Lector de llibres electrònics simple

Daniel Ferrer Garrido

Resum– Aquest document descriu el desenvolupament hardware i software d'un lector de llibres electrònics simple. El dispositiu es basa en un microcontrolador ARM i una pantalla electroforètica, és portable i es monta a una carcassa impresa en 3D. Pot llegir documents markdown des d'una SD a través d'un explorador d'arxius i mostrar el seu contingut. Tot no ser actualment compatible amb altres formats d'arxiu, pot servir de base per a futures extensions.

Paraules clau– Circuits digitals, microcontroladors, pantalles, freeRTOS, ereaders.

Abstract– This document describes the development of a simple ebook reader, both software and hardware-wise. The device is based on an ARM microcontroller and an electrophoretic display, is portable and fits in a 3D printed case. It can read markdown documents from an SD card through a file explorer and show its content. Although its currently limited file compaibility, it could be extended to support other formats in the future.

Keywords– Digital circuits, microcontrollers, displays, freeRTOS, ereaders.

1 Introducció

Els lectors de llibres electrònics sorgeixen a mitjans de la dècada dels 2000, fruit del desenvolupament i abaratiment de les pantalles de tinta electrònica[1], que ofereixen un baix consum energètic i un aspecte similar al paper al no emetre llum, fent-les idònies per a la lectura.

Aquestes característiques, juntament amb el relatiu baix cost, han permès als eReadrs sobreviure a l'auge de les tablets i els smartphones, amb nous models cada any i amb avenços recents com la incorporació de pantalles electroforètiques a tot color[2].

En aquest projecte s'explora la construcció des de zero d'un lector simple i de baix cost, des del disseny del prototip fins a la implementació del programari. S'utilitza un maquinari menys potent que a un disseny comercial, simplificant el desenvolupament.

Més enllà de l'abast d'aquest treball, s'espera que el disseny i la implementació fets durant el projecte, serveixin com a base per a futures ampliacions, sobretot pel que fa a la compatibilitat amb formats de fitxers, que permetin un us diari més còmode del dispositiu.

- E-mail de contacte: Daniel.FerrerGa@e-campus.uab.cat
- Menció realitzada: Enginyeria de Computadors
- Treball tutoritzat per: Màrius Montón Macián
- Curs 2020/2021

1.1 Antecedents

Des de fa un temps han sorgit alguns projectes similars, com The Open Book[3] o Inkplate[4], que han servit com a inspiració tot i les diferències de plantejament sobre els formats d'arxiu i el disseny dels dispositius. Altres projectes, com KOReader[5] o Plato[6], se centren en la part software, i requereixen d'un dispositiu comercial (amb kernel Linux) per a funcionar. En aquest projecte es busca una implementació més simple i eficient, per a ser executada en un microcontrolador.

Aquests projectes reben suport per part de la comunitat, que ha implementat aplicacions interessants sobre ells, des de la lectura de text fins a reports meteorològics, calendaris o calculadores. En aquest treball es busca facilitar això, amb una implementació modular, ben documentada i oberta.

Han resultat especialment utils per a la realització del projecte l'entrada al blog de Petteri Aimonen "Driving E-Ink display"[7], per l'explicació del funcionament i el control de les pantalles, i els projectes espeink i NekoCal, que han servit com a referència per al controlador de pantalla. Pel que fa a la renderització del text, s'han trobat diversos treballs rellevants com font-to-c de Roger Dahl[8].

2 Objectius

Degut al limitat temps de desenvolupament, els objectius del projecte se centraven en la construcció d'un prototip

funcional bàsic. Tot i que en un dispositiu comercial s'usaria un SoC Linux amb memòria RAM en l'ordre dels centenars de MB[9], això no és assolible en aquest projecte per la complexitat que suposa el desenvolupament de drivers i el disseny de circuits.

En canvi, s'utilitza un microcontrolador amb un sistema operatiu en temps real. Això permet un consum energètic més baix però restringeix les capacitats, fent impossibles algunes extensions com la renderització d'arxius PDF. Altres formats, com markdown, HTML o fins i tot epub, sí que es podrien implementar amb una compatibilitat prou bona ja que, en separar contingut d'estil, donen un major marge a simplificacions.

El projecte, però, se centra en markdown, un llenguatge de marques fàcil d'escriure i interpretar que permet la creació de publicacions digitals simples, suportant diversos nivells de títols, ressaltats, llistes i imatges entre altres coses. A més, tot i que això no s'ha tractat en aquest treball, és extensible mitjançant HTML incrustat[10].

Tot i les limitacions, markdown és un format prou complet per a la majoria de llibres, que es poden convertir a aquest format amb eines com calibre[11].

Els objectius principals del projecte es podrien resumir, doncs, en els següents:

1. Controlar una pantalla electroforètica usant un microcontrolador.
    (a) Implementació d'escala de grisos amb 4bpp de profunditat.
2. Lectura de fitxers markdown des d'una SD.
3. Disseny portàtil, amb unes dimensions tan ajustades com sigui possible i alimentat a partir d'una bateria recarregable d'ions de liti.
4. Renderitzat
    (a) Text pla.
    (b) Text amb format (títols, cursiva...).
    (c) Imatges
    (d) Altres elements (llistes, separadors, citacions)
5. Interfície d'usuari bàsica:
    (a) Explorador d'arxius.
    (b) Gestió d'historial.
    (c) Índex dels documents.

D'aquests objectius eren crítics el control de la pantalla, la lectura de fitxers markdown des de la SD i la renderització de text. Després d'aquests, l'escala de grisos, el text amb format, les imatges i l'explorador d'arxius es consideraven els més importants.

Durant el desenvolupament s'han tingut també en compte altres ampliacions que es poguessin fer més endavant, tractant que el disseny sigui compatible amb aquestes. Alguns exemples són la reproducció d'arxius de música, la lectura de documents epub, o la descàrrega de contingut via WiFi o Bluetooth.

3 METODOLOGIA

Com que una part important del projecte consisteix en la implementació del hardware, i no és un àmbit en què tingués massa experiència, a la primera part del projecte s'ha dissenyat un prototip modular, per tal de facilitar el testeig i les correccions o canvis de disseny que hi podien haver.

Durant aquesta primera fase, a més dels diferents circuits, s'han desenvolupat els drivers de dispositiu i el codi base, incloent el sistema operatiu FreeRTOS[12] i un bootloader USB. Com a base s'utilitza libopencm3[13], una biblioteca de dispositiu lliure per a microcontroladors basats en ARM, que proveeix una API per a les famílies més comuns i la majoria dels perifèrics. Tot i que la biblioteca és incompleta, és suficient per a aquest projecte. En comparació a la biblioteca oficial de STmicro, és més propera al HW real, i més portable a altres famílies de dispositius. Personalment la trobo més intuïtiva, i més conforme al manual de referència del micro.

Un cop provat tot el maquinari i desenvolupat el codi base, s'ha dissenyat un segon prototip unint les diferents parts. Aquest prototip inclou també un mòdul bluetooth i un DAC d'audio per I2S, de cara a ampliacions.

La idea inicial era desenvolupar tot el programari d'aplicació sobre aquest prototip. No obstant, una lesió al colze m'ha impedit manipular el prototip durant unes setmanes. Per tal de poder seguir amb el projecte, doncs, s'ha implementat un simulador del dispositiu, sobre el que s'ha desenvolupat la resta del codi del projecte. Finalment el codi ha sigut provat directament al dispositiu, comprovant que tant el simulador com la resta del codi funcionen correctament.

Pel que fa a la gestió de versions del codi s'utilitza git, amb el repositori allotjat a gitlab.com[14]. L'estructura del codi es divideix en tres directoris: *drivers*, que inclou tots els controladors de dispositius implementats durant el projecte; *src*, que inclou la part d'aplicació del projecte, i *extern*, que recull les biblioteques externes usades pel projecte. El codi inclou comentaris amb format doxygen[15] per tal de generar una documentació exhaustiva dels drivers i els diversos mòduls que conformen el projecte.

4 MAQUINARI

4.1 Pantalla

El component més important del dispositiu és la pantalla electroforètica, i avui dia hi ha múltiples models a l'abast. El principal problema d'aquestes pantalles és l'alimentació, ja que requereixen de diverses tensions positives i negatives, habitualment entre 22V i -20V[7].

