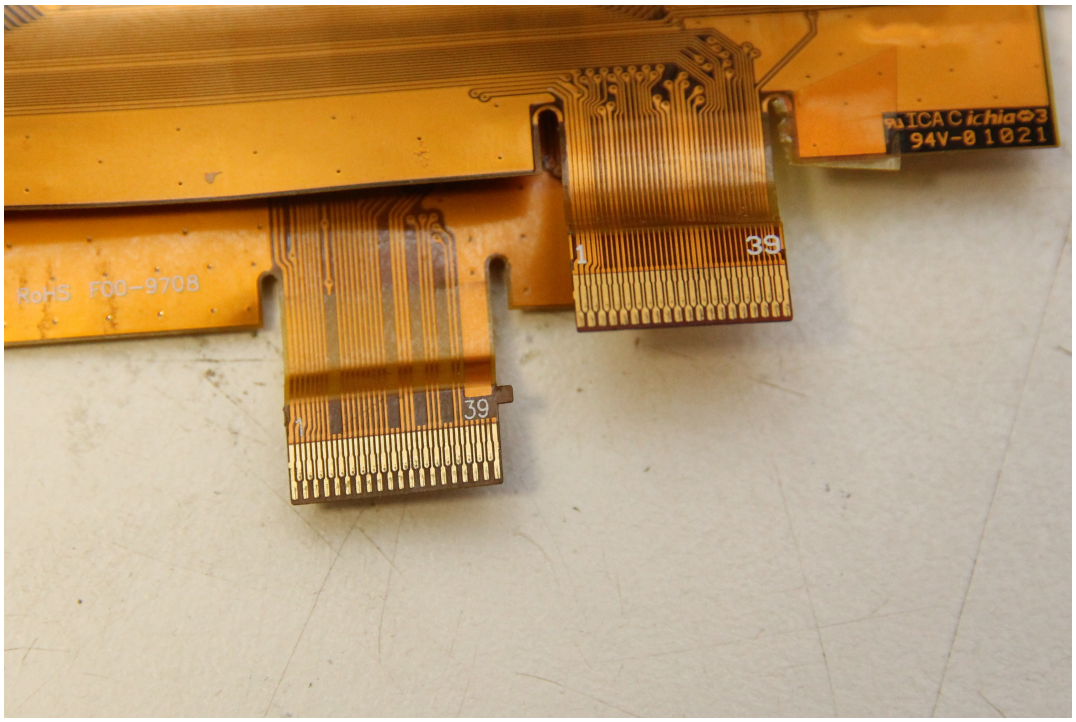
... najlepší je záver

Pre riadenie E-Ink („electronic ink“) displeja je potrebné zabezpečiť neštandardné napätia a digitálne signály s dodržaným časovaním.
Na rozdiel od TFT-displejov, E-ink displeje nemajú špecifikovanú minimálnu obnovovaciu frekvenciu - na ich riadenie je teda možné použiť aj pomalšie mikrokontroléry.

Dostal sa mi do rúk E-Ink displej z čítačky elektronických kníh KINDLE s rozlíšením 800x600 px. V tejto čítačke sa používa viacero verzií tohoto displeja.
Jeho hodnota sa na Ebay-i pohybuje okolo 20€.



img src: http://essentialscrap.com

Z obrázku je možné vidieť rozdiely medzi konektormi jednotlivých verzií.
Konektor vpravo prislúcha k verzii LF, vľavo bez.
Rozdiel spočíva hlavne v chýbajúcich vývodoch, t.j. napätiach, ktoré je nutné priviesť na tieto vynechané vývody.

