EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Enginyeria Tècn. Ind. Electrònica Ind. Pla 2002

Títol: Servidor d'impressió per a ordinadors a monedes

Document: 1. Memòria

Alumne: Ivan Lorca Aragó

Director/Tutor: Miquel Rustullet Reñé
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: Enginyeria de Sistemes i Automàtica

Convocatòria (mes/any): setembre/2012

Índex

1 INTRODUCCIÓ

1.1 Antecedents

Aquest projecte es va realitzar per a un client que disposa d'un conjunt d'uns 100 ordinadors per a accés a internet amb accés temporitzat per monedes, ubicats en diferents locals, i havia observat una creixent demanda del servei d'impressió, entre d'altres per a fer el check-in per a ryanair.

Fins al moment, el servei d'impressió, en els locals en què s'oferia es realitzava el control de la impressora i el cobrament de les impressions manualment pels propis treballadors del local, cosa que causava molèsties tant a clients com a treballadors, retards, conflictes en casos de mal funcionament o mal ús del servei, etc.

Aquest sistema feia que el servei d'impressió no fos rentable, i que els locals on s'oferia fos més una deferència cap al client que un negoci. Aquest fet provocava que ni els locals on hi havia els ordinadors, ni l'amo dels ordinadors apostessin per a oferir el servei d'impressió.

Per tal de poder fer rentable el servei d'impressió, s'havia de crear un sistema que permetés realitzar el cobrament de les impressions de forma automàtica i prèvia.

1.2 Objecte

L'objecte d'aquest projecte és donar solució al problema plantejat en els antecedents, creant un sistema que permeti realitzar el cobrament de les impressions de forma prèvia i automàtica, alhora que clara per al client final.

Aquest sistema ha de funcionar de forma autònoma, alliberant així al personal del local de les tasques de gestió de la impressora.

Per assolir aquest objectiu és proposa posar un servidor d'impressió es comunicarà amb els ordinadors controlats per monedes, de forma que al imprimir des dels mateixos, automàticament es descompti el preu de la impressió del crèdit restant en el moneder.

1.3 Especificacions i abast

Degut a la voluntat del client de posar Linux en alguns dels seus ordinadors, per tal d'estalviar llicències, i al parc d'ordinadors ja instal·lats que porten Windows XP, el sistema ha de funcionar tant en Linux com en Windows XP, o superior.

Es dissenyarà l'electrònica i el software corresponent corresponent al servidor de impressió, així com les comunicacions entre el servidor d'impressió i els ordinadors per monedes.

En quant als ordinadors controlats per monedes, s'implantarà la comunicació amb el moneder, per tal de controlar el crèdit disponible i descomptar el temps corresponent a les impressions. Per altre banda, es realitzarà una interfície d'usuari on es comunicarà a l'usuari el preu de les impressions, el temps que se li restarà, el, que té disponible i el que li restarà després de realitzar la impressió. En aquesta mateixa pantalla se li donarà la opció d'acceptar o rebutjar la impressió abans que li sigui descomptada del temps disponible.

Aquest projecte no inclou el disseny dels sistemes de seguretat per a impedir la manipulació de la impressora per a personal no autoritzat.

2 DESCRIPCIÓ DELS MÒDULS DEL SISTEMA

Com ja s'ha explicat en la introducció, el sistema proposat per a solucionar el projecte, consta dels següents mòduls: comunicació del PC amb el moneder, interfície d'usuari, servidor d'impressió, i comunicació de l'ordinador amb el servidor d'impressió.

A continuació es detallarà la funcionalitat de cada mòdul, i finalment s'exposarà les interaccions entre cadascun del mòduls i el workflow general del sistema.

2.1 Comunicació del PC amb el moneder

Aquest mòdul implanta la comunicació amb el moneder Telmicom TxD mitjançant el protocol definit pel fabricant, utilitzant el port RS-232.

Aquesta comunicació consisteix en la monitorització del temps disponible, i la modificació del mateix en quant sigui necessari.

Aquest mòdul s'executa en l'ordinador controlat per monedes i ha d'executar-se permanentment, en segon pla.

Degut a que ha de fer servir les comunicacions RS232, i ha de funcionar tant el Linux com en Windows, s'ha cregut oportú la programació del mateix en llenguatge C, i crear una interfície comú per a les comunicacions RS232, i la pertinent implantació per a cada sistema operatiu, deixant la resta del codi comú.

2.2 Interfície d'usuari

Com en el cas anterior, aquest mòdul ha d'executar-se en l'ordinador controlat per monedes, i, per tant, ha de funcionar tant en Linux com en Windows.

Tenint en compte això, s'ha buscat un sistema d'interfície gràfica que estigues disponible en els dos sistemes operatiu, per tal de poder utilitzar en mateix codi, i no haver de duplicar esforços.

Entre totes les opcions d'interfície gràfica disponibles el sistema escollit és GTK+, principalment per les següents raons: és software lliure i obert (Llicència GNU LGPL), funciona en llenguatge C, és multi-plataforma, i és el més utilitzat en el programari lliure, cosa que dóna garantia de la no obsolescència del mateix a llarg termini. Mes endavant en aquest projecte, s'explica el sistema GTK+ en detall.

La interfície d'usuari consisteix en una finestra emergent en quant es demanda una impressió, que mostra a l'usuari el nombre de pàgines a imprimir, el preu per pàgina, el preu total, el cost total en temps, el temps disponible i el temps restant un cop realitzada la impressió.

En el cas que el temps disponible sigui suficient per a pagar la impressió es permet a l'usuari acceptar o cancel·lar-la.

En el cas que no sigui suficient, es mostra el crèdit que manca per a poder fer la impressió, i es sol·licita que s'introdueixi. En aquest cas, només es permet a l'usuari cancel·lar la impressió.

2.3 Servidor d'impressió

El propòsit d'aquest mòdul és la creació d'un servidor d'impressió que ha de tenir les següents característiques: possibilitar el comptatge de pàgines a imprimir al rebre la comanda d'impressió, i poder-la retenir fins que es rebi la confirmació d'impressió. Així mateix, s'ha de minimitzar el cost en hardware tant pel què fa al servidor d'impressió mateix, com a la impressora que pot suportar.

La solució utilitzada és posar un microordinador tipus “intel” o x86, amb un sistema operatiu Linux mínim, que porti el servidor d'impressió CUPS integrat, i un petit programa que controli la cua d'impressió.

També s'ha contemplat la possibilitat de posar un microordinador basat en ARM, per tal de reduir el cost, però és dóna el cas que les impressores de gama baixa no suporten protocols estàndard com ara postscript o PCL, i els fabricants subministren els controladors per a Linux sobre intel, però no sobre ARM.

2.4 Comunicació de l'ordinador amb el servidor d'impressió

L'ordinador veu el servidor d'impressió com a una impressora de xarxa tipus postcript, i hi accedeix utilitzant el protocol IPP.

Un cop l'ordinador envia una impressió al servidor d'impressió, aquest detecta la adreça IP de l'ordinador que li ha enviat la impressió i inicia una connexió tipus socket contra l'ordinador, a un port determinat, mitjançant aquesta connexió, informa a l'ordinador del número de pàgines, el preu per pàgina i el preu total, i l'ordinador informa de si s'accepta o no la impressió.

2.5 Interaccions entre els elements i workflow general del sistema

A la figura 1 es pot veure el diagrama d'estats i les interaccions entre els diferents mòduls del sistema.

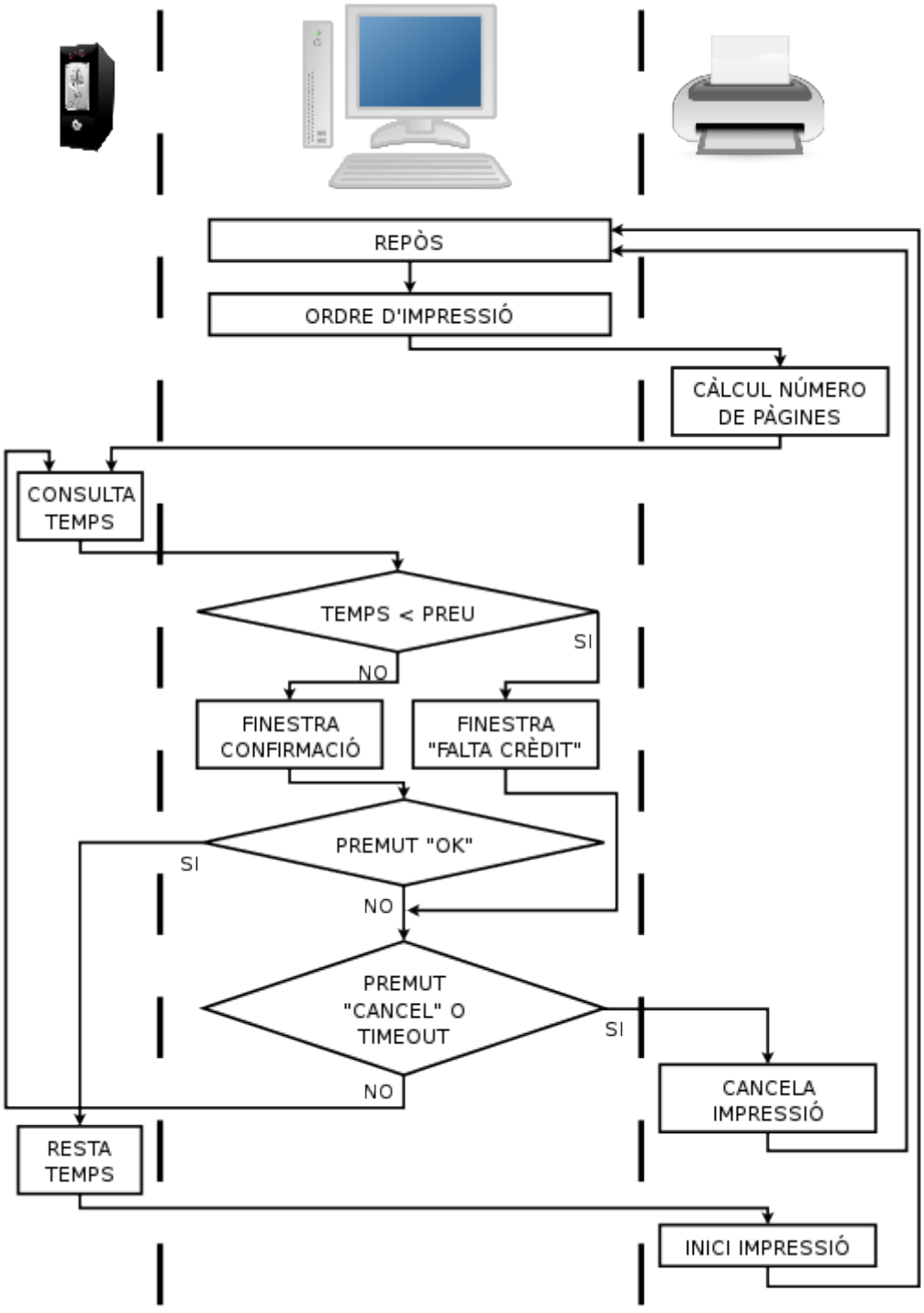

Figura 1. Diagrama d'estats

3 EL MONEDER TELMICOM TXD

3.1 Descripció general del moneder

El moneder Telmicom TxD proporciona múltiples formes de funcionament i varis paràmetres de configuració per adaptar-lo a les diferents necessitats de l'usuari.

En l'aplicació que ens ocupa, el moneder està programat en mode de funcionament tipus temporitzador, és a dir, manté un relé tancat durant un cert temps depenent de l'import introduït.

Aquesta funció és la que el client fa servir per a limitar l'ús que hom pot fer de l'ordinador, mitjançant la desconnexió dels ports USB de l'equip, impedint així la utilització del mateix (tenint en compte que el teclat i ratolí estan connectats per USB).

Aquest moneder disposa d'una pantalla exterior i 4 botons interiors, que es poden fer servir per a la configuració i gestió del mateix. També disposa de dos ports de comunicacions tipus sèrie, un TTL i l'altre RS232, que es poden utilitzar per a interactuar amb ell, mitjançant el protocol ccTalk.

3.2 El protocol ccTalk

El protocol ccTalk és un protocol de comunicació mitjançant un bus sèrie, desenvolupat per Coin Controls l'any 1996, actualment aquest protocol s'utilitza àmpliament en la indústria de la monètica, així doncs, és utilitzat per molts acceptadors de monedes, acceptadors de billets, dispensadors de monedes, màquines de canvi, telèfons de pagament, generadors de tiquets, etc.

