Verslag Practise Enterprise 2

Eneas Heynen – E-ICT EH

Introductie

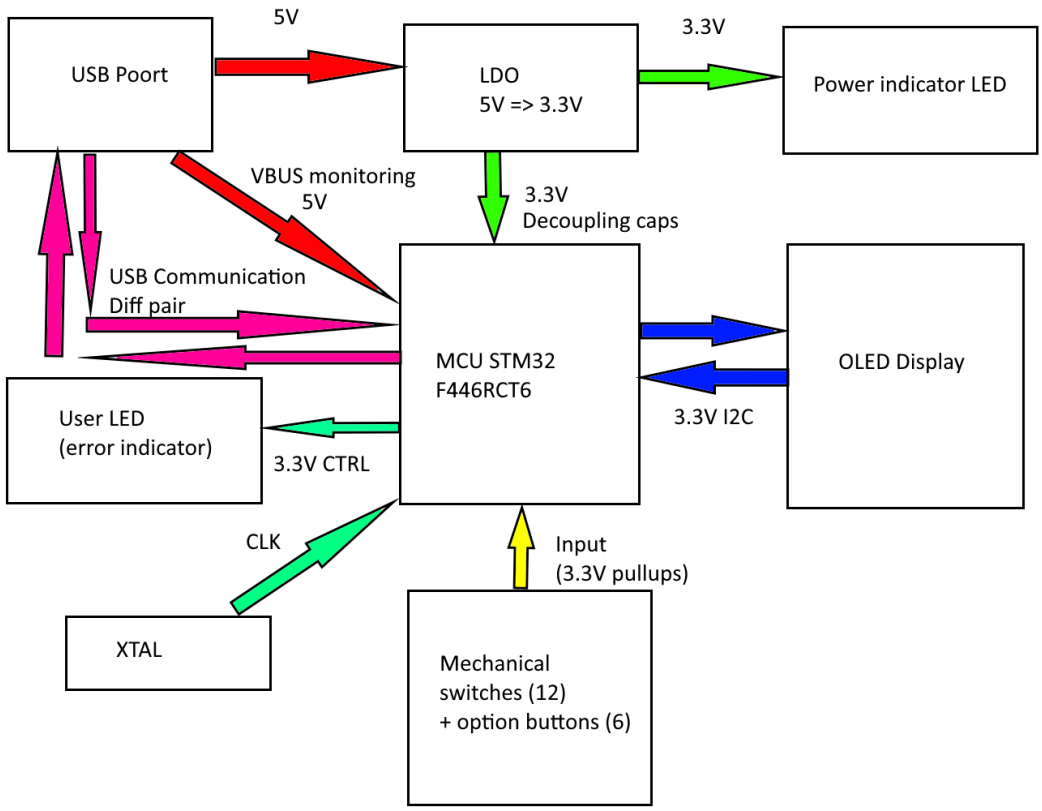
Ik heb gekozen om een “leverless” of “all-button” USB-gamecontroller te maken. In de competitieve fighting game scene (denk aan Street Fighter, Mortal Kombat, Tekken) was het voor lange tijd de standaard om op arcade-style controls te spelen (een grote joystick met grote tactiele knoppen). In de laatste jaren is er een shift gekomen naar controllers die volledig zonder joystick en enkel met de arcadeknoppen werken, o.a. om ergonomische redenen.

Ik speelde zelf ook op arcade joystick, maar had voor PE2 gekozen om de moderne trend te volgen, maar in plaats van arcadeknoppen te gebruiken koos ik voor mechanische keyboardswitches en een 3D-geprinte behuizing op basis van een opensourcebehuizing.

Een feature die op sommige controllers ook te vinden is is een OLED-display. Dit kan dienen om onmiddelijk defecte keyswitches op te sporen, of om aan toernooi-organizatoren te laten zien dat je controller aan bepaalde regels voldoet (geen rapidfire/turbo, links+rechts ingedrukt = neutral, ...).