Algunes pantalles incorporen conversors DC-DC, que s'encarreguen de generar aquestes tensions, però són força més costoses[16]. Una alternativa més assequible són les pantalles de recanvi per a dispositius comercials que, malgrat la baixa disponibilitat i la necessitat d'un conversor de tensió extern, són perfectes per a projectes d'aquest tipus, amb un cost d'entre 10 i 20€. Tot i que és difícil trobar documentació oficial, el seu funcionament es troba publicat a diverses pagines[7][17], fruit de l'enginyeria inversa d'altres usuaris.

La pantalla que s'ha escollit és el model ED060SC4[18], de 6" i 600x800 píxels. Es controla mitjançant dos registres de desplaçament integrats (vertical i horitzontal)[19][20] i 8 línies de dades (Fig.1)[7].

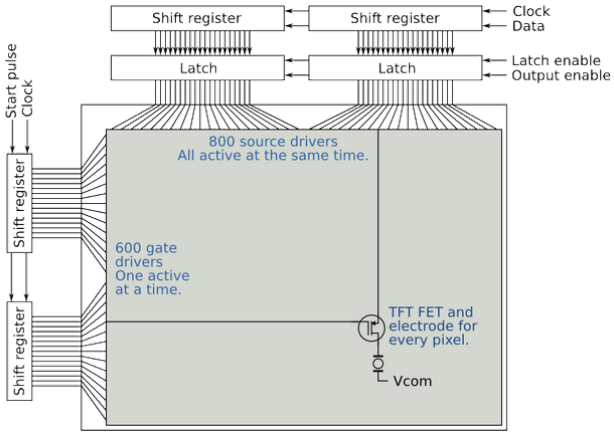


Fig. 1: Funcionament de la pantalla

4.2 Alimentació

La pantalla requereix de 6 entrades d'alimentació diferents: 22V, 15V, 3.3V, -15V, -20V i VCOM, que varia entre -0.5V i -2.5V depenent de la pantalla. El consum màxim d'aquestes alimentacions és de 36mA, o 0.54W[18].

Per tal de generar aquestes alimentacions es va seleccionar en un principi l'integrat TPS65186, un PMIC (circuit integrat de gestió d'alimentació) específic per a pantalles de tinta electrònica[21]. Aquest PMIC, però, no va arribar a ser estable, ja que de tant en tant s'aturava amb una interrupció, deixant de funcionar durant hores. Finalment es va escollir un PMIC més simple, l'LT3463, que va funcionar sense cap problema. Incorpora dos senyals de control per habilitar les tensions negatives i positives, que genera mitjançant dos conversors en commutació (tipus boost).

4.3 Microcontrolador

Com que, degut a restriccions de cost i temps, es va descartar usar un SoC Linux, la principal preocupació en quant al microcontrolador a usar era la capacitat de memòria, tant SRAM com flash. Tot i que no ha de desar-se el mapa de bits de la pantalla a memòria, ja que es pot renderitzar a cada escriptura, sí que cal emmagatzemar els elements a renderitzar, com els caràcters d'una pàgina i totes les estructures de dadesseves de paraula, línia o paràgraf. Pel que fa a la flash, ha de contenir els bitmaps de tots els caràcters a usar en els diferents tamanys i fonts.

Una altra restricció important era el nombre d'entrades i sortides. Tenint en compte tots els perifèrics i possibles ampliacions es necessitaven més de 40 entrades i sortides. A més, hauria d'incloure SDIO, USB i I2S, per poder usar àudio en futures expansions.

Altres aspectes que s'han tingut en compte són la disponibilitat de biblioteques, documentació, recursos, la velocitat del micro, la comunitat darrere de cada família de microcontroladors i la experiència pròpia amb aquesta.

Amb tot, la família més adient que s'ha trobat és la stm32f4, i en concret el model stm32f427[22], amb 256KB de SRAM i 1MB de flash. Malgrat no ser l'opció més assequible, compleix amb tots els altres requisits.

4.4 Esquemàtic

El disseny dels circuits s'ha efectuat amb l'eina kiCad[23], tant pels esquemàtics com per les PCB. Encara que hi ha hagut canvis entre els prototips, S'ha mantingut la majoria del disseny inicial.

El circuit del PMIC s'ha dissenyat seguint les pautes del seu datasheet, i s'han seleccionat els components passius més propers als recomanats pel fabricant. Com que només genera dos alimentacions (22V i -20V), s'han de generar les altres alimentacions a partir d'aquesta, usant els reguladors lm79l15 i lm78l15 per als -15V i 15V respectivament i un amplificador operacional per a VCOM.

El circuit s'havia de poder alimentar íntegrament des d'una bateria de liti. Es va seleccionar una bateria que inclogués protecció contra curtcircuits, i per a la seva càrrega es fa servir l'integrat MCP73831, que carrega la bateria començant a una intensitat constant regulable fins arribar als 4.2V, i mantenint aquesta tensió fins a una càrrega completa, quan la bateria deixa de consumir.

A més, no convé usar la bateria com a font d'alimentació mentre aquesta es carrega, pel que s'ha d'implementar un circuit que commuti entre aquesta i l'USB, en funció de si està o no carregant. També cal rebaixar aquestes tensions a uns 3,3V constants, tenint en compte que la caiguda de tensió limita l'aprofitament de la bateria.

Per a la selecció de font entre bateria i USB s'ha fet servir un MOSFET canal-P[24], amb una resistència en actiu molt baixa, i per a la regulació de tensió s'ha usat l'integrat MIC5504. Entre els dos s'obté una caiguda de 0,2V, que permet descarregar la bateria fins a 3,5V que equivalen a un ús d'aproximadament el 80% de la capacitat total[25].

El connexionat entre la pantalla i el micro ve definit pel pinout descrit al datasheet de la pantalla, però és incomplet, pel que s'han usat com a referència el projecte d'essentialscrap[7]. Als annexos es pot trobar l'esquemàtic complet.

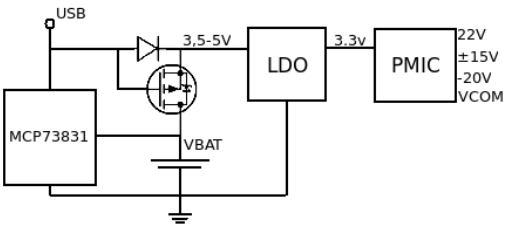


Fig. 2: Esquema d'alimentacions del segon prototip

4.5 Prototip

Pel primer prototip es van dissenyar 3 mòduls, un pel PMIC, un per a les alimentacions i un altre per a la pantalla. A finals d'abril, es va dissenyar i construir un altre prototip, amb l'idea que fos el prototip final del projecte. Es mantenia bona part de l'esquemàtic inicial introduint petites correccions. Per últim, a principis de juny es va haver de dissenyar i fabricar una altra placa, que seria la definitiva, incorporant el PMIC alternatiu. La placa incorpora circuiteria per al mòdul wireless i per al DAC d'audio per a futures expansions.

Per al disseny de la PCB s'ha tingut en compte la forma del dispositiu final, de forma que la placa cobreix tota l'àrea de la pantalla, deixant botons a les bores i amb un espai al mig per a la bateria. També s'han col·locat els components i traçat les connexions tractant de reduir al mínim les interferències, especialment al mòdul wireless però també a la SD, la pantalla i l'USB.

Per últim, s'ha dissenyat una carcassa en 3d amb FreeCAD i blender, i s'ha imprès en PLA, mitjançant FDM[26]. Un factor important és el gruix de la placa, d'1.6mm, que dona solidesa al conjunt.

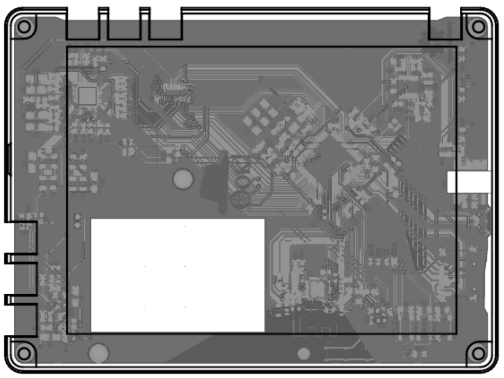


Fig. 3: Encaix entre placa i carcasa.

5 PROGRAMARI

Durant la primera fase del desenvolupament, amb el prototip inicial, es van desenvolupar els drivers de dispositiu, tot i que més endavant s'hi han efectuat canvis. La segona part ha consistit principalment en el desenvolupament d'un simulador i del codi d'aplicació.

