# Setting up Bluetooth controls for KOReader on Kobo Devices

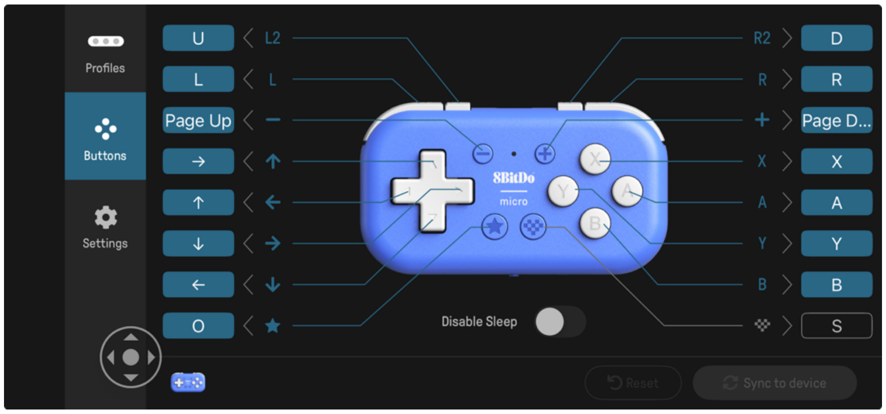
**Notes:**

- These instructions are setting up Bluetooth page-turn and other controls on KOReader on Kobo devices. I only have the Kobo Libra Colour, so I cannot vouch for other devices. As far as I can tell, the steps are pretty generic, so I imagine the instructions should work on other Kobo devices as well.
- The Bluetooth device I'm using is 8BitDo Micro. The instructions for other devices should be similar, and you should be able to follow this guide while making minor changes as needed.
- These instructions assume you already have NickelMenu and KOReader installed. There are plenty of instructions for installing those, and I didn't want to make the tutorial needlessly long.
- IMPORTANT NOTE: Currently there is a known bug in KOReader where exiting KOReader while Bluetooth is ON, reboots the device and often removes NickelMenu. You then have to transfer the KoboRoot.tgz file into the .kobo folder again to get NickelMenu back. This behavior can be avoided by choosing 'Reboot the device' in the exit menu, instead of 'Exit'. So, while this issue exists, remember to reboot the device to exit KOReader while you have Bluetooth ON
    - Reference Link -- https://github.com/koreader/koreader/issues/12739
- These instructions do not use any plug-in or different Bluetooth drivers. The default Kobo Bluetooth drivers work just fine.

**Features enabled:**

- This guide will enable the following features via Bluetooth -
    - Go to next page / previous page
    - Go to next chapter / previous chapter
    - Increase Font size / Decrease Font size
    - Increase brightness / Decrease brightness
    - Increase warmth / Decrease warmth
    - Go to next bookmark / previous bookmark
    - Rotate screen
    - Toggle Night mode
    - Toggle Bookmark
    - Toggle Status Bar
- I've not mapped every single of the above features to my controller, but this guide will show you how to map whichever features you want.

**Step 1 – Setting up your Bluetooth controller**

- Skip this step if you are not using a controller that allows for custom mapping. Since I use the 8BitDo micro, I've included instructions for that device.
- I've setup mapping assuming controller usage in vertical orientation. You may set it up any which way you like.
- 8BitDo micro will be used in keyboard mode. Go ahead and map random keyboard keys to all the buttons using the 8BitDo Ultimate Software. It does not matter which keys you map. We will be noting down codes for all the keys later. For reference, this is how I've mapped my controller.
- 

| | | | |
|---|---|---|---|
| U | L2 | R2 | D |
| L | L | R | R |
| Page Up | − | + | Page D... |
| → | ↑ | X | X |
| ↑ | ← | A | A |
| ↓ | → | Y | Y |
| ← | ↓ | B | B |
| O | ★ | ⁜ | S |

- Note that I've mapped the 'Keyboard right key' to the Up D-Pad, and 'Keyboard left key' to the Down D-pad. This is on purpose since keyboard left and right keys turn pages on the native Kobo readers without any additional setup. And since I hold the micro vertically, mapping it this way is more intuitive to me.

