

# X|B

Manual de usuario

# X-BOARD

# Raspberry Pi



# XIDE®

**XIDE®** es un kit de hardware integrado por módulos **X-NODE** y tarjetas de expansión **X-BOARD**, diseñado para incorporar en menos de 24 hrs, prototipos de hardware para proyectos de Internet de las Cosas **IoT**.



**XIDE®** es un proyecto realizado por **Microside Technology**, empresa orgullosamente mexicana, especializada en diseño y producción de soluciones tecnológicas para IoT.

# X-BOARD

## Características



Compatible con  
estándar de conexión  
mikroBUS™

QW | ST

Compatible con  
estándar Qwiic® y  
STEMMA QT®



Compatible con niveles  
lógicos de voltaje  
3.3V < > 5V

# X-BOARD

# Raspberry Pi

---

## I. Introducción

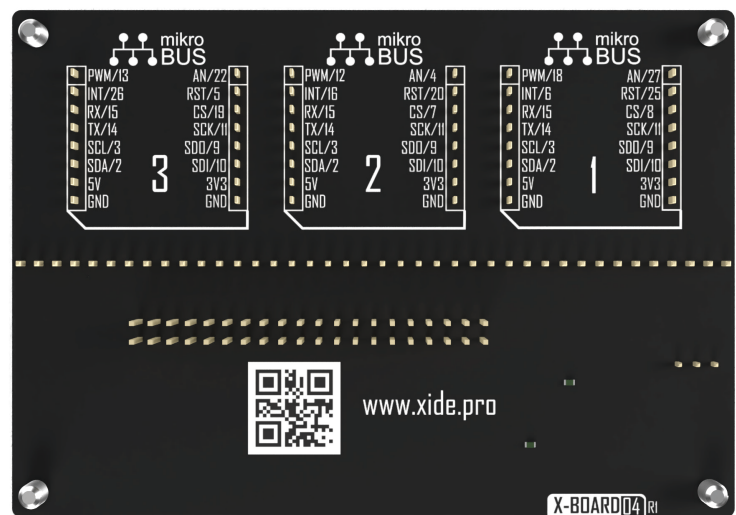
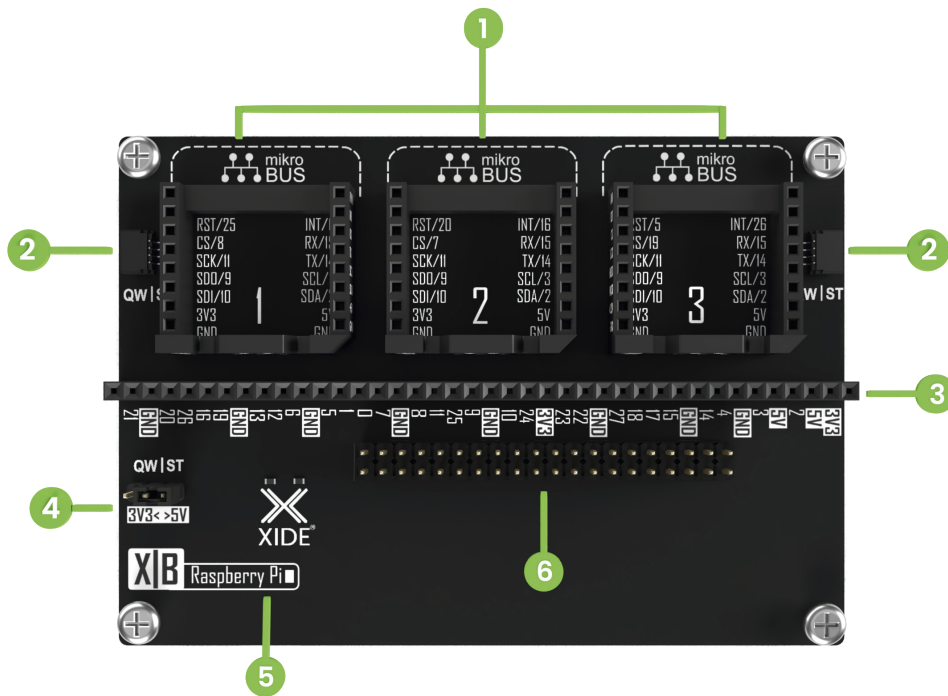
La **X-BOARD Raspberry Pi** es una tarjeta de expansión diseñada principalmente para evaluar de forma rápida sensores, actuadores o módulos de comunicación integrados en los **X-NODE** en conjunto con las tarjetas Raspberry Pi (modelos 4B, 3B+, Zero y Zero W). Es ideal para el desarrollo de prototipos y proyectos electrónicos donde se requiera una integración sencilla, tamaño compacto y compatibilidad con herramientas de desarrollo con los estándares **mikroBUS™**, **Qwiic®** y **STEMMA QT®**. Cuenta con un selector de voltaje entre 3.3 V y 5 V para los conectores JST, tres zócalos de conexión con estándar **mikroBUS™** y un header hembra con el acceso a todos los pines de la tarjeta Raspberry Pi.