5.1 Drivers i codi base

5.1.1 Pantalla

Com s'ha dit, la pantalla funciona internament amb dos registres de desplaçament, un per les línies i un altre per les columnes, i un registre de dades, que guarda tota una columna. El funcionament bàsic és el següent:

1. S'inicialitza el frame, posant el registre de desplaçament de columnes a zero.
2. S'inicialitza la columna, posant el registre de desplaçament de files a zero.
3. S'envien de 4 en 4 les dades a escriure, podent escriure cada píxel en blanc, en negre o deixant-lo com estigui.
4. S'escriu tota la columna d'una vegada.
5. Es torna al punt 2 mentre no acabem el frame.

Cada una d'aquestes accions es realitza generant en els senyals de control una forma d'ona predefinida, i algunes d'elles varien en funció del mètode d'escriptura (blanc i negre, grisos o esborrat). Quan volem escriure en escala de grisos, a més de variar les formes d'ona, hem d'escriure en diversos passos, escrivint un frame sencer per a cada tonalitat, i variant el temps d'escriptura en funció d'aquesta.

S'ha implementat un driver simple amb funcions per a esborrar i escriure la pantalla sencera o regions d'aquesta, tant en blanc i negre com en escala de grisos. En el segon prototip, però, no hi ha hagut temps d'ajustar els temps d'escriptura per a una bona gamma de grisos.

Més endavant es va usar aquest driver per a implementar un de més alt nivell d'abstracció, integrat amb el RTOS. Aquest driver s'executa en una tasca a part, que va rebent comandes i segments del bitmap de la pantalla mitjançant cues, imprimint-les a la pantalla. Això, a més d'oferir una interfície més simple al programari d'aplicació, permetrà renderitzar alhora que s'escriu per pantalla un cop s'implementi el driver amb timers i DMA, agilitzant tot el procés.

5.1.2 SD

El driver implementat ofereix funcions per a la inicialització del mòdul SDIO del microcontrolador i de la targeta SD, funcions de lectura i escriptura de blocs i de comprovació d'estat.

Aquest driver s'ha integrat amb FatFs[27], una biblioteca de sistema de fitxers compatible amb FAT32 i exFat. S'han implementat les funcions bàsiques, que permeten escriure i llegir blocs i comprovar l'estat de la SD, i faltarien funcions relacionades amb les marques de temps dels fitxers, ja que ara per ara no es disposa d'un driver del rellotge de temps real.

5.1.3 FreeRTOS

Tal com es va planificar, un cop implementats els drivers dels perifèrics es va afegir el kernel freeRTOS[12] al codi

i adaptar els drivers per tal que funcionessin correctament i de forma segura amb multitasking. A més es va implementar un bootloader i es va afegir suport per al sistema d'arxius de la SD, ja que freeRTOS només ofereix el kernel, a diferència d'altres sistemes més complets com nuttx o apache mynewt.

Es va escollir aquest sistema operatiu degut a la gran quantitat de documentació, recursos de suport i exemples que hi ha disponibles, a la simplicitat que ofereix en tractar-se només del kernel i a la independència que te respecte del dispositiu, els drivers i les llibreries usades.

Abans d'adaptar el codi existent a FreeRTOS es va generar una plantilla que compila i executa un exemple amb FreeRTOS i libopencm3. Realitzar aquesta plantilla no era trivial. A més de les múltiples variables de configuració que ofereix FreeRTOS, certes parts del codi s'havien d'adaptar a la llibreria libopencm3. Va ser problemàtic allotjar correctament les rutines de servei d'interrupció del sistema (systick entre altres) a la taula de vectors. S'ha optat per compilar FreeRTOS com una llibreria estàtica i incloure-la al linkador de l'aplicació, així que les rutines d'interrupció del SO s'han de cridar des d'una ISR definida al codi d'aplicació.

Per tal de reduir el consum energètic, s'apaga el nucli ARM quan no queda cap tasca en marxa, mitjançant una IDLE Hook de FreeRTOS. Aquí, es posa el micro en mode sleep amb la instrucció WFI, i torna a funcionar quan hi ha alguna interrupció.

A més de la gestió que faci el codi d'aquesta memòria, cal tenir present que en entorns encastats és perillós l'ús de memòria dinàmica. Es fa servir la implementació heap4 de FreeRTOS, que permet alliberar memòria, i la reserva seguint la norma best-fit. A més prevé la fragmentació combinant els espais alliberats adjacents, tot i que aquesta encara és possible que quedin petits fragments usats dispersos per la memòria si el codi no allibera correctament la memòria, com es veurà més endavant.

5.1.4 Bootloader

Pel que fa al bootloader s'ha optat per adaptar el codi d'exemple de DFU (Device Firmware Upgrade, estàndard USB[28]) de la lliberia libopencm3 al projecte. El codi d'exemple ens proporciona els descriptors de dispositiu, configuració, interfície i endpoints de USB per al bootloader, i s'ha implementat el control d'accés a aplicació i l'escriptura i esborrat de la flash. També s'ha implementat la part d'aplicació, que consisteix en una modificació dels descriptors anteriors que permet a l'ordinador reiniciar el dispositiu en mode DFU usant l'eina dfu-util. D'aquesta manera per programar el dispositiu no cal reiniciar manualment.

El micro usat té una petita regió de memòria que es manté entre resets sempre que es mantingui tensió a la línia Vbat. Són els registres de backup, i es poden usar per a la comunicació entre l'aplicació i el bootloader. Així, quan l'aplicació vulgui entrar en mode bootloader (perquè rep una petició des del PC), només ha d'escriure un valor concret en aquest registre i reiniciar. Quan el bootloader, que és sempre el primer que s'executa en arrencar, vegi aquest valor, sabrà que ha d'entrar en mode programació.

Quan aquest valor no hi sigui a l'arrencada, se saltarà a l'aplicació. Per fer-ho cal canviar l'adreça del vector d'interrupcions i reinicialitzar l'stack pointer.

5.2 Simulador

Tot i que no formava part de la planificació ni els objectius, s'ha implementar un simulador per tal de seguir amb el desenvolupament mentre no es podia fer servir el prototip, compilant i executant directament en Linux.

Està basat en la implementació per a POSIX de FreeRTOS, de manera que l'execució del codi hauria de ser la mateixa, incloent la gestió del heap. A més, per tal d'evitar diferències en el tamany de les dades per l'alineament que faci el compilador, es compila per a 32 bits (x86).

S'han implementat còpies dels drivers de dispositiu, que ofereixen la mateixa interfície però interaccionen amb el sistema operatiu del host. La pantalla i els botons s'implementen sobre GTK3 (Fig. 4), usant una tasca extra de FreeRTOS per al gtk_main(). Els segments de codi que accedeixen a la finestra estan protegits per locks per evitar fallades de segmentació.

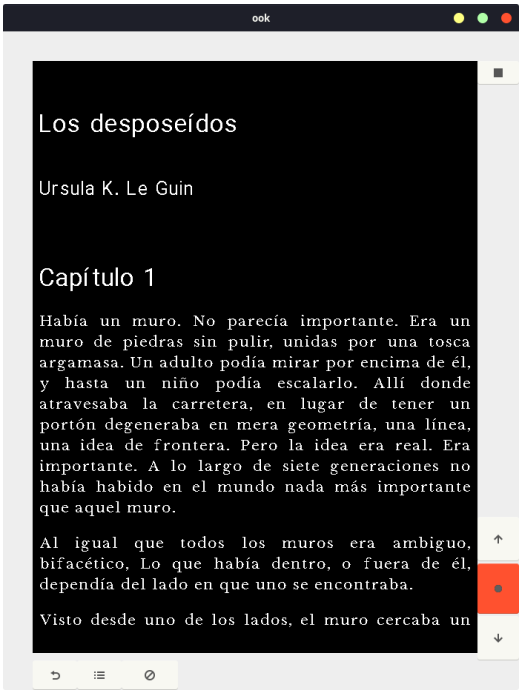


Fig. 4: Finestra del simulador.

Pel que fa al sistema d'arxius, s'han implementat una serie de macros i funcions que emulen la llibreria FatFS sobre el sistema d'arxius host. També s'ha integrat tot en un mateix makefile, de manera que per compilar per al simulador només s'ha d'executar *make pc*.

Per tot això, l'execució en PC i en el prototip hauria de ser la mateixa, tot i que sempre hi ha alguna diferència derivada de les diferències en el temps d'accés als drivers i al SO host. Aquestes diferències, no obstant, no haurien de notar-se si el codi de més alt nivell està ben implementat, sense fer assumpcions inadequades sobre els drivers.

5.3 Renderització

Per a la renderització s'ha usat C++, ja que un llenguatge orientat a objectes s'adapta molt més bé a aquest tipus de codi, on es requereixen múltiples estructures de dades similars però amb algunes diferències en contingut i gestió.

