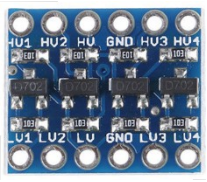
AR488 Bluetooth support

Bluetooth is a common short range wireless connection technology for connecting devices and sending data. Bluetooth devices can typically communicate over distances not exceeding 10 metres, although some devices can communicate over greater distances. In order to communicate with each other, Bluetooth devices must be paired. Once paired, the device facilities can be discovered and a connection established.

Bluetooth transceiver modules such as the HC05 and HC06 can be connected to an Arduino quite easily and manually configured to provide a wireless connection in place of the USB cable. These boards come as either the transceiver board itself with soldered edge connectors only and requiring a DC 3.3V supply, or, mounted on an "adapter" or "breakout" board (as shown opposite) with external pin connectors at one end and which can be supplied directly with 5V.

The AR488 supports auto-configuration of the HC05 Bluetooth breakout board for Arduino. This board can be just connected to the serial port of the Arduino and the configuration set in AR488_Config.h file. The HC05 board will be then configured automatically on start-up. Ideally the board would be connected to the RX and TX pins of a second serial port, e.g. Serial1 on the 32u4 or Serial1/Serial3 on the Mega 2560. The HC06 module does not support master mode, so auto-configuration is not possible with the HC06. However, either the HC05 and HC06 breakout boards or modules can be configured manually if required and used with the AR488.

The HC05 and HC06 modules have a very similar appearance, but the HC05 can be identified by the fact that it has six pins, while the HC06 has only four. There is no EN or STATUS pin on the HC06.



Unlike most Arduino boards, which are supplied with 5V DC as is common to all USB devices, the HC05 module requires a 3.3V supply. The "adapter" or "breakout" board provides a 3.3V regulator for this purpose so it can be supplied with 5V, however, the serial input and output as well as the enable pin still operate at a 3.3V signalling level as indicated by the 'LEVEL:3.3V' marked on the board. Applying 5V to these pins can damage the module and while some tutorials do show these connected directly to the RX/TX pins on the Arduino, this is not recommended and ideally, a level shifter should be used. This can be made quite easily but ready made boards such as the one shown opposite can also be purchased. Such boards usually have 4 "channels" (HV1-HV4 and corresponding LV1-LV4) which is sufficient for our purposes. There are 4 transistors across the centre and 6 soldered connectors along each edge. They board is usually supplied with a row of pins which can be soldered to the connectors if required or wires can be soldered to the board directly. The HV side is connected to the Arduino and the LV side to the HC05.

The level shifter board is easily supplied from the 5V and 3.3V pins on the Arduino with the 5V pin connected to HV and 3.3V pin connected to LV. The GND connector must be connected to the Arduino GND pin as well as the HC05 GND pin. Either one or both GND connectors on the level shifter can be used as they are linked together. The Arduino TX, RX and enable pin connections are made to one of the HV side channels and connections to the HC05 Bluetooth board are made to the corresponding LV side channel. It does not matter which actual channel numbers are used for which function so long as TX on the Arduino connects with RX on the HC05, RX on the Arduino to the TX on the HC05 and EN on the HC05 connects to the GPIO pin configured for this purpose on the Arduino, all connections being made via correspondingly numbered HV/LV channel on the level shifter.

The HC05 breakout board requires 5 connections in total including 5V DC power to VCC, GND, TXD, RXD and EN or “Enable” that enables the Arduino master mode required for configuration. In the example shown below, the EN pin is connected via the level shifter to pin 6 on the Uno, but any spare Arduino GPIO pin can be used. The pin marked STATE is left disconnected.

Factory set serial communication speeds also seem to vary and while some modules operate at 38400 baud, others seem to require a 9600 baud connection. The AR488 Bluetooth module will automatically detect the default speed and set it to the configured speed.

The Bluetooth HC05 transceiver breakout board requires three channels. The diagram below is an example of how an Arduino Uno might be wired to the HC05 module:

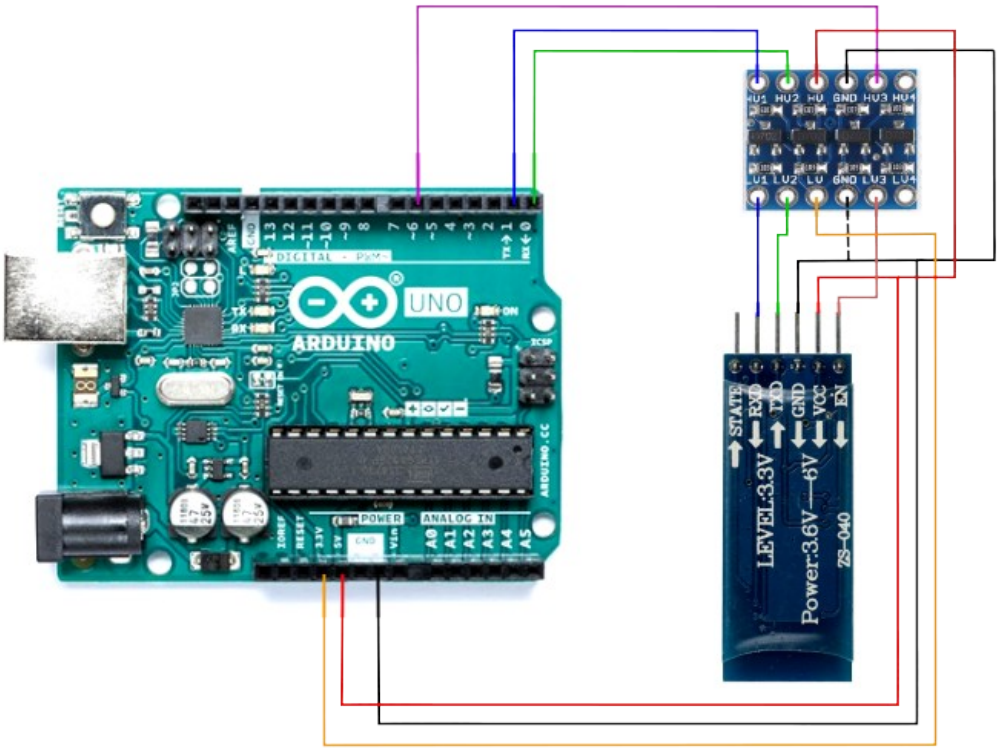


Diagram of the Arduino to HC05 connections

Clones of the Pro Micro board do not have a 3.3V regulator on board, so a 3.3V supply will have to be provided separately.

Enabling Bluetooth on the AR488

The Bluetooth feature is disabled by default, but can be enabled by removing the comment characters '//' preceding the definition for AR_BT_EN within the "Bluetooth (HC05) support" section in the AR488_Config.h file:

```
//#define AR_BT_EN 6            // Bluetooth enable and control pin
#ifdef AR_BT_EN
  #define AR_BT_BAUD 115200     // Bluetooth module preferred baud rate
  #define AR_BT_NAME "AR488-BT" // Bluetooth device name
  #define AR_BT_CODE "488488"   // Bluetooth pairing code
#endif
```

so that it becomes:

```
#define AR_BT_EN 6            // Bluetooth control enable pin
#ifdef AR_BT_EN
  #define AR_BT_BAUD 115200     // Bluetooth module baud rate
  #define AR_BT_NAME "AR488-BT" // Bluetooth device name
  #define AR_BT_CODE "488488"   // Bluetooth pairing code
#endif
```

This will activate the Bluetooth auto-configuration. Set the pin number to the number of the GPIO pin that will be used to control the EN pin on the HC05.

The default baud rate can be set to anything that the Arduino and the HC05 board will support. Please note that there are no double quotes around this parameter.

The default pairing device name is *AR488-BT*, but this can be changed to anything desired by the following line:

```
#define AR_BT_NAME "AR488-BT"
```

where *AR488-BT* is replaced by whatever string is desired.

The default paring code is *488488*, but this can be changed by modifying the number in quotes after the *AR_BT_CODE* keyword:

```
#define AR_BT_CODE "488488"
```

The code must be enclosed in double quotes and must be at least 6 digits long.

The diagram above shows a Uno wired up to a HC05 connector. It should be noted that the Uno and Nano have only one UART and the 328p has only one set of Tx/Rx pins. Since the serial protocol was designed for 1:1 connections, on these Arduino boards, USB and Bluetooth should not be used at the same time. The sketch must be uploaded to the Arduino prior to the The HC05 being connected to the Tx/Rx pins, otherwise the upload will fail. Once the sketch has been uploaded, the module can then be connected and both Arduino and HC05 module powered up. This is not a problem where an Arduino has additional serial ports available such as the 32u4 and the Mega 2560. The HC05 can be connected to a secondary port and the USB port used to program as normal.