**Step 2 – Pair your device**

- Pair the controller using the Bluetooth settings page on the Kobo
- Side note on 8BitDo micro - Reconnecting the device to Kobo seems to be a bit finicky. I find it best to turn Bluetooth ON on the settings page, and then turning on the controller after a few seconds. The controller then usually connects on first go. Otherwise, it takes three or four tries to connect the controller. I do not use the Nickelmenu option to Turn Bluetooth ON, as that never automatically connects the controller (even though Bluetooth does turn ON, on the Kobo).

**Step 3 – Confirm 'event#' for your controller and note event keys**

- Make sure your device is connected via Bluetooth. Open KOReader.
- Log into KOReader via SSH
  - KOReader → Gear icon → Network → SSH Server

- Tick ‘Login without password (DANGEROUS)’, and then tick ‘SSH server’. (Of course you can choose to set up SSH credentials, but we’ll directly log in for sake of convenience).
- Take note of the device’s internal IP address on the pop-up screen
- On your computer open Terminal if on Mac or Linux, Command Prompt if on Windows
- Enter **ssh -p 2222 root@(ip-address-you-noted above)**
- You will be asked if you want to continue, and no password is required (just hit enter)
- Enter **ls /dev/input**
- This will show inputs connected to your Kobo, something like –
    - By-path event0 event1 event2 event3 event4
- Typically, the last one will be your Bluetooth device if you haven’t connected multiple Bluetooth devices to the Kobo. To check which one is your controller, you can try this step with and without your controller connected. When it’s not connected, that event# will not be there. In my case, ‘event4’ shows up when the 8BitDo controller is connected, and is missing when it’s not. This confirms the 8BitDo controller is ‘event4’.
- Type **evtest /dev/input/event4** (replace ‘event4’ with whichever one you think is your connected controller)
- The display will show a long list of Event codes, and then wait on ‘Testing … (interrupt to exit)’
    - At this point, if you press any key on the controller, you will an event response. This confirms that your controller is indeed 'event4' (or whichever one.
    - Press Ctrl+C to exit
- Scroll up to see the long list of Event codes. Now make note of whichever keys you mapped on your Bluetooth controller. For example, the line ‘Event code 20 (T)’ means that keyboard key ‘T’ has event code 20.
    - In our case, the keys we mapped on the 8BitDo micro are as follows –
        - Event code 103 (Up) — mapped to Left D-Pad
        - Event code 105 (Left) — mapped to Down D-Pad
        - Event code 106 (Right) — mapped to Up D-Pad
        - Event code 108 (Down) — mapped to Right D-Pad
        - Event code 104 (PageUp) — mapped to minus
        - Event code 109 (PageDown) — mapped to plus
        - Event code 21 (Y) — mapped to Y
        - Event code 45 (X) — mapped to X
        - Event code 48 (B) — mapped to B

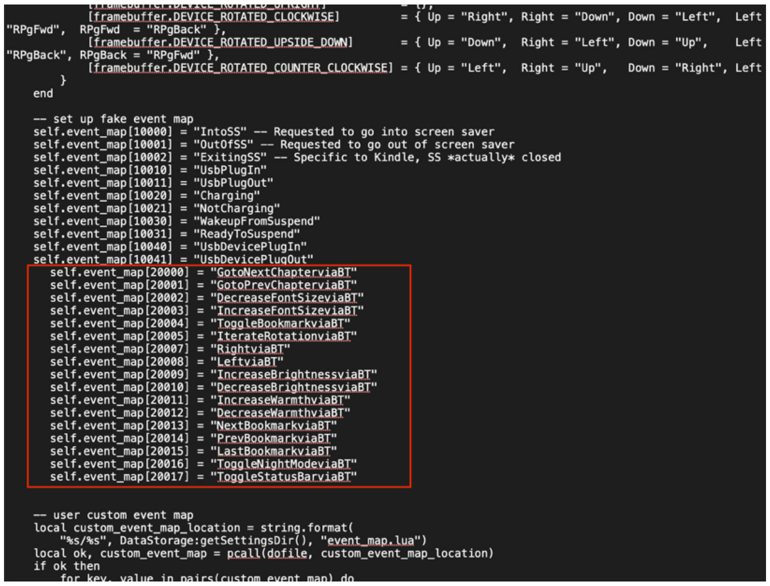
        - Event code 30 (A) — mapped to A
        - Event code 38 (L) — mapped to Left Trigger L
        - Event code 22 (U) — mapped to L2
        - Event code 32 (D) — mapped to R2
        - Event code 19 (R) — mapped to Right Trigger R
        - Event code 24 (O) — mapped to Star button
    - Turn off the SSH server
    - Exit KOReader (remember to use Reboot due to the KOReader bug I mentioned earlier)