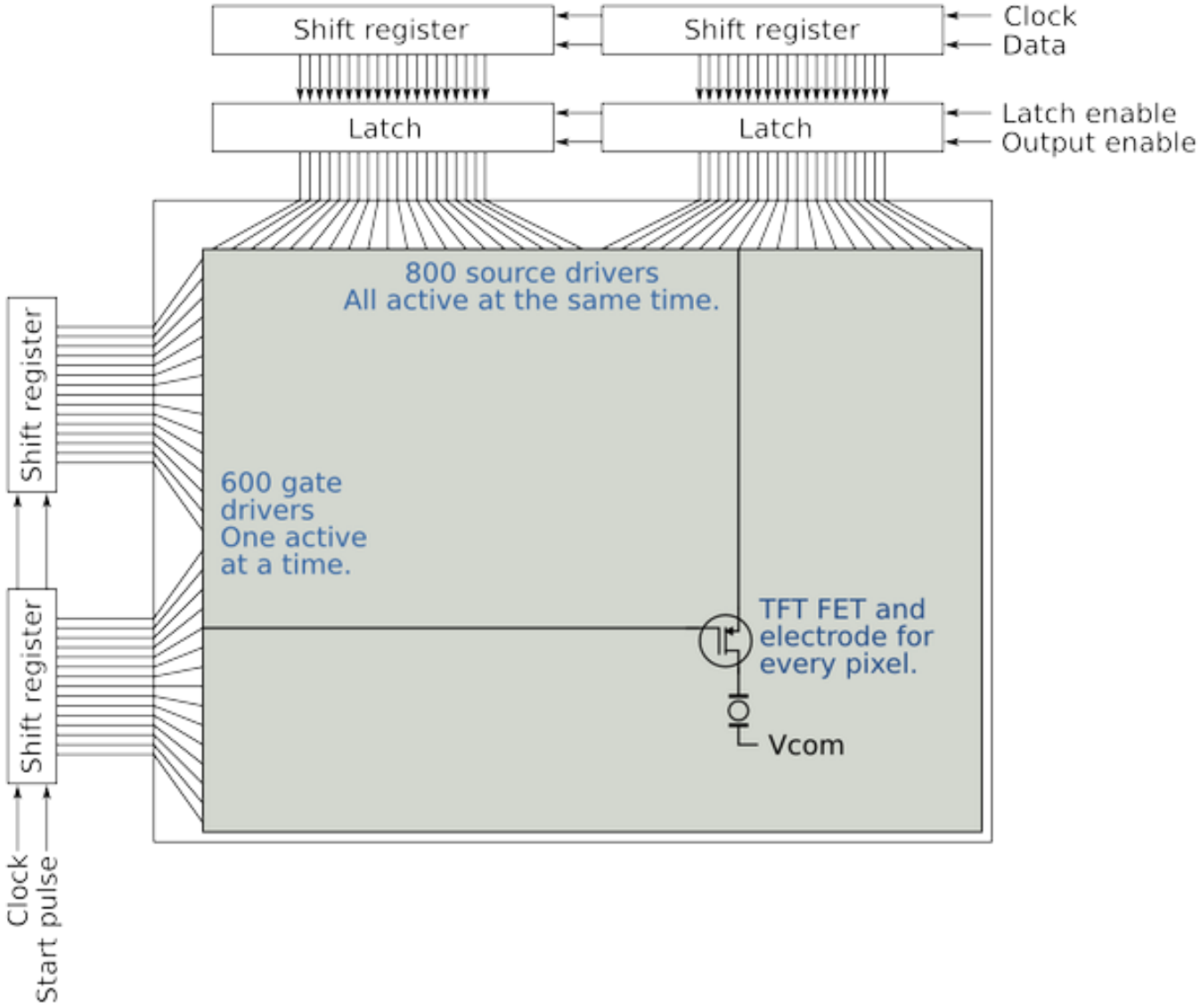
Tento displej mi prišiel ako fajn zobrazovadlo k môjmu zámeru - zobrazovať graf teploty a hodín v miestnosti.

Ale späť k displeju.