Setting up the hardware

Once the sketch has been configured and uploaded to the board and the HC05 Bluetooth module connected up, the AR488 can be powered up. On start-up it will automatically detect the baud rate of the HC05 module and configure it. By default, the AR488 will appear as a Bluetooth device called *AR488-BT*, but otherwise should appear with the configured name. On Windows, it may appear as "Other device" until the screen is refreshed.

The first time that the HC05 board is used, the LED on the HC05 board will initially blink slowly for a few seconds. The internal LED on the Arduino should then blink twice to indicate that the baud rate has been successfully detected, and following that, 3 more times to indicate that configuration has been successful. A few moments after this, the LED on the HC05 should begin flashing rapidly to indicate that it has switched to slave mode and is ready for pairing.

On subsequent power-up, the LED on the HC05 board will initially blink slowly for a few seconds. The Arduino internal LED will blink twice to indicate successful auto-detection of the baud rate, and then once to indicate that the configuration has not changed. The LED on the HC05 will then blink rapidly to indicate that it has switched to slave mode and the module is ready for pairing.

If the any of the Bluetooth configuration parameters are changed in the sketch, the AR488 will perform the auto-configuration process again. If nothing has changed, the auto-configuration process is skipped and the HC05 board just goes into slave mode.

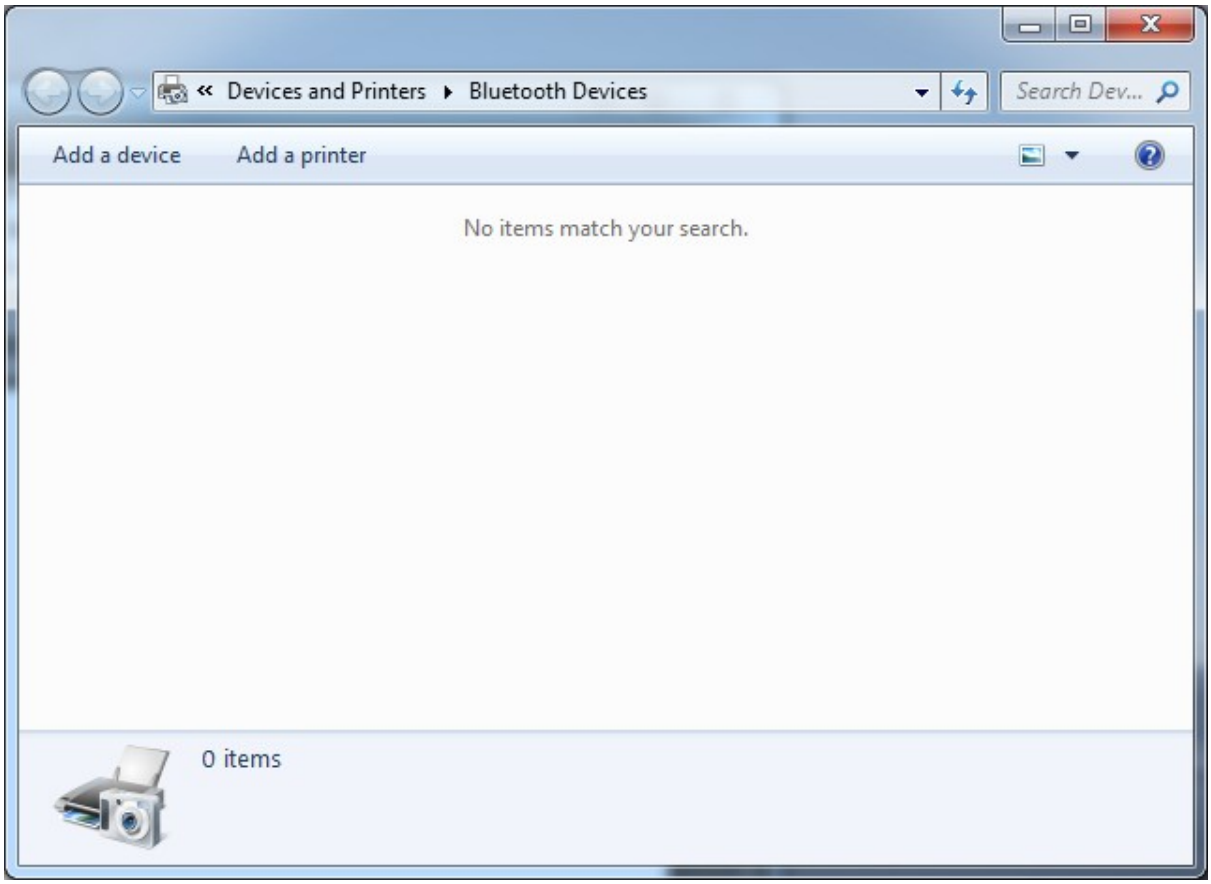
Once paired, the LED on the HC05 board will blink twice every few seconds.

Pairing the AR488 Interface with Bluetooth

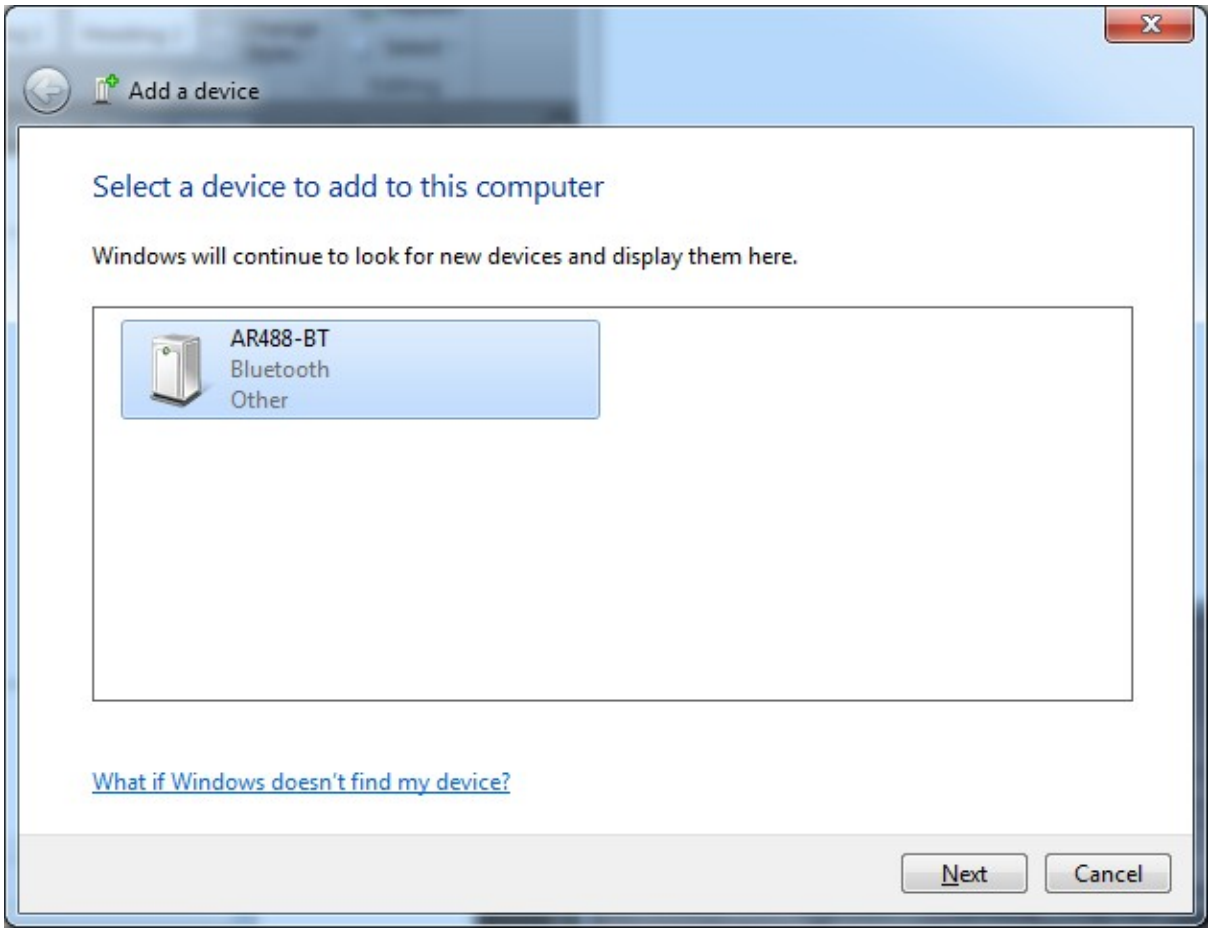
Windows 7

Go to the task bar and from the hidden icons panel select the Bluetooth icon:

Click *Show Bluetooth Devices*

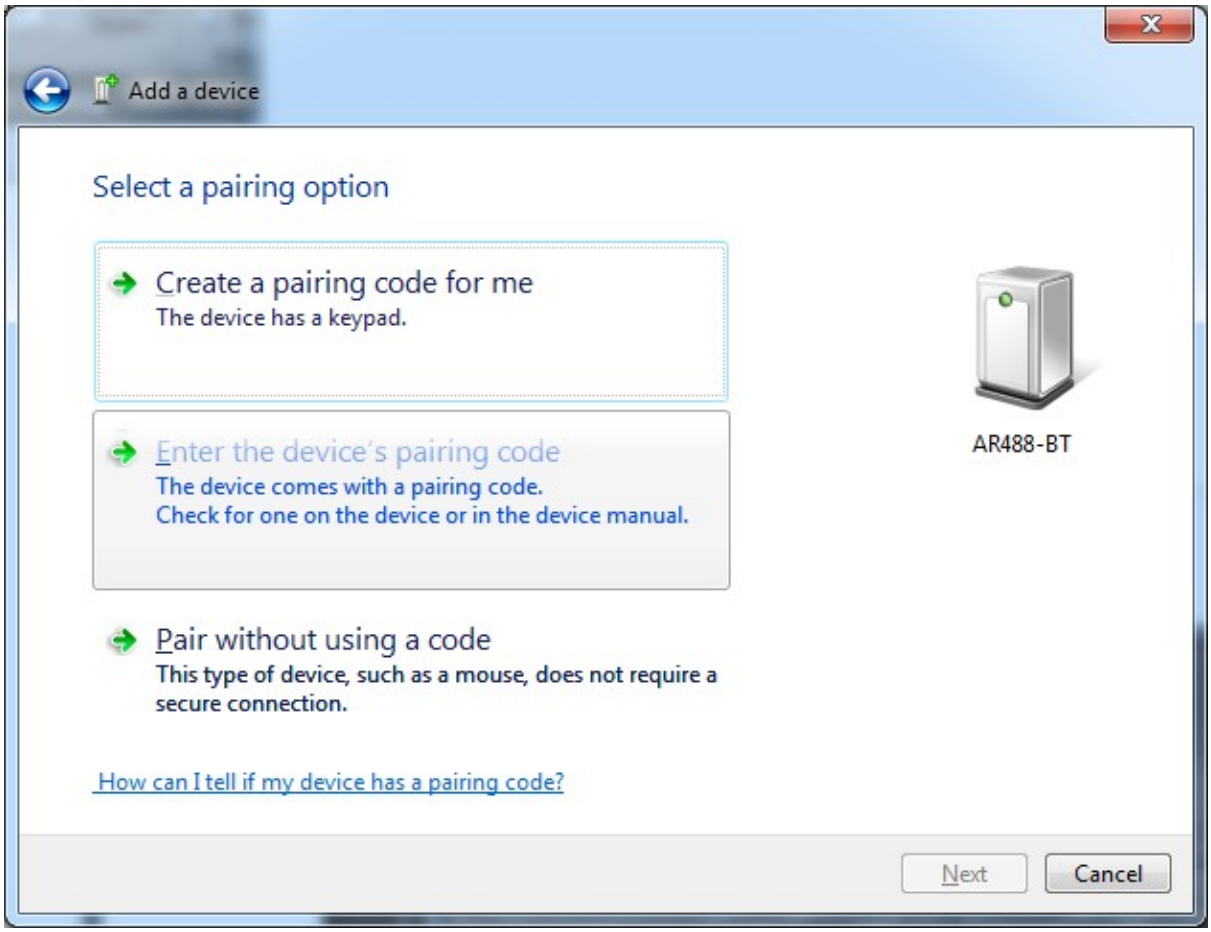


This will open a windows showing your Bluetooth devices. If you have no other devices connected, the window will be empty. Click *Add a Device*

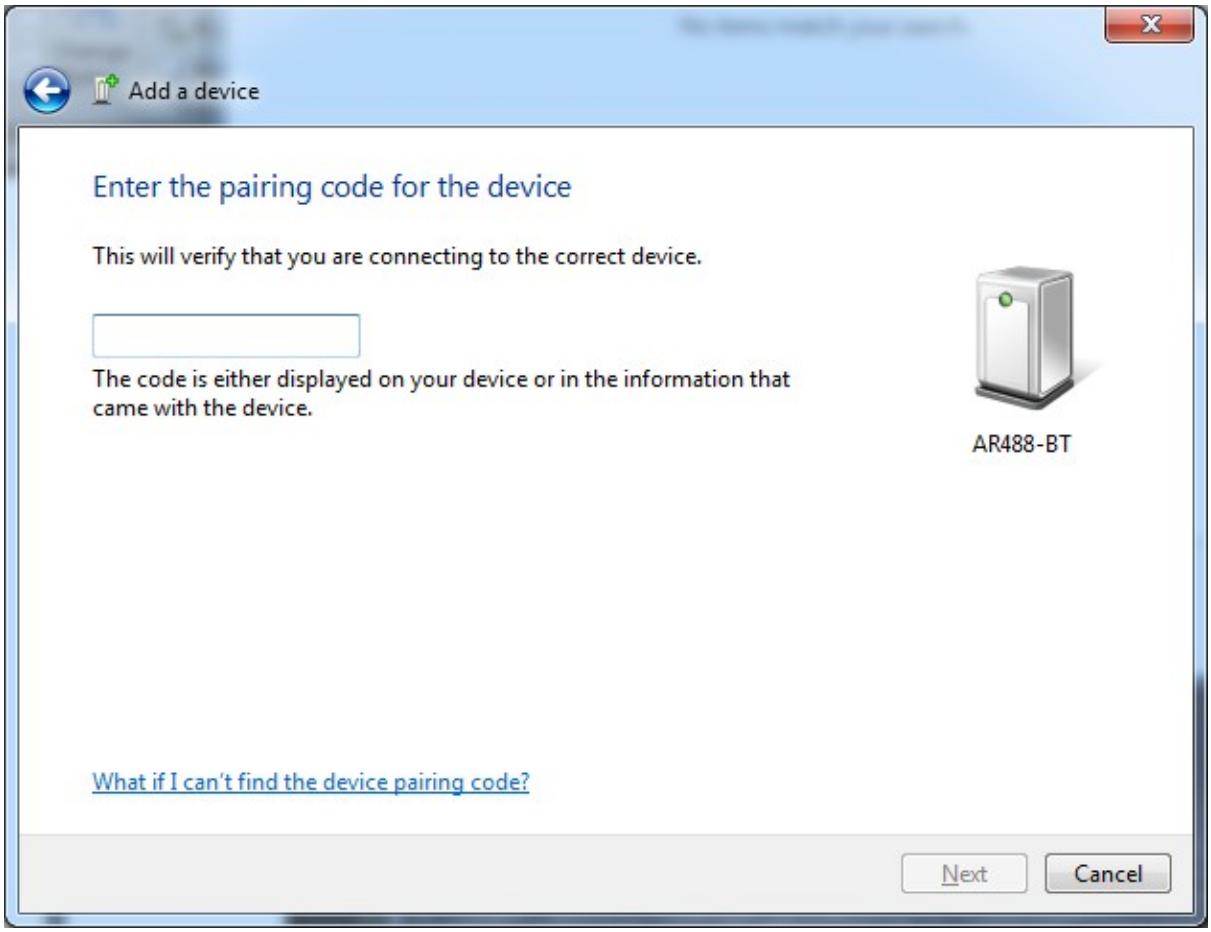


After a few moments, the AR488 -BT device should be detected. Please not that it may initially show simply as *Other*, but when you click on it, the name should be updated to AR488-BT.