## Step 4 – Update input.lua file

- Connect your ereader to you computer via USB
- Navigate to KOBOeReader/.adds/koreader/frontend/device
- Take a backup of the input.lua file, so you can place the original back and start over if anything goes wrong.
- Open input.lua
    - Scroll down to the section which has the header – *set up fake event map (*This section has lines that look like Self.event_map[10000] = "IntoSS")
    - Add the following lines to this section –

```
self.event_map[20000] = "GotoNextChapterviaBT"
self.event_map[20001] = "GotoPrevChapterviaBT"
self.event_map[20002] = "DecreaseFontSizeviaBT"
self.event_map[20003] = "IncreaseFontSizeviaBT"
self.event_map[20004] = "ToggleBookmarkviaBT"
self.event_map[20005] = "IterateRotationviaBT"
self.event_map[20007] = "RightviaBT"
self.event_map[20008] = "LeftviaBT"
self.event_map[20009] = "IncreaseBrightnessviaBT"
self.event_map[20010] = "DecreaseBrightnessviaBT"
self.event_map[20011] = "IncreaseWarmthviaBT"
self.event_map[20012] = "DecreaseWarmthviaBT"
self.event_map[20013] = "NextBookmarkviaBT"
self.event_map[20014] = "PrevBookmarkviaBT"
self.event_map[20015] = "LastBookmarkviaBT"
self.event_map[20016] = "ToggleNightModeviaBT"
self.event_map[20017] = "ToggleStatusBarviaBT"
```

- The numbers in the brackets above are just random numbers I picked. They don't have any significance here.
- Screenshot for reference –