Princíp činnosti
Displej obsahuje jednoduchú internú logiku a to posuvný register na voľbu riadku a zobrazovaných hodnôt (pixelov).
Register pracuje s 2-bitovým kódovaním farieb. Čierna 0b01 a biela 0b10, pričom ostatné hodnoty sú ignorované. Áno, ani hodnoty 0xFF a 0x00 nemenia obsah.
Postupným posúvaním je takto možné prepísať celý displej.

Ak sa pozrieme do dátového listu displeja, všimneme si hlavne napätia, ktoré je potrebné pripojiť: +22V, +15V, -20V a -15V.
Okrem nich displej potrebuje aj 3.3V pre napájanie logiky a Vcommon napätie -3V až -1V pre nastavenie kontrastu. Hodnota je väčšinou napísaná na nálepke na pripojenom flex kábli.
V mojom prípade išlo o hodnotu -1.46V.

V realite hodnoty napätí nie sú až také dramatické. Optoelektrický materiál pozostáva z veľmi malých „atramentových“ častí. Ak je privedené dostatočne veľké napätie, posunú sa k/od neho a práve na to slúži spomínaných +/-15V. Jednotlivé časti elektród sú prepojené cez veľmi tenké FET tranzistory. Napätia +22V a -20V slúžia na ich riadenie.
Naopak, tranzistory na riadenie TFT technológie sú „ideálne“ a vyžadujú pomerne vysoké napätia na kompletné prepnutie. V konečnom dôsledku je toto napätie väčšie ako napätie potrebné na prepnutie FET tranzistorov.

