親指二軸ジョイスティックによる
ホームポジション維持型ポインティングデバイスの開発

- 何を作ったか?
  - Arduino Nano（USB-CDC/シリアル）を入力源とする
    Linux向けの自作ポインティングデバイスを開発した

  - ユーザーデーモンではなく、Linuxカーネルモジュールとして実装

  - input subsystem に仮想マウスを登録し、
    カーソル移動・スクロール・クリック・ズームを扱える設計

- リポジトリはここにあります。

https://github.com/mikio815/nanostick-kmod

入力がGUIに反映される流れ

①ハード

↓　（普通は）HID Report　今回は別

②デバドラ、ldiscなど（USB, HIDドライバ類）　⇦これ作りました

↓　input_report_rel(), input_report_key(), など..

③Input subsystem, evdev

↓　/dev/input/eventX に移動などの値を持ったイベントが飛ぶ

④libinput, compositor(Waylandの場合)

↓　イベントを読んでカーソルなどを更新

⑤GUI

Arduino NanoはCH340というUSBモジュールを介している

→標準的な入力のプロトコルとして求められるHID形式ではバイト列は送れない

→Arduino Nano に HID デバイスとして振る舞うUSB ポートを増設すれば、HIDとして出力して読むことはできる

しかし、普通にHIDを読んでPC側で解釈するだけでは実装に新規性がなくあまり面白くない

→HIDレポートではない従来のUSB-CDCの形式のまま受け取り、それをモジュールで独自の入力プロトコルとして解釈したい

何をする必要があるか？

・CH340から受け取った入力はカーネル内の
”ch341.ko”というドライバによって/dev/ttyUSBに流れてくる

・ttyUSBの内容をいい感じにパースして送ってくれるユーザーデーモンを書けば全て解決するが、つまらない

→どうにかして/dev/ttyUSBをカーネルから読んで全部やらせる

じゃあどう流すか

- ドライバ自体(ch341.ko)をラップして、/dev/ttyUSBだけではなくinput subsystemにも直接流すように改造してしまえばいいのでは？

→これは多分苦しいのでやりたくない

・ch341などからUSB-CDCを剥がす層だけ流用して、別で直接inputに流せるドライバ作ればいいのでは？

→発表の4日前に気づいた

実装量を少なくしたかったので、

ldisc(Line Discipline)を置き換え

てttyからinput subsystemへと渡す

ldisc(Line Discipline)とは何か

・ttyから受信したバイト列をどう解釈するかの中間層
→改行処理、エコーバックなど

・デフォルトはN_TTY
・CH340が投げてくるtty(/dev/ttyUSB?)のfdに対応するldiscだけ差し替えて、バイト列がinput subsystemにも流れるようにする

ldiscを採用するメリット

・既存のドライバ(ch341.koなど)を流用できる
→実装量が少ない
・ttyからinput_relに爆速で流せる

ユーザーデーモンはより簡単だがなんかカッコよくない
ch341の改造はなかなか辛い＋将来の更新で破滅する可能性がある

プロトコルの詳細

- フレーム固定16バイト・LE

バイト	長さ	フィールド名	サイズ	
0	2	magic	u16	マジックナンバー
2	2	seq	u16	パケットの順番管理
4	2	lx	s16	左スティックのX軸
6	2	ly	s16	左スティックのY軸
8	2	rx	s16	右スティックのX軸
10	2	ry	s16	右スティックのY軸
12	1	buttons	u8	左右ボタンの押されたフラグ
13	1	flags	u8	予約用
14	2	crc16	u16	CRC-16/CCITT-FALSEでエラー検出

ここで問題がある

ldiscの切り替えは基本的にユーザーランドから、対象のttyに対して

```
ioctl(fd, TIOCSETD, &ldisc_num);
```

を呼ぶ必要がある

fd：対象のtty,
TIOCSETD：ldiscを切り替えるコマンド
$ldisc_num：切り替え先のldiscのポインタ

- カーネルモジュールが書きたかっただけではあるので、

ldiscの切り替えもカーネルに任せるのは諦めてsystemdにやらせる

→ttyに新しいデバイスが刺さったときにそのfdをnanostickとして登録し、それを元にioctlを発行してldiscを切り替える

→udevルールをもとにCH340のみにデバイスを絞る

まとめ　作ったところ

・input subsystemに操作を流し込むまでの流れ

① プロトコルに沿ったバイト列を/dev/ttyUSBで受け取る

②ldiscを書き換えてttyに流れてきたバイト列を適当な構造体(ns_action)に詰め、値に応じた意味付けをしてinput subsystemに流す

③ns_actionを元にinput_report_rel()などのカーネルAPIを叩く

イベントが飛んできている様子

Event: time 1767801399.646170, type 1 (EV_KEY), code 272 (BTN_LEFT), value 1
Event: time 1767801399.646170, ---------- SYN_REPORT ----------
Event: time 1767801407.882230, type 1 (EV_KEY), code 272 (BTN_LEFT), value 0
Event: time 1767801407.882230, ---------- SYN_REPORT ----------
Event: time 1767804012.360843, type 2 (EV_REL), code 0 (REL_X), value -40
Event: time 1767804012.360843, ---------- SYN_REPORT ----------
Event: time 1767804012.575086, type 2 (EV_REL), code 0 (REL_X), value -30
Event: time 1767804012.575086, ---------- SYN_REPORT ----------
Event: time 1767804012.789080, type 2 (EV_REL), code 0 (REL_X), value -20
Event: time 1767804012.789080, ---------- SYN_REPORT ----------
Event: time 1767804013.004038, type 2 (EV_REL), code 0 (REL_X), value -10
Event: time 1767804013.004038, ---------- SYN_REPORT ----------
Event: time 1767804013.431825, type 2 (EV_REL), code 0 (REL_X), value 10