Click the *Next* button. This will momentarily show a *Connecting to devices* status and then open the *Add a Device* window:



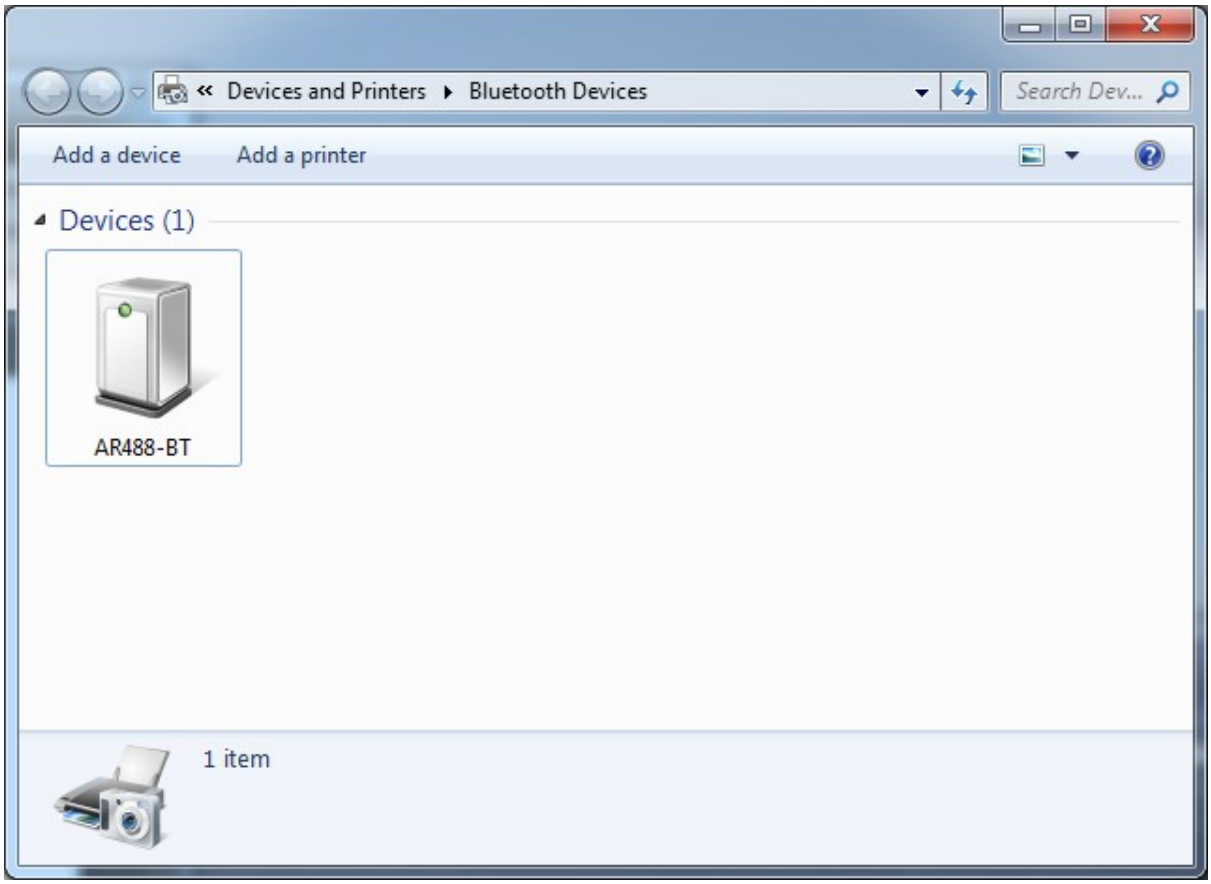
Select *Enter the device's pairing code* panel. The next dialogue will prompt you to enter the device pairing code:



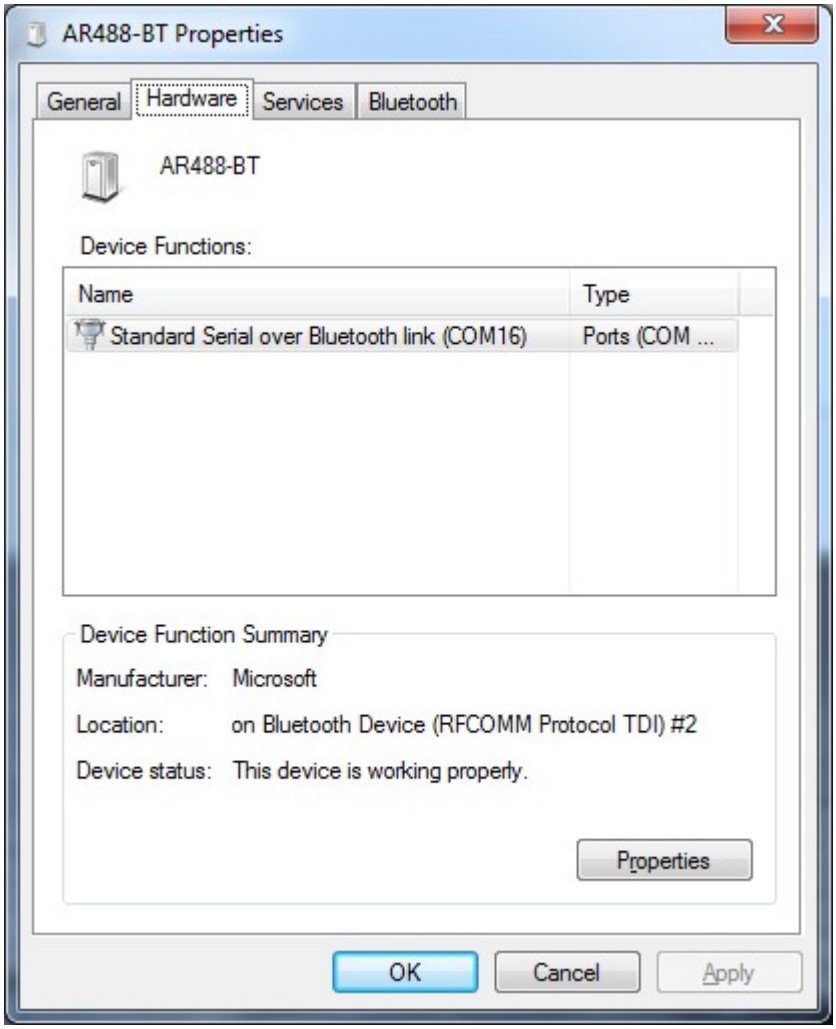
Enter the configured pin code, for example *488488* and then click *Next*.

Windows should now connect the device and install a virtual serial port. A confirmation should now be displayed to indicate that the device has been successfully paired:

After clicking *Close*, you should now see the connected device:

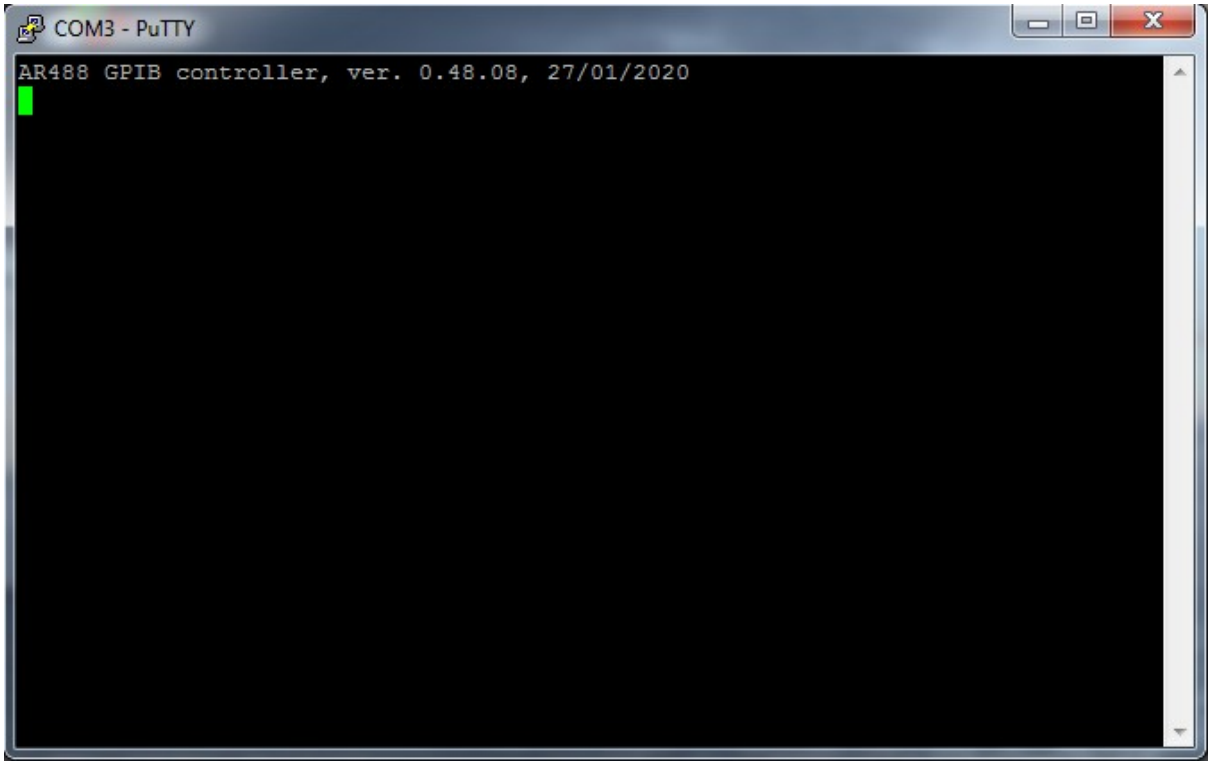


Double click on the device and open the *Hardware* tab:



This should display the virtual serial port that has been assigned to the device, e.g. in the above example it is COM16.