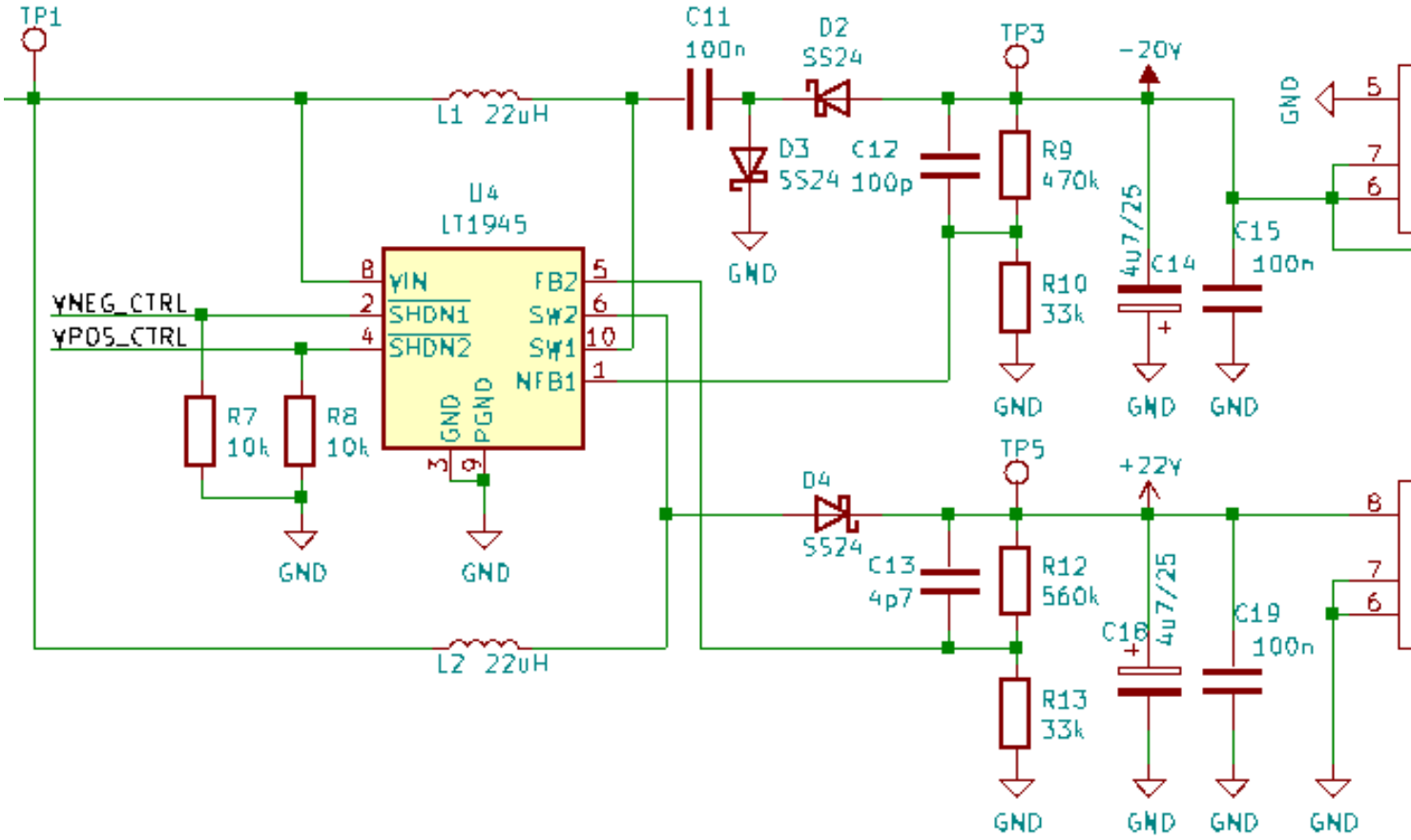


Samozrejme existujú aj iné spôsoby. Netreba sa báť pohľadať.

Zdroj napätí

Generovanie týchto neštandartných napätí sa môže zdať problematické, ale ja som poľahky našiel 2 obvody, ktoré to za pomoci okolitej bižutérie pomerne jednoducho dokážu.

Jedným z nich je LT3463 a druhým je LT1945. Jedná sa o dvojité boost DCDC meniče. Ja som použil LT1945, pretože som ho mal v šuplíku