1. Commercieel verkrijgbare leverless controller

Blokschema



Testen

Ik heb niet echt veel dingen moeten testen op mijn print. Ik heb getest:

- 5V van de USB

- 3.3V van de regulator

- De warmte van de regulator met een hittecamera (ten hoogste 56°C)

- Of de interne pullups werkten voor I2C en inlezen van knoppen

- ST-Link connectie

- Blinkyprogramma

- Blinky terwijl STM op XTAL loopt

- Bootswitch en resetknop

- Een enkele keyswitch inlezen en LED aansturen op basis daarvan

- Standaard USB-muis example van ST (klikken via drukknop)

- USB herkend laten worden als gamepad (USB descriptor)

- UART communicatie

- I2C communicatie

- I2C library voor scherm

Problemen en oplossingen

Mijn eerste probleem was Altium zelf. Na de zoveelste crash waarbij mijn projectdata corrupt werd kreeg ik een paniekaanval, en achteraf besloot ik om van nul te herbeginnen in KiCad. Na enkele dagen vloeken had ik al mijn componenten opnieuw geïnstalleerd en begon ik te wennen aan alle shortcuts, en aan de manier waarop de user interface in elkaar zit. Daarna ging de rest van het ontwerp relatief vlot (buiten enkele footprints die schijnbaar incorrect leken op basis van de datasheet – deze heb ik in de footprint editor aangepast).

Pin layout van de controller heb ik via de IOC tool in de CubeIDE gekozen, in plaats van door de datasheet constant op en neer te moeten scrollen. Dit maakte de PCB-layout veel makkelijker.

Ik heb mijn print aangepast om van de assembly bij JLCPCB gebruik te kunnen maken, en mijn componenten vervangen met degene die zij bij LCSC in stock hadden. De footprint van mijn USB-poort was incorrect (cutout stond op de verkeerde layer en was dus niet uitgesneden), dus ik moest nog op zoek naar een vervanger, die natuurlijk op de footprint paste én bij LCSC in stock was. Gelukkig kon ik er nog een vinden waar er nog juist 2 van in stock waren (minimum afleverhoeveelheid bij JLC assembly is 2 bestukte prints). Ook was het soms onhandig om door de datasheets van LCSC te scrollen omdat sommige in het Chinees (Kantonees? Mandarijns?) waren.

Door de plotse vervanging van de USB-poort moest ik ook de STL files van mijn case aanpassen, dit deed ik in freecad. Ik had eerder de files al aangepast om plaats uit te snijden voor de componenten op de print, zoals de boot- en resetcontrols, en de headers.

Ik koos voor de STM32F446RCT6 simpelweg omdat ik er vervangstukken voor heb liggen, dat mocht ik een controller opblazen ik op z'n minst er een nieuwe op kon zetten zonder nog bij digikey of dergelijke te bestellen. Achteraf gezien denk ik dat dit project ook kan draaien op een F1 controller, misschien eventueel met lagere USB pollrate (een van mijn persoonlijke doelen was een 1000Hz USB polling rate te behalen voor minimale latency en best mogelijke same-frame performance – dat de input op dezelfde frame wordt ingelezen in de PC als op het beeldscherm vertoond wordt). Om dit te kunnen doen heb ik de I2C snelheid best ver opgekrikt.

Als voorbeeld voor de controllerfunctionaliteit baseerde ik mij op Eazyjoy, een project voor een STM blue pill bord. Dit heeft echter geen functionaliteit voor een DPAD, of in USB termen een point-of-view directional hat. Dit is technisch gezien de properste manier om digitale richtingsdata door te sturen, en daarom heb ik de USB descriptor aangepast zodat ik geen analoge axissen moest faken.

De DPAD is wel licht raar opgebouwd, maar in mijn implementatie komt het erop neer dat een getal tussen 0 en 7 de richting aangeeft, met 0 recht naar boven, dan met de klok mee

tot 7 (=linksboven). Dit maakt de logica voor SOCD cleaning (Simultaneous Opposite Cardinal Directions – een toernooiregel die bepaald hoe de controller moet reageren op de inputs Links+Rechts en Boven+Onder) wel iets slordiger. Er zal wel een relatief elegante methode voor bestaan, maar ik kon deze niet vinden, dus er zullen wel enkele if-statements uit mijn code weggeoptimalizeerd kunnen worden. Het impact de performance echter niet – mijn USB-error lampje blijft uit (dus 1000Hz pollrate is gehaald).