Com que C++ és gairebé un superconjunt estricte de C, es pot compilar el codi previ com a C++ sense massa problemes. S'han hagut d'introduir alguns canvis en la sintaxi de declaració d'estructures tipus *bit field*, però les biblioteques usades ja eren compatibles amb C++. Pel que fa al compilador, com que s'usa gcc només cal canviar a g++ i canviar alguns paràmetres de compilació. Per l'ús de memòria dinàmica s'han sobrecarregat els operadors *new* i *delete*, redirigint-los a les funcions del heap de FreeRTOS.

Per als elements de renderització s'han implementat diverses classes. La classe *element* serveix com a base per a tots els elements imprimibles, i se'n deriven les classes *bloc* i *segment*, per a elements que ocupin tot l'ample de la pantalla, com un paràgraf i elementqs que només n'ocupin una part, com una paraula. La classe *frame* serveix per gestionar i imprimir el conjunt d'elements que ocupen una pàgina.

La funció més important d'aquestes classes és render(), que renderitza una regió de l'element segons la regió de la pantalla que s'escrigui en aquell moment.

5.3.1 Text

S'ha pogut imprimir text per pantalla usant diferents fonts (text, títols, negreta, cursiva...). També s'ha aconseguit implementar kerning (separació dinàmica entre lletres) tant al generar la font com al renderitzar-la, i justificació al centre, a dreta i esquerre o emplenant la línia (Fig. 4).

Com que no és possible renderitzar fonts vectorials com TTF en un microcontrolador com aquest en un temps adequat, es converteixen en temps de compilació. Per a això s'utilitza un script en python que converteix fonts ttf a estructures de dades en C, incloent arrays amb els seus bitmaps en el format adequat per a la pantalla (2bpp) i les estructures de control, que contenen informació sobre el posicionament de cada lletra i el seu tamany entre altres. L'script llegeix un fitxer de configuració XML, en el qual s'especifiquen les fonts ttf d'origen, el tamany, les variants usades de cada font i l'àmbit en què s'utilitzarà (text, títols...).

Aquest script també calcula el kerning, que permet ajustar l'espai entre caràcters dinàmicament, depenent del seu contorn. Per a això es divideix l'alçada de les lletres en 4 segments, i per a cada un d'ells es desa el punt més proper al límit del bitmap (Fig. 5). Un cop guardats aquests valors, per a calcular la distància entre dos caràcters, el codi del lector només ha de comparar els límits dels dos caràcters, un a l'esquerra i l'altre a la dreta. El desplaçament del caràcter estarà per tant limitat pels píxels més propers entre els dos caràcters.

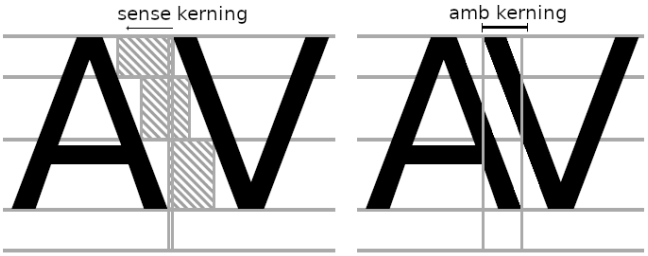


Fig. 5: Funcionament del kerning.

La implementació a la banda del micro utilitza 3 classes: paràgraf, línia i paraula. La gestió dels finals de línia i pàgina ha de tenir en compte el text que sobra i readaptar les dades d'aquest a una nova pàgina. S'han implementat les classes de tal manera que es puguin reaprofitar per a altres elements, com ara el text dels menús.

Totes 3 classes incorporen una funció addchar, que gestiona l'addició d'un nou caràcter a l'objecte i, quan calgui, el final d'aquest objecte i/o la creació d'un de nou. Com que tot això es fa amb memòria dinàmica, cal tenir present les possibles pèrdues. Un error que es va cometre va ser no declarar el destructor de element com a virtual, cosa que provocava que mai s'esborressin les dades de les classes heretades si el punter usat era del tipus de la classe base. Usant l'eina valgrind[29] es va poder trobar l'error i ara funciona com s'espera.

Pel que fa a la codificació del text, com que no es pot usar un gran nombre de caràcters degut a la limitació de memòria, s'ha restringit els caràcters usats a l'estàndard ISO-8859-15, que inclou l'alfabet llatí amb tots els accents i símbols d'ús comú a Europa. S'ha implementat també la conversió d'UTF-8 a aquest estàndard, i seria força senzill canviar a un altre estàndard per a altres llengües i regions. També hi ha espai per afegir fins a 32 símbols addicionals a les parts no imprimibles de l'estàndard.

5.3.2 Altres elements

També s'ha implementat una classe listEntry que permet imprimir per pantalla llistes. Pel que fa al text, només faltaria implementar la renderització de cites i codi, dins un requadre. També s'ha implementat la descodificació d'imatges jpg a bitmap, usant la llibreria picojpeg, però per falta de temps no s'ha pogut implementar l'escalat d'aquestes ni l'escriptura en escala de grisos a l'últim prototip. El driver de pantalla en escala de grisos va ser provat al prototip modular, imprimint bitmaps des de la memòria ROM del micro, pel que aquesta part no hauria de complicar-se.

5.4 Lectura de documents

S'ha començat la lectura de fitxers markdown, implementant de zero el parser. S'ha decidit fer així ja que és un format força senzill d'implementar, almenys els elements bàsics, i perquè més endavant s'haurà d'unir al parser d'HTML per als elements incrustats.

El parser s'ha implementat com una classe, i utilitza bàsicament les següents funcions:

- readPage(n): Llegeix (parseja) una pàgina. Ara per ara, quan es vol anar enrere s'ha de llegir tot el document de nou, ja que no es manté registre d'on comença cada pàgina.
- render(): renderitza la pàgina actual. S'ha de cridar després de readPage.
- parse_body(): Llegeix el document fins que troba algun element imprimible. Es retorna aquí cada cop que acaba l'element actiu.
- parse_paragraph(): Inicialment llegia només paràgrafs, però per no repetir codi pràcticament idèntic, també interpreta títols i llistes.
- start(): Bucle principal, on es van llegint pàgines a mesura que l'usuari les demana.

Els elements suportats fins ara són 6 nivells de títols (començats per #), els paràgrafs, la cursiva i negreta, usant tant asteriscs com guions baixos, i les llistes no ordenades, començades per un asterisc a cada entrada. Faltarien principalment els enllaços i les imatges.

La classe tal com s'ha implementat carrega una sola pagina cada cop, però en un futur s'ampliarà, de forma que es puguin llegir diverses pàgines abans d'imprimir-les per pantalla per avançar-se al pas de pàgines de l'usuari. Ara per ara, quan s'avança la pàgina simplement se segueix llegint l'arxiu, tenint en compte el tipus d'element (text, títol...) en que es troba. Quan es retrocedeix, en canvi, s'ha de tornar a llegir tot l'arxiu fins la pàgina demanada.

Pel que fa a HTML encara no s'ha començat, però s'implementaran els mateixos elements que a markdown usant com a base la llibreria yxml[30], que, enlloc de generar un DOM o usar callbacks, interpreta l'arxiu caràcter per caràcter, cosa que facilita la lectura element a element i el final de les pàgines. Els dos parsers seran força semblants, pel que bona part del codi es podrà heretar amb la classe.

5.5 Interfície d'usuari

S'ha implementat un sistema de menús d'usuari que serveix com a base per a un explorador d'arxius (Fig. 7)]. El menú permet afegir opcions fins a omplir la pàgina, de manera que els menús multi-pàgina s'han d'implementar a part. Això és així per evitar menús amb massa opcions que ocupin massa memòria. El bucle principal de la classe menú va llegint les pulsacions de tecles fins que se selecciona un ítem, es canvia de pàgina o es torna enrere, i retorna un valor diferent en cada cas.

Pel que fa a l'explorador d'arxius, llegeix un directori fins a omplir una pàgina del menú i espera. Si s'ha de canviar de pàgina, es torna a omplir amb les següents entrades. Si se selecciona una carpeta, es repeteix el procés, i si se selecciona un fitxer amb extensió ".md"s'obre un parser per a aquest arxiu.

5.5.1 Memòria cau

Hi ha diversos punts del projecte on es necessita fer ús d'una memòria cau que emmagatzemi dades a llarg termini sobre els documents llegits i l'ús del dispositiu. Un d'ells és l'historial de lectura, que permet reprendre un document per allà on es va deixar.