Aquest protocol utilitza una aproximació d'un únic Mestre i varis Esclaus, és a dir, que un dels equips és el propietari del bus, i és ell qui pren la iniciativa a l'hora d'establir comunicació amb els altres elements.

Cadascun dels elements connectats al bus té assignada una adreça. Aquesta adreça és arbitrària i no indica res sobre l'element connectat. L'adreça 0 és fa servir per a broadcast, és a dir, tots els equips hi responen sigui quina sigui la seva adreça, i la 1 es fa servir típicament per al Mestre. Per a la resta d'equips queden disponibles les adreces de la 2 a la 255.

A la taula 1 es pot veure l'estructura dels missatges, i una breu descripció de cadascun dels camps. Cal notar que cada camp està format per un byte. El format del missatge és el mateix tant per a l'enviament des del Mestre com per a la resposta de l'esclau.

Camp	**Descripció**
Adreça de destí	Adreça de l'equip al què va destinat el missatge.
Número de bits	Longitud de les dades del missatge. Valor de 0 a 252.
Adreça d'origen	Adreça de l'equip que envia el missatge
Capçalera	Indica l'acció a realitzar. El valor '0' indica que el missatge és una resposta.
Dada 1	
...	
Dada N	
Suma de comprovació	És una simple comprovació de suma de 8 bits (mòdul 256) de tots els bytes del missatge, de forma què, la suma total, incloent el byte de suma de comprovació, ha de ser 0. En cas que sigui diferent de 0, indica un error de recepció del missatge

Taula 1. Format dels missatges ccTalk

3.3 Principals comandes disponibles en el TxD

3.3.1 Eco

Permet saber si el moneder està funcionant i ben connectat i configurat. Respon un ACK.

Comanda: 0xFE

3.3.2 Petició d'identificació del Fabricant

Respon l'identificador del fabricant de l'equip, en aquest cas TMC

Comanda: 0xF6

3.3.3 Petició d'identificació del tipus de producte

Respon l'identificador del producte, en aquest cas TxD

Comanda: 0xF4

3.3.4 Petició del número de sèrie

Respon el número de sèrie del producte

Comanda: 0xF2

3.3.5 Reset

Força l'equip a reiniciar-se

Comanda: 0x01

3.3.6 Petició de monedes acceptades

Respon un bitmap de les monedes que acceptarà el moneder. A la taula 2 és pot veure la correspondència entre cada bit i cada moneda.

Comanda: 0xE6

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
--	--	2€	--	--	1€	50c	--	--	20c	--	10c	5c	2c	--	--

Taula 2.Correspondència del mapa de bits amb les monedes

3.3.7 Modificació de monedes acceptades

Modifica les monedes que acceptarà el moneder. Se li envia un bitmap igual que el de l'apartat anterior, descrit a la taula 2.

Comanda: 0xE6

3.3.8 Petició de paràmetres avançats

Respon el temps total disponible, i el valor del totalitzador de diners introduïts, i un paràmetre reservat.

Comanda: 0xFF 0x63. Aquesta comanda és composta, és a dir, comparteix el camp Header (0xFF) amb altres comandes, i defineix la funcionalitat segone el valor del primer byte de dades.

3.3.9 Modificació de paràmetres avançats

S'envia la mateixa trama que es rep en el punt anterior, per tal de canviar el valor d'aquests paràmetres.

Comanda: 0xFF, 0x65

3.3.10 Petició tarifa

Retorna el preu configurat en cèntims d'euro per hora.

Comanda: 0xFF, 0x68

3.3.11 Modificació de tarifa

Canvia el preu hora configurat, se li envia el preu en cèntims d'euro

Comanda: 0xFF, 0x69

3.3.12 Petició d'última moneda introduïda

Retorna el valor en cèntims d'euro de l'última moneda introduïda al moneder. Si se li envia un 0, el moneder reseteja aquest registre i en successives consultes retornarà 0 fins que no se li introdueixi una nova moneda. En cas contrari, es segueix llegint el mateix valor fins que una nova moneda és introduida.

4 SISTEMA D'IMPRESSIÓ CUPS

4.1 Introducció

El sistema d'impressió CUPS és un sistema d'impressió modular per sistemes operatius tipus UNIX, aquest sistema d'impressió utilitza com a base el protocol Internet Printing Protocol i el llenguatge de impressió postscript, i converteix l'ordinador en un servidor d'impressió. Així doncs, un ordinador amb el CUPS funcionant pot rebre treballs d'impressió d'ordinadors client, processar-los i enviar-los a la impressora corresponent.

Aquest sistema va ser desenvolupat originalment per Michael Sweet, propietari de Easy Software Products, fins que l'any 2.002 va ser adquirit per Apple Inc. Aquest software està llicenciat sota la “GNU General Public License” i la “GNU Lesser General Public License” versió 2.

4.2 Estructura del sistema CUPS

El sistema CUPS està format per tres mòduls principals: La cua i el planificador d'impressió, el sistema de filtres i els destins d'impressió.

Aquest sistema modularitzat fa possible que es puguin generar treballs d'impressió amb diferents formats, per exemple, postscript, HP/GL, PDF, gràfics tipus mapa de bits (PNG, JPEG, GIF, etc), gràfics vectorials (EPS, SVG, etc), i que tots siguin processats per mateix sistema, i pel mateix controlador d'impressió. Així mateix, permet que cada fabricant crei el seu propi controlador d'impressió per a les seves impressores i sigui integrat en el sistema de forma que es pugui utilitzar transparentment per a tots els clients del servidor d'impressió CUPS.

A continuació es farà una descripció de cadascun dels tres mòduls principals anomenats anteriorment.

Finalment, a la figura 2 es pot veure un diagrama de blocs mes detallat del sistema CUPS, amb cadascun dels actors que intervenen segons cada escenari i el flux de les dades.

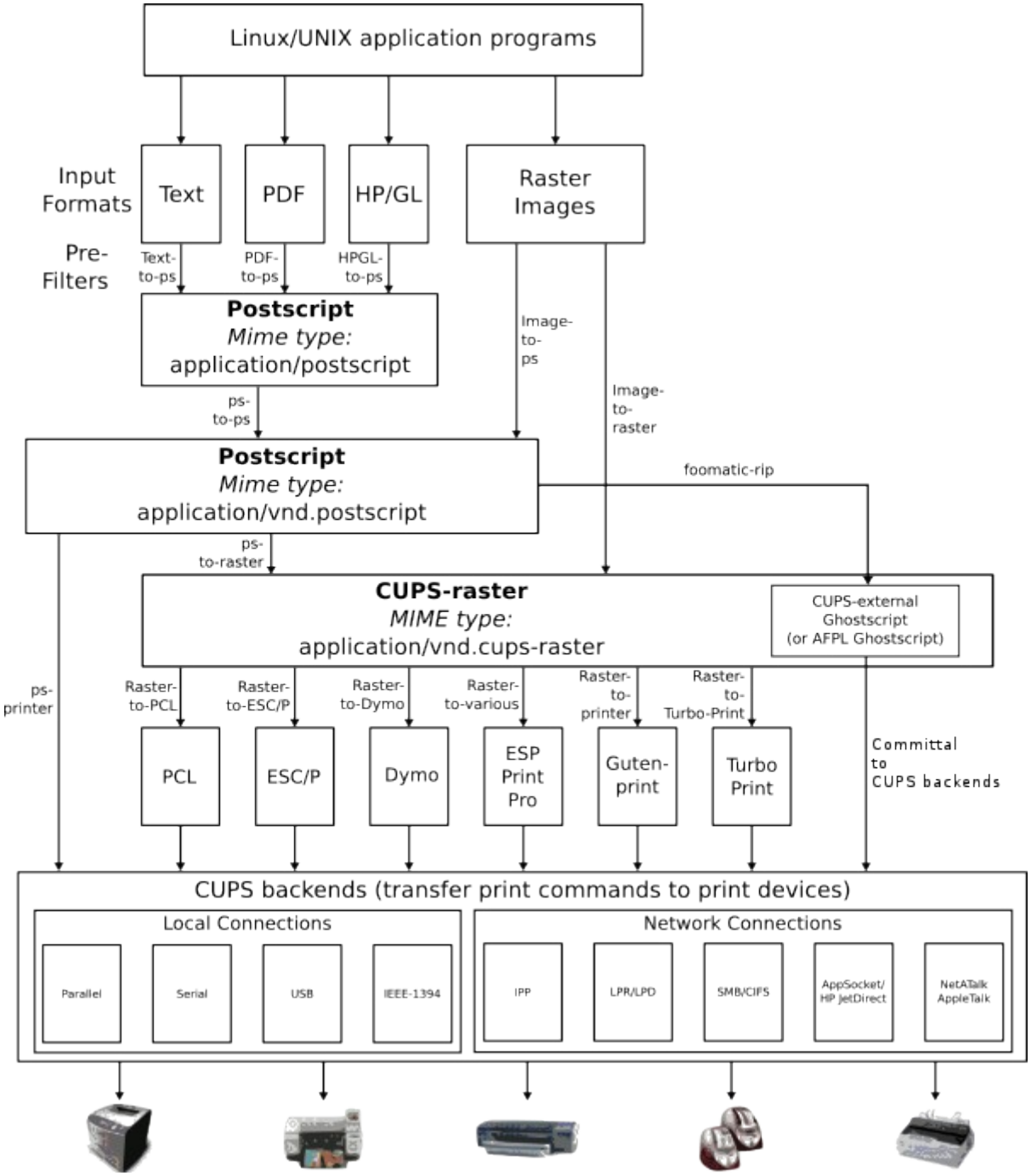

Figura 2.Diagrama de blocs del sistema CUPS

4.2.1 La cua i el planificador d'impressió

La cua d'impressió del CUPS implementa el Internet Printing Protocol (IPP) sobre HTTP/1.1. Incorpora també un sistema d'autorització, per tal de gestionar quins missatges estan permesos i poden ser processats pel sistema i quins no.

Un cop els missatges han estat autoritzats, són processats i enviats al mòdul de IPP, on se'ls assigna un URI (Identificador Uniforme de Recurs), que serà utilitzat durant tota la vida del missatge.

La cua d'impressió suporta classes d'impressores, és a dir, es poden enviar peticions d'impressió a una classe, i aquesta impressió s'enviarà a la primera impressora que quedi lliure de la classe.

Un cop s'ha autoritzat, processat l'ordre d'impressió, aquesta s'envia a la cua d'impressió corresponent, i un cop en surt, es dirigeix al sistema de filtres que es descriu en el punt següent.

4.2.2 El sistema de filtres

Com ja s'ha esmentat abans, el sistema CUPS suporta diferents tipus de dades d'entrada a l'hora d'acceptar una ordre d'impressió. La conversió des del format d'entrada fins al format utilitzat per la impressora es fa mitjançant l'encadenament de diferents filtres. El sistema utilitza els tipus MIME per tal d'identificar cadascun dels formats de fitxer.

El sistema de filtres utilitza dues fonts de configuració, que descriurem a continuació.

En primer lloc té una configuració que permet identificar el format de les dades d'entrada, ja sigui mitjançant l'extensió del fitxer d'entrada, com el que es conegut com a MAGIC NUMBER, es a dir, identificar el format del fitxer segons el seu contingut.

En segon lloc té una taula que relaciona una conversió de tipus, mitjançant una taula que relaciona tipus d'entrada, tipus de sortida, filtre (o programa) per a fer la conversió i cost de la conversió. D'aquesta forma, es pot buscar la forma mes òptima de convertir d'un format a un altre unes dades donades.

La conversió de format de les dades es pot fer directament des de les dades d'entrada cap al format acceptat per la impressora, o utilitzant el format intermig de ràster del CUPS. Aquest darrer sistema és el més utilitzat per a les impressores que utilitzen formats propietaris.

4.2.3 Els destins d'impressió

Els destins d'impressió són les formes en les que el sistema d'impressió pot enviar les dades a les impressores, altrament dit, transport de les dades.

El CUPS suporta diferents destins d'impressió tant d'impressores locals, com de xarxa. Per les impressores locals els destins poden ser port sèrie, port paral·lel, port USB, bluetooth, i per xarxa suporta els protocols IPP, JetDirect (AppSocket), Line Printer Daemon (LDP) i SMB.