Voor het aansturen van de display wilde ik gebruik maken van de reeds ingelezen pinnen (het opnieuw inlezen van de pinnen leek mij bad practice, omdat ik deze vlak voor de DrawAll functie net binnengelezen zijn, en pinnen inlezen mogelijk te veel tijd in beslag nam). Dit had als voordeel dat ik de SOCD-cleaning niet opnieuw moest doen. Voor de DPAD leek een simpel switch-statement optimaal. Voor het tekenen van de actieknoppen (een holle cirkel voor niet ingedrukt, een volle cirkel voor wel ingedrukt) vond ik een methode met functiepointers via conditional statements, omdat ze dezelfde argumenten namen.

De displaylayout had ik op voorhand in paint dot net ontworpen, op een 64x128 pixel grid.

Conclusie

Dit project heeft me vooral getoond dat een PCB ontwerpen geen pijnlijke bedoening moet zijn, waarbij elke stap van het proces probeert tegen te werken. Blijkbaar kwamen de meeste van mijn problemen van Altium, en de manier waarop de user interface opgebouwd is. KiCad spoort gewoon veel beter met hoe ik intuitief wil handelen.

Verder verzuip ik veel minder in de datasheets, en weet ik beter waar ik naar moet zoeken om iets functioneel te maken. Ik kan makkelijker het kaf van het koren scheiden zeg maar.

Het 3D-ontwerp van de case heb ik gelukkig grotendeels niet zelf moeten doen, maar dit is de eerste keer dat ik zelf in contact kom met de hele CAD wereld. De extrusions maken in FreeCad (eerst een sketch maken, constraints definen, en dan via trial en error er een stuk uitsnijden, ...) was wel een moeilijk proces, ik denk dat andere software gebruiksvriendelijker is.

Bijlagen

- Berekening XTAL: $Load\ Capacitance\ 20pF = \frac{33pF*33pF}{33pF+33pF} + Stray\ Capacitance\ 3.5pF$