Per a aquest tipus de dades s'utilitzen taules en brut, que corresponen a estructures de C, guardades en fitxers de la SD. S'ha implementat un codi de control amb una interfície força simple, i una estructura base per a les entrades que pot servir per a qualsevol taula.

Cada element de la taula consta almenys de dos entrades: l'índex, de 32bits i una suma de comprovació de 16bits (el mateix que a IPv4). S'accedeix als elements pel seu índex i es comprova sempre el checksum, tot i que es retornen les dades també quan hi ha error. Per a l'escriptura es comprova primer si l'índex està present i, en aquest cas, l'actualitza. No hi ha camp de longitud, pel que totes les entrades d'una taula hauran de ser de la mateixa mida, especificada al generar el fitxer taula i al començar-ne la lectura.

Per a escriure o llegir dades només s'ha de generar una struct on els primers elements siguin l'índex i el checksum, i incorporant tants camps com calgui. De fet, hi ha una estructura base *dbdata* de la qual s'en pot derivar aquesta capçalera per a altres taules. L'índex pot ser qualsevol cosa, i en el cas de l'historial correspon a un hash (MurMur2) del nom del fitxer. Com que el hash és de 32bits és poc probable que hi hagi cap col·lisió. Aquest esquema, servirà també per a altres taules, com la d'imatges descodificades o arxius descoprimits. Per a taules més grans es podrien implementar, si calguessin, taules hash dividides en diversos fitxers basades en aquesta mateixa idea, per tal de millorar el temps d'accés.

6 RESULTATS

S'ha dissenyat un prototip amb tot el necessari per aquest projecte (objectiu 3.), incloent la gestió de la bateria, el control de la pantalla electroforètica i la seva alimentació, l'USB i la SD, i algun extra com el mòdul bluetooth o el DAC d'àudio, tot dins d'una carcassa còmoda impresa en 3D. El prototip final funciona correctament després del canvi de PMIC, i només falta provar les ampliacions del circuit, que no eren objectius d'aquest projecte.

Pel que fa al software s'ha pogut implementar controladors per als diferents perifèrics usats i per a la pantalla (objectiu 1.). Tal com hem vist, el dispositiu és capaç de

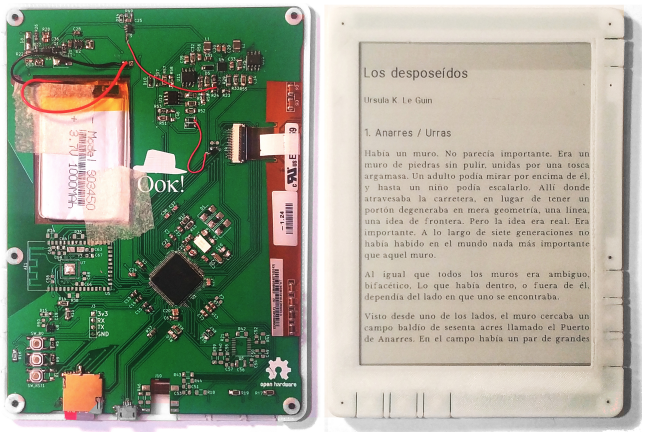


Fig. 6: Protorip en funcionament.

llegir documents markdown des d'una SD (2.) i mostrar-los per pantalla, podent renderitzar text marcat, amb títols i ressaltat, a més de llistes multinivell (4.). Tot i que falta la renderització d'imatges (4.c), aquests elements són suficients per a la lectura de la majoria de llibres literaris. També s'han implementat menús d'usuari que son la base per a un explorador d'arxius (5.). Per tant, el dispositiu és ja perfectament usable per a la lectura.

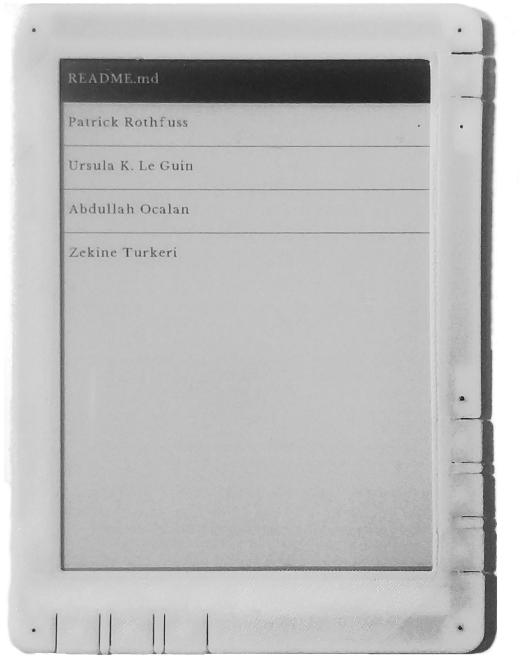


Fig. 7: Explorador d'arxius.

A diferència de projectes similars com The Open Book o Inkplate, aquest dispositiu és compatible amb un format estandarditzat (markdown), el codi desenvolupat fins ara servirà com a base per a formats més complexes, com HTML i, per extensió ePub, i no depèn d'un PC per a la renderització, com és el cas d'Inkplate amb el mòdul ePub. El dispositiu, a més, té un cost unitari molt ajustat, pel que es pot convertir en una alternativa completament oberta a aplicacions com KOreader.

També s'ha implementat un simulador que , tot i no estar previst, ha resultat molt útil, accelerant el desenvolupament i simplificant la depuració del codi. Juntament amb l'eina *valgrind*, ens permet trobar problemes en l'us de memòria dinàmica, cosa especialment important en un sistema amb una memòria tan limitada.

Una de les principals preocupacions al principi del projecte era l'ús de memòria, ja que només es disposa de 256KB. Actualment s'està usant entorn a un 10% d'això, a més d'un mínim de 5kB de memòria dinàmica per a freertos. Cada pàgina carregada a memòria ocupa entre 10 i 20kB, pel que es podrien carregar algunes pàgines abans que l'usuari ho demanés, per tal d'accelerar el canvi. També les fonts bitmap ocupen menys flash del que s'esperava, pel que es podrien incloure vàries fonts seleccionables (sans/serif) i/o un alfabet més estès per donar suport a més llengües.

Com que s'usa memòria dinàmica, es pot aprofitar aquest espai per altres tasques mentre no estigui obert un document, cosa que permetria, per exemple, descomprimir arxius epub o reproduir arxius de so. Per tant, no sembla que la memòria hagi de ser un impediment en desenvolupaments futurs, tot i que s'haurà de veure el rendiment amb altres formats d'arxiu.

Dels objectius marcats, s'han complert els següents:

- Controlar una pantalla electroforètica usant un microcontrolador.
- Lectura de fitxers markdown des d'una SD
- Disseny portàtil, amb una mida reduïda i alimentat des d'una bateria.
- Renderitzat de text pla i amb format.
- Explorador d'arxius
- Historial de lectura

6.1 Cost

Tot i no tractar de competir amb productes comercials, és important tenir en compte que el preu del dispositiu estigui dins d'uns límits raonables tenint en compte les seves característiques. A la següent taula es mostra un desglòs del cost aproximat de construir-ne un. El preu pot veure's afectat per l'actual escassetat de semiconductors a nivell global.

Placa de circuit imprès	3€
Microcontrolador	12€
PMIC i reguladors	7€
Bateria	5€
Pantalla	16€
Connectors	1€
Components Passius	6€
Altres components	6€
Carcassa	2€
Total	**53€**

A això, no obstant, caldria afegir el cost d'enviament dels components que, de voler construir un sol dispositiu,

pujaria uns 30€.

Pel que fa al cost de desenvolupament, es van preveure en un principi uns 270€ en total, però s'han acabat gastant aproximadament 350€. Això ha estat degut a la compra de més plaques de les previstes, a la ruptura de dues pantalles durant el desenvolupament i a la compra d'una pistola d'aire calent per a canviar el PMIC del segon prototip.

7 Conclusions

Com s'ha anat veient en aquest informe, els objectius principals del projecte s'han pogut assolir. Degut a alguns imprevistos i a una planificació massa optimista, no s'ha pogut acabar el control de pantalla en escala de grisos i el renderitzat d'imatges, i no s'ha pogut començar amb la generació d'índexs dels documents, tot i que aquests objectius són perfectament assolibles a futur.