---

## II. ¿Cómo funciona?

Para una comunicación con herramientas de desarrollo que poseen estándares **mikroBUS™**, **Qwiic®** o **STEMMA QT®**, simplemente bastará con insertar la tarjeta Raspberry Pi en el puerto dedicado y los módulos **X-NODE** o tarjetas compatibles en los zócalos o conectores de la **X-BOARD Raspberry Pi**. El usuario podrá seleccionar el voltaje de operación en los conectores JST y usar el header hembra para disponer de los pines de la tarjeta Raspberry Pi colocada, de esta forma puede utilizar un analizador lógico en diversas pruebas de funcionamiento.

La tarjeta **X-BOARD Raspberry Pi** es compatible con el estándar **mikroBUS™** de Mikroe® para una conexión fácil e integra conectores JST que admiten el estándar **Qwiic®** de SparkFun® y el estándar **STEMMA QT®** de Adafruit® para una comunicación entre diversos módulos y tarjetas de desarrollo por medio del protocolo I<sup>2</sup>C de manera rápida y sencilla.



### III. Descripción del hardware

1. Conectores estándar **mikroBUS™**
2. Conectores JST compatibles con **Qwiic®** y **STEMMA QT®**
3. Header para disposición de los pines de cada conector **mikroBUS™**
4. Selector de voltaje en el conector JST 3.3V <> 5V
5. Modelo de X-BOARD
6. Conector para tarjeta **Raspberry Pi**

---

## IV. Especificaciones técnicas

<b>Modelo</b>	Raspberry Pi
<b>Módulos compatibles</b>	Raspberry Pi Modelo 4B, 3B+, Zero y Zero W.
<b>Estándares compatibles</b>	Estándar mikroBUS™, estándar Qwiic® y estándar STEMMA QT®
<b>Características</b>	3 Zócalos de conexión con estándar mikroBUS™, 2 conectores JST compatibles con el estándar Qwiic® o estándar STEMMA QT® y un header de 40 pines para disposición de los pines de las tarjetas Raspberry Pi.
<b>Tamaño</b>	72 x 102 x 14 mm
<b>Voltaje</b>	3.3V o 5V

---

## V. Ejemplo de uso

A continuación encontrarás ejemplos prácticos para el uso de la **X-BOARD Raspberry Pi** utilizando Python y la terminal Putty en el sistema operativo Raspbian.

### Habilitar puerto serial

Antes de iniciar es necesario habilitar el puerto serial en nuestra **Raspberry Pi**, para ello ejecutamos el siguiente comando:

```
ls -l /dev
```