5 INTERFÍCIE PER AL CONTROL DE LA CUA D'IMPRESSIÓ DEL SISTEMA CUPS

5.1 Introducció

Per a tal de poder controlar la cua d'impressió del sistema CUPS, i poder fer el comptatge de pàgines a imprimir, s'han utilitzat dos programes que es detallen tot seguit.

5.2 Tea4CUPS

Aquest programa permet punxar el sistema CUPS per tal de fer diferents accions en el procés de vida d'una tasca d'impressió, permetent la execució de programes externs durant el mateix.

El Tea4CUPS permet executar els programes externs de tres formes, o, el què és el mateix, en tres estats del procés d'impressió.

La primera forma és en forma de filtre, tal com s'ha descrit anteriorment el funcionament dels filtres en el sistema CUPS. Aquesta forma permet la manipulació de les dades a imprimir. Només es suporta un sol filtre per a cada cua d'impressió.

La segona és l'anomenada prehook, i permet executar un programa abans que s'enviïn les dades a la impressora. En el cas que el programa retorni -1 la impressió es cancel·la.

Finalment, la tercera és l'anomenada posthook, que s'executa un cop efectuada la impressió, i que permet saber l'estat resultat de la mateixa.

En el nostre cas, com que el procés que volem realitzar implica actuacions abans de la impressió, i hem de tenir la possibilitat de cancel·lar el treball d'impressió, i donat que no hem de fer cap altre procés ni modificació sobre el contingut a imprimir, el que s'adequa millor a les nostres necessitats és el prehook, donat que el posthook s'executa quan la impressió ja s'ha realitzat, i el filtre s'utilitza per a modificar el contingut a imprimir. Així dons, s'establirà un filtre prehook que faci el comptatge de les pàgines a imprimir, calculi el preu d'impressió, es comuniqui amb l'ordinador que ha iniciat la impressió i, un cop rebi la resposta, permeti o cancel·li la impressió.

Aquests programes externs poden obtenir la informació referent a la tasca d'impressió que s'està processant mitjançant certes variables d'entorn, les quals estan llistades i descrites a la taula 3.

Paràmetre	Descripció
TEAPRINTERNAME	Nom de la cua d'impressió
TEADIRECTORY	Directori de treball del Tea4CUPS
TEADATAFILE	Nom complert del fitxer de treball del Tea4CUPS (dins del directori)
TEAJOBSIZE	Pes del treball en bytes
TEAMD5SUM	Suma de comprovació MD5 del fitxer de dades
TEACLIENTHOST	Adreça de xarxa del client que ha iniciat el treball
TEAJOBID	Identificador del treball
TEAUSERNAME	Nom d'usuari del qui ha iniciat el treball
TEATITLE	Títol del treball
TEACOPIES	Número de copies del treball
TEAOPTIONS	Opcions del treball
TEAINPUTFILE	Nom complert del fitxer de dades
TEABILLING	Codi de facturació del treball
TEACONTROLFILE	Nom complert del fitxer de missatges IPP
TEASTATUS	Valor d'estat del resultat de la impressió. Només per a posthook

Taula 3.Llistat de variables d'entorn que proporciona el Tea4CUPS als programes externs

El programa Tea4CUPS està desenvolupat per Jerome Alet en llenguatge python, i està llicenciat segons la GNU General Public Licence versió 3.

5.3 El pkpgcounter

Tal com el programa anterior, Tea4CUPS, aquest programa està desenvolupat per Jerome Alet en llenguatge python, i està llicenciat segons la GNU General Public Licence versió 3.

El propòsit d'aquest programa és realitzar el comptatge de pàgines d'un document.

Suporta varis formats de documents d'entrada, entre el que destaquem: Postscript, PDF i PCL (versions 3 a 6).

6 LLIBRERIA GRÀFICA GTK+

6.1 Introducció

La llibreria gràfica GTK+ és la més utilitzada en entorn UNIX. Va ser desenvolupada originalment per a ser utilitzada en la creació del GNU Image Manipulation Program (GIMP) a qui deu el nom (Gimp Tool Kit), a partir de l'any 1.997. Des d'aleshores aquesta llibreria ha anat evolucionant, i, juntament amb la llibreria Qt, s'ha estès com a l'estàndard de facto per a la programació d'interfícies gràfiques en el món UNIX i GNU.

Avui dia, la llibreria GTK+ està disponible amb suport natiu per a entorns UNIX-X11, Windows i Mac OS X. Després de 25 anys de desenvolupament, es pot considerar una plataforma estable i fiable.

La llibreria GTK+ està programada utilitzant el llenguatge C, per tant el suport per aquest llenguatge és natiu, tot i que també hi ha interfícies per a utilitzar-la des d'altres llenguatges de programació, com ara C++, python, perl, java, etc.

Aquesta llibreria està llicenciada sota la GNU Lesser General Public License versió 2.1.

6.2 Estructura de la llibreria GTK+

A la figura 3 es pot veure l'estructura de llibreries que componen l'entorn de desenvolupament per a GTK+. A continuació es farà una breu descripció de cadascun dels mòduls.

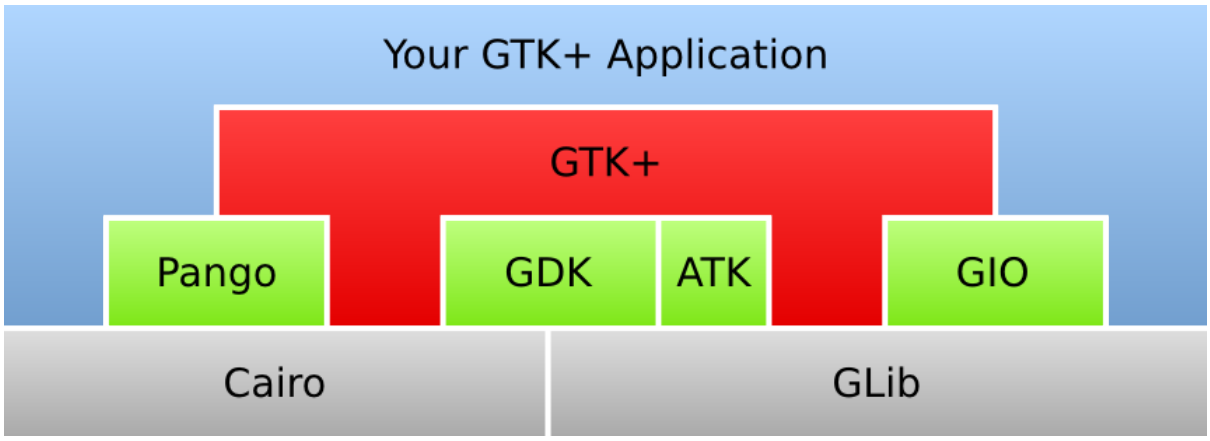

Figura 3.Estructura d'una aplicació GTK+

6.2.1 Glib

Aquesta és una llibreria de baix nivell que és la base de GTK+. Aporta les estructures de manegament de dades per a C, les interfícies per a altres llenguatges de programació, així com les funcionalitats bàsiques com ara bucle de processat d'esdeveniments, suport per a fils d'execució, càrrega dinàmica de mòduls i el sistema d'orientació a objecte.

6.2.2 Pango

La funció de la llibreria Pango és proveir d'un sistema per a el tractament i presentació del text, amb un especial èmfasi en la internacionalització. És la base del tractament del text i del tipus de lletra en GTK+.

6.2.3 Cairo

Aquesta és una llibreria de gràfics 2D, que suporta diferents interfícies gràfiques (X11, Windows, etc), mantenint una coherència en totes elles i aprofitant les acceleracions gràfiques per hardware quan estan disponibles.

6.2.4 ATK

Aquest mòdul proveeix accessibilitat a l'aplicació per a poder ser utilitzada amb eines per a gent impedida, com ara lectors de pantalla, sistemes amb funcionalitat de lupa i mecanismes d'entrada alternatius.

6.3 Creació d'interfícies gràfiques d'usuari amb la Llibreria GTK+

La llibreria GTK+ proporciona tot un seguit de ginys, o widgets, que poden ser utilitzats per a construir la interfície gràfica. Aquests ginys estan programats utilitzant orientació a objecte, i es poden crear, eliminar i modificar dinàmicament des del codi.

Els ginys de la llibreria GTK+ proveeixen tot tipus de funcionalitat típica de una llibreria gràfica, com ara botons, llistes de selecció, imatges, barres de desplaçament, desplegables, eina de selecció de fitxers, marcs, menús, sistemes de pestanyes, barres de progrés, barres de botons, menús, etc.

El sistema d'esdeveniments permet definir dinàmicament quines funcions s'han de cridar en cadascun dels objectes per a cadascun dels ginys.

Aquesta llibreria també proporciona la possibilitat de afegir una icona a la barra de sistema tant de Windows com de Linux, utilitzant la mateixa interfície i proporcionant les mateixes funcionalitats en ambdues plataformes.

6.3.1 L'eina de creació d'interfícies Glade

Com ja s'ha esmentat existeix la possibilitat de crear interfícies gràfiques directament des del programa, creant els objectes pertinents, enllaçant-los, etc. Aquest procés pot ser molt farragós i procliu a errors, a part de suposar que s'ha de recompilar l'aplicació per a cada petit canvi que s'hagi de fer.

Per a solucionar aquests problemes existeix una eina gràfica i visual de creació d'entorns gràfics basats en GTK+ anomenada Glade.

Aquesta aplicació permet dissenyar la interfície gràfica utilitzant un sistema WYSIWYG (What You See Is What You Get), proporcionant tots els ginys de gtk, i podent definir les funcions a cridar per a cada esdeveniment de cada giny o element. A la figura 4 es pot veure la interfície d'aquesta eina de desenvolupament.

La definició de la interfície s'emmagatzema utilitzant un format XML, que en temps d'execució serà processat pel programa i es generarà automàticament la interfície gràfica del mateix, i els enllaços amb les funcions de manegament d'esdeveniments.

Aquest sistema permet que en molts casos es pugui canviar l'aparença gràfica del programa només modificant aquest fitxer XML, sense haver de recompilar tota l'aplicació.

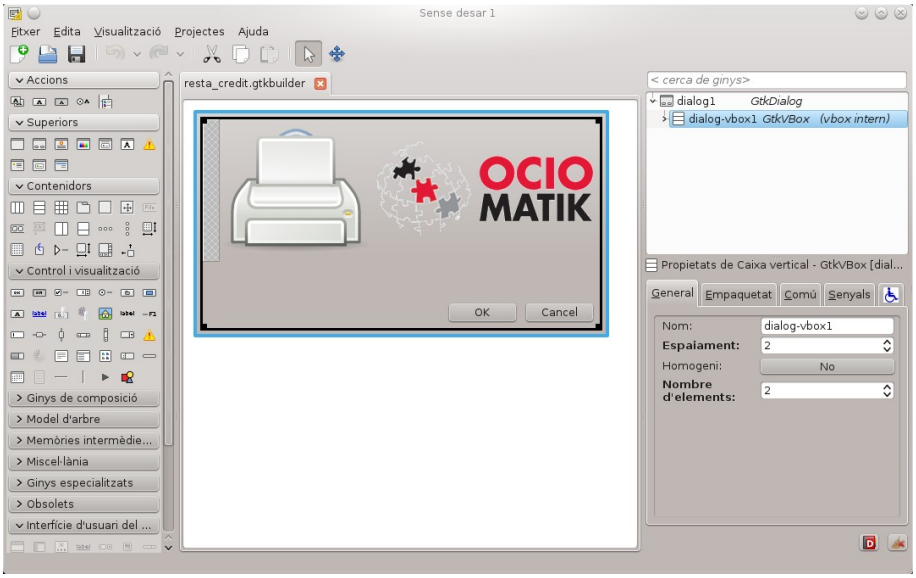

Figura 4.Captura de pantalla de la eina Glade

7 SOFTWARE REALITZAT PER A L'ORDINADOR

7.1 Introducció

El software realitzat per a l'ordinador té tres funcions principals: La interfície d'usuari, La comunicació amb el moneder i la comunicació amb el servidor d'impressió.

Cal tenir en compte que aquest programa ha de funcionar tant en Linux com en Windows, degut a això, es va decidir utilitzar les llibreries gràfiques GTK+ per tal de fer la interfície d'usuari, ja que aquestes llibreries estan disponibles en els dos sistemes operatius.

En quant a la comunicació amb el moneder telmicom, aquesta es realitza mitjançant el port sèrie de l'ordinador, i la forma d'accedir-hi és significativament diferent en cada sistema operatiu, per tant, el que s'ha fet és crear una interfície amb les funcions genèriques per a accedir a aquest port, que seran les utilitzades per la resta del programa, i una implantació diferent d'aquesta interfície per a cada sistema operatiu.