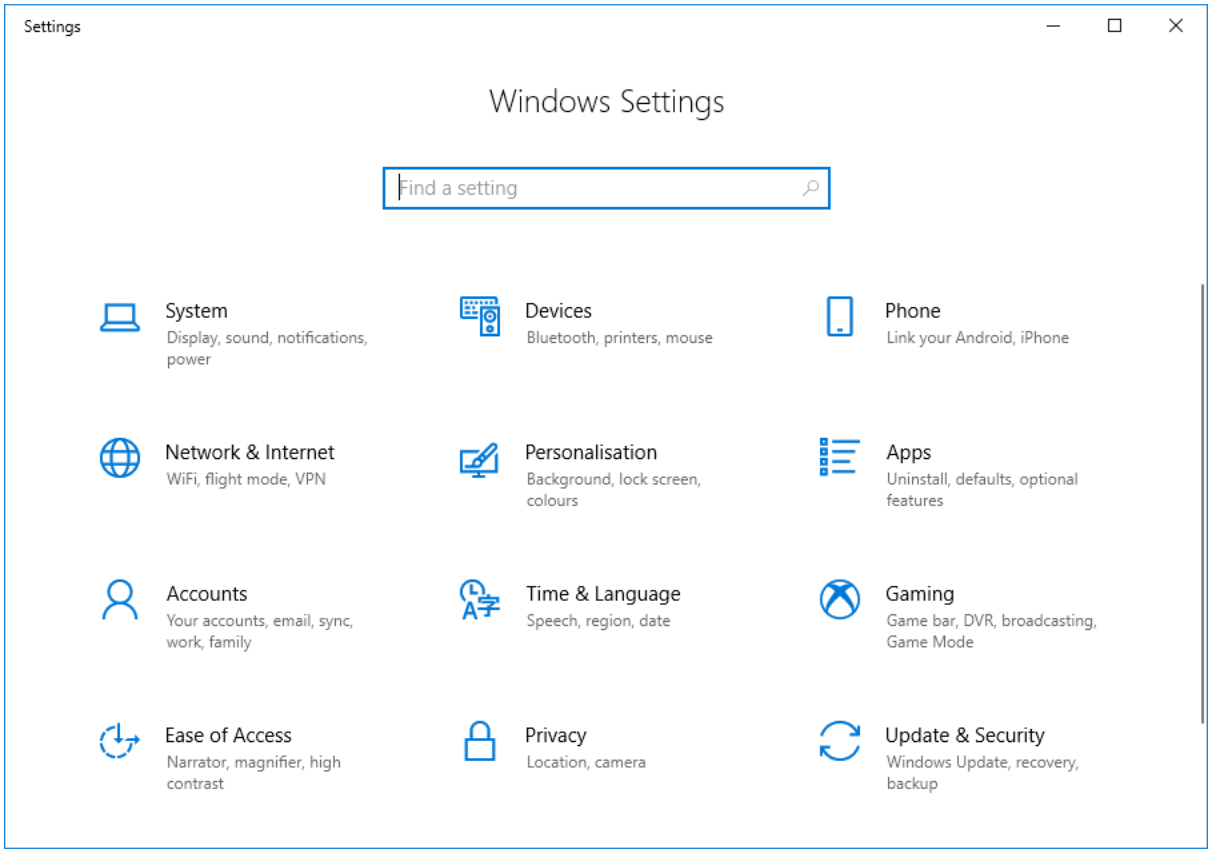
Open a terminal session to the device using the baud rate that was configured in the sketch:



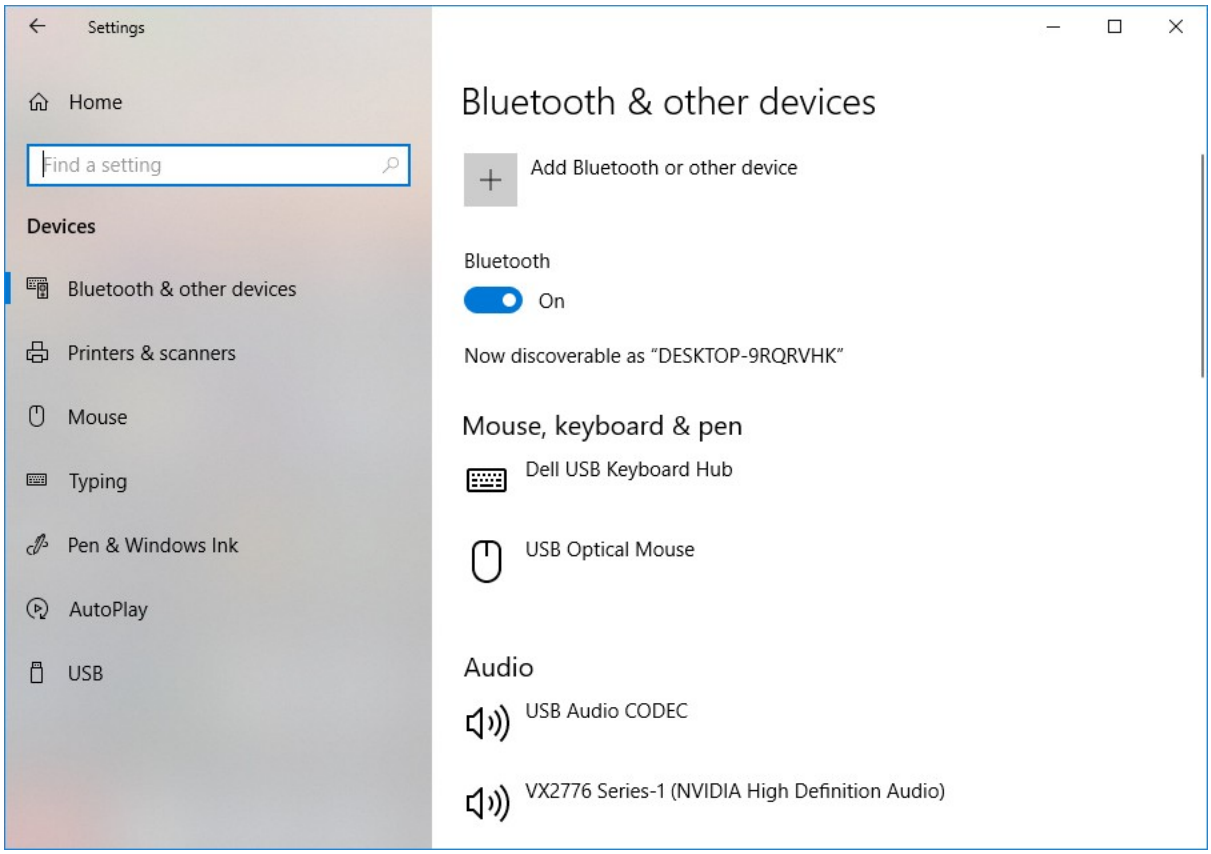
The interface should respond as normal.

