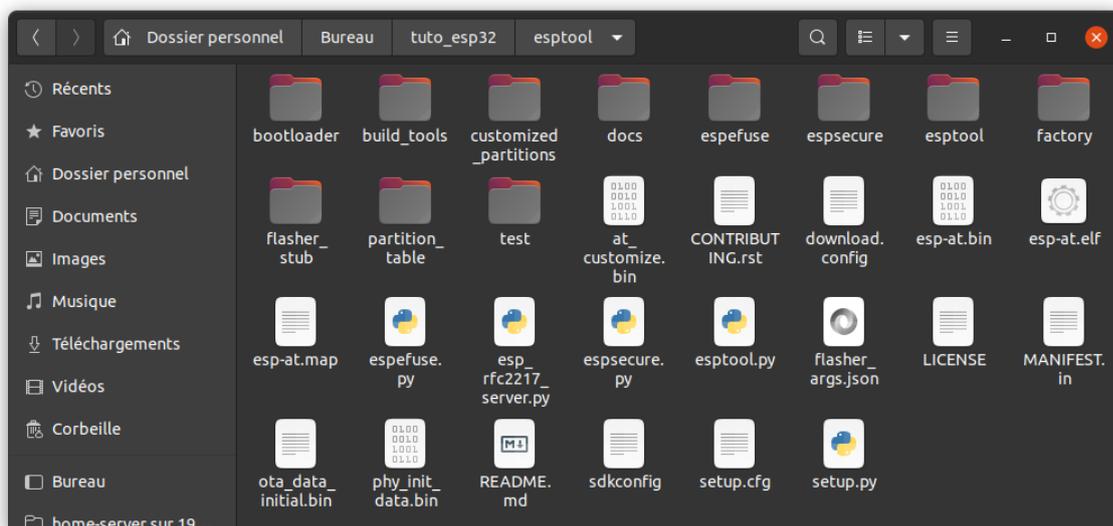
Ouvrez le zip, et naviguez dans le dossier ESP32-C3-MINI-1-V2.3.0.0 voici les fichier qu'il contient :



Décompresser le contenu de ce dossier dans le dossier esptool.
Voici le dossier avant la manipulation :



et après :



Avant de continuer éteignez et rallumer la carte.

Ensuite lancez la commande :

```
> esptool.py --chip auto --port /dev/ttyUSB0 --baud 115200 --before default_reset --after hard_reset write_flash -z download.config
```

En remplacez download.config avec le contenu du fichier, dans mon cas :

```
> esptool.py --chip auto --port /dev/ttyUSB0 --baud 115200 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_size 4MB 0x8000 partition_table/partition-table.bin 0xd000 ota_data_initial.bin 0xf000 phy_init_data.bin 0x0 bootloader/bootloader.bin 0x60000 esp-at.bin 0x1e000 at_customize.bin 0x1f000 customized_partitions/ble_data.bin 0x3a000 customized_partitions/mqtt_key.bin 0x27000 customized_partitions/server_key.bin 0x3c000 customized_partitions/mqtt_ca.bin 0x2d000 customized_partitions/client_key.bin 0x2b000 customized_partitions/client_cert.bin 0x31000 customized_partitions/factory_param.bin 0x2f000 customized_partitions/client_ca.bin 0x38000 customized_partitions/mqtt_cert.bin 0x29000 customized_partitions/server_ca.bin 0x25000 customized_partitions/server_cert.bin
```

Normalement le programme devrait s'écrire dans la flash de l'ESP32.

Éteignez votre ESP32, puis connecter la pin IO8 et IO9 à 1 afin qu'il démarre en mode SPI (enlever le jumper).