Finalment, la comunicació amb el servidor d'impressió es fa mitjançant un socket, o connexió TCP. En aquest cas la forma de tractar el socket dels dos sistemes operatius és força més similar que en el cas anterior, tot i que hi ha alguna diferència, pel que també s'ha fet una part comú i una part específica per a cada sistema operatiu.

Per a garantir la veracitat de les comunicacions, cada missatge entre el servidor d'impressió i l'ordinador (i viceversa) porta una signatura digital que realitza mitjançant realitzar un hash MD5 del missatge, juntament amb una marca de temps i una clau compartida entre el servidor i l'ordinador.

A la figura 5 es pot veure l'esquema dels mòduls que formen part del software, i les interaccions interaccions entre ells. Ens els següents apartats es fa una descripció detallada de cadascun dels mòduls.

Finalment, a la figura 6 es mostra el diagrama funcional del software del PC.

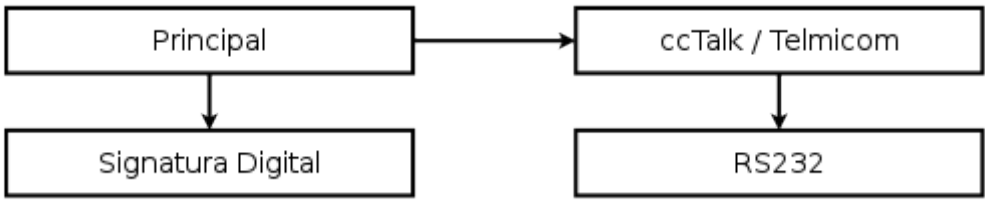

Figura 5. Esquema de mòduls del software del PC

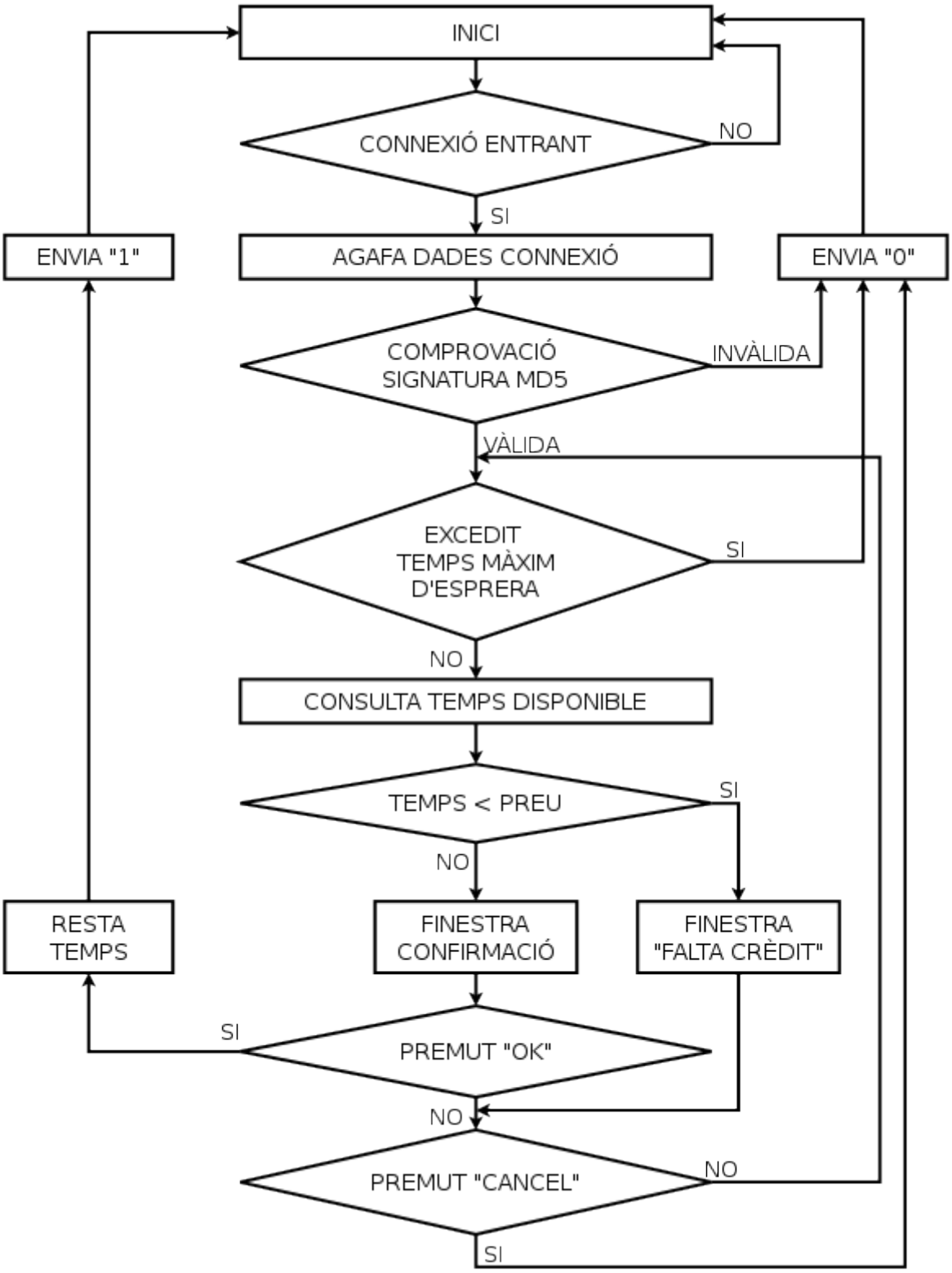

Figura 6. Diagrama funcional del software del PC

7.2 Mòdul RS232

Com ja s'ha comentat, la forma d'accedir als ports sèrie és molt diferent en Windows que en Linux, es significativament diferent, es per això que s'ha creat una interfície on hi ha la descripció de les funcions que proporcionen la funcionalitat, i s'ha creat una implantació d'aquestes funcions diferents per a cada sistema operatiu.

A continuació es descriuen les principals funcions del mòdul.

7.2.1 Read

Aquesta funció permet obtenir les dades rebudes pel port.

7.2.2 Write

Aquesta funció permet enviar dades per port

7.2.3 HasData

Aquesta funció retorna 1 s'has rebut dades.

7.2.4 Flush

Aquesta funció neteja les dades rebudes fins al moment.

7.3 Mòdul ccTalk / Telmicom

Aquest mòdul implanta el protocol ccTalk, i les funcions necessàries per a la comunicació amb el moneder Telmicom TxD.

En quant al protocol ccTalk, implanta la recepció i enviament de missatges, en mode master, utilitzant el mòdul de RS232 abans descrit i calcula la suma de comprovació, tant en enviament , com per en recepció per a validar la integritat de les dades rebudes.

Per a la representació de les dades d'una comanda o resposta ccTalk s'ha creat el tipus telmicom_data_t, que conté les dades d'una trama ccTalk.

La part referent al moneder Telmicom, implanta funcions per a formatar les dades tant en recepció com en enviament, i implanta les següents funcions de comunicació amb el moneder:

A continuació es detallen les funcions principals d'aquest mòdul

7.3.1 telmicom_SendCommand

Aquesta funció rep dos punters a elements tipus telmicom_data_t, el primer és la comanda a enviar, i en el segon es guardarà la resposta rebuda. Aquesta funció envia calcula la suma de comprovació de la comanda, envia la comanda, espera la resposta i comprova la suma de comprovació de la resposta.

Aquesta funció retorna 0 si tot va bé o l'error corresponent en cas d'error.

7.3.2 telmicom_temps

Aquesta comanda consulta el temps disponible al moneder

7.3.3 telmicom_get_temps

Aquesta comanda sobreescriu el temps disponible al moneder

7.3.4 telmicom_get_tarifa

Aquesta comanda consulta la tarifa (preu / hora) del moneder

7.4 Mòdul de signatura digital

La funció d'aquest mòdul és la de signar digitalment les comunicacions entre el servidor d'impressió i l'ordinador. Aquesta signatura es fa mitjançant un hash MD5 de les dades enviades, mes una clau compartida entre el servidor d'impressió i els ordinadors.

Degut a l'aleatorietat del càlcul MD5 respecte les dades d'entrada es fa molt difícil per a algú que no conegui l'algoritme de generació de les dades, i la clau continguda a les mateixes pugui crear una signatura vàlida per a un missatge determinat.

Aquest mòdul utilitza la llibreria md5lib creada per Ulrich Drepper l'any 1.995.

7.5 Mòdul principal

Aquest mòdul és l'encarregat d'implantar les funcionalitats del programa abans descrites. Amb aquest propòsit utilitza la resta de mòduls.

Aquest mòdul també es el que implanta el socket en mode escolta, a on es rebran les connexions provinents del servidor d'impressió. Quan rep una connexió, la processa, comprova que sigui vàlida, mostra la pantalla de confirmació i finalment retorna el resultat al

servidor d'impressió.

7.6 Configuració de la impressora a l'ordinador

Per tal que el sistema funcioni correctament, s'ha d'instal·lar una impressora de xarxa en els ordinadors. El controlador de dispositiu per a aquesta impressora ha de ser tipus postscript genèric, sigui quina sigui la impressora connectada al servidor d'impressió. Això és així perquè la comanda d'impressió és rebuda pel servidor d'impressió i per tal de poder fer el comptatge de pàgines cal que estigui en format postscript. Un cop rebuda la autorització d'impressió, serà el propi sistema CUPS el que processarà el postscript per tal d'imprimir correctament a la impressora que te connectada.

En el cas de Linux, existeix ja el propi driver postscript genèric, en el cas de Windows, es recomana utilitzar el driver per a la impressora Apple LaserWriter.

L'adreça de connexió de la impressora és http://{adreça del servidor}:631/printers/{nom de la impressora}, i el protocol és el IPP.

8 SOFTWARE PER AL SERVIDOR D'IMPRESSIÓ

8.1 Introducció

Per al servidor d'impressió s'han creat dues peces fonamentals de software: el sistema de gestió de la cua d'impressió del CUPS i una interfície de configuració per web, amb autentificació d'usuari.

En quant a la interfície de configuració té dues pantalles, una per a la configuració de la xarxa, i l'altre per a l'administració de les impressores i el preu de la impressió.

Per tal de facilitar la gestió del sistema, s'ha configurat el sistema operatiu del servidor d'impressió per tal que mostri per pantalla un navegador en mode text, amb la interfície de configuració web. Per accedir a aquesta web també es requereix autentificació.

8.2 Gestió de la cua d'impressió

Tal com s'ha comentat anteriorment, per a fer la gestió de la cua d'impressió del CUPS s'utilitzarà el programa Tea4CUPS, en el qual se li crearà un filtre prehook.

Un cop dins el procés del filtre, s'utilitzarà el programa pkpgcounter per a comptar les pàgines de la impressió. Al programa pkpgcounter se li passa com a paràmetre la variable d'entorn $TEADATAFILE, que és el nom del fitxer temporal de la impressió en curs, per tal que realitzi el comptatge sobre aquest fitxer.

Per tal de obtenir l'import total en euros de la impressió, es multiplicarà el número de pàgines a imprimir pel preu per pàgina configurat per la impressora en qüestió.

Coneixent el numero de pàgines i el preu total de la impressió, s'establirà la comunicació amb l'ordinador que ha iniciat l'ordre d'impressió, i se li comunicarà el número de pàgines i el preu total, i s'esperarà a la resposta per part de l'ordinador. En quant l'usuari accepti la impressió es desbloquejarà aquest element de la cua d'impressió, fent-se aquesta efectiva.

En cas que hi hagués algun problema amb les comunicacions, o que l'usuari no accepti la impressió o que s'excedeixi el temps d'espera, el treball d'impressió es cancel·larà.

A la figura 7 es pot veure un diagrama funcional d'aquest procés.