- ○ 

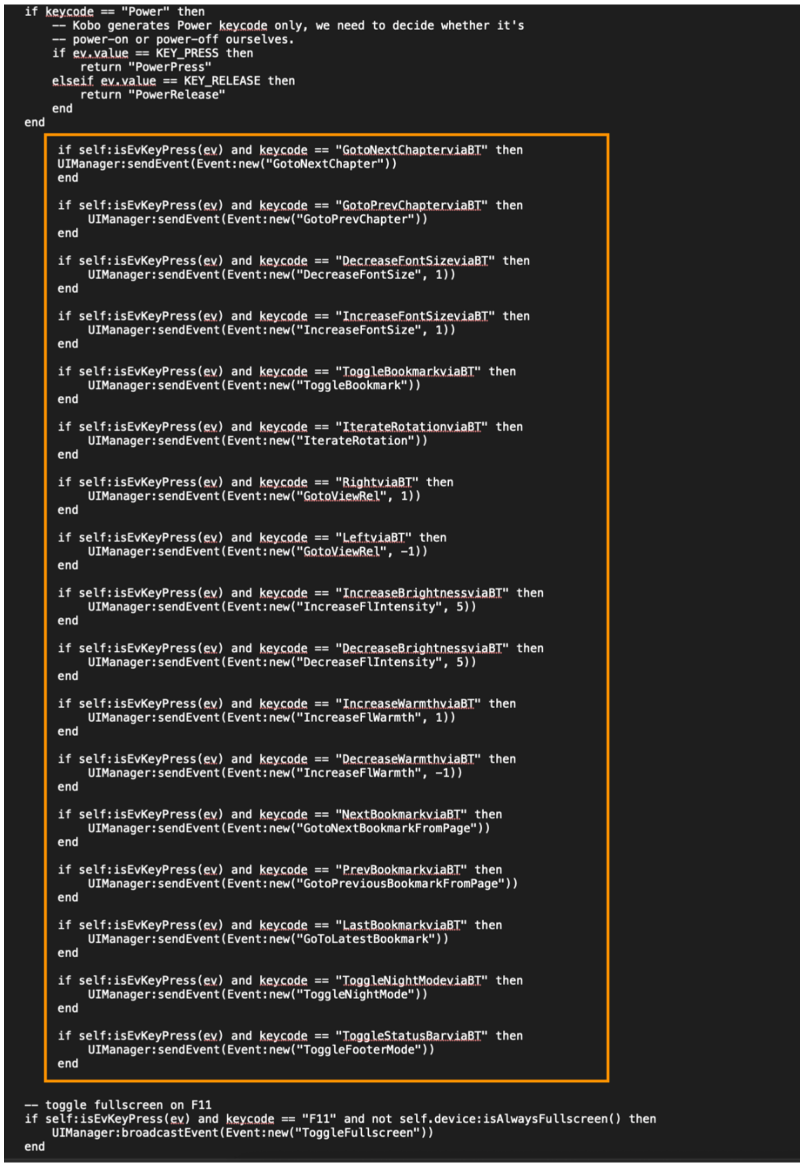
```
            [framebuffer.DEVICE_ROTATED_CLOCKWISE]         = { Up = "Right", Right = "Down", Down = "Left",  Left
"RPgFwd",  RPgFwd  = "RPgBack" },
            [framebuffer.DEVICE_ROTATED_UPSIDE_DOWN]       = { Up = "Down",  Right = "Left", Down = "Up",    Left
"RPgBack", RPgBack = "RPgFwd" },
            [framebuffer.DEVICE_ROTATED_COUNTER_CLOCKWISE] = { Up = "Left",  Right = "Up",   Down = "Right", Left
        }
    end

    -- set up fake event map
    self.event_map[10000] = "IntoSS" -- Requested to go into screen saver
    self.event_map[10001] = "OutOfSS" -- Requested to go out of screen saver
    self.event_map[10002] = "ExitingSS" -- Specific to Kindle, SS *actually* closed
    self.event_map[10010] = "UsbPlugIn"
    self.event_map[10011] = "UsbPlugOut"
    self.event_map[10020] = "Charging"
    self.event_map[10021] = "NotCharging"
    self.event_map[10030] = "WakeupFromSuspend"
    self.event_map[10031] = "ReadyToSuspend"
    self.event_map[10040] = "UsbDevicePlugIn"
    self.event_map[10041] = "UsbDevicePlugOut"
      self.event_map[20000] = "GotoNextChapterviaBT"
      self.event_map[20001] = "GotoPrevChapterviaBT"
      self.event_map[20002] = "DecreaseFontSizeviaBT"
      self.event_map[20003] = "IncreaseFontSizeviaBT"
      self.event_map[20004] = "ToggleBookmarkviaBT"
      self.event_map[20005] = "IterateRotationviaBT"
      self.event_map[20007] = "RightviaBT"
      self.event_map[20008] = "LeftviaBT"
      self.event_map[20009] = "IncreaseBrightnessviaBT"
      self.event_map[20010] = "DecreaseBrightnessviaBT"
      self.event_map[20011] = "IncreaseWarmthviaBT"
      self.event_map[20012] = "DecreaseWarmthviaBT"
      self.event_map[20013] = "NextBookmarkviaBT"
      self.event_map[20014] = "PrevBookmarkviaBT"
      self.event_map[20015] = "LastBookmarkviaBT"
      self.event_map[20016] = "ToggleNightModeviaBT"
      self.event_map[20017] = "ToggleStatusBarviaBT"

    -- user custom event map
    local custom_event_map_location = string.format(
        "%s/%s", DataStorage:getSettingsDir(), "event_map.lua")
    local ok, custom_event_map = pcall(dofile, custom_event_map_location)
    if ok then
```