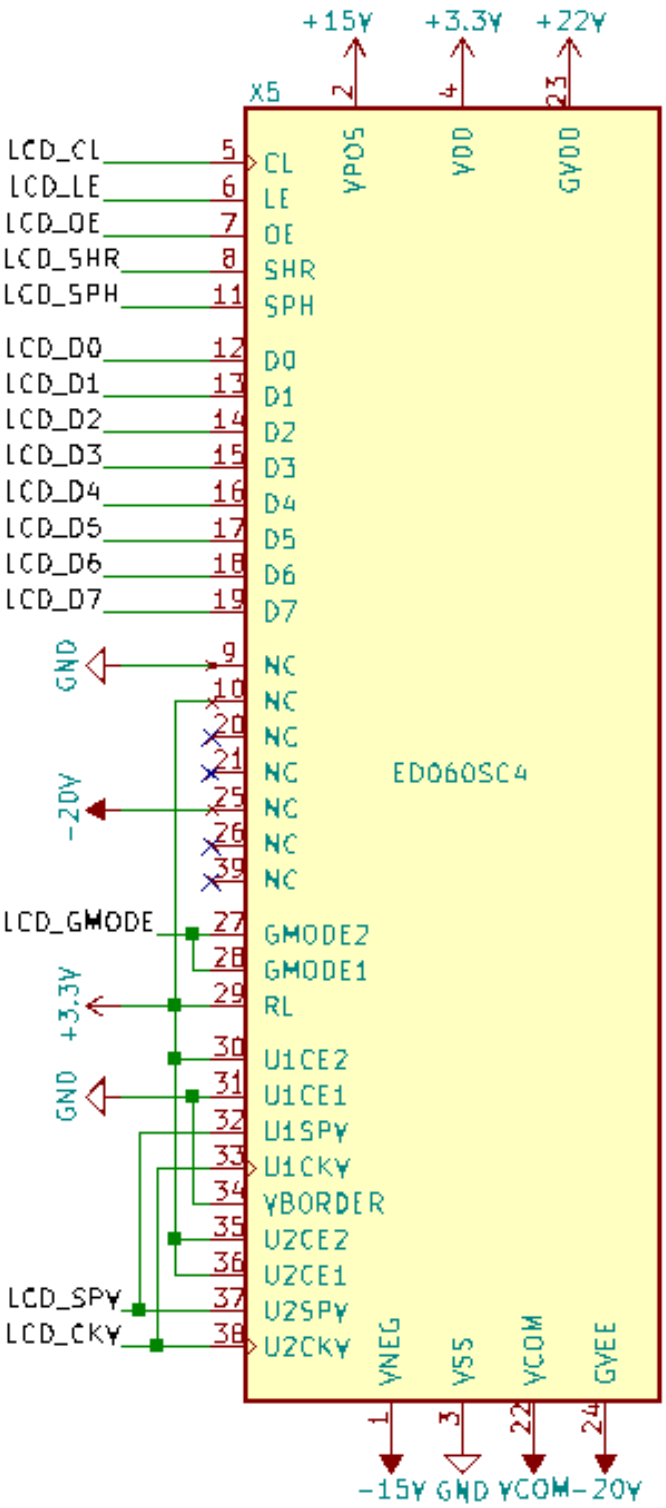


Na vstup meniča (vývod č. 8) je privedené napájacie napätie 3.7V (na napájanie som použil Li-ion batériu), ktoré sa v meniči naboostuje na +22V a zároveň druhý výstup meniča dáva potrebných -20V. Hodnota napätia je daná spätnovazobným deličom R12/R13, resp. R9/R10. Nasleduje filter v podobe kapacít C16 a C14.
Tieto napätia sú ďalej zastabilizované na +15V a -15V pomocou stabilizátor LM7815 resp. LM7915, a opäť mierne vyfiltrované kapacitami C21 a C17. Pre istotu som pridal aj zaťažovací rezistor R14 a R11.
Celá spotreba „meniča“ sa pohybuje ~50mA.

Výhodou LT1945 je možnosť ovládať jednotlivé výstupy tohto meniča pomocou signálov SHDN1 a SHDN2.
Toto ovládanie je dôležité, pretože **pozitívne napätie by malo byť privedené skôr ako negatívne. To má vplyv na korektné zobrazenie obrazca.**
Signály SHDN 1 a 2 sú pomocou Pull-Down rezistorov potiahnuté ku GND, aby po pripojení zdroja neaktivovali výstup. Napätie VCOM (Vcommon) je tvorené odporovým deličom - pre jednoduchosť trimrom.

Digitálna časť

Použitý E-Ink displej má osadených niekoľko čipov po obvode na hranách. Ako som už spomínal ide o source/gate driver-y, ktoré slúžia na ovládanie jednotlivých pixelov displeja.



Ich ovládanie je pomocou signálov:
Gate drivery: U1CE1, U1CE2, U2CE1, U2CE2, U1SPV, U2SPV, U1CKV, U2CKV, GMODE1, GMODE2 a RL
Source drivery: CL, LE, OE, SPH, SHR, D0-D7

Činnosť gate drivera je trochu záhadná. Veľké množstvo signálov umožňuje rôznymi spôsobmi spojiť polovice displeja, no nikde nie je popísané ako na to.
Takže som bol odkázaný na to, čo som mal k dispozícii.
Spojením U1SPV&U2SPV, U1CKV&U2CKV a GMODE1&GMODE2 som dosiahol postupné zapisovanie pixelov riadok po riadku z jednej polovice do druhej.

Oproti tomu, je source driver o niečo jednoduchší.
K činnosti potrebuje individuálne taktovacie CLK signály, ovládací signál pre preklápací obvod (Latch), ovládanie výstupného

buffera a štart/stop signál.
Zjednodušene - ide o 2-bitový 800B register s preklápacím obvodom. Jeden riadok je takto zapísaný do spomínaného posuvného registra a následne pomocou preklopenia do výstupného buffera je zobrazený pod sklom.
Čo sa týka SHR a RL signálov, tieto slúžia na zmenu smeru zápisu.