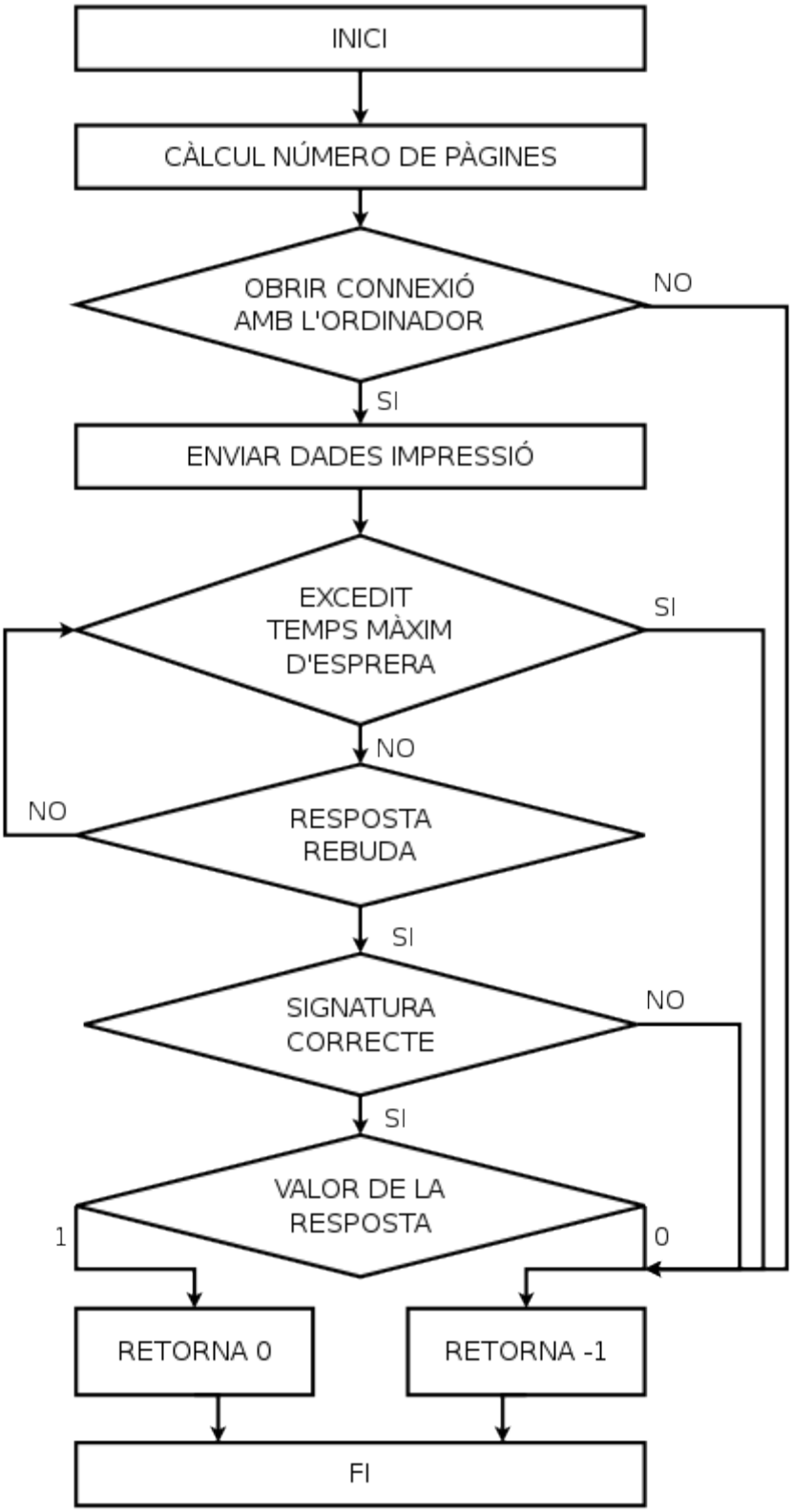

Figura 7. Diagrama funcional del control de la cua d'impressió

8.3 Configuració de la xarxa

Per a fer la configuració de la xarxa, s'ha creat una senzilla pàgina web, on es poden configurar els paràmetres bàsics de la xarxa, nom de la màquina, DHCP o adreça estàtica, adreça IP, màscara de xarxa, adreça de la passarel·la per defecte, i servidors de noms, tal i com es pot veure a la figura 8.

Aquesta web està creada utilitzant un CGI creat en shell-script, que quan se li passen els paràmetres, actua sobre els fitxers propis de configuració de la xarxa de Linux.

Per tal d'evitar accessos no autoritzats a la web de configuració, s'ha protegit l'accés amb usuari i password.

L'adreça per accedir a la configuració de la xarxa és: http://{adreça del servidor d'impressió}/cgi/ip.cgi .

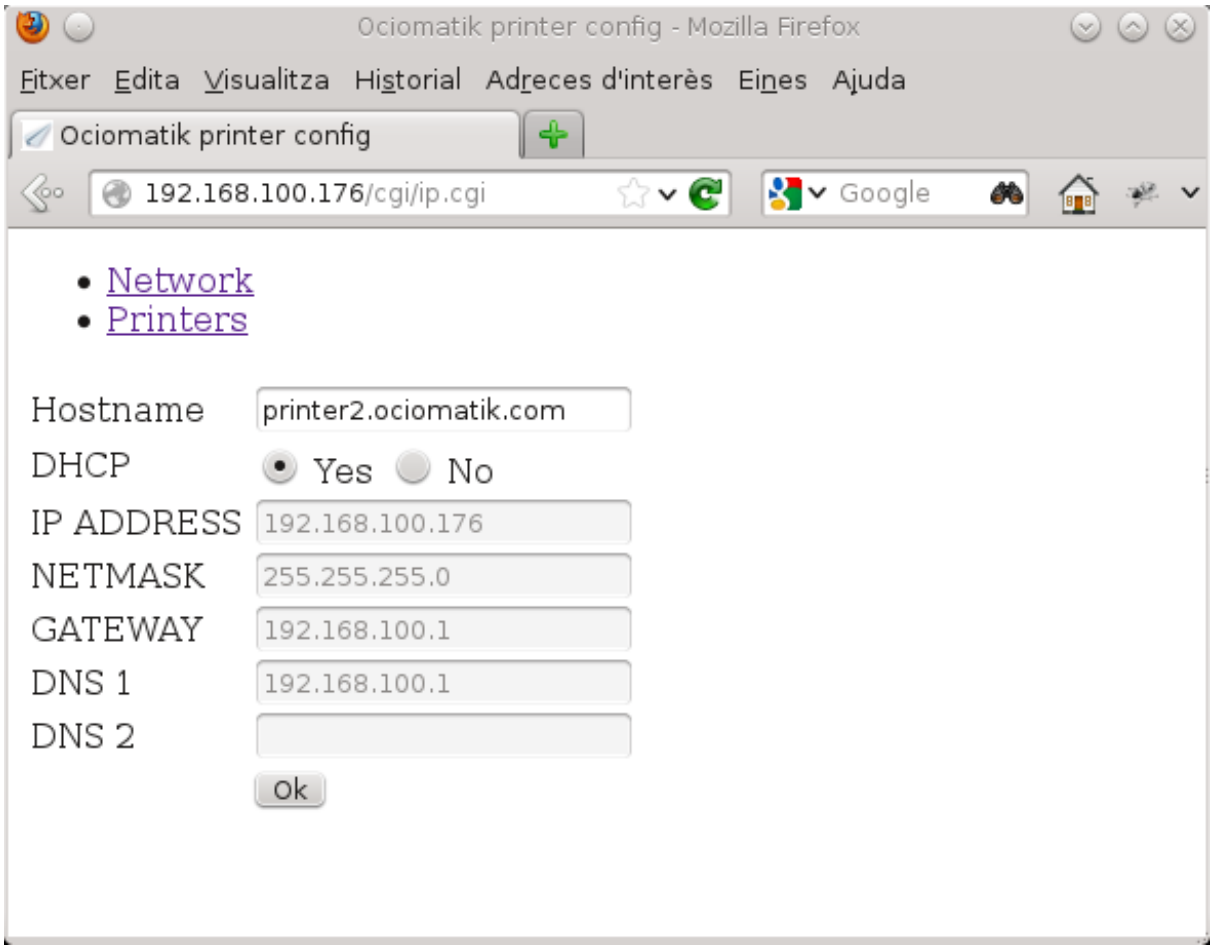

Figura 8. Web de configuració de la xarxa

8.4 Configuració de les impressores

Tal com en l'apartat anterior, s'ha creat una pàgina web per a l'administració de les impressores connectades a el servidor d'impressió.

En aquesta web, es veuen les impressores configurades, i, al final de tot, dóna la possibilitat de crear-ne una de nova. En la interfície de creació d'impressora, hi ha els camps nom de la impressora (que serà el nom de la cua, i el què s'haurà d'utilitzar al crear la impressora en l'ordinador a monedes, tal com s'ha descrit en el capítol anterior), seleccionar impressora, on apareix una llista de les impressores USB connectades al servidor, dos camps informatius, un procés de selecció del controlador de la impressora, agrupat segons, marca de la impressora, model i controlador (hi pot haver més d'un controlador disponible per a un mateix model d'impressora), i, finalment, el preu per pàgina.

Les impressores que ja estan configurades se'ls pot canviar els paràmetres de configuració, excepte el nom de la impressora, i eliminar-les.

A la figura 9 es pot veure una captura de pantalla de la web de configuració de les impressores.

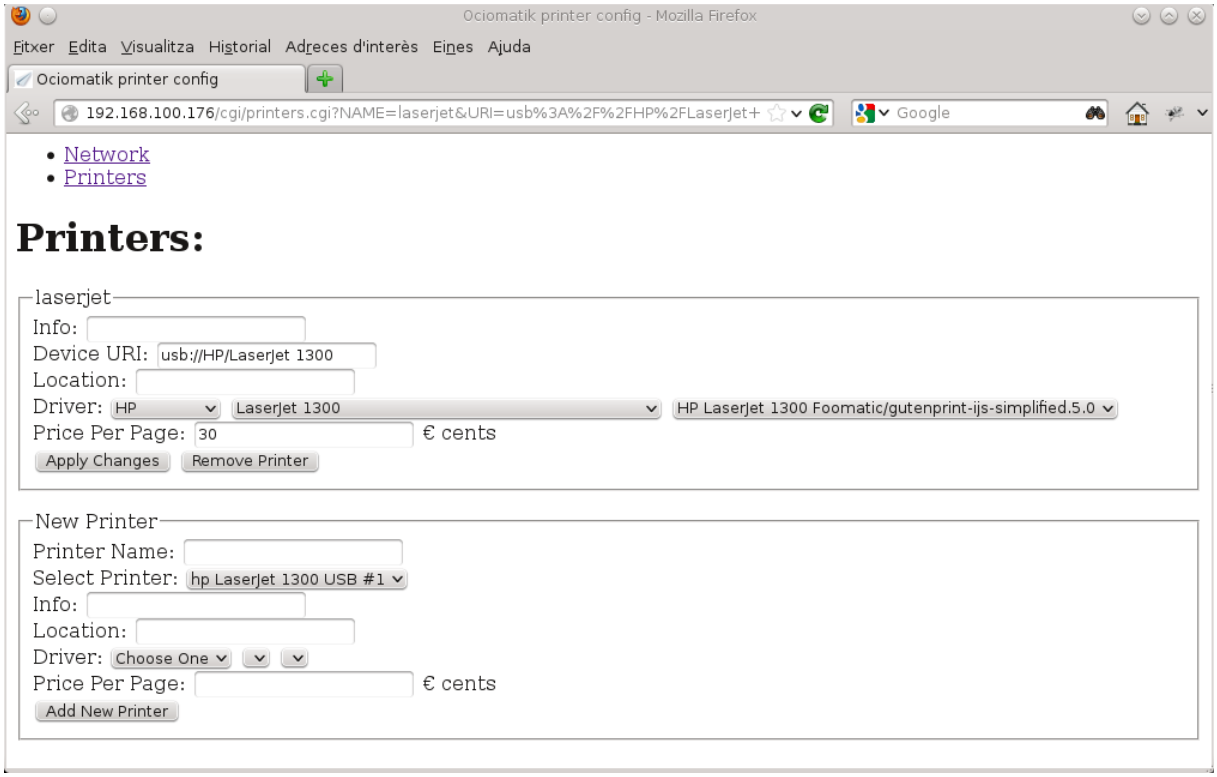

Figura 9. Web de configuració d'impressores

Un cop s'omple aquest formulari i es crea la impressora, automàticament es refà la configuració del CUPS i del Tea4CUPS, per tal que tot funcioni correctament.

Com en el cas anterior, per tal d'evitar accessos no autoritzats a la web de configuració, s'ha protegit l'accés amb usuari i password.

L'adreça per accedir a la configuració de les impressores és: http://{adreça del servidor d'impressió}/cgi/printers.cgi .