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ ls -l /dev  
total 0  
crw-r--r-- 1 root root 10, 235 Nov 13 20:17 autofs  
drwxr-xr-x 2 root root 580 Nov 13 20:17 block  
crw----- 1 root root 10, 234 Nov 13 20:17 btrfs-control  
drwxr-xr-x 3 root root 60 Jan 1 1970 bus  
crw----- 1 root root 10, 62 Nov 13 20:17 cachefiles  
drwxr-xr-x 2 root root 3260 Nov 13 20:17 char  
crw----- 1 root root 5, 1 Nov 13 20:17 console  
crw----- 1 root root 10, 203 Nov 13 20:17 cuse  
drwxr-xr-x 7 root root 140 Nov 13 20:17 disk  
drwxr-xr-x 2 root root 80 Jan 1 1970 dma_heap  
drwxr-xr-x 3 root root 120 Nov 13 20:17 dri  
crw-rw---- 1 root video 29, 0 Nov 13 20:17 fb0  
lrwxrwxrwx 1 root root 13 Feb 14 2019 fd -> /proc/self/fd  
crw-rw-rw- 1 root root 1, 7 Nov 13 20:17 full  
crw-rw-rw- 1 root root 10, 229 Nov 13 20:17 fuse  
crw-rw---- 1 root gpio 254, 0 Nov 13 20:17 gpiochip0  
crw-rw---- 1 root gpio 254, 1 Nov 13 20:17 gpiochip1  
crw-rw---- 1 root gpio 246, 0 Nov 13 20:17 gpiomem  
crw----- 1 root root 243, 0 Nov 13 20:17 hidraw0  
crw----- 1 root root 243, 1 Nov 13 20:17 hidraw1  
crw----- 1 root root 243, 2 Nov 13 20:17 hidraw2  
crw----- 1 root root 243, 3 Nov 13 20:17 hidraw3