Riadenie displeja

Na riadenie som použil 32-bitový mikrokontrolér (MCU) PIC32MX575F512H, pretože obsahuje až 64 KB RAM pamäte, ktorú som potreboval ako video RAM.
Počítal som, že displej s rozlíšením 800x600 pixelov potrebuje 480 000 bitov na plné zobrazenie 1:1, čo je pomerne veľa.
Tak som hodnotu podelil 800/100 a vzniklo z toho rozlíšenie 600x100, čo je 60 KB.
Pri použití spomínaného MCU, mi zostali ešte 4 KB k dobru  .

Samozrejme, rozlíšenie je možné podeliť aj iným spôsobom a tak dosiahnúť menšie obsadenie pamäte a tým pádom aj menší/jednoduchší/lacnejší mikrokontrolér.

O riadení som si čo-to prečítal z dostupných informácií z internetu, no nie všetky spomínané ovládané displeje boli kompatibilné s tým mojim.
Dočítal som sa napríklad, že autor Petteri Aimonen, od ktorého som hlavne čerpal inšpiráciu, vychádzal z funkčného zapojenia čítačky a následným reverzným inžinierstvom pomocou LA (logického analyzátora) namapoval potrebné signály.

To, ale nebol môj prípad, nakoľko som nemal k dispozícii funkčnú čítačku. Tým pádom som bol odkázaný len na teoretické informácie a dúfal, že to bude aj fungovať.

Na stránke autora http://essentialscrap.com/eink/waveforms.html som si naštudoval postup.
No hneď prvý krok bol neúspešný.

Náhodné generovanie obsahu s priamym zápisom na dátové/signálové linky nefungovalo (aké nečakané).
Na základe tohto som usúdil, že niečo je zle... Upriamil som svoju pozornosť na source drivery, nakoľko ich riadenie je jednoduchšie.

Po niekoľkých chvíľach (konkrétne 2 dňoch) som došiel na problémy a vyriešil ich.
Prvý problém som odstránil tým, že som našiel správnu sekvenciu riadenia.
Druhý, no veľmi zásadný, spočíval v tom, že displej je nutné najprv „vymazať“ a „vykontrastovať“.

Pri riadení je potrebné dodržať určité časy, čo bol jeden z hlavných problémov.
Ak som aj zavolal funkciu __delay_us, tj. čakaj určitý počet mikrosekúnd (napr. 1us), čas bol rádovo väčší. Časy pre milisekundy pomocou funkcie __delay_ms sedeli vcelku fajn. Ani časy generované pomocou časovača nedodávali požadovaný výsledok.
Nakoniec som to vyriešil pomocou osciloskopu a starej-dobrej „nop“ inštrukcie (ja viem - tfuj!).

Fungujúci popis k displeju som nakoniec našiel na stránke https://www.circuitvalley.com.

Celý zápis do displeja, som implementoval 3 sekvenciami - štart, zápis a stop. Toto opakujem pre každý riadok (600x).

Jednoduchý program potom vyzerá nasledovne:

```
1.
   static void EINK_delay(uint8_t              us) {
2.
      while(us--) {
3.
         __asm("nop");
4.
         __asm("nop");
```

```
5.        __asm("nop");
6.        __asm("nop");
7.    }
8.  }
9.
10. static void EINK_vClk(void) {
11.    LCD_CKV=0x00;
12.    EINK_delay(1);
13.    LCD_CKV=0x01;
14.    EINK_delay(1);
15. }
16.
17. static void EINK_hClk(void) {
18.    LCD_CL=0x01;
19.    LCD_CL=0x00;
20. }
21.
22. static void EINK_hClks(uint16_t            unit) {
23.    while(unit--)            EINK_hClk();
24. }
25.
```