- ○ Scroll down further till you reach a function called Input:handleKeyBoardEv(ev). The following lines need to be within this function. I placed them just before a section with the header -- *toggle fullscreen on F11.*

```
if self:isEvKeyPress(ev) and keycode == "GotoNextChapterviaBT" then
      UIManager:sendEvent(Event:new("GotoNextChapter"))
end

if self:isEvKeyPress(ev) and keycode == "GotoPrevChapterviaBT" then
      UIManager:sendEvent(Event:new("GotoPrevChapter"))
end

if self:isEvKeyPress(ev) and keycode == "DecreaseFontSizeviaBT" then
      UIManager:sendEvent(Event:new("DecreaseFontSize", 1))
end

if self:isEvKeyPress(ev) and keycode == "IncreaseFontSizeviaBT" then
      UIManager:sendEvent(Event:new("IncreaseFontSize", 1))
end

if self:isEvKeyPress(ev) and keycode == "ToggleBookmarkviaBT" then
      UIManager:sendEvent(Event:new("ToggleBookmark"))
end

if self:isEvKeyPress(ev) and keycode == "IterateRotationviaBT" then
      UIManager:sendEvent(Event:new("IterateRotation"))
end

if self:isEvKeyPress(ev) and keycode == "RightviaBT" then
      UIManager:sendEvent(Event:new("GotoViewRel", 1))
end
if self:isEvKeyPress(ev) and keycode == "LeftviaBT" then
      UIManager:sendEvent(Event:new("GotoViewRel", -1))
end

if self:isEvKeyPress(ev) and keycode == "IncreaseBrightnessviaBT" then
      UIManager:sendEvent(Event:new("IncreaseFlIntensity", 5))
end
```

```
if self:isEvKeyPress(ev) and keycode == "DecreaseBrightnessviaBT" then
        UIManager:sendEvent(Event:new("DecreaseFlIntensity", 5))
end

if self:isEvKeyPress(ev) and keycode == "IncreaseWarmthviaBT" then
UIManager:sendEvent(Event:new("IncreaseFlWarmth", 1))
end

if self:isEvKeyPress(ev) and keycode == "DecreaseWarmthviaBT" then
        UIManager:sendEvent(Event:new("IncreaseFlWarmth", -1))
end

if self:isEvKeyPress(ev) and keycode == "NextBookmarkviaBT" then
UIManager:sendEvent(Event:new("GotoNextBookmarkFromPage"))
end

if self:isEvKeyPress(ev) and keycode == "PrevBookmarkviaBT" then
        UIManager:sendEvent(Event:new("GotoPreviousBookmarkFromPage"))
end

if self:isEvKeyPress(ev) and keycode == "LastBookmarkviaBT" then
        UIManager:sendEvent(Event:new("GoToLatestBookmark"))
end

if self:isEvKeyPress(ev) and keycode == "ToggleNightModeviaBT" then
        UIManager:sendEvent(Event:new("ToggleNightMode"))
end

if self:isEvKeyPress(ev) and keycode == "ToggleStatusBarviaBT" then
        UIManager:sendEvent(Event:new("ToggleFooterMode"))
end
```