- GitHub: https://github.com/ene-h/PE2/tree/master

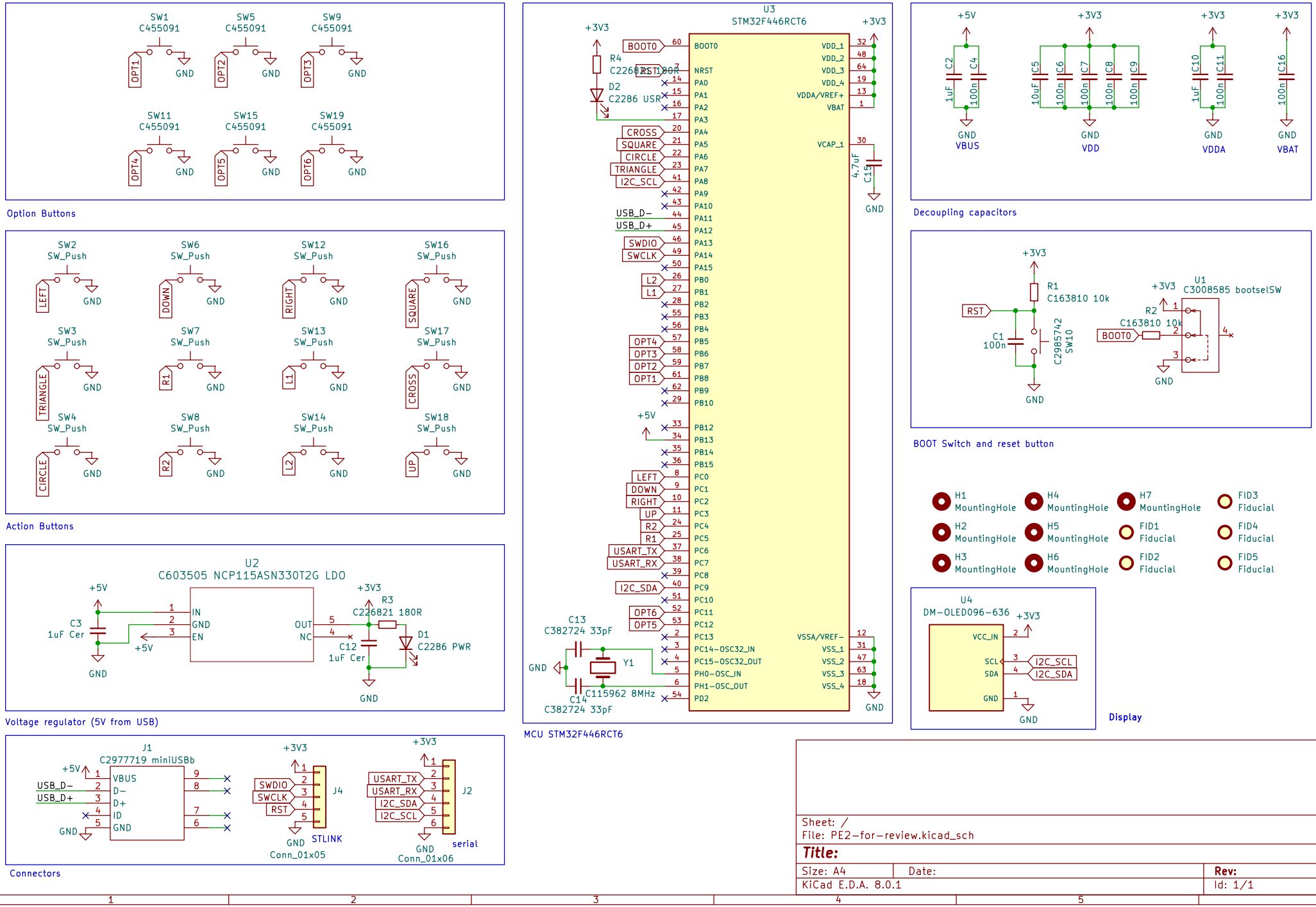
- Demos: Demo in game

Demo in windows control panel + OLED Display

- Schema's en datasheets: https://github.com/ene-h/PE2/tree/master/Misc

- USB Descriptor:

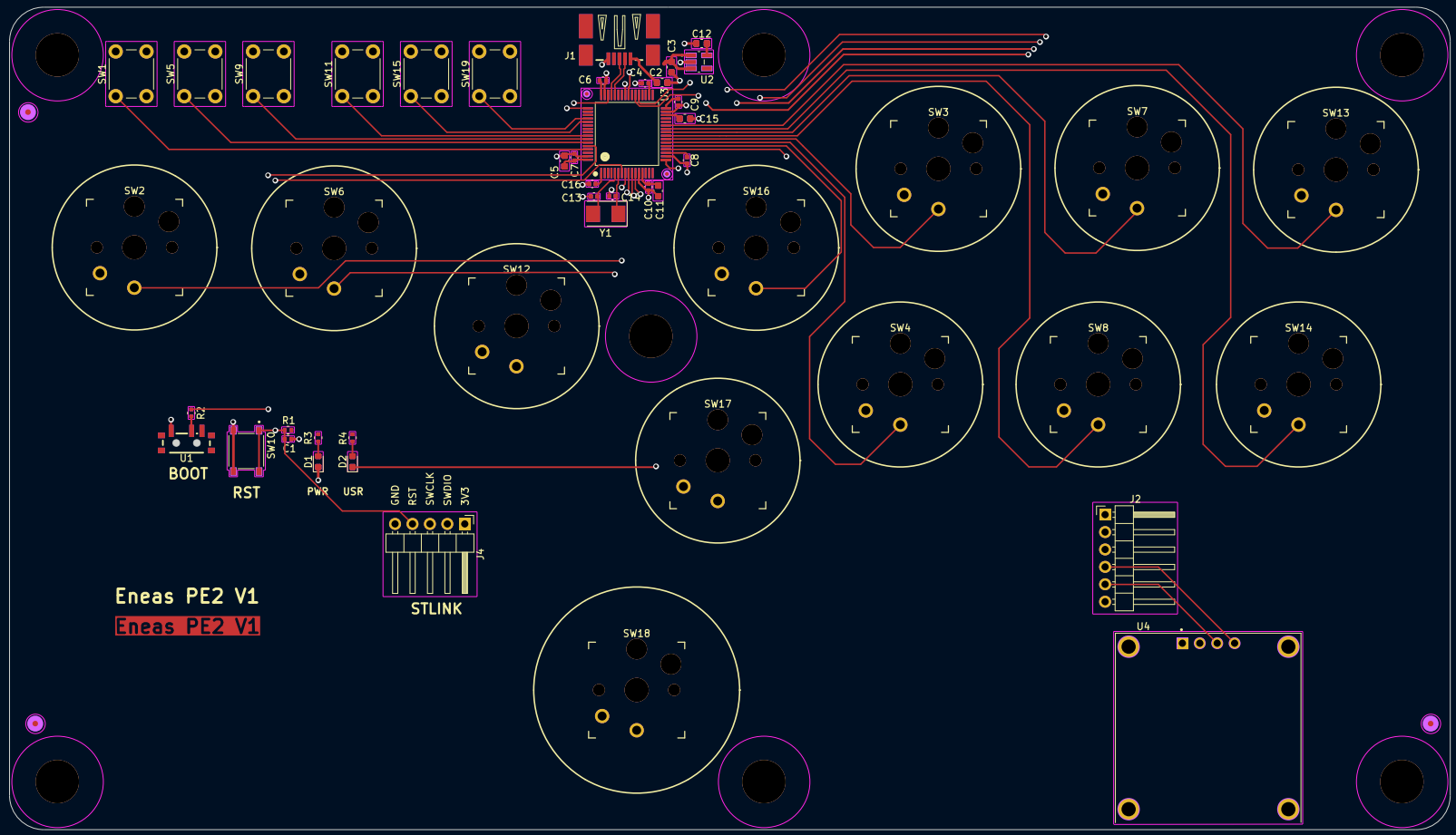
```
{
    0x05, 0x01,                    // USAGE_PAGE (Generic Desktop)
    0x09, 0x04,                    // USAGE (Joystick)
    0xa1, 0x01,                    // COLLECTION (Application)

#if (BUTTONS_NUM > 0)
    0x05, 0x09,                    //   USAGE_PAGE (Button)
    0x19, 0x01,                    //   USAGE_MINIMUM (Button 1)
    0x29, BUTTONS_NUM,             //   USAGE_MAXIMUM (Button x)
    0x15, 0x00,                    //   LOGICAL_MINIMUM (0)
    0x25, 0x01,                    //   LOGICAL_MAXIMUM (1)
    0x75, 0x01,                    //   REPORT_SIZE (1)
    0x95, 0x10,                                        //   REPORT_COUNT (16)
    0x81, 0x02,                    //   INPUT (Data,Var,Abs)
#endif //     (BUTTONS_NUM > 0)
//DPAD goes here, size = 18
       0x05, 0x01,                    // USAGE_PAGE (Generic Desktop)
       0x09, 0x39,                    // USAGE (Hat switch)
       0x15, 0x00,                    // LOGICAL_MINIMUM (0)
       0x25, 0x07,                    // LOGICAL_MAXIMUM (7)
       0x75, 0x08,                    // REPORT_SIZE (8)
       0x95, 0x01,                    // REPORT_COUNT (1)
       0x65, 0x14,                    // UNIT (Eng Rot:Angular Pos)
       0x55, 0x00,                    // UNIT_EXPONENT (0)
       0x81, 0x42,                    // INPUT (Data,Var,Abs,Null)
       0xc0,                          // END_COLLECTION
}
```