9 HARDWARE

9.1 Introducció

En el plantejament del projecte es preveia la necessitat de construir algun tipus d'electrònica d'interfície entre els diferents elements involucrats, però a mesura que es va anar desenvolupant, es va veure que es podia aconseguir tota la funcionalitat desitjada conectant adequadament els diferents equips, i a partir de software.

9.2 Comunicacions

Com ja s'ha explicat en el capítol 3 EL MONEDER TELMICOM TXD, el moneder Telmicom TxD disposa s'un port sèrie RS232, que permet la comunicació amb l'ordinador. Degut a això, s'ha triat una placa base per l'ordinador que disposi d'un port sèrie. A la figura 10 es pot veure la situació del port de comunicacions, i a la taula 4 la interconnexió entre aquest port i el port RS232 del PC.

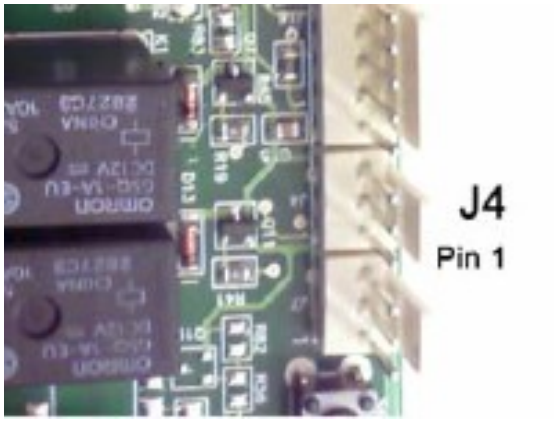

Figura 10. Connector J4 del Telmicom TxD

Telmicom TxD J4	Ordinador COM1 (DB9)
1	3
2	2
3	5

Taula 4. Cable de comunicacions TxD a RS232

9.3 Tall dels ports USB

En quant al tall dels USB per a que no funcionin el teclat i el ratolí quan s'ha esgotat el temps disponible, inicialment es va plantejar la construcció de una petita placa addicional dos relés i dos parells de connectors USB mascle i femella, de forma que es connectes aquesta placa entre l'ordinador i el teclat. Aquest sistema tenia la desavantatge que si hom connectava un teclat i ratolí al port USB que es deixa lliure per a que l'usuari pugui connectar el seu pendrive, telèfon mòbil, etc, podria utilitzar l'ordinador. La solució òbvia era posar més relés, complicant la placa i augmentant el preu.

Finalment, es va descobrir que la placa disposa d'uns pins de configuració que permeten alimentar els ports USB sempre que l'ordinador estigui connectat a la xarxa elèctrica o només quan aquest està en funcionament. Aquests pins de connexió són els que tenen l'etiqueta “1” a la figura 12 i el pinout a la taula 5. En el cas que es deixin a l'aire, els ports USB deixen de funcionar.

Així dons, connectant el relé K3 del TxD, tal com es pot veure a la figura 11) entre el comú d'aquests pins i un dels dos altres, tots els ports USB deixaven de funcionar quan s'esgotava el temps de l'usuari, i tornaven a funcionar al tornar a posar moneda. Així doncs, es va utilitzar aquesta solució.

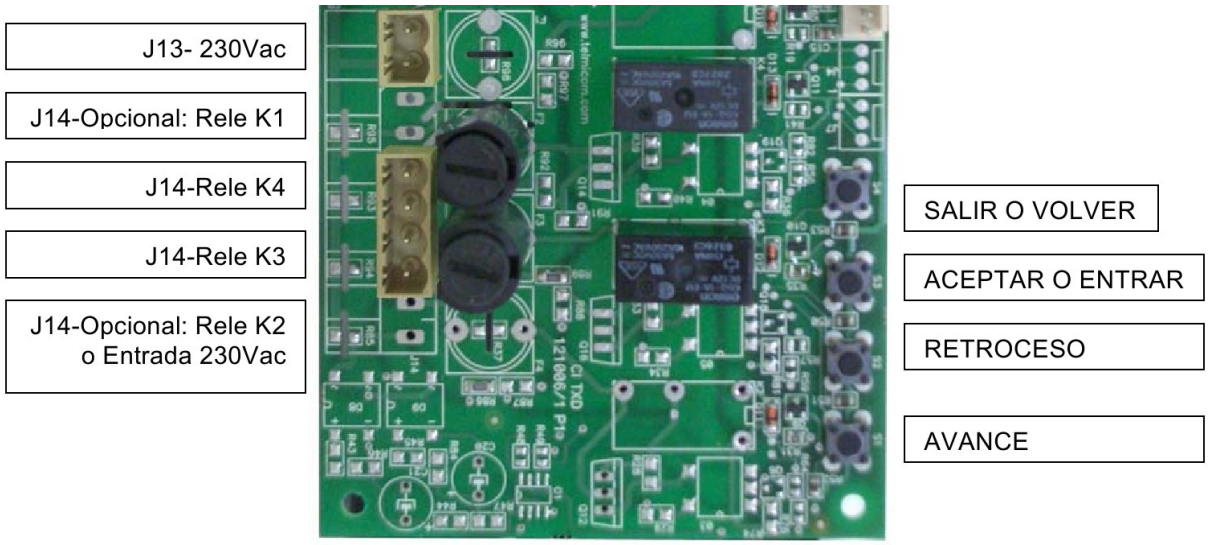

Figura 11. Connectors d'alimentació i dels relés del Telmicom TxD

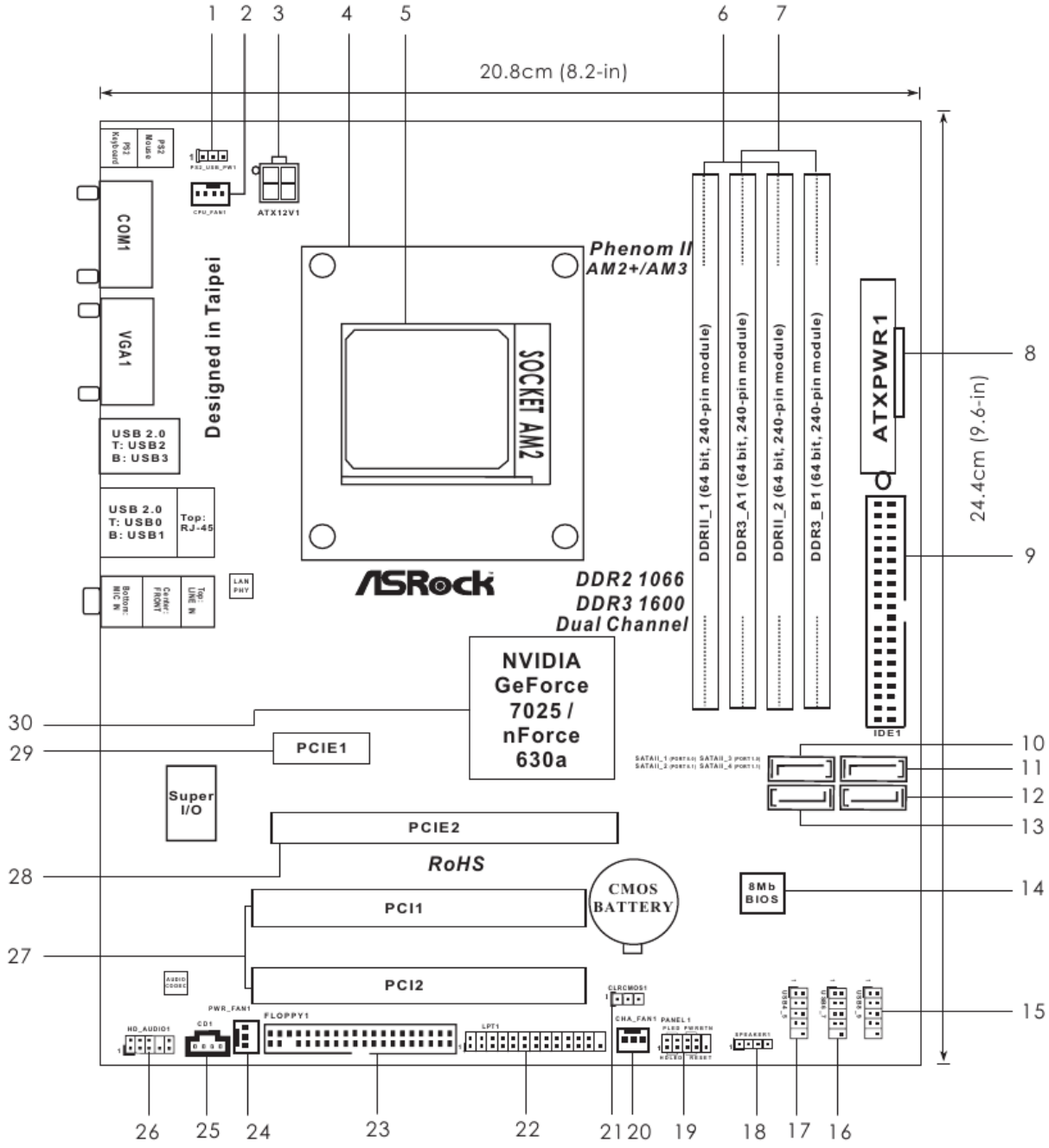

Figura 12. Esquema d'elements de la placa base ASRock N68C-S

PIN	1	2	3
Funció	+5V	Comú	+5VSB (en repós

Taula 5. Pinout del connector 1 de la placa base

10 RESUM DEL PRESSUPOST

El desenvolupament del servidor d'impressió per a ordinadors a monedes, tindrà un cost de set mil nou-cents catorze euros i setanta-vuit cèntims, abans d'impostos.

11 CONCLUSIONS

Per tal d'assolir els objectius d'aquest projecte, s'ha optat per a la integració de varis elements ja existents, per tal de minimitzar el cost tant en desenvolupament de software, com de hardware, i també, el cost unitari del producte final, ja que degut a les previsions de fabricació de poques unitats, no és econòmicament viable assumir el cost de disseny de hardware, certificacions, homologació CE, fabricació de motlles, etcètera.

La solució desenvolupada és una de les possibles, de ben segur que hi ha moltes alternatives a l'aproximació triada, totes amb les seves avantatges i inconvenients, s'ha intentat que la solució adoptada fos la més òptima de cara a la situació que se'ns presentava.

El fet que l'aproximació realitzada sigui la d'integració de sistemes ja existents, fa que el gruix del contingut del projecte sigui de software.

Durant el desenvolupament del projecte, s'han explorat diferents aproximacions possibles per a solucionar cadascun dels mòduls.

En primer lloc, es va avaluar la possibilitat de, en comptes de posar un servidor d'impressió, utilitzar una impressora de xarxa i desenvolupar un driver per a la impressora que fes el comptatge de les pàgines i el cobrament. Aquesta opció presentava l' avantatge d'eliminar el cost del servidor d'impressió, però era més susceptible de ser piratejat (només calia substituir el driver per l'original de la impressora), a part que les impressores de xarxa són més cares que les USB. També cal tenir en compte que s'hauria d'haver desenvolupat el driver tant per Linux com per Windows, cosa que faria que hi hagués molta menys part comuna en el software.

Un cop decidit posar el servidor d'impressió, es va estudiar fer-lo utilitzant un sistema basat en ARM, com ara un router compatible amb openwrt (què no deixa de ser un linux), però presentava el problema que les impressores de baix cost utilitzen un protocol propietari que per a linux només està disponible en binari per a arquitectura tipus intel, i que les limitacions en memòria i velocitat de procés podrien haver limitat la funcionalitat.

Finalment, es va optar per fer els servidors basats en plaques base intel atom, posant el sistema operatiu en un llapis USB, tot i que més tard es van utilitzar micro ordinadors PC Engines ALIX, basats en AMD Geode (compatibles amb intel), per una qüestió de preu.

En quant a les comunicacions amb el moneder, en un principi s'estudiava la possibilitat de

comunicar-s'hi mitjançant un conversor sèrie TTL a USB, que proporciona el mateix fabricant, però al trobar la solució per a tallar tots els ports USB de la placa, i degut a que tant el moneder com la placa disposaven de RS232, es va decidir utilitzar aquest últim. En el cas que en el futur es volgués utilitzar el conversor USB, no seria un problema, ja que aquest es tractat perl sistema operatiu com si d'un port RS232 es tractés.

Per acabar, en quant a la part de software de d'interfície d'usuari, es van buscar un sistema que pogués funcionar tant el linux com en windows, per evitar haver de fer codi específic per a cada plataforma, i degut a que s'havien de fer els comunicacions amb el port sèrie, es van descartar les solucions basades en java. Es van avaluar les llibreries GTK+, Qt i wxWidgets, i finalment es va optar per la primera degut a la seva major implantació.