- Screenshot for reference –

- 

```
if keycode == "Power" then
    -- Kobo generates Power keycode only, we need to decide whether it's
    -- power-on or power-off ourselves.
    if ev.value == KEY_PRESS then
        return "PowerPress"
    elseif ev.value == KEY_RELEASE then
        return "PowerRelease"
    end
end
     if self:isEvKeyPress(ev) and keycode == "GotoNextChaptervia BT" then
     UIManager:sendEvent(Event:new("GotoNextChapter"))
     end

     if self:isEvKeyPress(ev) and keycode == "GotoPrevChaptervia BT" then
         UIManager:sendEvent(Event:new("GotoPrevChapter"))
     end

     if self:isEvKeyPress(ev) and keycode == "DecreaseFontSizeviaBT" then
         UIManager:sendEvent(Event:new("DecreaseFontSize", 1))
     end

     if self:isEvKeyPress(ev) and keycode == "IncreaseFontSizeviaBT" then
         UIManager:sendEvent(Event:new("IncreaseFontSize", 1))
     end

     if self:isEvKeyPress(ev) and keycode == "ToggleBookmarkviaBT" then
         UIManager:sendEvent(Event:new("ToggleBookmark"))
     end

     if self:isEvKeyPress(ev) and keycode == "IterateRotationviaBT" then
         UIManager:sendEvent(Event:new("IterateRotation"))
     end

     if self:isEvKeyPress(ev) and keycode == "RightviaBT" then
         UIManager:sendEvent(Event:new("GotoViewRel", 1))
     end

     if self:isEvKeyPress(ev) and keycode == "LeftviaBT" then
         UIManager:sendEvent(Event:new("GotoViewRel", -1))
     end

     if self:isEvKeyPress(ev) and keycode == "IncreaseBrightnessviaBT" then
         UIManager:sendEvent(Event:new("IncreaseFlIntensity", 5))
     end

     if self:isEvKeyPress(ev) and keycode == "DecreaseBrightnessviaBT" then
         UIManager:sendEvent(Event:new("DecreaseFlIntensity", 5))
     end

     if self:isEvKeyPress(ev) and keycode == "IncreaseWarmthviaBT" then
         UIManager:sendEvent(Event:new("IncreaseFlWarmth", 1))
     end

     if self:isEvKeyPress(ev) and keycode == "DecreaseWarmthviaBT" then
         UIManager:sendEvent(Event:new("IncreaseFlWarmth", -1))
     end

     if self:isEvKeyPress(ev) and keycode == "NextBookmarkviaBT" then
         UIManager:sendEvent(Event:new("GotoNextBookmarkFromPage"))
     end

     if self:isEvKeyPress(ev) and keycode == "PrevBookmarkviaBT" then
         UIManager:sendEvent(Event:new("GotoPreviousBookmarkFromPage"))
     end

     if self:isEvKeyPress(ev) and keycode == "LastBookmarkviaBT" then
         UIManager:sendEvent(Event:new("GoToLatestBookmark"))
     end

     if self:isEvKeyPress(ev) and keycode == "ToggleNightModeviaBT" then
         UIManager:sendEvent(Event:new("ToggleNightMode"))
     end

     if self:isEvKeyPress(ev) and keycode == "ToggleStatusBarviaBT" then
         UIManager:sendEvent(Event:new("ToggleFooterMode"))
     end

-- toggle fullscreen on F11
if self:isEvKeyPress(ev) and keycode == "F11" and not self.device:isAlwaysFullscreen() then
    UIManager:broadcastEvent(Event:new("ToggleFullscreen"))
end
```

- Save and close the file

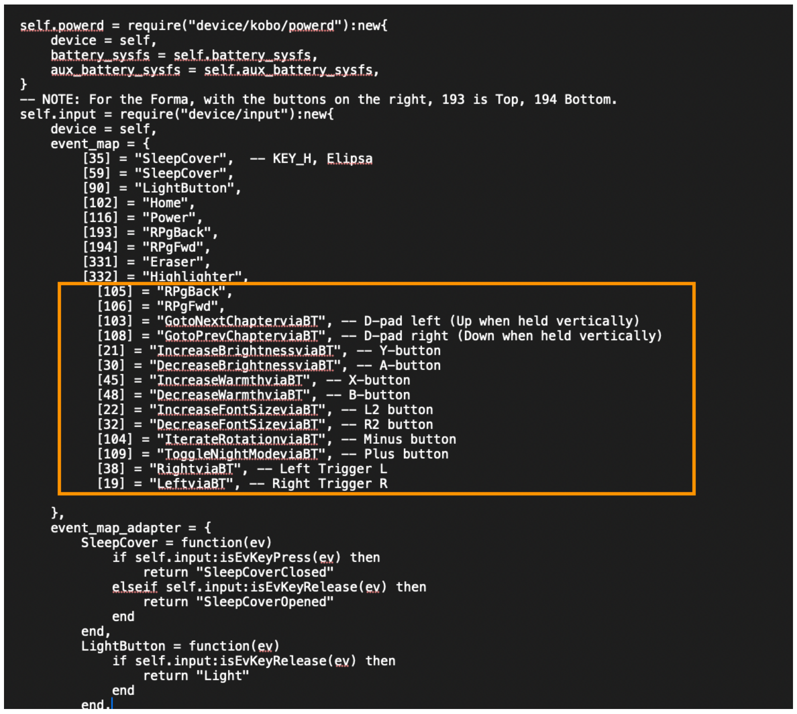
**Step 5 – Update device.lua file**

- Navigate to KOBOeReader/.adds/koreader/frontend/device/kobo
- Take a backup of the device.lua file, so you can place the original back and start over if anything goes wrong.
- Navigate to the section that has event maps (Starts with *event_map = {*)
- Here you will add lines for the features that you want, and map them against the buttons you've picked. So, for example if we want to map increase font size to the L2 button. On the 8BitDo app we mapped L2 button on the controller to keyboard key 'U'.