Durant el projecte s'han explorat temes importants en el desenvolupament de sistemes encastats en general, com el disseny de circuits digitals i d'alimentació o l'ús de sistemes operatius en temps real com freertos i el desenvolupament de drivers. S'ha dissenyat, des del hardware al software d'aplicació, un dispositiu coherent amb els objectius, que compleix amb les característiques essencials d'un lector de llibres electrònics, sent perfectament usable per a la seva tasca.

Amb tot, durant el projecte s'ha vist que les capacitats d'un microcontrolador com el que s'ha usat son força grans si s'implementa pensant en les restriccions de memòria, pel que s'espera poder ampliar aquest projecte en les línies previstes.

7.1 Línies de continuació

Per tal de ser usable en el dia a dia de forma més còmode, encara hi ha una sèrie de qüestions que es poden tractar. Primer, s'hauria d'acabar la renderització d'imatges, per tal de cobrir tots els objectius del projecte. També cal millorar la gestió de l'historial, guardant la posició (byte) d'inici de cada pàgina i variables d'estat, per tal de poder reprendre la lectura i retrocedir pàgines sense haver de llegir tot el document de nou.

També serà necessari veure com es comporta el dispositiu amb un ús diari, i corregir possibles *bugs*. En aquest sentit seria interessant millorar el sistema de logs per USB.

Després, cal implementar el parser d'HTML, principalment per tal de poder renderitzar taules incrustades a markdown, i com a base per a llegir llibres en format epub més endavant. També és necessari incloure alguna implementació de l'algorisme inflate per descomprimir tant els arxius epub com imatges en format png[31].

S'ha de millorar també el driver USB, permetent la càrrega d'arxius a la SD per mitjà d'aquest, implementant una interfície MSC[32]. En la mateixa línia, cal provar el mòdul bluetooth i la comunicació amb el micro principal, i començar a implementar una aplicació android que permeti enviar-li contingut. Una opció interessant que obre això és la lectura de pdf, que es podria renderitzar al telèfon mòbil, enviant el bitmap al lector i rebent comandes des d'aquest per al pas de pàgines. També s'ha de provar el DAC d'àudio, i el rendiment de diversos codecs com OGG, FLAC o WAV.

Referències

[1] Contributors to Wikimedia projects, "E-reader - Wikipedia," https://en.wikipedia.org/w/index.php?title=E-reader&oldid=1025653891, May 2021, [Online; accessed 14. Jun. 2021].

[2] Eink, "Eink color technologies," https://www.eink.com/color-technology.html.

[3] joeycastillo, "The-Open-Book," https://github.com/joeycastillo/The-Open-Book, Jun 2021, [Online; accessed 14. Jun. 2021].

[4] "Getting Started – Inkplate," https://inkplate.io/getting-started, Jun 2021, [Online; accessed 15. Jun. 2021].

[5] "KOReader," https://koreader.rocks, Sep 2020, [Online; accessed 15. Jun. 2021].

[6] baskerville, "plato," https://github.com/baskerville/plato, Jun 2021, [Online; accessed 15. Jun. 2021].

[7] "Driving E-ink display – Essential scrap," http://essentialscrap.com/eink;https://epdiy.readthedocs.io/en/latest/, Jun 2021, [Online; accessed 14. Jun. 2021].

[8] rogerdahl, "font-to-c," https://github.com/rogerdahl/font-to-c, Jun 2021, [Online; accessed 19. Jun. 2021].

[9] Contributors to Wikimedia projects, "Comparison of e-readers - Wikipedia," https://en.wikipedia.org/w/index.php?title=Comparison_of_e-readers&oldid=1018565412, Apr, [Online; accessed 15. Jun. 2021].

[10] "Markdown Reference," https://commonmark.org/help, Mar 2021, [Online; accessed 15. Jun. 2021].

[11] "calibre - E-book management," https://calibre-ebook.com, Jun 2021, [Online; accessed 19. Jun. 2021].

[12] "FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions," https://freertos.org, May 2021, [Online; accessed 14. Jun. 2021].

[13] "LibOpenCM3 by libopencm3," https://libopencm3.org, May 2021, [Online; accessed 14. Jun. 2021].

[14] D. F. Garrido, "Repositori del projecte," https://gitlab.com/dferrg/ereader, Jun 2021, [Online; accessed 14. Jun. 2021].

[15] Doxygen, "Doxygen," https://www.doxygen.nl/index.html, Apr 2021, [Online; accessed 14. Jun. 2021].

[16] "4.2inch e-Paper Module - Waveshare Wiki," https://www.waveshare.com/wiki/4.2inch_e-Paper_Module, Jun 2021, [Online; accessed 15. Jun. 2021].

[17] "Sprites mods - Wifi E-ink display - Introduction," https://spritesmods.com/?art=einkdisplay, Jun 2021, [Online; accessed 14. Jun. 2021].

[18] PrimeView, "ED060SC4 datasheet," http://essentialscrap.com/eink/ED060SC4V2.pdf, 2008.

[19] micronix, "MXE1480 datasheet."

[20] IXYS, "MXEI2240 datasheet," http://essentialscrap.com/eink/eink_gate_driver.pdf, 2012.

[21] T. Instruments, "TPS65186 datasheet."

[22] STMicroelectronics, *STM32F407 Reference Manual*, https://www.st.com/resource/en/reference_manual/dm00031020-stm32f405-415-stm32f407-417-stm32f427-437-and-stm32f429-439-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf, 2021.

[23] "KiCad EDA," https://www.kicad.org, May 2021, [Online; accessed 19. Jun. 2021].

[24] "Fdn340p datasheet," https://www.onsemi.com/pdf/datasheet/fdn340p-d.pdf.

[25] "State-of-the-art of battery state-of-charge determination," https://www.researchgate.net/figure/Li-ion-cells-voltage-curve-at-different-discharge-rates_fig5_228624305, Oct 2020, [Online; accessed 19. Jun. 2021].

[26] dferrg, "Ook! case," https://www.thingiverse.com/thing:4886255, Jun 2021, [Online; accessed 15. Jun. 2021].

[27] "FatFs - Generic FAT Filesystem Module," http://elm-chan.org/fsw/ff/00index_e.html, Apr 2021, [Online; accessed 14. Jun. 2021].

[28] U. I. Forum, "USB Device Firmware Upgrade class specification."

[29] "Valgrind Home," Jun 2021, [Online; accessed 29. Jun. 2021]. [Online]. Available: https://www.valgrind.org

[30] "Yxml - A small, fast and correct∗ XML parser," May 2021, [Online; accessed 29. Jun. 2021]. [Online]. Available: https://dev.yorhel.nl/yxml

[31] Contributors to Wikimedia projects, "Deflate - Wikipedia," Feb 2021, [Online; accessed 20. Jun. 2021]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Deflate&oldid=1007667711