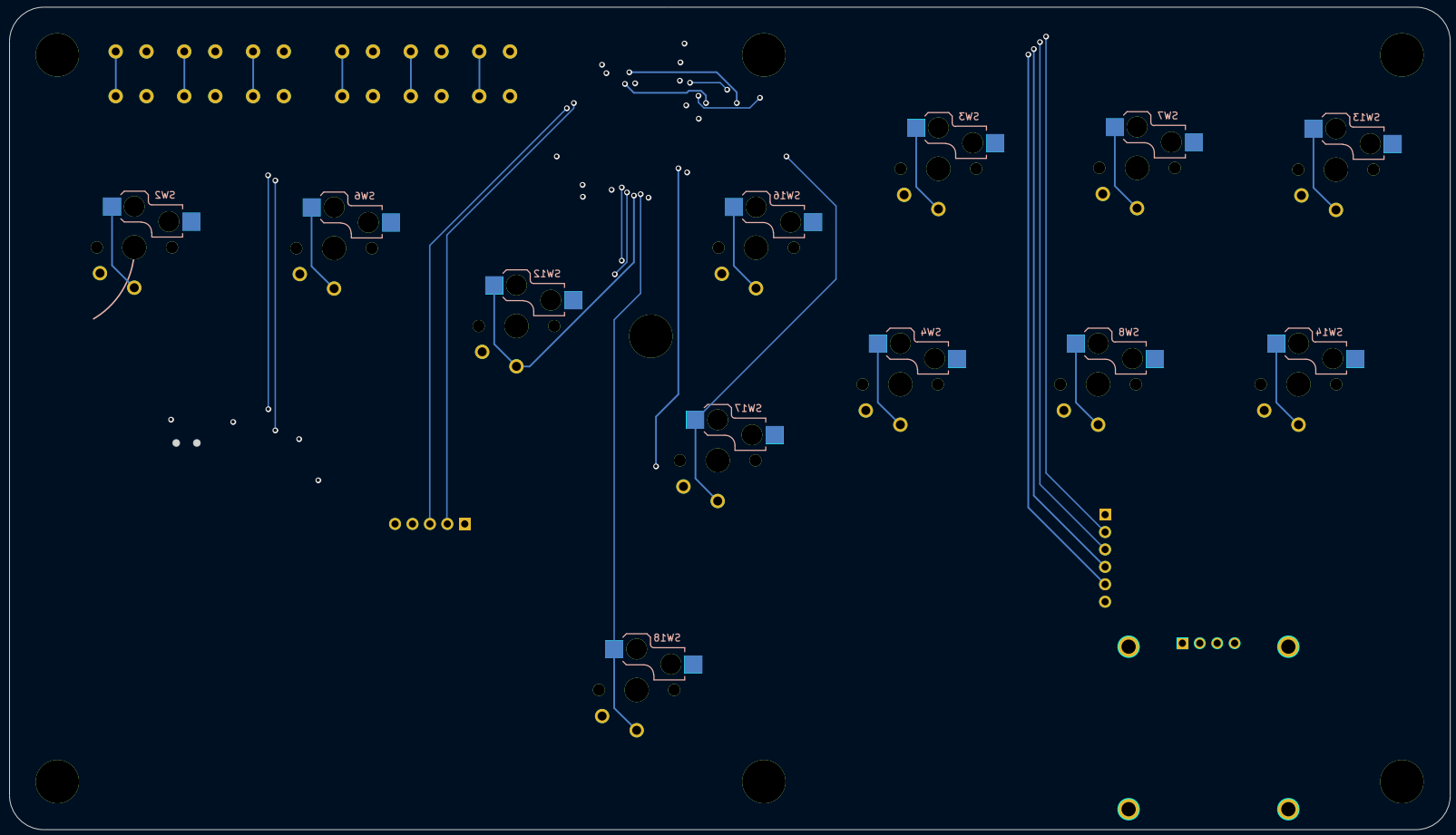
Sheet: /
File: PE2-for-review.kicad_sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 1/1



Sheet:		
File: PE2-for-review.kicad_pcb		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 1/1



Sheet:		
File: PE2-for-review.kicad_pcb		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 1/1