And we noted in our event map codes that keyboard key U had event code 22. And so here we will add the event "IncreaseFontSizeviaBT" against code [22]. See format below.

- I'm putting down lines I added below, but you can change these to whichever keys and features you prefer. The "RightviaBT"/"LeftviaBT" are functionally same as "RPgFwd"/"RPgBack". RPgFwd/RPgBack functions may be eReader model specific, so feel free to use the RightviaBT / LeftviaBT instead.

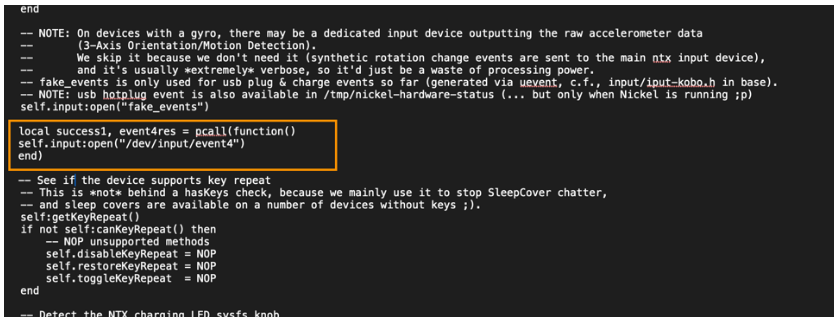
```
[105] = "RPgBack",
[106] = "RPgFwd",
[103] = "GotoNextChapterviaBT", -- D-pad left (Up when held vertically)
[108] = "GotoPrevChapterviaBT", -- D-pad right (Down when held vertically)
[21] = "IncreaseBrightnessviaBT", -- Y-button
[30] = "DecreaseBrightnessviaBT", -- A-button
[45] = "IncreaseWarmthviaBT", -- X-button
[48] = "DecreaseWarmthviaBT", -- B-button
[22] = "IncreaseFontSizeviaBT", -- L2 button
[32] = "DecreaseFontSizeviaBT", -- R2 button
[104] = "IterateRotationviaBT", -- Minus button
[109] = "ToggleNightModeviaBT", -- Plus button
[38] = "RightviaBT", -- Left Trigger L
[19] = "LeftviaBT", -- Right Trigger R
```

- Screenshot for reference -
  - 

```
self.powerd = require("device/kobo/powerd"):new{
    device = self,
    battery_sysfs = self.battery_sysfs,
    aux_battery_sysfs = self.aux_battery_sysfs,
}
-- NOTE: For the Forma, with the buttons on the right, 193 is Top, 194 Bottom.
self.input = require("device/input"):new{
    device = self,
    event_map = {
        [35] = "SleepCover",  -- KEY_H, Elipsa
        [59] = "SleepCover",
        [90] = "LightButton",
        [102] = "Home",
        [116] = "Power",
        [193] = "RPgBack",
        [194] = "RPgFwd",
        [331] = "Eraser",
        [332] = "Highlighter",
          [105] = "RPgBack",
          [106] = "RPgFwd",
          [103] = "GotoNextChapterviaBT", -- D-pad left (Up when held vertically)
          [108] = "GotoPrevChapterviaBT", -- D-pad right (Down when held vertically)
          [21] = "IncreaseBrightnessviaBT", -- Y-button
          [30] = "DecreaseBrightnessviaBT", -- A-button
          [45] = "IncreaseWarmthviaBT", -- X-button
          [48] = "DecreaseWarmthviaBT", -- B-button
          [22] = "IncreaseFontSizeviaBT", -- L2 button
          [32] = "DecreaseFontSizeviaBT", -- R2 button
          [104] = "IterateRotationviaBT", -- Minus button
          [109] = "ToggleNightModeviaBT", -- Plus button
          [38] = "RightviaBT", -- Left Trigger L
          [19] = "LeftviaBT", -- Right Trigger R

    },
    event_map_adapter = {
        SleepCover = function(ev)
            if self.input:isEvKeyPress(ev) then
                return "SleepCoverClosed"
            elseif self.input:isEvKeyRelease(ev) then
                return "SleepCoverOpened"
            end
        end,
        LightButton = function(ev)
            if self.input:isEvKeyRelease(ev) then
                return "Light"
            end
        end,
```

