Bluetooth keyboard

From ArchWiki

This article describes how to set up a Bluetooth HID keyboard with Arch Linux, bluez version 5.

Contents

Pairing process

Login to the affected computer by a wired keyboard or by ssh.

First, make sure the local BT controller (e.g. a BT dongle the built in BT radio) is recognized:

```
# lsusb
```

```
Bus 001 Device 004: ID 0a12:0001 Cambridge Silicon Radio, Ltd Bluetooth Dongle (HCI mode)
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp. LAN9500 Ethernet 10/100 Adapter / S
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

The above output is from a Raspberry-Pi revision 'B' with archlinux-arm and a Keysonic BT Dongle.

Start bluetooth service with *systemctl*, or even better, enable it permanently in the start up process. See systemd.

Three items worth remembering:

- BT devices (keyboard) and controllers (dongle) need to be paired once.
- The BT controller needs to be powered up after every boot.
- The BT controller needs to be told to connect to the keyboard after every boot.

Pairing is a one time process, required only once. There are BT keyboards sold with a BT dongle which come already paired, but that's not certain. We will use the `bluetoothctl` command from bluez5 to pair our dongle and the keyboard.

Power up can be done with `bluetoothctl`, or with `hciconfig` which is more suitable for scripting. See below.

Same for *connecting*, either `bluetoothctl` or `hcitool` can be used, the latter is more useful for scripting.

We will use `bluetoothctl` for the pairing process:

```
# bluetoothctl -a
```

```
[bluetooth]#
```

puts you at the `[bluetooth]#` prompt. If you are on a colour console: the word "bluetooth" is in the default colour as long as no devices are available, and blue as soon as required devices and/or controllers have been found.

While in *bluetoothctl* power up the controller:

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

```
Changing power on succeeded
[CHG] Controller 06:05:04:03:02:01 Powered: yes
```

Next, tell `bluetoothctl` to look only for keyboards, and make that the default agent:

```
[bluetooth]# agent KeyboardOnly
```

```
Agent registered
```

```
[bluetooth]# default-agent
```

```
Default agent request successful
```

Next, put your controller (the local dongle) in *pairable* mode:

```
[bluetooth]# pairable on
```

```
Changing pairable on succeeded
```

Next, put your keyboard in an active mode, where it is *discoverable*, i.e. pairable. Some keyboards have a special button for this on the underside, or require a special key combination to be pressed. See the documentation of your keyboard. Please note that this *discoverability* of a device is time limited, some devices are only visible 30 seconds, other for 2 minutes. Your mileage may vary.

Next, let the controller scan the BT frequencies for a suitable device:

```
[bluetooth]# scan on
```

```
Discovery started
[CHG] Controller 06:05:04:03:02:01 Discovering: yes
```

After a few seconds the adress of the keyboard should be listed as found. This line will repeat over and over, but won't stop you from entering new commands.

Next, actually do the pairing. The address used is the BT-MAC address of the keyboard:

```
[bluetooth]# pair 01:02:03:04:05:06
```

```
Pairing successful
```

Next, make this a trusted device (this allows the device to establish the connection on itself). Again, the BT-MAC address is the address of the keyboard device:

```
[bluetooth]# trust 01:02:03:04:05:06
```

```
Trusted
```

Next and finally connect to the device (keyboard). Again, the BT-MAC address is the address of the keyboard device:

```
[bluetooth]# connect 01:02:03:04:05:06
```

```
Connection successful
```

Done. Leave the `bluetoothctl` utility:

```
[bluetooth]# quit
```

Now the external device (i.e. keyboard) and the USB BT dongle are paired permanently, unless you break the pairing intenionally. This does not mean that the keyboard will connect automatically to your BT device after a boot. *Power up* of the controller and *connecting* device and controller needs to be done after every startup of your computer.

Manually enabling a Bluetooth Keyboard

Although the device and the controller are now paired (see above), you need to connect them every time the computer starts.

First the BT controller (i.e. BT Dongle) needs to be powered up. This is done with the `hciconfig` utility. We assume that you have only one BT device connected, and this one has the symbolic name `hci0` .

```
# hciconfig hci0 up
```

Next, make the connection, now using the `hcitool` utility. Make sure that the keyboard is powered up and connectable, i.e. not in a power saving sleep state.

```
# hcitool cc 01:02:03:04:05:06
```

Your BT keyboard should be useable now, even if an error message ("Can't create connection: Input/output error") is shown. Next we will discuss how to automate this process with systemd.

Automatically enabling a Bluetooth Keyboard

Here is one way to activate a Bluetooth keyboard after a system start. There are certainly other, more elegant ways, but for the moment this has to do. For these tests you need to be logged in to the affected computer, perhaps by a wired keyboard, or by ssh.

We will write an custom systemd service file. Your need to know:

- the MAC address of the BT keyboard. Find out with the **scan on** command of the `bluetoothctl` utility.
- the HCI device identifier of the controller. Find out with

```
# hcitool dev
```

```
Devices:
  hci0   06:05:04:03:02:01
```

We are looking for the identifier 'hci0' in this case. The MAC address shown here is the address of the controller, not the address of the keyboard itself.

Now create a config file `/etc/btkbd.conf` which contains these two data items, the hci-identifier and the keyboard MAC address.

```
# Config file for btkbd.service
# change when required (e.g. keyboard hardware changes, more hci devices are connected)
BTKBDMAC = <mac address here>
HCIDEVICE = <hci-device identifier here>
```

Next, create a new service file at `/etc/systemd/system/btkbd.service`.

```
[Unit]
Description=systemd Unit to automatically start a Bluetooth keyboard
Documentation=https://wiki.archlinux.org/index.php/Bluetooth_Keyboard
Requires=dbus-org.bluez.service
After=dbus-bluez.org.service
ConditionPathExists=/etc/btkbd.conf
ConditionPathExists=/usr/bin/hcitool
ConditionPathExists=/usr/bin/hciconfig

[Service]
Type=oneshot
EnvironmentFile=/etc/btkbd.conf
ExecStart=
ExecStart=/usr/bin/hciconfig ${HCIDEVICE} up
# ignore errors on connect, spurious problems with bt? so start next command with -
ExecStart=-/usr/bin/hcitool cc ${BTKBDMAC}
```

Now start this service with the usual systemd utility and see what happens. Since my keyboard always reports an error on startup, the second command in the service unit file starts with a '-', ignoring any errors from hcitool. YMMV, test with `hcitool` on the command line and see what happens.

If all is ok, permanently enable this service with systemd's tools.

Troubleshooting

- What if the BT controller does not show up in `lsusb` ?
- What if the BT controller is not visible in `bluetoothctl` ?
- My BT keyboard still does not work. What to do?
  - Check: Does it have power?
  - Check: Did it connect to the BT controller? If not, try with another controller or your smart phone.

Xorg

Device should be added as `/dev/input/event*` and your Xorg should add it automatically if you did not disable such feature.

Retrieved from "https://wiki.archlinux.org/index.php?title=Bluetooth_keyboard&oldid=328579"
Categories: Bluetooth | Keyboards