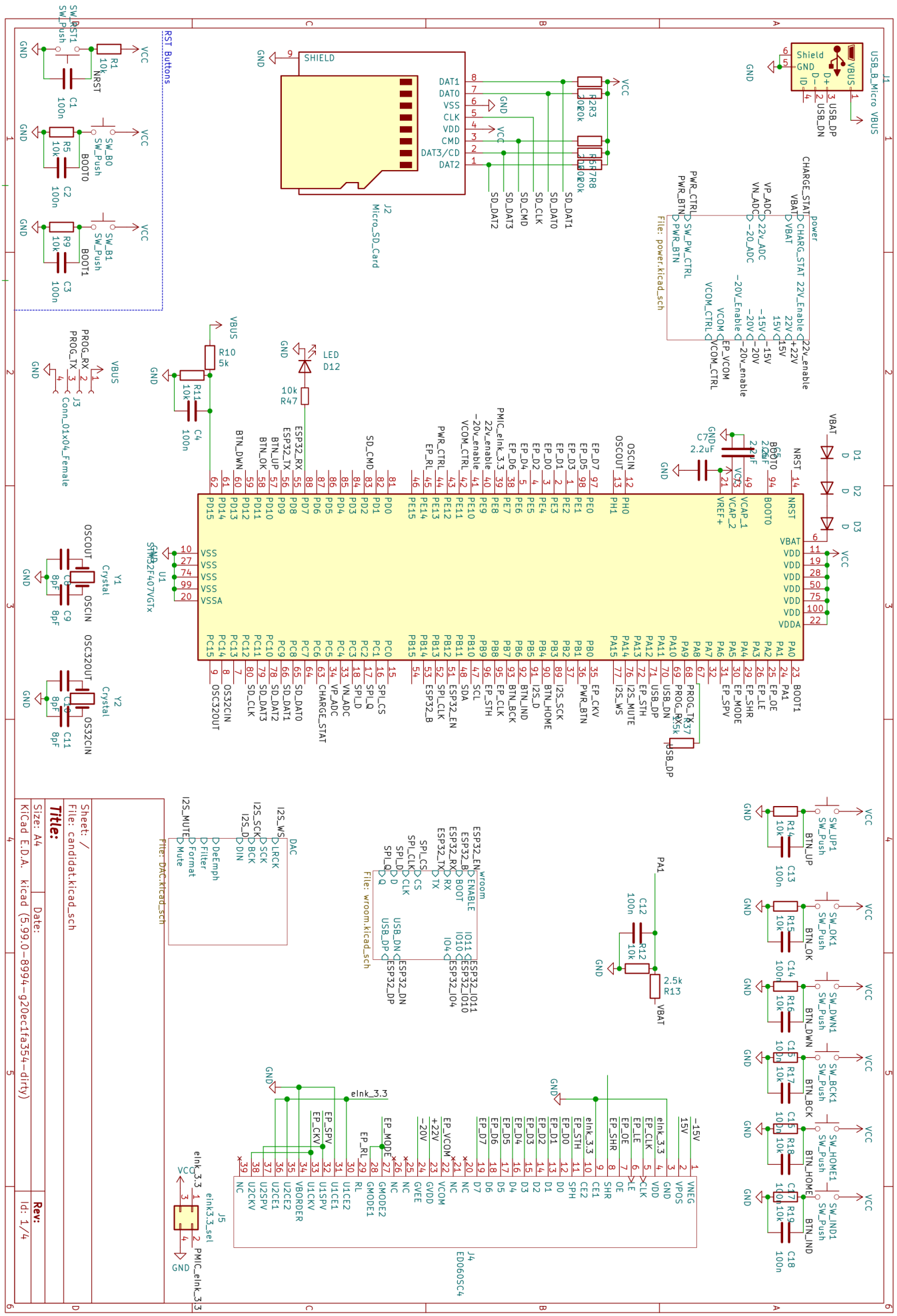
[32] "Mass Storage Class Specification Overview 1.4 | USB-IF," Jun 2021, [Online; accessed 20. Jun. 2021]. [Online]. Available: \url{https://www.usb.org/document-library/mass-storage-class-specification-overview-14}

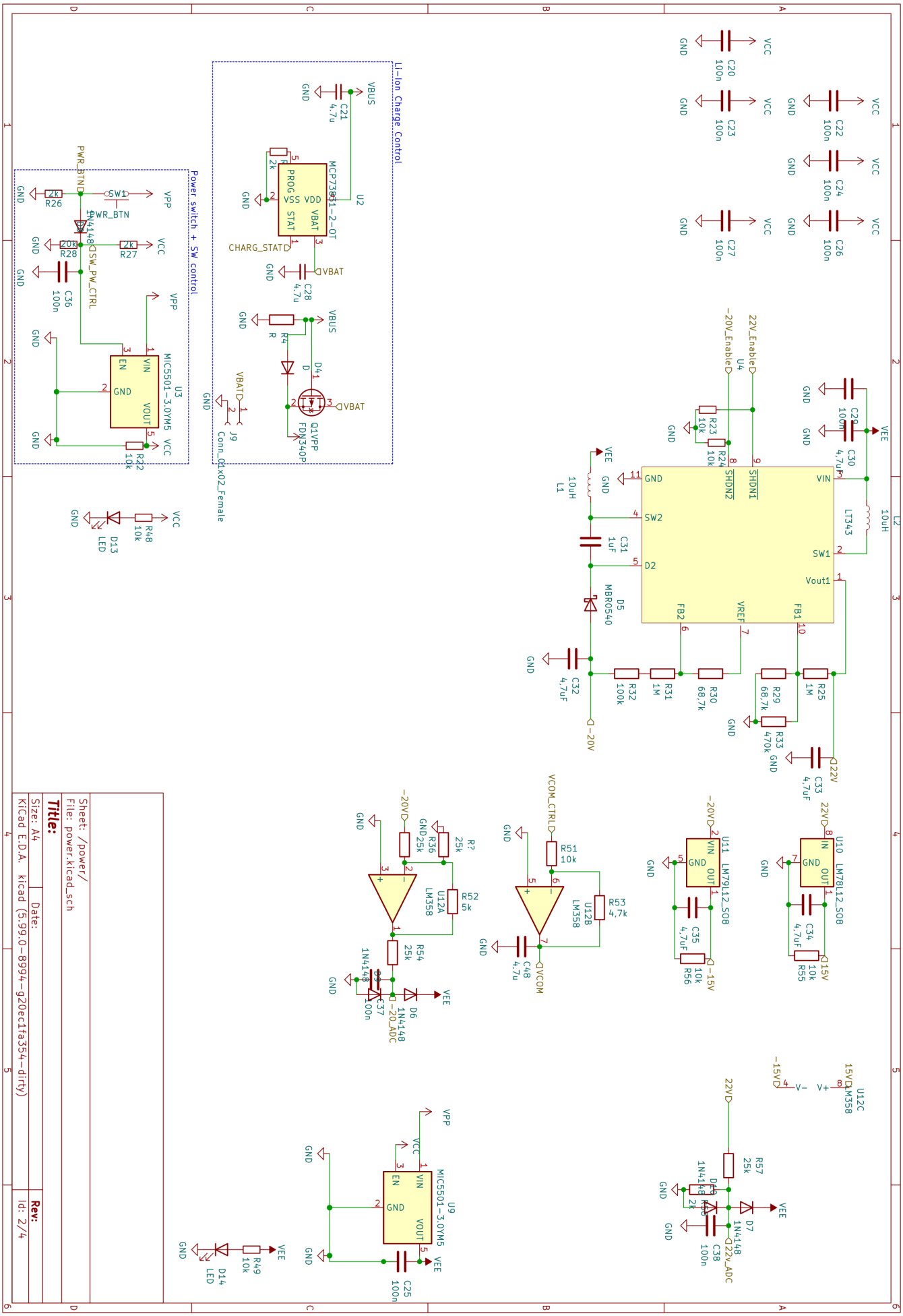
APÈNDIX

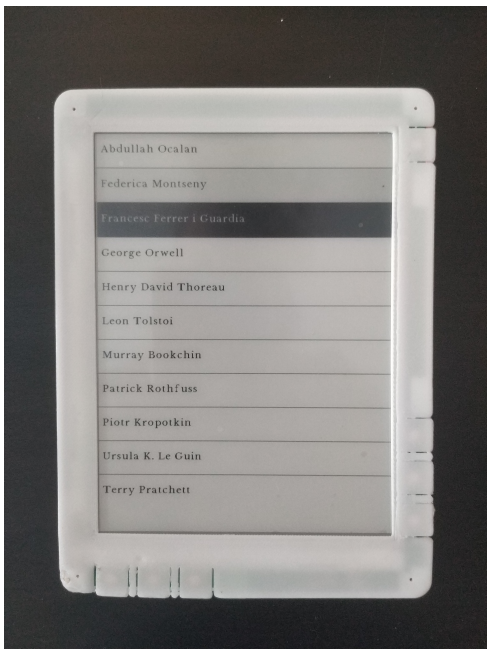
A RESUM SINTAXI DE MARKDOWN

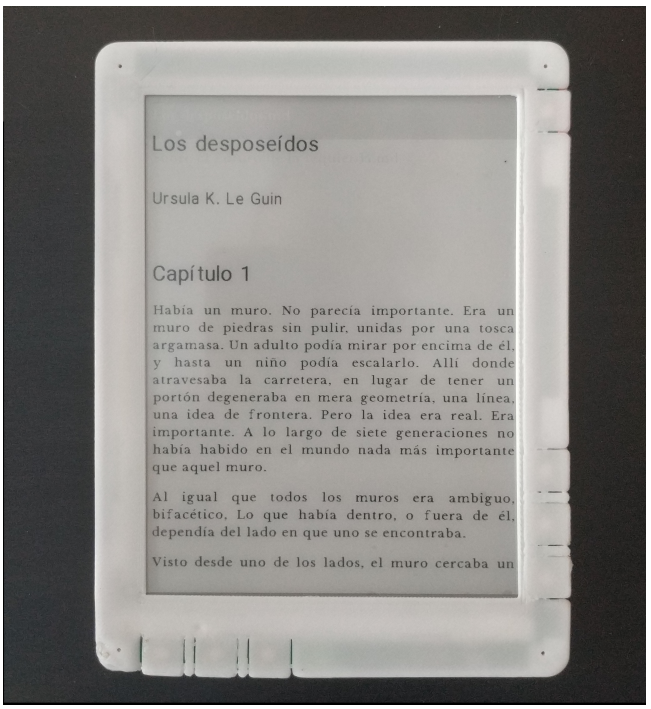
Text normal	Text normal
Negreta	**Negreta**
Cursiva	*Cursiva*
# Títol 1	Títol 1
## Títol 2	Títol 2
### Títol 3	Títol 3
> Cita de text > o codi incrustat	Cita de text o codi incrustat