Windows 10

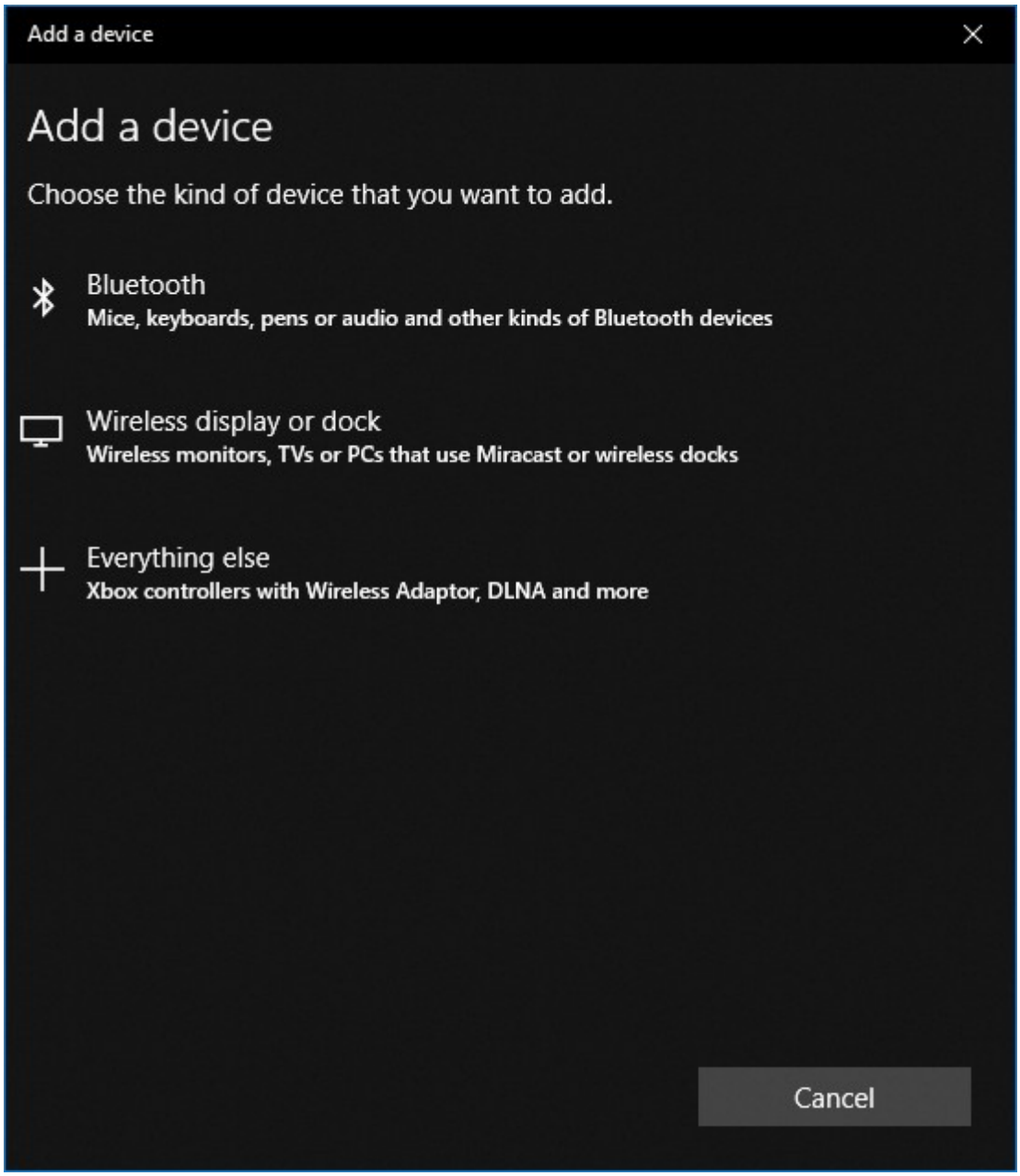
Go to *Windows Settings* (the cog icon on the left of the windows menu) and select *Devices*:



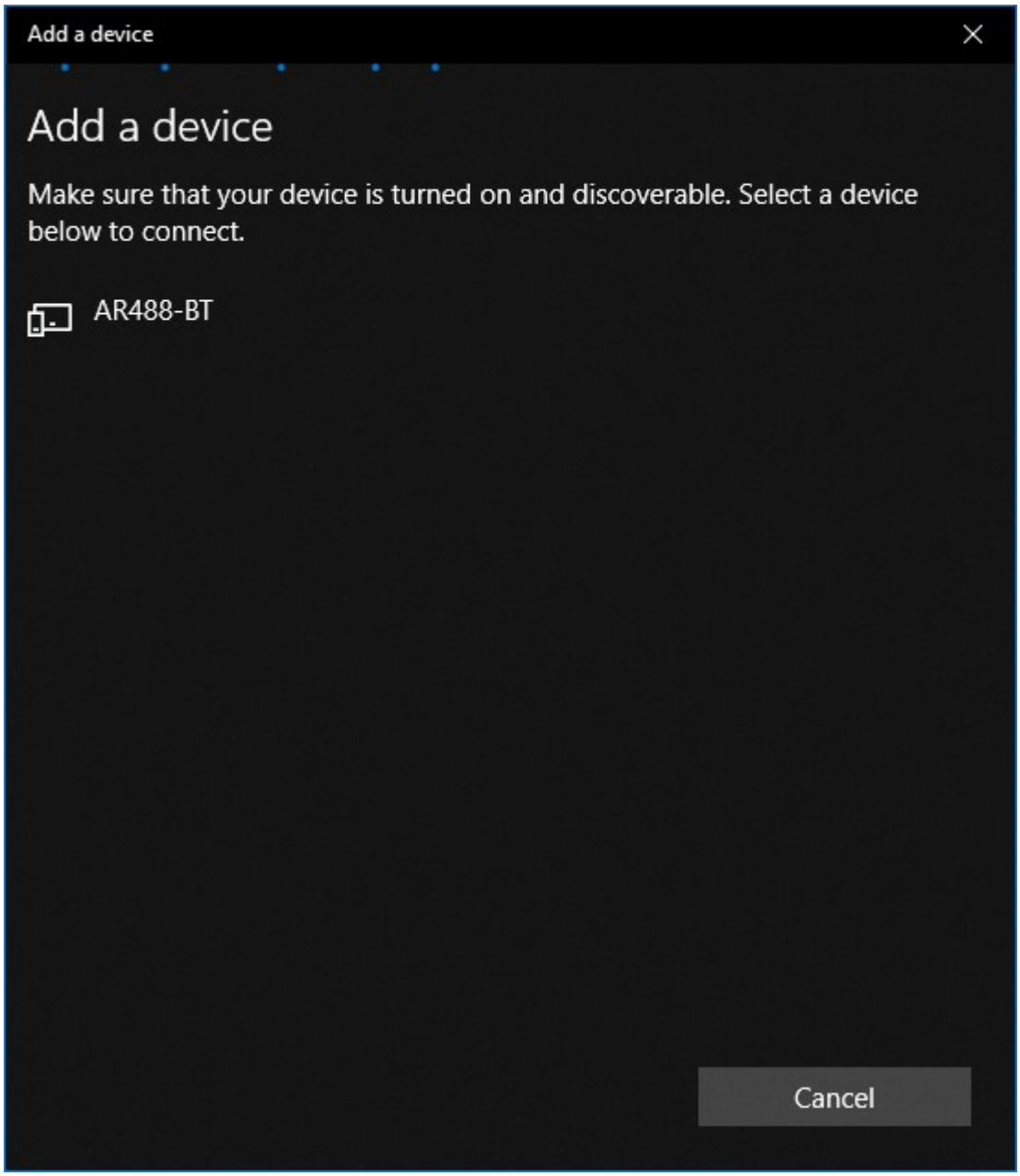
This opens the *Bluetooth & other devices* dialogue:



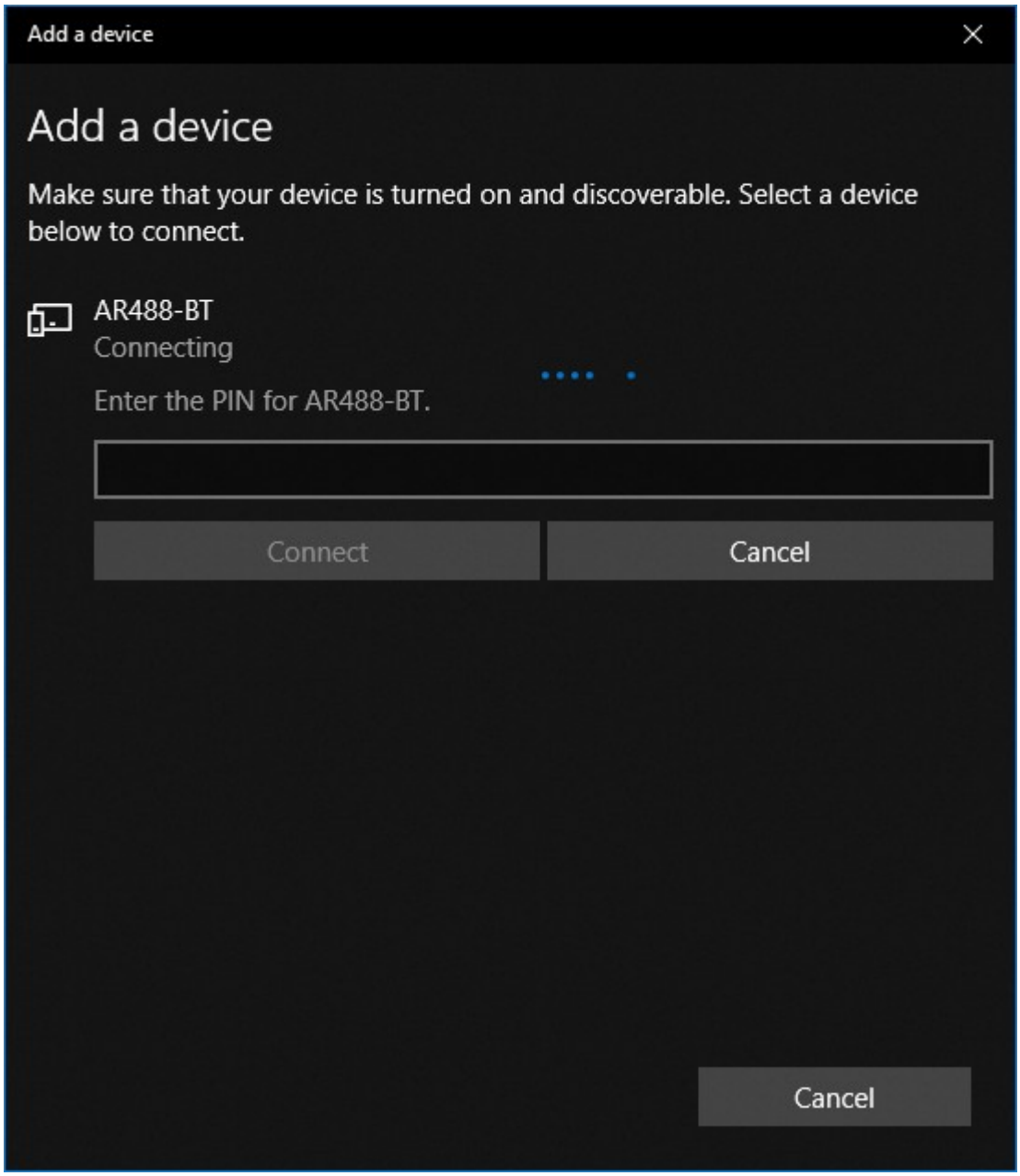
Make sure that Bluetooth is turned on. Click *Add Bluetooth or other device*. This will open another Window:



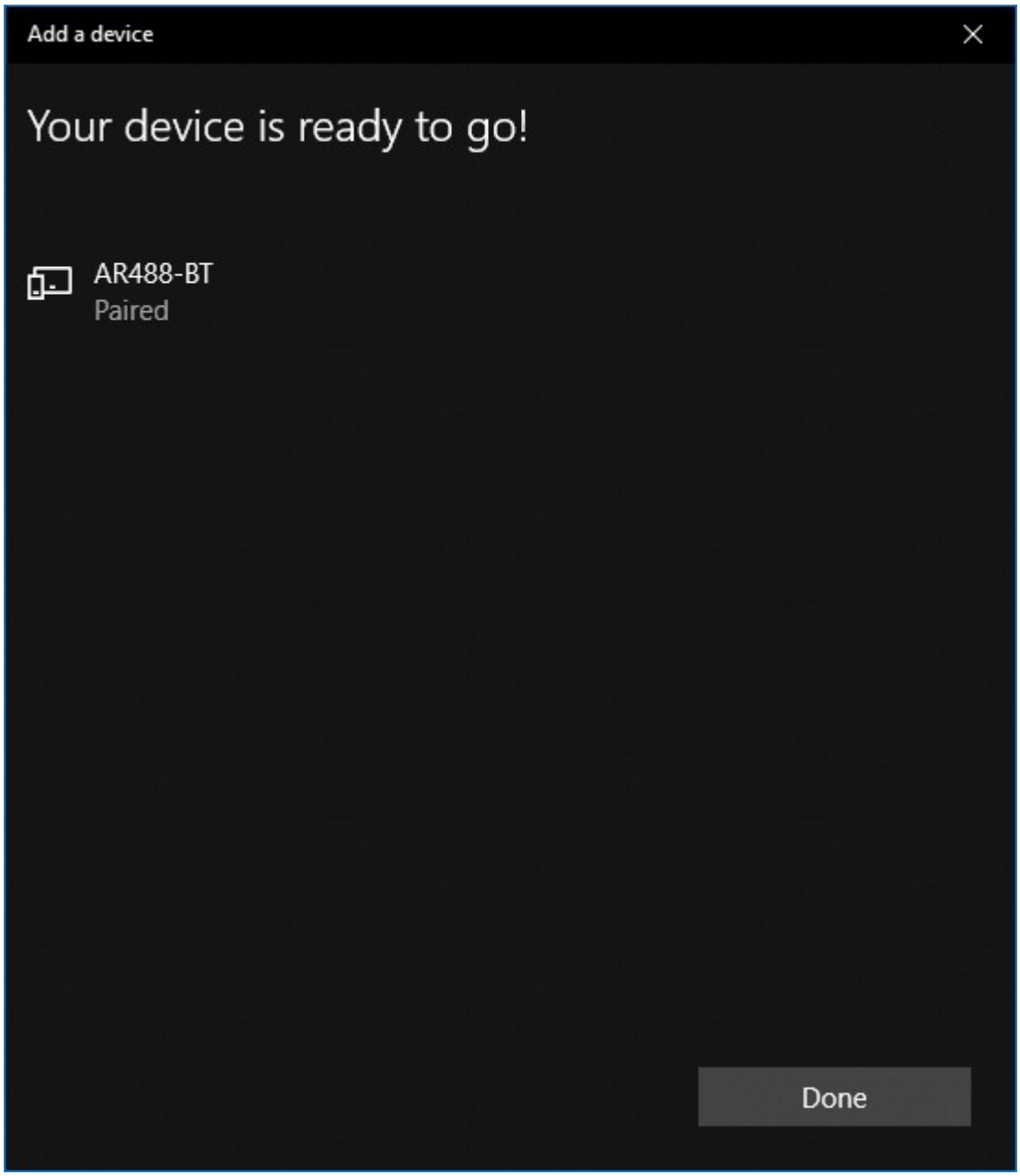
Click *Bluetooth*. Windows should now look for devices. It should momentarily show the AR488-BT device as *unknown device*, but this should quickly change to *AR488-BT*.



Click the AR488-BT device. After a few moments prompt will appear requesting the pin.



Enter the pin and click *Connect*. If it times out, the dialogue may show *"Try connecting your device again"*. Click on the device again to try once more. Once successful, this will be clearly confirmed:

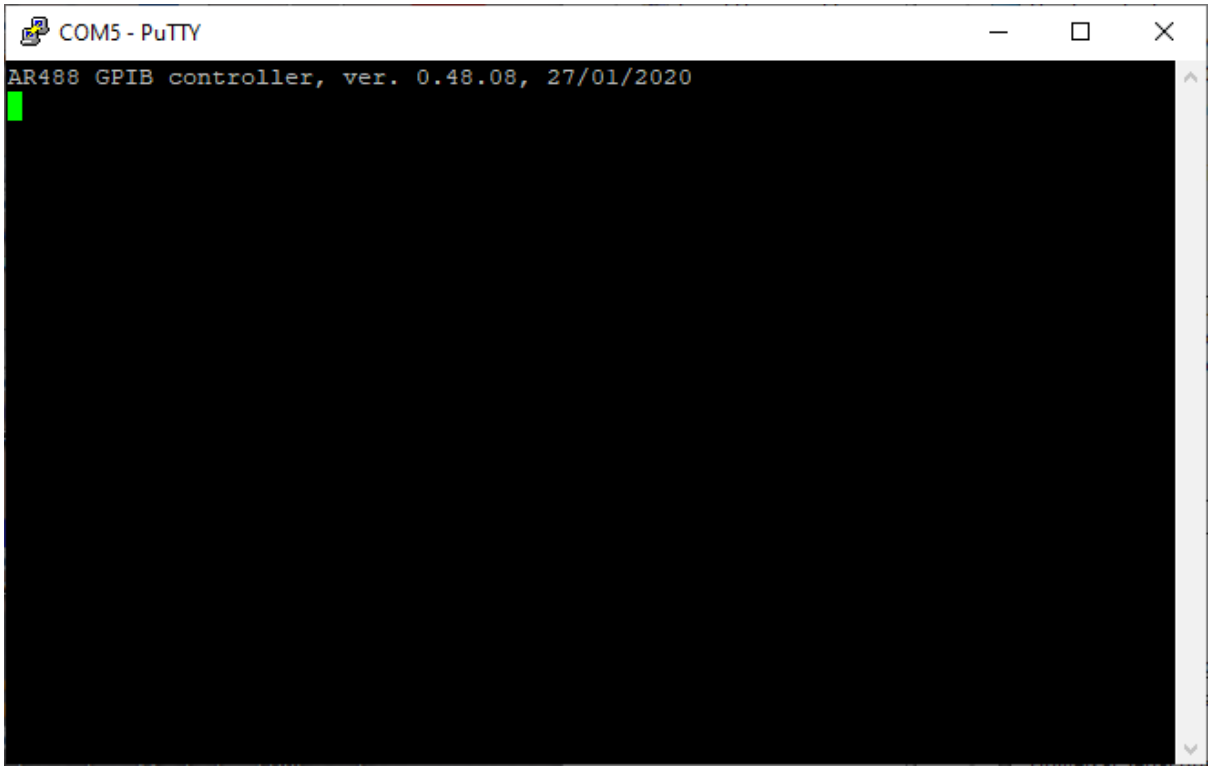


The device status should now be shown as *Paired*. Click the *Done* button to close the device configuration window.