Com a línies de futur, es podria ampliar el sistema per a permetre amb una sola impressora realitzar impressions en color o en blanc i negre, amb un preu diferent per a cada cas, estudiar de posar el servidor d'impressió en un micro-ordinador basat en ARM per, com per exemple el Raspberry Pie, per tal d'abaratir costos de hardware. Finalment, el que arriba des del sector és que el negoci dels ordinadors per monedes va a la baixa, mentre que l'accés a internet mitjançant wifi de pagament està en auge, així dons, una bona línia de futur seria la integració del sistema d'impressió amb algun software de hot-spot, així com possibilitar la impressió des de dispositius mòbils (telèfons, tauletes ...).

Finalment, dir que l'objectiu del projecte ha estat assolit amb èxit, ja que hi ha una desena de sistemes funcionant en diferents establiments del client des de fa ja uns anys, que no han donat problemes.

Ivan Lorca Aragó

Enginyeria tècnica industrial, especialitat en electrònica industrial

Girona, a 1 de agost de 2012

12 RELACIÓ DE DOCUMENTS

Aquest projecte consta de quatre documents: la memòria, el plec de condicions, l'estat d'amidaments i el pressupost.

13 BIBLIOGRAFIA

APPLE INC. CUPS Documentation. (http://www.cups.org/documentation.php, 25 de juliol de 2.012)

JEROME ALET. pkpgcounter's homepage (http://www.pykota.com/software/pkpgcounter, 25 de juliol de 2.012)

JEROME ALET. Tea4CUPS HomePage. (http://www.pykota.com/software/tea4cups, 25 de juliol de 2.012)

MONEY CONTROLS. cctalk Serial Communication Protocol. Generic Specification. Issue 4.4. (http://www.elektronika.rs.ba/includes/projekti/ccTalk/cctalk44-1.pdf, 25 de juliol de 2.012)

TELMICOM. Documentación Técnica. Protocolo para el control remoto del equipo TxD. Rev. 1.

TELMICOM. Manual Técnico TxD. Rev 4.

THE GTK+ TEAM. The GTK+ Project Documentation (http://www.gtk.org/documentation.php, 25 de juliol de 2.012)

14 GLOSSARI

CGI (Common Gateway Interface): És un sistema estàndard (dels servidors webs per a delegar la generació del contingut a un fitxer executable extern.

GIF (Graphics Interchange Format): És un format per a la compressió d'arxius d'imatge més populars a Internet i fou desenvolupat per CompuServe

GNU (GNU's Not Unix): És un sistema operatiu a l'estil d'Unix de caràcter lliure, entenent com a sistema operatiu una col·lecció d'aplicacions, biblioteques i eines de programació, a més d'un programa que allotja els recursos i es comunica amb el maquinari, anomenat nucli.

HP/GL (Hewlett-Packard Graphics Language): És un llenguatge de control d'impressora desenvolupat per Hewlett Packard per als seus plòters, que més tard s'ha convertit en estàndard de facto per a la gran majoria de plòters.

HTTP (HyperText Transfer Protocol): Estableix el protocol per a l'intercanvi de documents d'hipertext i multimèdia al web.

IPP (Internet Printing Protocol): És un protocol estàndard per a impressions remotes així com per manegar els treballs d'impressió, els tamanys de paper, la resolució d'impressió, etc.

JPEG (Joint Photographic Experts Group): és un algorisme dissenyat per a comprimir imatges estacionàries amb 24 bits de profunditat o en escala de grisos.

LPD (Line Printer Daemon): És un protocol per crear tasques d'impressió a una impressora remota. La primera implantació fou al Berkeley printing system, al sistema operatiu BSD UNIX.

Llenguatge C: Llenguatge de programació creat per Dennis Ritchie i Ken Thompson als Laboratoris Bell d'AT&T, a principis de la dècada dels 70.

Llenguatge Python: Python és un llenguatge de programació d'alt nivell de propòsit general creat per Guido van Rossum l'any 1.991.

Llicència GNU GPL (GNU General Public Licene): És la llicència de la majoria dels programes del Projecte GNU i de més de la meitat dels paquets de programari lliure.

Llicència GNU LGPL (GNU Lesser General Public Licene): S'utilitza en algunes (no pas totes) biblioteques del Projecte GNU i permet l'ús de la biblioteca en programes privatius.

Mac OS X: El Mac OS X és una versió del sistema operatiu que utilitzen els ordinadors Macintosh, i que es basa en un nucli Unix.

MD5: És un algorisme criptogràfic de reducció de 128 bits que permet l'obtenció d'una petita "signatura" o resum característic per a cada bloc de dades original que el representa.

PCL (Printer Command Language): És un llenguatge de descripció de pàgines molt sofisticat, desenvolupat per Hewlett-Packard (HP) per a impressores d'injecció de tinta.

PDF (Portable Document Format): És una forma d'emmagatzematge de documents, desenvolupat per l'empresa Adobe. És un dels formats més extesos tant a Internet com en empreses i governs.

PNG (Portable Network Graphics): Format d'imatge pensat especialment per a pàgines web, desenvolupat pel W3C amb la intenció de substituir el format GIF, que estava subjecte a patents, a Internet.

Postscript: Llenguatge de descripció de pàgines, utilitzat en moltes impressores i, de manera usual, com a format de transport de fitxers gràfics en tallers d'impressió professional.

RS232: És una interfície que designa una norma per l'intercanvi en sèrie de dades binaries entre un equip terminal de dades i un DCE (Data Communication Equipment), tot i que existeixen altres situacions on pot ser utilitzat.

Shell-script: És un programa que l'executa l'interpret de línia de comandes d'un sistema operatiu interpretant directament el codi font.

SMB (Server Message Block): És un protocol de xarxa desenvolupat per Microsoft, que permet compartir fitxers i impressores dins d'una xarxa informàtica.

Socket: Designa un concepte abstracte pel qual dos programes (possiblement situats a ordinadors diferents) poden intercanviar qualsevol flux de dades, generalment de manera fiable i ordenada. Tot socket està definit per una adreça de socket. L'adreça de socket és una combinació de tres elements: una adreça IP, un protocol de transport i un número de port.

TCP (Transmission Control Protocol): És un protocol orientat a la connexió dintre del nivell de transport del model OSI que permet l'entrega de paquets de manera fiable.

Tipus MIME (Multi-Purpose Internet Mail Extensions): És un estàndard per adjuntar arxius a missatges de correu electrònic, com ara gràfics, documents de processadors de text, arxius de so, etc. A més de programari de correu electrònic, l'estàndard MIME s'usa per identificar els arxius que s'envien a clients Web, nous formats de fitxers es poden acomodar simplement actualitzant la llista de browsers de parells de tipus MIME i el programari apropiat per a gestionar cada tipus.

TTL (Transistor-Transistor Logic): És una tecnologia de construcció de circuits electrònics digitals. La seva tensió d'alimentació característica es troba compresa entre els 4,75 v i els 5,25 V. Els nivells lògics venen definits per el rang de tensió compresa entre 0,2 V i 0,8 V per l'estat L (baix) i entre 2,4 V i Vcc per l'estat H (alt).

UNIX: És un sistema operatiu creat el 1969 a l'empresa AT&T Bell, amb la participació de Ken Thomson, Dennis Ritchie i Douglas McIlroy, entre d'altres.

URI (Uniform Resource Identifier): Text curt que identifica sense equivocació qualsevol recurs (servei, pàgina, document, direcció de correu electrònic, enciclopèdia ...) accessible en una xarxa.

USB (Universal Serial Bus): És un port que serveix per connectar perifèrics a un ordinador. Va ser creat el 1996 per set empreses (que actualment en formen el consell directiu): IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation i NEC.

Web: És una xarxa de pàgines escrites en hipertext mitjançant el llenguatge de marcatge HTML i connectades entre sí mitjançant vincles, de manera que formin un sol cos de coneixement pel qual s'hi pot navegar fàcilment.

Windows: És una sèrie de sistemes operatius i interfícies gràfiques d'usuari produïts per Microsoft.

WYSIWYG (What You See Is What You Get): En l'àmbit de la programació i el disseny es refereix a la tecnologia informàtica que permet que el que es veu durant l'edició o programació es correspongui més o menys acuradament amb el resultat final (imprès o en pantalla).

X11: És un sistema de finestres per mostrar mapes de bits. És la interfície gràfica estàndard en els sistemes Unix, *NIX (Linux, BSD, ...) i OpenVMS, i és disponible per a la majoria dels sistemes operatius moderns.

XML (eXtensible Markup Language): és un metallenguatge extensible, d'etiquetes, desenvolupat pel World Wide Web Consortium (W3C).

A CODI INFORMÀTIC

A.1 Programa de l'ordinador

A.1.1 Fitxer serial.h

```
#ifndef __SERIAL_H__
#define __SERIAL_H__

#define SERIAL_PARITY_NONE  0
#define SERIAL_PARITY_EVEN  1
#define SERIAL_PARITY_ODD   2
#define SERIAL_PARITY_MARK  3
#define SERIAL_PARITY_SPACE 4

struct serial_config {
       int speed;
       int stop_bits;
       int word_size;
       int parity;
};

typedef struct serial serial_t;
typedef struct serial_config serial_config_t;
typedef void (*serial_handler_t)(serial_t *serial, int serial_event);

serial_t *serial_init();
serial_handler_t serial_notification_handler(serial_t *serial, serial_handler_t
handler);
int serial_open(serial_t *serial, const char *device, serial_config_t *config);
int serial_close(serial_t *serial);
void serial_release(serial_t *serial);
int serial_read(serial_t *serial, void *data, int length, long timeout);
int serial_write(serial_t *serial, void *data, int length);
void serial_empty_data(serial_t *serial);

serial_config_t *serial_config_create(int speed, const char *options);
#define serial_config_create()   serial_config_create(-1, NULL)
void serial_config_release(serial_config_t *config);

char *serial_device(int port_number, int usb);
int serial_read_bytes_retrys(serial_t * serial, void * buff, int length, int
retries, long tout);
```

```
int serial_write_bytes_retrys(serial_t * serial, void * buff, int length, int
retries);
int serial_write_bytes(serial_t * serial, void * buff, int length);
int serial_read_bytes(serial_t * serial, void * buff, int length);
#endif
```

A.1.2 Fitxer serial-common.c

```
#include "serial.h"

int serial_read_bytes(serial_t * serial, void * buff, int length)
{
	return serial_read_bytes_retrys(serial, buff, length, 2, 200);
}

int serial_read_bytes_retrys(serial_t * serial, void * buff, int length, int
retries, long tout)
{
	int readed = 0, tmp,i = 0;
	while ((readed < length) && (i < retries)) {
			tmp = serial_read(serial, buff + readed, length - readed, tout);
			if(tmp < 0) return -1;
			if(tmp == 0) i++;
			readed+=tmp;
		}
	return readed;
}
int serial_write_bytes_retrys(serial_t * serial, void * buff, int length, int
retries)
{
	int written = 0, tmp,i = 0;
	while ((written < length) && (i < retries)) {
			tmp = serial_write(serial, buff + written, length - written);
			if(tmp < 0) return -1;
			if(tmp == 0) i++;
			written+=tmp;
		}
	return written;
}

int serial_write_bytes(serial_t * serial, void * buff, int length)
{
```

```
	return serial_write_bytes_retrys(serial, buff, length, 1);
}
```

A.1.3 Fitxer serial-win32_overlapped.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <windows.h>
#include "serial.h"
#define CHARPTR(ptr)    ((char *)(ptr))
#define MIN(a, b) ((a) >= (b) ? (b) : (a))

#undef serial_config_create
#undef serial_has_data

struct serial {
	HANDLE h;
	serial_handler_t handler;
	OVERLAPPED read_overlapped;
	OVERLAPPED write_overlapped;
	int using_overlapped;
	char olbuf[1024];
	DWORD readed;
};

serial_t *serial_init()
{
	serial_t *s;

	if (!(s = (serial_t *)malloc(sizeof(struct serial)))) return 0;
	memset(s, 0, sizeof(struct serial));
	s->h = INVALID_HANDLE_VALUE;

	return s;
}

serial_handler_t serial_notification_handler(serial_t *serial, serial_handler_t
handler)
{
	serial_handler_t old;
```

