Input problems

Under construction.

ROMs take a long time to load

Some controllers (including Xbox One and 8-bitdo ones) can cause certain cores to take a long time to load a ROM, upwards of thirty seconds. In these situations, it can be worked around by setting the controller to D-input mode (may be referred to as “Android mode” by some manuals). Read more about it in this Retropie forum post.

Bluetooth connection

This is a phenomena causes with some issues and like input lag (especially if you connect more than one BT controller)

1. Try reducing distance from your controller to Raspberry (helps sometimes)
2. Try to disable internal WiFi - this helps to extent signal strength, you can use `rfkill wifi` to disable/enable Wifi (several users report PS3 controller connections problems are vanished since)
3. Try forcing a lower baud rate in `/etc/init.d/S31emulationstation` or `/etc/init.d/S32bluetooth`. Smaller values of 115200 or 230400 have been known to alleviate input lag on a Pi 3B.
4. Buy a new BT adapter and put it to your Raspberry. Add `dtoverlay=pi3-disable-bt` in `/boot/config.txt` to disable internal BT module (ultima ratio, speeds up communication, extents range and speed depending on your BT adapter model)

No controller detected even when connected via USB

Few things to try:

- Use a different USB port
- Ensure controller itself is even functional
- Disable IOMMU setting in BIOS
- Ensure there's no other obvious settings in the BIOS that could be disabling USB functionality
- Clean the contacts with rubbing alcohol (when everything is turned off and disconnected from power), let it dry, and try again
- If using a third-party controller, switch the input mode on it (refer to your controller's manual)
    - D-input typically works best, followed by X-input, then “Mac” and Android modes
- Scream

If all else fails, please provide a detailed issue report so that the devs can look into it. Here's a real example.

Controller connected and detected, but no inputs

Try configuring the controller by manually editing the `/userdata/system/configs/emulationstation/es_input.cfg` file.

Controller connected, detected and making inputs, but they are unexpected/ghost inputs

Ensure that all of the controller's inputs are in a neutral position when plugging in/connected the controller.

If that still gives unexpected ghost inputs during mapping, try to repair the mapping process by renaming `/userdata/system/.sdl2` to `/userdata/system/.sdl2_bad` and immediately restarting ES (`[Alt] + [F4]` or restart via SSH). If this fixes the issue, it could be because of Batocera's neutral position detection method failing, please report such issues to the Github and the steps to reproduce.

Run `cat /userdata/system/.sdl2/<GUID><name of controller>.cache` to grab the neutral values that have been detected.

Diagnose Joysticks Issues

Physical joystick ←→ Linux driver ←→ Linux events stack ←**A**→ SDL2 library ←**B**→ EmulationStation

From your joystick to Batocera's menu interface (EmulationStation), there are several technical steps. There are two tools to analyze two separate parts of the sequence: `evtest` and `sdl2-jstest`.

`evtest` acts on point **A** in the above diagram, used to debug very low-level issues (like a button's state change not being registered by the controller or an incorrect amount of inputs being registered). `sdl2-jstest` acts on point **B** in the above diagram, used to debug high-level issues (like EmulationStation not reacting correctly when button X is pressed, certain emulator not “seeing” a controller's inputs which otherwise work fine in other programs, etc.).

evtests

`events` in the linux kernel are the way to handle mouses, keyboards, joysticks, etc... sdl2 is a library to help to code program compatible with linux, windows, mac os x, ... it converts linux events to a common way to define events. On linux, events are define in /usr/include/linux/input-event-codes.h. For example, the button “yellow” if your joystick has one is :

```
#define KEY_YELLOW      0x190 (1*16*16 + 9*16 + 0 = 400)
```

On Windows, possibly that this button is not defined, or at least, for sure, with an other value, while

on Linux, the value is arbitrary. Thus, SDL doesn't use the code. Instead, SDL uses button numbers, thus, this button may be “BUTTON 1”.

Let's see an example :

1. Connect via ssh to batocera
2. run evtest to check that linux sees your joysticks :

In my case, i've only 1 joystick plugged, it displays :

```
# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:  Bigben Interactive Bigben Game Pad
Select the device event number [0-0]:
```

Press 0 to see linux events on this joystick. In my case, when i press one button (button BTN_C), it displays :

```
Event: time 1492442283.255053, type 1 (EV_KEY), code 306 (BTN_C), value 1
Event: time 1492442283.255053, type 1 (EV_KEY), code 306 (BTN_C), value 0
```

value 1 is when the button is pressed. value 0 is when the buton is released.

Some general information, like the list of buttons available are provided by :

```
evtest --info /dev/input/event0
udevadm info -q all -n /dev/input/event0
```

sdl2-jstest

Run `sdl2-jstest --list` to check what EmulationStation sees when you press buttons In my case, it displays :

```
# export DISPLAY=:0.0
# sdl2-jstest --list
Found 1 joystick(s)

Joystick Name:     'Bigben Interactive Bigben Game Pad'
Joystick Path:     '/dev/input/event0'
Joystick GUID:     030000006b1400000209000011010000
Joystick Number:    0
Number of Axes:     4
Number of Buttons: 13
Number of Hats:     1
...
Button code  0:   304
Button code  1:   305
Button code  2:   306
```

```
...
```

To see the events, run sdl2-jstest -e 0 (0 is the joystick number) When I press C, it displays :

```
...
SDL_JOYBUTTONDOWN: joystick: 0 button: 2 state: 1 code:306
SDL_JOYBUTTONUP: joystick: 0 button: 2 state: 0 code:306
...
```

You can see a graphical representation of your controller by running:

```
sdl2-jstest --test #
```

where # is the number of the joystick you want to test.