```

Nos mostrará una lista de puertos, buscaremos el puerto (ttyS0), en el caso que no se encuentre, seguir lo siguientes pasos para habilitarlo

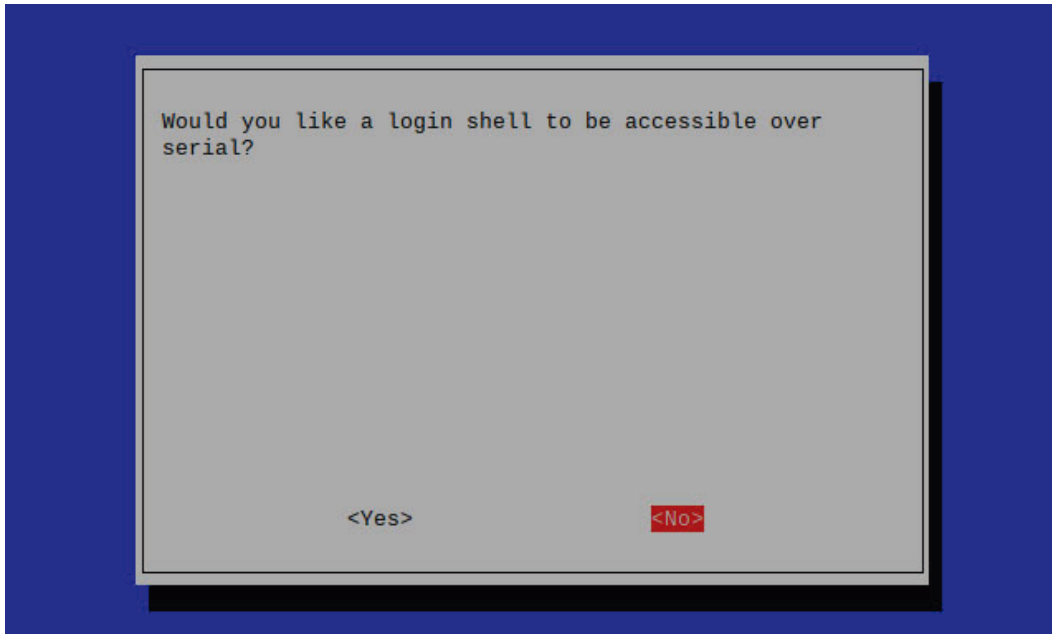
```
pi@raspberrypi: ~  
File Edit Tabs Help  
brw-rw---- 1 root disk 1, 8 Nov 15 16:19 ram8  
brw-rw---- 1 root disk 1, 9 Nov 15 16:19 ram9  
crw-rw-rw- 1 root root 1, 8 Nov 15 16:19 random  
drwxr-xr-x 2 root root 60 Jan 1 1970 raw  
crw-rw-r-- 1 root netdev 10, 242 Nov 15 16:19 rfkill  
crw-rw---- 1 root video 238, 0 Nov 15 16:19 rpidvid-h264mem  
crw-rw---- 1 root video 240, 0 Nov 15 16:19 rpidvid-hevcmem  
crw-rw---- 1 root video 239, 0 Nov 15 16:19 rpidvid-intcmem  
crw-rw---- 1 root video 237, 0 Nov 15 16:19 rpidvid-vp9mem  
lrwxrwxrwx 1 root root 7 Nov 15 16:19 serial1 -> ttyAMA0  
drwxrwxrwt 2 root root 40 Feb 14 2019 shm  
drwxr-xr-x 3 root root 180 Nov 15 16:19 snd  
lrwxrwxrwx 1 root root 15 Feb 14 2019 stderr -> /proc/self/fd/2  
lrwxrwxrwx 1 root root 15 Feb 14 2019 stdin -> /proc/self/fd/0  
lrwxrwxrwx 1 root root 15 Feb 14 2019 stdout -> /proc/self/fd/1  
crw-rw-rw- 1 root tty 5, 0 Nov 15 16:19 tty  
crw--w---- 1 root tty 4, 0 Nov 15 16:19 tty0  