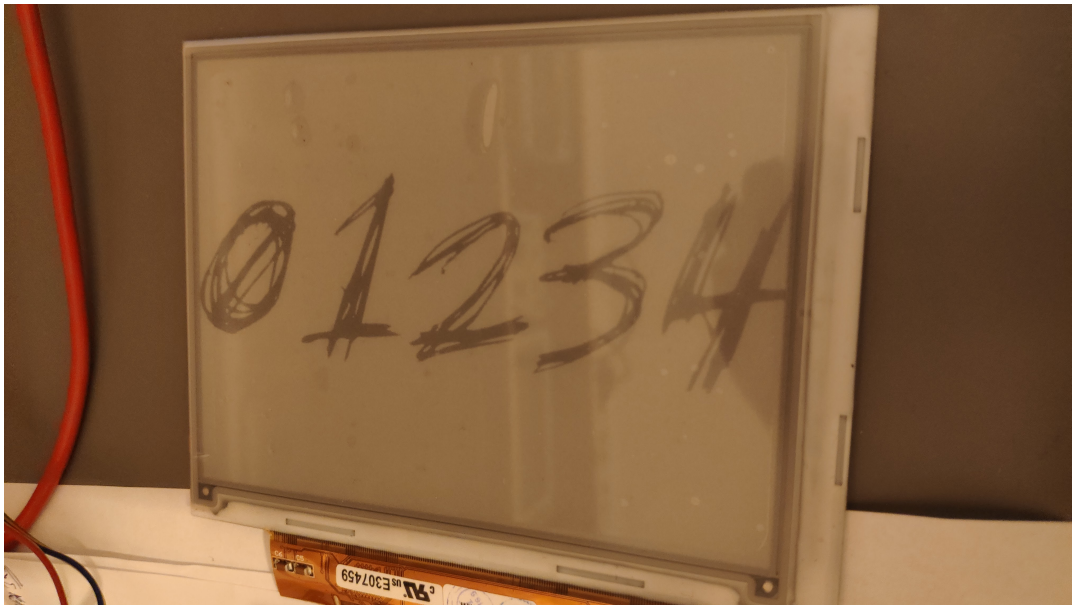
```
26. void EINK_vScanStart(void)          {
27.     LCD_GMODE=0x01;
28.     LCD_CKV=0x01;
29.     LCD_OE=0x01;
30.     LCD_LE=0x00;
31.     EINK_delay(100);
32.     LCD_SPV=0x01;
33.     EINK_delay(100);
34.
35.     LCD_SPV=0x00;
36.     EINK_delay(1);
37.     LCD_CKV=0x00;
38.     EINK_delay(25);
39.     LCD_CKV=0x01;
40.     EINK_delay(1);
41.     LCD_SPV=0x01;
42.     EINK_delay(25);
43.
44.     EINK_vClk();
45.     EINK_delay(25);
```

```
46.     EINK_vClk();
47.     EINK_delay(25);
48.     EINK_vClk();
49.   }
50.
51.   void EINK_vScanStop(void)           {
52.     EINK_vClk();
53.     EINK_vClk();
54.     EINK_vClk();
55.     EINK_vClk();
56.     EINK_vClk();
57.     EINK_vClk();
58.     EINK_vClk();
59.     EINK_vClk();
60.     LCD_CKV=0x00;
61.     EINK_delay(50);
62.     LCD_CKV=0x01;
63.     EINK_delay(10);
64.   }
65.
```