Back on the *Bluetooth & other devices* window, scroll down. The AR488-BT device should be listed under *Other Devices*. The window can now be closed.

Right-click *This PC* and select *Manage*. Click on *Device Manager* and expand the *Port (COM & LPT)* section. The device should be shown as *Standard Serial over Bluetooth link (COMx)* where COMx will be the assigned COM port.

Open a terminal session to the assigned COM port and test communication with the device:



The interface should respond as normal.

Linux MINT

These instructions should work on Linux Mint 19 and above and probably Ubuntu. Depending on your distro of Linux, you may need to download the *bluez* or *bluez5* tools package or compile it from source. On Linux Mint 19 and likely Ubuntu this will be installed by default or can be readily downloaded from the repository using *apt or Software Manager*.

Make sure that your Bluetooth dongle or built-in device is working correctly on your computer or laptop.

Make sure that Bluetooth is turned on and your system can identify your Bluetooth hardware.

Once you have confirmed that your Bluetooth hardware is working, open a terminal and at the command prompt type:

```
% bluetoothctl
```

This should list any known Bluetooth devices, show *Agent registered* at the end of the list, and a `[bluetooth]#` prompt:

```
$ bluetoothctl
[NEW] Controller 00:80:98:94:AB:7E agabus [default]
[NEW] Device 78:3A:84:93:BC:B9 iPad
[NEW] Device 10:2F:6B:BD:49:F1 N930 phone
Agent registered
[bluetooth]#
```

If your device is not listed, then at the prompt type:

```
scan on
```

This should initiate a scan for new devices. In a few seconds any new devices should be listed:

```
[bluetooth]# scan on
Discovery started
[CHG] Controller 00:80:98:94:AB:7E Discovering: yes
[NEW] Device 98:D3:31:F9:4E:6D AR488-BT
```

The AR488-BT device should be detected and its mac address listed.

To pair the device type:

```
[bluetooth]# pair 98:D3:31:F9:4E:6D
```

where the MAC address is the address of YOUR Bluetooth device. The *bluetoothctl* utility should respond with something like:

```
[bluetooth]# pair 98:D3:31:F9:4E:6D
Attempting to pair with 98:D3:31:F9:4E:6D
[CHG] Device 98:D3:31:F9:4E:6D Connected: yes
```

```
Request PIN code
[AR481m[agent] Enter PIN code:
```

Enter the pairing code configured on the HC05 device. The default pairing code is *488488*, but if a custom six digit code has been configured then that should be provided instead. The *bluetoothctl* utility will now attempt to pair with the device. If successful, the output should be something like:

```
[AR481m[agent] Enter PIN code: 488488
[CHG] Device 98:D3:31:F9:4E:6D UUIDs: 00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 98:D3:31:F9:4E:6D ServicesResolved: yes
[CHG] Device 98:D3:31:F9:4E:6D Paired: yes
Pairing successful
[AR488-BT]#
```

At this point the device is paired, but there is no serial port associated with it. Another tool called *rfcomm* can be used to associate a serial port with the Bluetooth device. First exit the *bluetoothctl* utility by typing:

```
[bluetooth]# exit
```

The utiltity will respond with the following and return to the system prompt:

```
Agent unregistered
[DEL] Controller 00:80:98:94:AB:7E agabus [default]
$
```

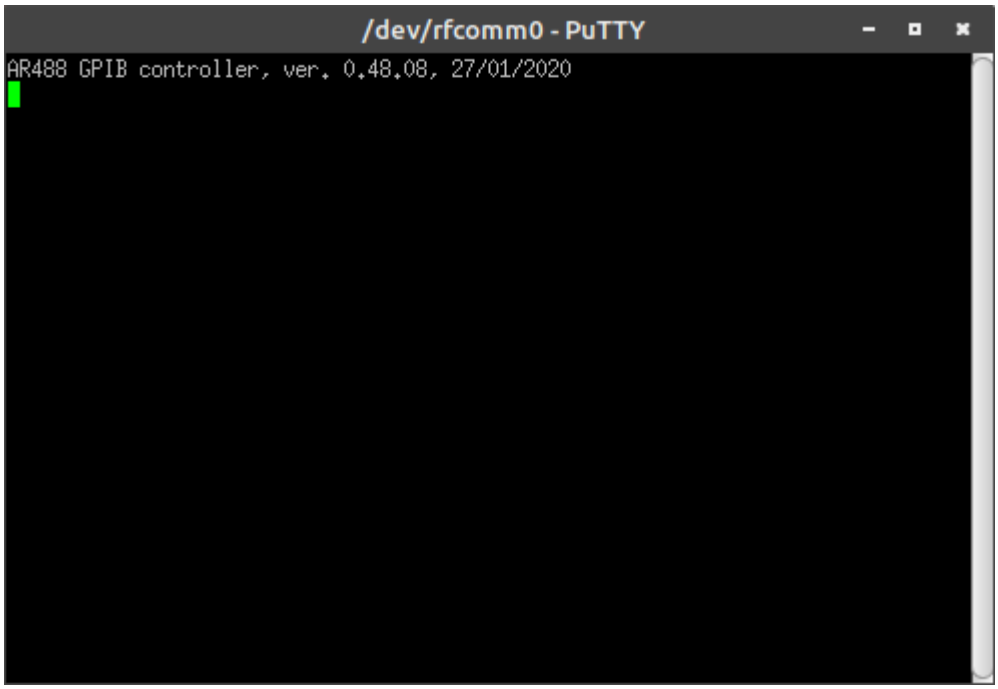
Type the following to associate a serial port with the Bluetooth device:

```
$ sudo rfcomm bind 0 98:D3:31:F9:4E:6D 1
```

You may be prompted for the sudo password. After this has command has completed, a serial port called */dev/rfcomm0* will be created. To confirm that the port binding has been established can be done with the following command:

```
$ ls /dev/rfcomm*
/dev/rfcomm0
```

The repose confirms that the port is now available, so we can open a TTY terminal such as PuTTY and try to establish a connection to it:



The terminal should connect and the device should respond in the usual manner.

To unbind the /dev/rfcomm0 port at the end of the session, from the system `$` prompt type:

```
$ sudo rfcomm unbind 0 98:D3:31:F9:4E:6D 1
```

Further troubleshooting information

A noteworthy point is that *bluetoothctl* is launched while a connected session is in progress, then the prompt will show the name of the currently connected device:

```
[AR488-BT]#
```

To get further information about a device in *bluetoothctl*, when at the `[bluetooth]#` prompt type:

```
info 98:D3:31:F9:4E:6D
```

where the MAC address is the address of the device you would like further information on. The program should respond with something like this:

```
[bluetooth]# info 98:D3:31:F9:4E:6D
Device 98:D3:31:F9:4E:6D (public)
       Name: AR488-BT
       Alias: AR488-BT
       Class: 0x00001f00
```

```
      Paired: yes
      Trusted: no
      Blocked: no
      Connected: yes
      LegacyPairing: yes
      UUID: Serial Port               (00001101-0000-1000-8000-00805f9b34fb)
[bluetooth]#
```

You can also list all the devices that have been paired with:

```
[bluetooth]# paired-devices
Device 98:D3:31:F9:4E:6D AR488-BT
Device 10:2F:6B:BD:49:F1 N930 phone
[bluetooth]#
```

To remove a paired device type:

```
[bluetooth]# remove 98:D3:31:F9:4E:6D
[DEL] Device 98:D3:31:F9:4E:6D AR488-BT
Device has been removed
[bluetooth]#
```

If *bluetoothctl* responds with:

```
Failed to start discovery: org.bluez.Error.NotReady
```

This may indicate that the device is not powered on or blocked. Try this first:

```
[bluetooth]# power on
```

If this reports:

```
Failed to set power on: org.bluez.Error.Blocked
```

Then exit *bluetoothctl* and run this command to check for blocked devices:

```
$ rfkill list
1: hci0: Bluetooth
      Soft blocked: yes
      Hard blocked: no
```

If, as shown above, Bluetooth is 'Soft blocked' then it might be possible to unblock it with:

```
rfkill unblock all
```

Now run *bluetoothctl* again.

Otherwise check that device drivers were properly loaded.

If a device is ‘Hard blocked’ then there is a hardware problem, e.g. a toggle switch may be in the wrong position, faulty cable, power off etc. or the device may be disabled in BIOS.