- Scroll down a bit further and place the following lines after *self.input:open("fake_events")*. The exact placement shouldn't matter as long as it's as long as it’s within the function Kobo:init(), and placed between logic blocks in that function –

```
local success1, event4res = pcall(function()
self.input:open("/dev/input/event4")
end)
```

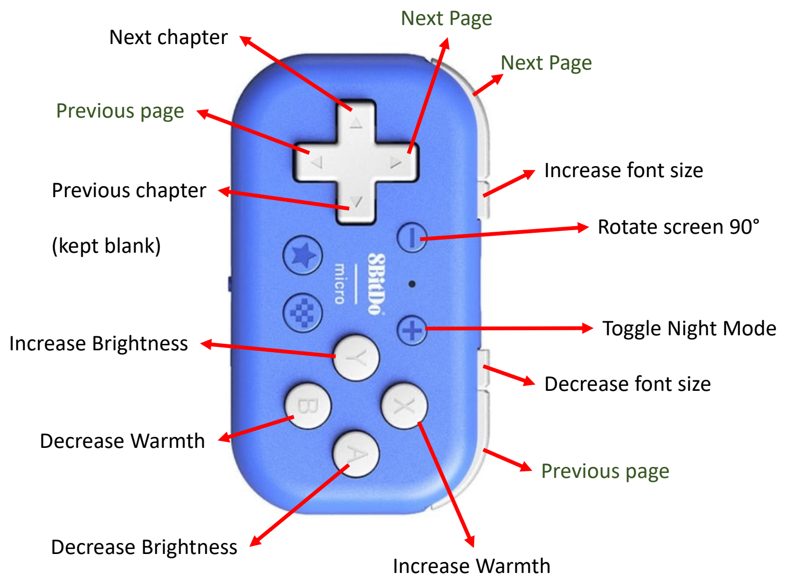
- Screenshot for reference –
    - 

```
end

-- NOTE: On devices with a gyro, there may be a dedicated input device outputting the raw accelerometer data
--       (3-Axis Orientation/Motion Detection).
--       We skip it because we don't need it (synthetic rotation change events are sent to the main ntx input device),
--       and it's usually *extremely* verbose, so it'd just be a waste of processing power.
-- fake_events is only used for usb plug & charge events so far (generated via uevent, c.f., input/iput-kobo.h in base).
-- NOTE: usb hotplug event is also available in /tmp/nickel-hardware-status (... but only when Nickel is running ;p)
self.input:open("fake_events")

local success1, event4res = pcall(function()
self.input:open("/dev/input/event4")
end)

-- See if the device supports key repeat
-- This is *not* behind a hasKeys check, because we mainly use it to stop SleepCover chatter,
-- and sleep covers are available on a number of devices without keys ;).
self:getKeyRepeat()
if not self:canKeyRepeat() then
    -- NOP unsupported methods
    self.disableKeyRepeat = NOP
    self.restoreKeyRepeat = NOP
    self.toggleKeyRepeat  = NOP
end

-- Detect the NTX charging LED sysfs knob
```

- Save and close the file

**Step 6 – Wrap up**

- You are now done. Eject the Kobo ereader from your computer.
- Reboot just to be safe.
- Since this method does not rely on any plug-ins, you will need to connect the controller before opening KOReader. And again, till the KOReader bug mentioned earlier is fixed, remember to reboot the device to exit KOReader.
- Here is the final config I’ve set up, just for reference.



**Additional Notes :**

- If you don’t care about advanced features, and only care about basic page turn buttons, you can skip the entirety of Step 4. On Step 5, you only need to enter the two lines for mapping “RPgFwd” and “RPgBack” and the “local success1…” lines.
- The Next Chapter/Previous Chapter functionality doesn’t work perfectly if you change the font-size after opening the book. I imagine this is a bug in KOReader, where chapter locations are not updated dynamically after changing the font size. If you close the book and open it again after changing font size, the chapter skips work perfectly again.

**Hope this helps! Best of luck!**