crw----- 1 pi tty 4, 1 Nov 15 16:19 tty1  
crw--w---- 1 root tty 4, 10 Nov 15 16:19 tty10  
crw--w---- 1 root tty 4, 11 Nov 15 16:19 tty11  
crw--w---- 1 root tty 4, 12 Nov 15 16:19 tty12  
crw--w---- 1 root tty 4, 13 Nov 15 16:19 tty13  
crw--w---- 1 root tty 4, 14 Nov 15 16:19 tty14  
crw--w---- 1 root tty 4, 15 Nov 15 16:19 tty15
```

Para habilitar el puerto serial (ttyS0) vamos a ejecutar el siguiente comando sudo raspi-config e inmediatamente nos abrirá una ventana.

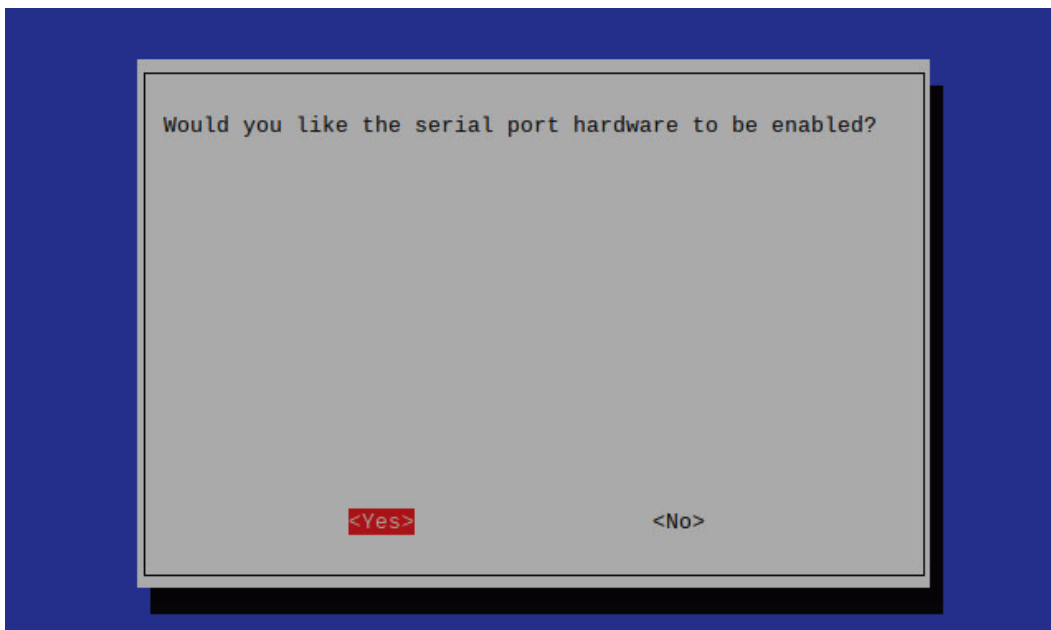




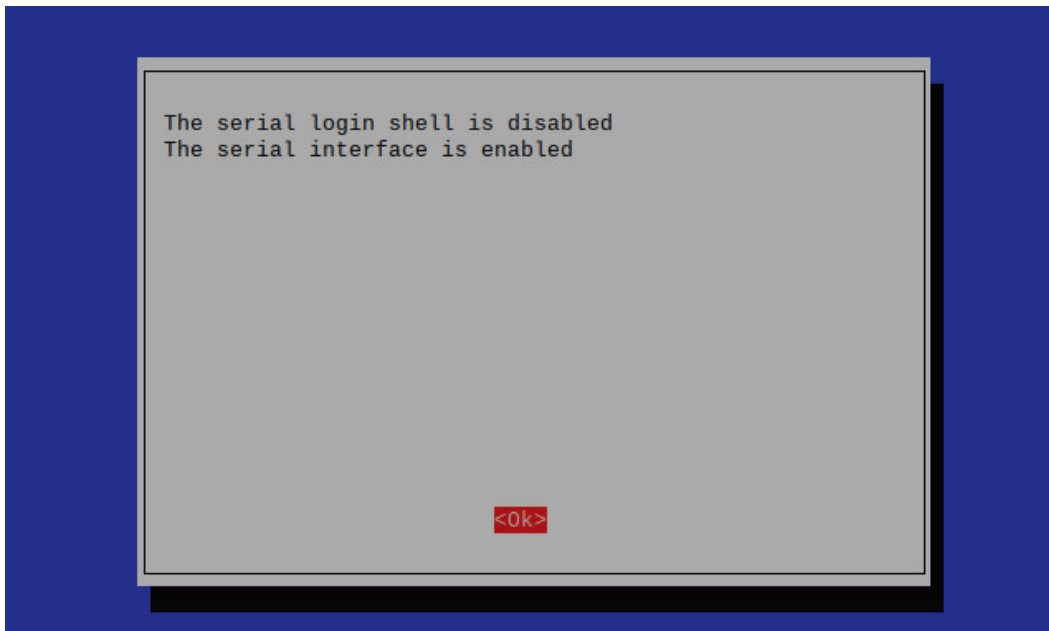
Después mostrará otra ventana donde daremos "ENTER" en "No".



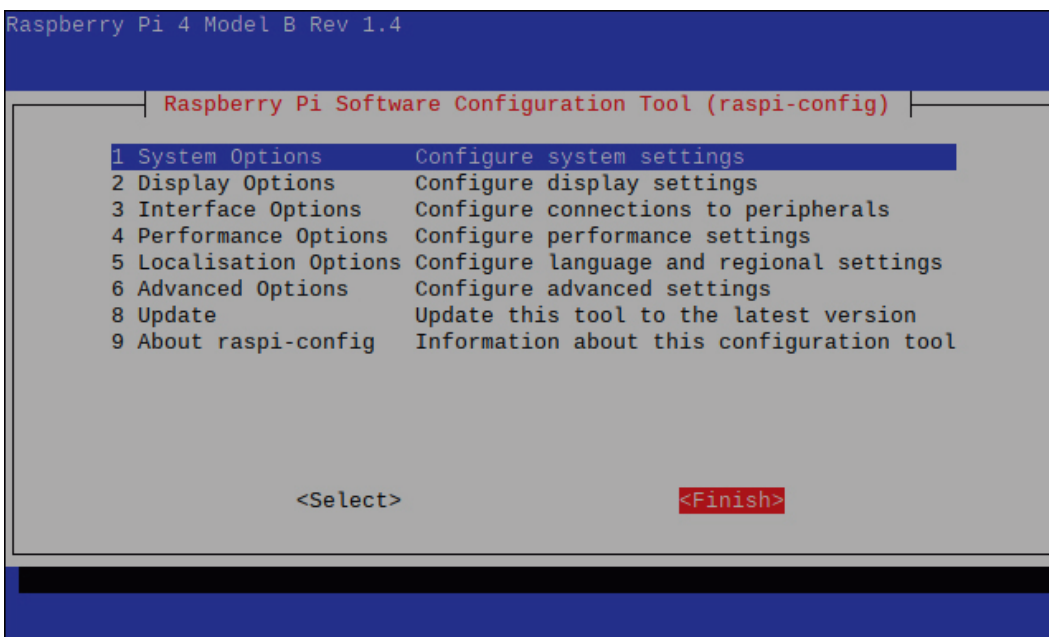
Ahora daremos "ENTER" en Yes para habilitar el puerto serial.



Nos saldrá la siguiente ventana, en la cual mostrará que el puerto serial se ha habilitado, daremos "ENTER" para continuar.



Esto nos regresará a la ventana principal y simplemente nos iremos a la opción <Finish> y daremos "ENTER" para finalizar el proceso.



Finalmente daremos "ENTER" en Yes y para ello nos solicitará reiniciar el sistema.



Una vez que encienda nuevamente la Raspberry Pi ejecutamos el comando:

```
ls -l /dev
```

para verificar que el puerto serial (ttyS0) este habilitado.

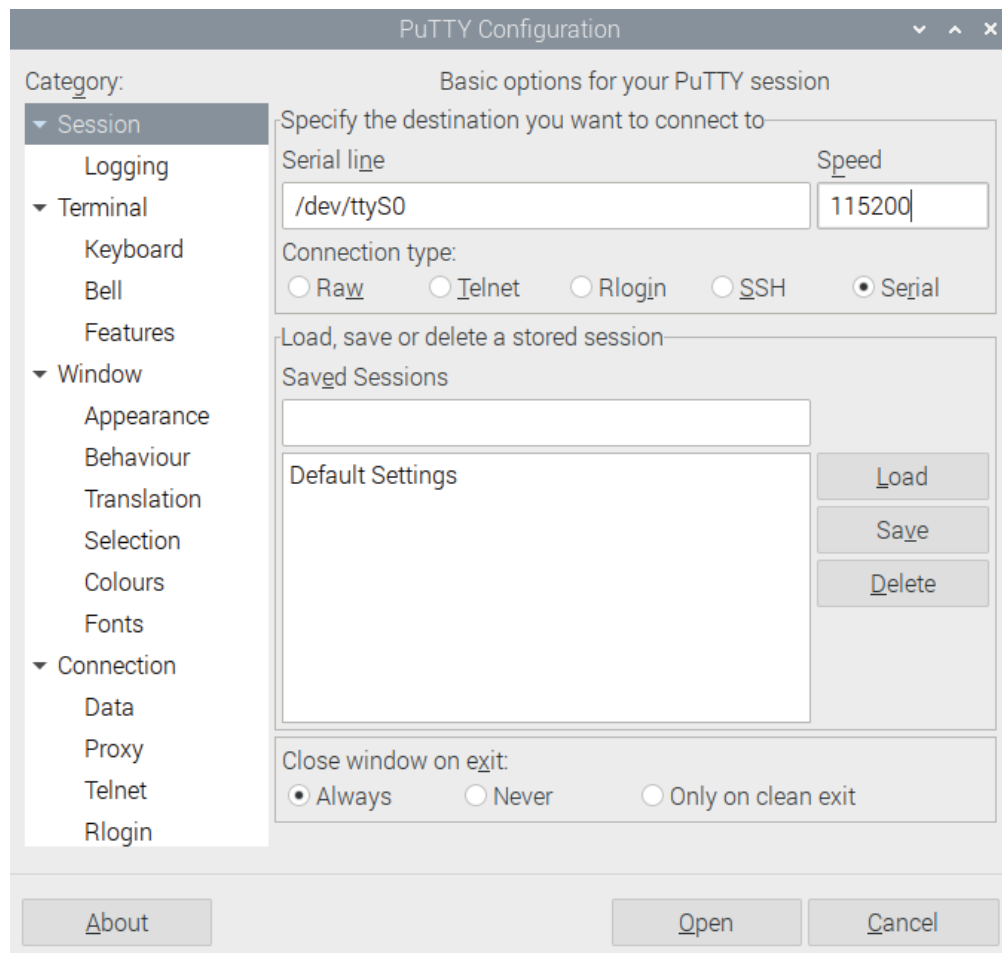
```
pi@raspberrypi: ~  
File Edit Tabs Help  
brw-rw---- 1 root disk 1, 7 Nov 13 20:17 ram7  
brw-rw---- 1 root disk 1, 8 Nov 13 20:17 ram8  
brw-rw---- 1 root disk 1, 9 Nov 13 20:17 ram9  
crw-rw-rw- 1 root root 1, 8 Nov 13 20:17 random  
drwxr-xr-x 2 root root 60 Jan 1 1970 raw  
crw-rw-r-- 1 root netdev 10, 242 Nov 13 20:17 rfkill  
crw-rw---- 1 root video 238, 0 Nov 13 20:17 rpidvid-h264mem  
crw-rw---- 1 root video 240, 0 Nov 13 20:17 rpidvid-hevcmem  
crw-rw---- 1 root video 239, 0 Nov 13 20:17 rpidvid-intcmem  
lrwxrwxrwx 1 root root 5 Nov 13 20:17 serial0 -> ttyS0  
drwxrwxrwt 2 root root 40 Nov 15 15:45 shm  
drwxr-xr-x 3 root root 180 Nov 13 20:17 snd  
lrwxrwxrwx 1 root root 15 Feb 14 2019 stderr -> /proc/self/fd/2  
lrwxrwxrwx 1 root root 15 Feb 14 2019 stdin -> /proc/self/fd/0  
lrwxrwxrwx 1 root root 15 Feb 14 2019 stdout -> /proc/self/fd/1  
crw-rw-rw- 1 root tty 5, 0 Nov 13 20:17 tty  
crw--w---- 1 root tty 4, 0 Nov 13 20:17 tty0  
crw----- 1 pi tty 4, 1 Nov 13 20:17 tty1  
crw--w---- 1 root tty 4, 10 Nov 13 20:17 tty10  
crw--w---- 1 root tty 4, 11 Nov 13 20:17 tty11  
crw--w---- 1 root tty 4, 12 Nov 13 20:17 tty12  
crw--w---- 1 root tty 4, 13 Nov 13 20:17 tty13
```

## Ejemplo de uso con Terminal PuTTY

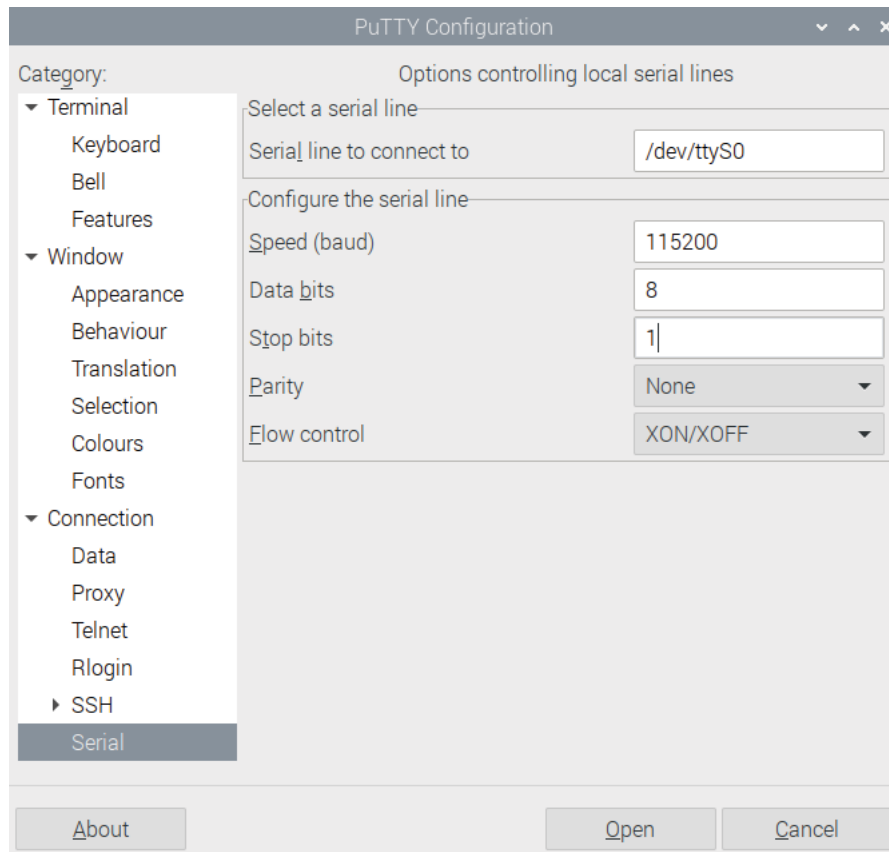
Realizamos la instalación de PuTTY a través del siguiente comando:

```
sudo apt install putty
```

Una vez instalado lo abrimos y aparecerá la siguiente ventana donde seleccionaremos "Serial" en tipo de conexión y cambiaremos el "Speed" por 115200.

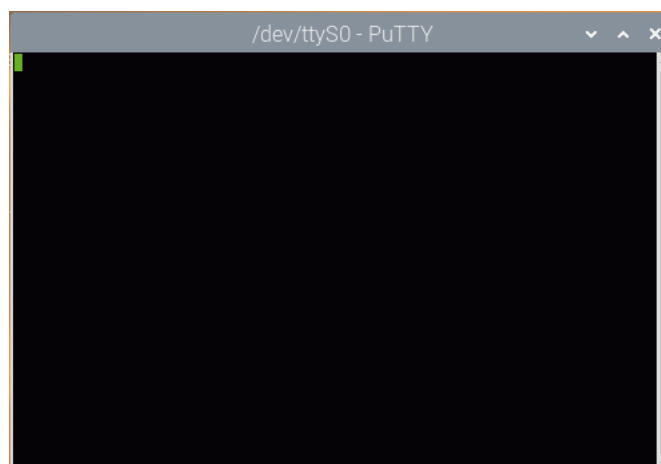


Para revisar las configuraciones que actualmente se tienen en esta opción, nos desplazamos sobre la columna de la parte izquierda y damos clic sobre “Serial”, verificamos que está configurado de la siguiente manera y después damos clic en Open.



Se nos abrirá la siguiente ventana, en la cual ya podremos ingresar comandos.

**Nota:** Por motivos de seguridad, los comandos no se podrán visualizar, únicamente se podrán ver las respuestas que nos responda el hardware.



En este ejemplo, vamos a verificar si ya existe comunicación entre un **X-NODE 4 x 24V Input (XN046)** y la **Raspberry Pi**, para ello vamos a enviar el comando:

```
XN046A?
```

Como respuesta debemos obtener un OK, confirmando dicha comunicación entre ambos dispositivos .



```
/dev/ttyS0 - PuTTY
OK
```

Ahora ingresamos el comando:

```
XN046A+G
```

Esto con el objetivo de mostrar el estado actual de las entradas en el conector, por lo cual mostrará 1 cuando haya señal en la entrada y 0 cuando haya ausencia.



```
/dev/ttyS0 - PuTTY
XN046A=1000
```

La respuesta que nos regresa es 1000, indicándonos que solo la entrada 1 tiene presencia de señal.

---

## Ejemplo de uso con Python

También es posible enviar comandos a través de código Python, para ello debemos instalarlo con los siguientes comandos:

```
sudo apt install python3
```

```
sudo apt install idle3
```

Para utilizar los comandos en Python, es necesario instalar la librería Serial, para ello ejecutaremos el siguiente comando:

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

Una vez instalado, procedemos a abrir Python desde la sección Programming e ingresar el siguiente código:

```
import serial
import time
import subprocess
import os, sys

puerto = serial.Serial(port = '/dev/ttyS0', baudrate = 115200, bytesize = serial.EIGHTBITS,
parity = serial.PARITY_NONE, stopbits = serial.STOPBITS_ONE)

puerto.write('XN007A?' + '\n\r')

port=puerto.read()

print port
```

En este caso enviaremos el comando XN046A? y como resultado nos dará un OK de confirmación.

```
Serial.py x
1 import serial
2
3 puerto = serial.Serial('/dev/ttyS0',
4                         baudrate = 115200,
5                         bytesize = serial.EIGHTBITS,
6                         parity = serial.PARITY_NONE,
7                         stopbits = serial.STOPBITS_ONE)
8
9
10 puerto.write(b'XN009A?\n\r')
11 respuesta=puerto.readline()
12 print(respuesta)
13
14
```

```
Shell
>>> %Run Serial.py
b'OK\r\n'
>>>
```

Ahora en nuestro código ingresamos el comando:

XN046A+G

Esto con el objetivo de mostrar el estado actual de las entradas en el conector, por lo cual mostrará 1 cuando haya señal en la entrada y 0 cuando haya ausencia.

```
Serial.py x
1 import serial
2
3 puerto = serial.Serial('/dev/ttyS0',
4                         baudrate = 115200,
5                         bytesize = serial.EIGHTBITS,
6                         parity = serial.PARITY_NONE,
7                         stopbits = serial.STOPBITS_ONE)
8
9
10 puerto.write(b'XN046A+G\n\r')
11 respuesta=puerto.readline()
12 print(respuesta)
13
14
```

```
Shell
>>> %Run Serial.py
b'XN046A=1000\r\n'
>>>
```

Python 3.7.3

La respuesta que nos regresa es 1000, indicándonos que solo la entrada 1 tiene presencia de señal.



# X|B



X|IDE<sup>®</sup>

[www.xide.pro](http://www.xide.pro)