Ensuite pour vérifier que tout s'est bien passé ouvrez un picocom sur votre ESP32, puis rallumez la carte, voici ce que vous devriez obtenir :

```
1/1 + ...y@tanguy-Ubuntu: ~/Bureau/tuto_esp32/esptool
1: tanguy@tanguy-Ubuntu: ~/Bureau/tuto_esp32/esptool
ESP-ROM:esp32c3-api1-20210207
Build:Feb 7 2021
rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:2
load:0x3fcd6100,len:0x1948
load:0x403ce000,len:0x8dc
load:0x403d0000,len:0x2de0
entry 0x403ce000
I (31) boot: ESP IDF v4.1.1-354-g00d34e5 2nd stage bootloader
I (32) boot: Compile Time: 11:21:16
I (33) boot: SPI read flash: 3
I (34) boot: esp32c3: SPI Speed : 40MHz
I (35) boot: esp32c3: SPI Mode : DIO
I (36) boot: esp32c3: SPI Flash Size : 4MB
I (38) boot: Enabling 8MB early entropy source...
I (39) boot: Partition Table:
I (39) boot:


| Part Label      | Usage     | Type | ST | Offset   | Length   |
|-----------------|-----------|------|----|----------|----------|
| 0 ota_data      | OTA data  | 01   | 00 | 00000000 | 00002000 |
| 1 phy_init      | PHY data  | 01   | 01 | 00001000 | 00001000 |
| 2 factory_param | WIFI data | 01   | 02 | 00010000 | 00001000 |
| 3 at_customize  | unknown   | 00   | 00 | 00010000 | 00001000 |
| 4 ota_0         | OTA app   | 00   | 10 | 00000000 | 00100000 |
| 5 ota_1         | OTA app   | 00   | 11 | 00100000 | 00100000 |


I (40) boot: End of partition table
I (41) boot: No factory image, trying OTA 0
I (42) esp_image: segment 0: paddr=00000010 vaddr=3c140010 size=39000h (148032) map
I (43) esp_image: segment 1: paddr=00000000 vaddr=3f100000 size=00d70h ( 1268) load
I (44) esp_image: segment 2: paddr=0000c000 vaddr=40300000 size=01100h ( 1264) load
I (45) esp_image: segment 3: paddr=00000010 vaddr=42000010 size=120700h (1247076) map
I (46) esp_image: segment 4: paddr=001c070c vaddr=0010110c size=0d100h ( 5408) load
I (47) esp_image: segment 5: paddr=001c0d1c vaddr=00000000 size=00000h ( 0) load
I (48) esp_image: segment 6: paddr=001c0d28 vaddr=00000000 size=000100h ( 10) load
I (49) boot: Loaded app from partition at offset 0x00000
I (49) boot: Set actual ota_seq1 to ota_data[0]
I (49) boot: Enabling 8MB early entropy source...
module_name:MINI-1
max tx power=78,ret=0
2.3.0
```