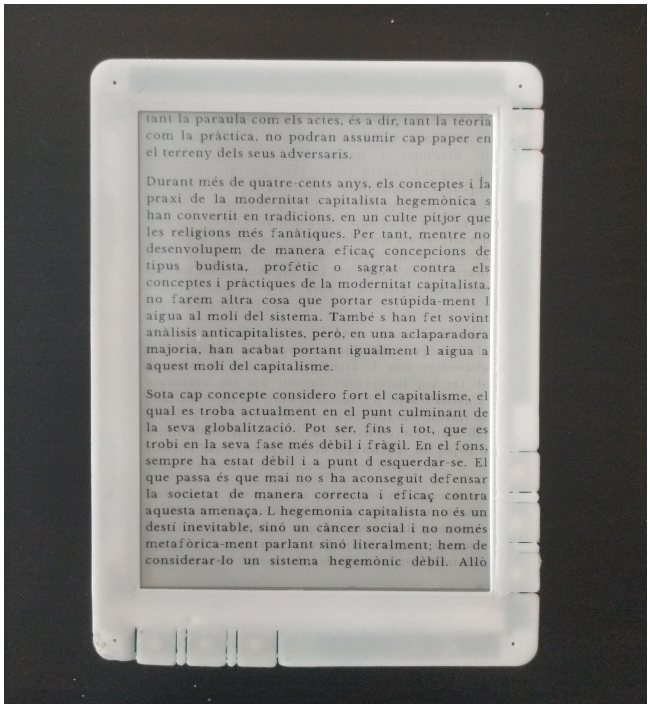
- Entrada de llista
∗Entrada de llista
+ Entrada de llista
  + Segon nivell de llista
	• Entrada de llista
	• Entrada de llista
	• Entrada de llista
	  • Entrada de llista


RST Buttons
SW_RST1
SW_Push
R1
10k
NRST
C1
100n
SW_B0
BOOT0
R5
C2
SW_B1
BOOT1
R9
C3
SHIELD
DAT1
DAT0
VSS
CLK
VDD
CMD
DAT3/CD
DAT2
J2
Micro_SD_Card
R2
R3
R7
R8
SD_DAT1
SD_DAT0
SD_CLK
SD_CMD
SD_DAT3
SD_DAT2
USB_B_Micro VBUS
Shield
GND
D+
D-
ID
VBUS
USB_DP
USB_DN
power
CHARGE_STAT
VBAT
CHARG_STAT
22V_ADC
-20V_ADC
22V_Enable
-20V_Enable
VCOM
EP_VCOM
VCOM_CTRL
PWR_CTRL
PWR_BTN
SW_PW_CTRL
File: power.kicad_sch
C7
2.2uF
R10
5k
R11
10k
C4
100n
LED
D12
10k
R47
PROG_RX
PROG_TX
J3
Conn_01x04_Female
OSCOUT
OSCIN
Y1
Crystal
C8
8pF
C9
OSC32OUT
OSC32CIN
Y2
C10
C11
U1
STM32F407VGTx
VSS
VSSA
VDD
VDDA
VBAT
NRST
BOOT0
VCAP_1
VCAP_2
VREF+
ESP32_RX
ESP32_TX
BTN_UP
BTN_OK
BTN_DWN
PMIC_eink_3.3
PWR_CTRL
EP_RL
SD_CMD
SD_DAT0
SD_DAT1
SD_DAT2
SD_DAT3
SD_CLK
EP_D0
EP_D1
EP_D2
EP_D3
EP_D4
EP_D5
EP_D6
EP_D7
BTN_IND
BTN_BCK
BTN_HOME
EP_CKV
PWR_BTN
I2S_SCK
I2S_WS
I2S_MUTE
I2S_D
EP_OE
EP_LE
EP_SHR
EP_MODE
EP_SPV
EP_STH
EP_CLK
USB_DN
USB_DP
PROG_RX
PROG_TX
R37
5k
USB_DP
SW_UP1
BTN_UP
R14
C13
SW_OK1
BTN_OK
R15
C14
SW_DWN1
BTN_DWN
R16
C15
SW_BCK1
BTN_BCK
R17
C16
SW_HOME1
BTN_HOME
R18
C17
SW_IND1
BTN_IND
R19
C18
PA1
C12
100n
R12
10k
2.5k
R13
VBAT
wroom
ESP32_EN
ESP32_IO0
ESP32_RX
ESP32_TX
ENABLE
BOOT
RX
TX
IO11
IO10
IO4
ESP32_IO11
ESP32_IO10
ESP32_IO4
SPI_CS
SPI_CLK
SPI_D
SPI_Q
CS
CLK
D
Q
ESP32_DN
ESP32_DP
File: wroom.kicad_sch
DAC
I2S_WS
I2S_SCK
I2S_D
LRCK
SCK
BCK
DIN
DeEmph
Filter
Format
Mute
File: DAC.kicad_sch
J4
ED060SC4
eink_3.3
-15V
15V
VNEG
VPOS
GND
VDD
CE1
CE2
SPH
D0
D1
D2
D3
D4
D5
D6
D7
NC
VCOM
GVDD
GVEE
SHR
CL
LE
OE
CKV
SPV
CE1
CE2
RL
GMODE1
GMODE2
VBORDER
U1CKV
U1SPV
U1CE1
U1CE2
U2CKV
U2SPV
U2CE1
U2CE2
J5
eink_3.3_sel
PMIC_eink_3.3
Sheet: /
File: candidat.kicad_sch
Title:
Size: A4
Date:
KiCad E.D.A. kicad (5.99.0-8994-g20ec1fa354-dirty)
Rev:
Id: 1/4


Li-Ion Charge Control
Power switch + SW control
Sheet: /power/
File: power.kicad_sch
Title:
Size: A4
Date:
KiCad E.D.A. kicad (5.99.0-8994-g20ec1fa354-dirty)
Rev:
Id: 2/4


Abdullah Ocalan
Federica Montseny
Francesc Ferrer i Guardia
George Orwell
Henry David Thoreau
Leon Tolstoi
Murray Bookchin
Patrick Rothfuss
Piotr Kropotkin
Ursula K. Le Guin
Terry Pratchett


Los desposeídos
Ursula K. Le Guin
Capítulo 1
Habia un muro. No parecia importante. Era un muro de piedras sin pulir, unidas por una tosca argamasa. Un adulto podía mirar por encima de él, y hasta un niño podía escalarlo. Allí donde atravesaba la carretera, en lugar de tener un portón degeneraba en mera geometría, una línea, una idea de frontera. Pero la idea era real. Era importante. A lo largo de siete generaciones no había habido en el mundo nada más importante que aquel muro.
Al igual que todos los muros era ambiguo, bifacético. Lo que había dentro, o fuera de él, dependía del lado en que uno se encontraba.
Visto desde uno de los lados, el muro cercaba un


tant la paraula com els actes, és a dir, tant la teoria com la pràctica, no podran assumir cap paper en el terreny dels seus adversaris.
Durant més de quatre-cents anys, els conceptes i la praxi de la modernitat capitalista hegemònica s han convertit en tradicions, en un culte pitjor que les religions més fanàtiques. Per tant, mentre no desenvolupem de manera eficaç concepcions de tipus budista, profètic o sagrat contra els conceptes i pràctiques de la modernitat capitalista, no farem altra cosa que portar estúpida-ment l aigua al molí del sistema. També s han fet sovint anàlisis anticapitalistes, però, en una aclaparadora majoria, han acabat portant igualment l aigua a aquest molí del capitalisme.
Sota cap concepte considero fort el capitalisme, el qual es troba actualment en el punt culminant de la seva globalització. Pot ser, fins i tot, que es trobi en la seva fase més dèbil i fràgil. En el fons, sempre ha estat dèbil i a punt d esquerdar-se. El que passa és que mai no s ha aconseguit defensar la societat de manera correcta i eficaç contra aquesta amenaça. L hegemonia capitalista no és un destí inevitable, sinó un càncer social i no només metafòrica-ment parlant sinó literalment; hem de considerar-lo un sistema hegemònic dèbil. Allò