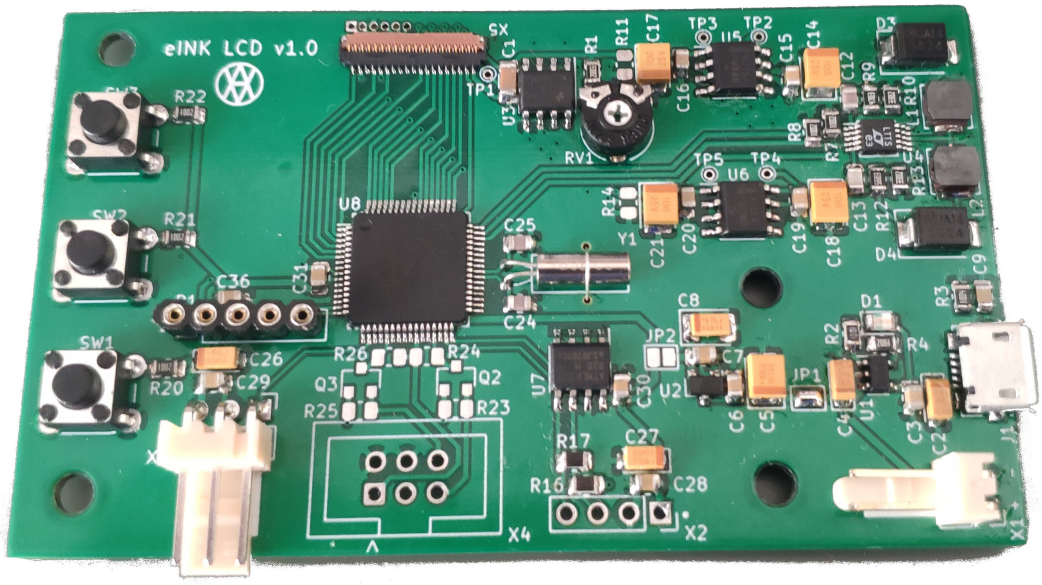
```
66. void EINK_hScanStart(void)          {
67.     EINK_hClk();
68.     EINK_hClk();
69.     LCD_SPH=0x00;
70. }
71.
72. void EINK_vScanWrite(void)          {
73.     LCD_CL=0x01;
74.     LCD_CKV=0x00;
75.     LCD_CL=0x00;
76.
77.     //LCD_OE=0x00; //not happning
78.     EINK_delay(1);
79.     // LCD_OE=0x01; //not happning
80.     LCD_CKV=0x01;
81.     EINK_delay(1);
82.
83.     LCD_LE=0x01;
84.     __asm("nop");
85.     LCD_LE=0x00;
86.
```

```
    EINK_delay(1);
87. }
88.
89.
90.
91. void EINK_clrscr(unsigned char             bw)
92. {
93.   for (uint8_t i=0x00;i<0x05;i++) {
94.     EINK_vScanStart();
95.     for (uint16_t j=0x00;j<LCD_HEIGHT;j++) {
96.       EINK_hScanStart();
97.       LCD_DATA_PORT=bw?BYTE_WHITE:BYTE_BLACK;
98.       EINK_hClks(LCD_WIDTH/0x04);
99.       EINK_hScanStop();
100.      EINK_vScanWrite();
101.    }
102.    EINK_vScanStop();
103.  }
104. }
```

Samozrejme toto bol len fungujúci nástrel a optimalizácii sa medze nekladú. Mne hlavne šlo o to, aby som rozbehal komunikáciu, čo sa mi nakoniec aj podarilo.



Displej som riadil z vlastnej elektroniky. Tejto:



A pohromade to bolo takto nejak:

Ten červený káblik je vyvedená zem pre meracie prístroje.
A tie dva biele na MCU sú vylepšenia, ktoré budú zahrnuté do ďalšej verzie .

Pár tipov nakoniec

- Samotný FFC kábel je veľmi chúlostivý a je nutné k nemu pristupovať opatrne.

- Pre kvalitnejšie zobrazenie je vhodné najprv displej „vyfarbiť“ načierno a následne bielou farbou vyfarbiť časť, ktorá nemá byť zobrazená.

- Po malých úpravách je možné dosiahnuť pomerne pekné zobrazenie aj priamych fontov či obrázkov. Naneštastie sa mi nepodarilo projekt dokončiť, nakoľko sa mi z nezistených príčin podarilo displej poškodiť. Teda spraviť aj poriadne merania s kompletnými priebehmi.