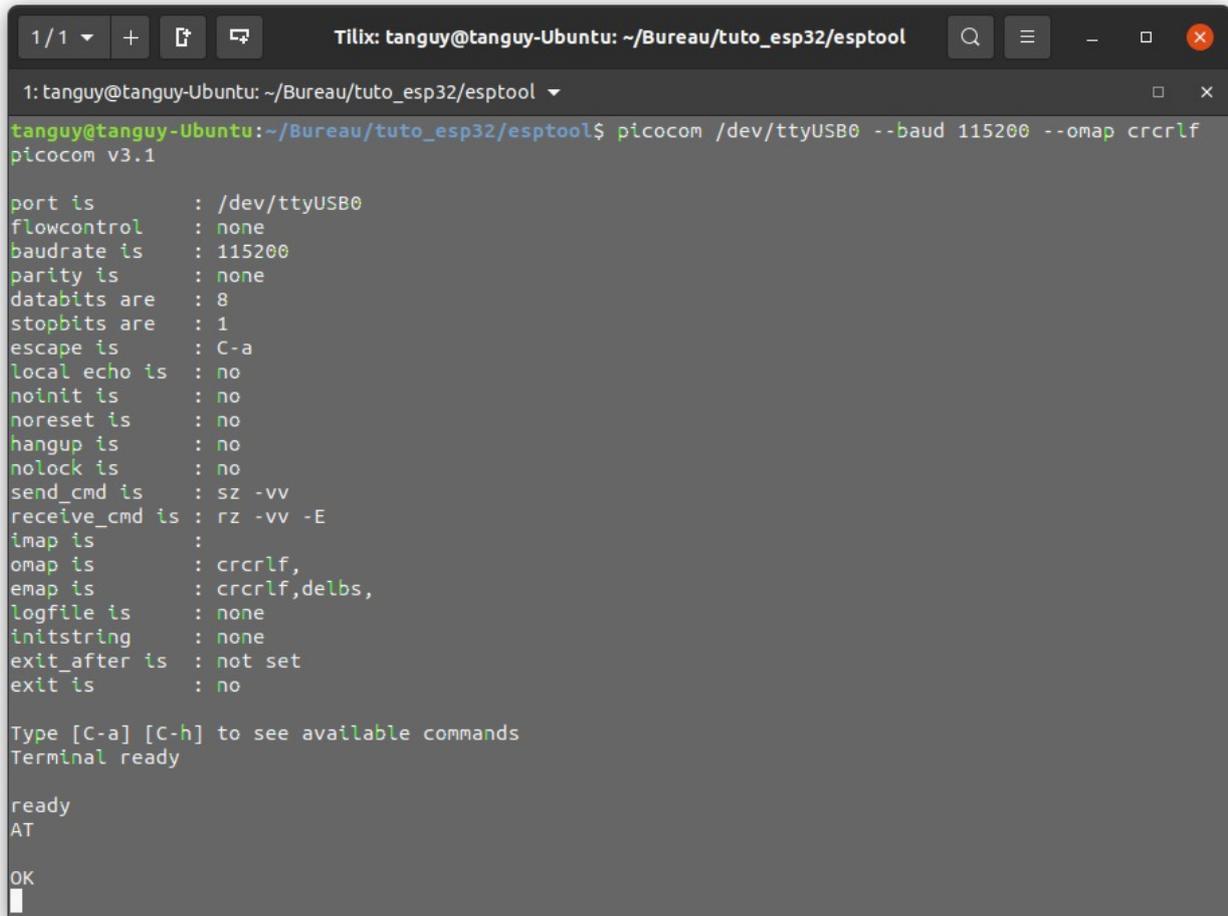
Si tout c'est bien passé, l'ESP32 démarre bien sur le firmware flashé, il ne reste plus qu'à changer d'UART pour communiquer avec les commande AT.

Connecter votre convertisseur USB-UART au pin IO6 et IO7

Ouvrez un picocom avec cette commande :

```
> picocom /dev/ttyUSB0 --baud 115200 --omap crclrf
```

Éteignez et rallumer la carte, un ready devrait apparaître, entrez AT, puis faite ENTER, la carte devrait répondre OK



```
Tilix: tanguy@tanguy-Ubuntu: ~/Bureau/tuto_esp32/esptool
1: tanguy@tanguy-Ubuntu: ~/Bureau/tuto_esp32/esptool
tanguy@tanguy-Ubuntu:~/Bureau/tuto_esp32/esptool$ picocom /dev/ttyUSB0 --baud 115200 --omap crclrf
picocom v3.1

port is       : /dev/ttyUSB0
flowcontrol   : none
baudrate is   : 115200
parity is     : none
databits are  : 8
stopbits are  : 1
escape is     : C-a
local echo is : no
noinit is     : no
noreset is    : no
hangup is     : no
nolock is     : no
send_cmd is   : sz -vv
receive_cmd is : rz -vv -E
imap is       :
omap is       : crclrf,
emap is       : crclrf,delbs,
logfile is    : none
initstring    : none
exit_after is : not set
exit is       : no

Type [C-a] [C-h] to see available commands
Terminal ready

ready
AT
OK
█
```

La carte est prête à être programmée avec les commande AT, bonne chance.