```
	old = serial->handler;
	serial->handler = handler;

	return old;
}

int serial_open(serial_t *serial, const char *device, serial_config_t *config)
{
	DCB dcb;
	DWORD dwError;
	BOOL fSuccess;
	COMMTIMEOUTS commTimeouts;
	HANDLE hCom;

	hCom = CreateFile(device,
		GENERIC_READ | GENERIC_WRITE,
		0,	/* comm devices must be opened w/exclusive-access */
		NULL, /* no security attrs */
		OPEN_EXISTING, /* comm devices must use OPEN_EXISTING */
		FILE_ATTRIBUTE_NORMAL | FILE_FLAG_WRITE_THROUGH |
FILE_FLAG_OVERLAPPED,	/* not overlapped I/O */
		NULL  /* hTemplate must be NULL for comm devices */
	);

	if (hCom == INVALID_HANDLE_VALUE) {
		logger("Error: unable to open port %s handle", device);
		return -1;
	}

	logger("Seting Comm properties");
	if (!GetCommState(hCom, &dcb)) {
		logger("Error: unable to retrieve port %s properties", device);
		CloseHandle(hCom);
		return -1;
	}

	dcb.BaudRate = config->speed;
	dcb.ByteSize = config->word_size;
	switch (config->parity) {
		case SERIAL_PARITY_NONE:
			dcb.Parity = NOPARITY;
```

```
                        break;

                case SERIAL_PARITY_EVEN:
                        dcb.Parity = EVENPARITY;
                        break;

                case SERIAL_PARITY_ODD:
                        dcb.Parity = ODDPARITY;
                        break;

                case SERIAL_PARITY_MARK:
                        dcb.Parity = MARKPARITY;
                        break;

                case SERIAL_PARITY_SPACE:
                        dcb.Parity = SPACEPARITY;
                        break;

                default:
                        logger("Error: Invalid parity (%d) at port %s.", config->parity,
device);
                        return -1;
        }

        switch (config->stop_bits) {
                case 1: dcb.StopBits = ONESTOPBIT; break;
                case 15: dcb.StopBits = ONE5STOPBITS; break;
                case 2: dcb.StopBits = TWOSTOPBITS; break;
                default:
                        logger( "Error: Invalid stop bits (%f) at port %s. Valid values
are 1 15 2", config->stop_bits, device);
                        return -1;
        }

        if (!SetCommState(hCom, &dcb)) {
                logger( "Error: unable to set port %s properties", device);
                CloseHandle(hCom);
                return -1;
        }

        logger( "Setting Comm Timeouts   ");
        if (!GetCommTimeouts(hCom, &commTimeouts)) {
```

```
		logger("Error: unable to get port %s timeouts", device);
		CloseHandle(hCom);
		return -1;
	}

	commTimeouts.ReadIntervalTimeout=0;		//timeout between chars in
ms
	commTimeouts.ReadTotalTimeoutMultiplier = 0;	//total timeout =
	commTimeouts.ReadTotalTimeoutConstant = 1500;  // ReadTotalTimeoutConstant +
numchar*ReadTotalTimeoutMultiplier
	commTimeouts.WriteTotalTimeoutMultiplier = 0;
	commTimeouts.WriteTotalTimeoutConstant = 1500;

	if (!SetCommTimeouts(hCom, &commTimeouts)) {
		logger("Error: unable to set port %s timeouts", device);
		CloseHandle(hCom);
		return -1;
	}

	logger("Serial: Connected to port %s, handle: %p", device, hCom);
	serial->h = hCom;

	SetCommMask(hCom, EV_RXCHAR);
	serial->using_overlapped = 1;
	serial->read_overlapped.Offset = 0;
	serial->read_overlapped.OffsetHigh = 0;
	serial->read_overlapped.hEvent = 0;
	serial->write_overlapped.Offset = 0;
	serial->write_overlapped.OffsetHigh = 0;
	serial->write_overlapped.hEvent = 0;

	return 0;
}

int serial_close(serial_t *serial)
{
	CloseHandle(serial->h);
	serial->h = INVALID_HANDLE_VALUE;

	return 0;
}
```

```
void serial_release(serial_t *serial)
{
	if (serial->h != INVALID_HANDLE_VALUE) serial_close(serial);
	free(serial);
}

int serial_write(serial_t *serial, void *data, int length)
{
	DWORD written;
	DWORD err;
	if (!WriteFile(serial->h, data, length, &written, &(serial-
>write_overlapped))) {
		err = GetLastError();
		if(err == ERROR_IO_PENDING) {
//			logger("Serial: Write Async\n");
			WaitForSingleObject(serial->h,INFINITE);
			err  = GetOverlappedResult(serial->h, &(serial-
>write_overlapped), &written,FALSE);
			FlushFileBuffers(serial->h);
//			logger("Serial: Wrote  %d of %d %x\n", written, length, err);
			return length;
		}
//		logger("Serial: Error: unable to write. error=%d", err);
		return -1;
	}

	return written;
}

int serial_read(serial_t *serial, void *data, int length, long timeout)
{
	DWORD readed;
	DWORD err;

	if (!ReadFile(serial->h, data, length, &readed, &(serial->read_overlapped)))
{
		err = GetLastError();
		if(err == ERROR_IO_PENDING) {
//			logger("Serial: Read Async\n");
			if((err=WaitForSingleObject(serial->h, timeout?
timeout:INFINITE)) != WAIT_OBJECT_0) {
//				logger("Serial: Read timeout! %x \n", err);
```

```
                        CancelIo(serial->h);
                }
                GetOverlappedResult(serial->h, &(serial->read_overlapped),
&readed,FALSE);
//              printf("Serial: Readed  %d of %d\n", readed, length);
                return readed;
        }
//      logger("Serial: Error: unable to read. error=%d", err);
        return -1;
    }

    return readed;
}


serial_config_t *serial_config_create(int speed, const char *options)
{
    struct serial_config cfg;
    serial_config_t *r;

    if (!options) {
        return (serial_config_t *)malloc(sizeof(struct serial_config));
    }

    cfg.speed = speed;

    switch (options[0]) {
        case '7':
        case '8':
            cfg.word_size = options[0] - '0';
            break;

        default:
            return 0;
    }

    switch (options[1]) {
        case 'N':
        case 'P':
            break;

        default:
```

```
                return 0;
        }

        switch (options[2]) {
                case '1':
                case '2':
                        cfg.stop_bits = options[2] - '0';
                        break;

                default:
                        return 0;
        }

        if (options[3] != 0) {
                return 0;
        }

printf("speed: %d\nword: %d\nstop: %d\n", cfg.speed, cfg.word_size, cfg.stop_bits);

        if (!(r = (serial_config_t *)malloc(sizeof(struct serial_config)))) {
                return 0;
        }
        memcpy(r, &cfg, sizeof(struct serial_config));
        return r;
}

void serial_config_release(serial_config_t *config)
{
        free(config);
}

char *serial_device(int port_number, int usb)
{
    char buf[30];
    sprintf(buf, "COM%d", port_number);
    return strdup(buf);
}
```

A.1.4 Fitxer serial-linux.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include "serial.h"

#undef serial_config_create
#undef serial_has_data

struct serial {
	int fd;
	serial_handler_t handler;
};

serial_t *serial_init()
{
	serial_t *s;

	if (!(s = (serial_t *)malloc(sizeof(struct serial)))) return 0;
	memset(s, 0, sizeof(struct serial));
	s->fd = -1;

	return s;
}

serial_handler_t serial_notification_handler(serial_t *serial, serial_handler_t
handler)
{
	serial_handler_t old;

	old = serial->handler;
	serial->handler = handler;

	return old;
}


struct serial_speed {
	int id;
	int baudrate;
};

struct serial_speed serial_speeds[] = {
```

```
	{ B0, 0 },
	{ B50, 50 },
	{ B75, 75 },
	{ B110, 110 },
	{ B134, 134 },
	{ B150, 150 },
	{ B200, 200 },
	{ B300, 300 },
	{ B600, 600 },
	{ B1200, 1200 },
	{ B1800, 1800 },
	{ B2400, 2400 },
	{ B4800, 4800 },
	{ B9600, 9600 },
	{ B19200, 19200 },
	{ B38400, 38400 },
	{ B57600, 57600 },
	{ B115200, 115200 },
	{ B230400, 230400 },

	{ -1, -1 }
};


int internal_serial_speed(int baudrate)
{
	struct serial_speed *p;

	for (p = serial_speeds; p->baudrate != -1; p++) {
//		printf("id=%d; baudrate=%d;\n", p->id, p->baudrate);
		if (p->baudrate == baudrate) return p->id;
	}

	return -1;
}


int serial_open(serial_t *serial, const char *device, serial_config_t *config)
{
	struct termios tconfig;
	int flags;
	int speed;
```

```
	if ((speed = internal_serial_speed(config->speed)) < 0) {
		return -1;
	}

	switch (config->word_size) {
		case 8:
			flags = CS8;
			break;
		case 7:
			flags = CS7;
			break;
		case 6:
			flags = CS6;
			break;
	}

	switch (config->stop_bits) {
		case 1:
			break;
		case 2:
			flags |= CSTOPB;
			break;
	}

	switch (config->parity) {
		case SERIAL_PARITY_NONE:
			break;

		case SERIAL_PARITY_EVEN:
			flags |= PARENB | PARODD;
			break;

		case SERIAL_PARITY_ODD:
			flags |= PARENB;
			break;

//		case SERIAL_PARITY_MARK:
//		case SERIAL_PARITY_SPACE:
		default:
			close(serial->fd);
			serial->fd = -1;
```

```
                    return -1;
        }
        if ((serial->fd = open(device, O_RDWR | O_NOCTTY)) < 0) {
                return -1;
        }
        printf("serial open %s fd: %d\n", device,serial->fd);

//printf("flags=%x; speed=%d;\n", flags, speed);
        tcgetattr(serial->fd, &tconfig);
        tconfig.c_cflag = flags;
        tconfig.c_iflag = 0;
        tconfig.c_oflag = 0;
        tconfig.c_lflag = 0;
        tconfig.c_cc[VMIN] = 1;
        tconfig.c_cc[VTIME] = 0;
        cfsetispeed(&tconfig, speed);
        tcsetattr(serial->fd, TCSANOW, &tconfig);

        return 0;
}


int serial_close(serial_t *serial)
{
        close(serial->fd);
        serial->fd = -1;

        return 0;
}

void serial_release(serial_t *serial)
{
        if (serial->fd != -1) serial_close(serial);
        free(serial);
}

void serial_empty_data(serial_t *serial)
{
        lseek(serial->fd, 0, SEEK_END);
}

int serial_read(serial_t *serial, void *data, int length, long timeout)
```

```
{
	printf(" serial_has_read %d %d %d\n", serial->fd,length, timeout);
	struct timeval tv;
	fd_set fds;
	FD_ZERO(&fds);
	FD_SET(serial->fd, &fds);
	tv.tv_sec = 0;
	tv.tv_usec = timeout * 1000;
	if (timeout> 0) {
		if (select(serial->fd + 1, &fds, NULL, NULL, &tv) != 1) {
			return -1;
		}
	} else {
		if (select(serial->fd + 1, &fds, NULL, NULL, NULL) != 1) {
			return -1;
		}
	}

	int reb = read(serial->fd, data, length);
	int i;
	unsigned char* d = (unsigned char* ) data;
	printf(" ** serial_read rebut (%d/%d):",reb,length);
	for (i=0; i < reb; i++) printf(" %02x",d[i]);
	printf("\n");
	return reb;
}

int serial_write(serial_t *serial, void *data, int length)
{
	int i;
	int t;
	unsigned char* d = (unsigned char* ) data;
	printf(" ** serial_write (%d) enviant:", serial->fd);
	for (i=0; i < length; i++) printf(" %02x",d[i]);
	i = write(serial->fd, data, length);
	perror("HOLA");
	printf(" %s (%d)\n",i>0?"OK":"KO", i);
	return i;
}

serial_config_t *serial_config_create(int speed, const char *options)
{
```

```
struct serial_config cfg;
serial_config_t *r;

if (!options) {
	return (serial_config_t *)malloc(sizeof(struct serial_config));
}

cfg.speed = speed;

switch (options[0]) {
	case '7':
	case '8':
		cfg.word_size = options[0] - '0';
		break;

	default:
		return 0;
}

switch (options[1]) {
	case 'N':
	case 'P':
		break;

	default:
		return 0;
}

switch (options[2]) {
	case '1':
	case '2':
		cfg.stop_bits = options[2] - '0';
		break;

	default:
		return 0;
}

if (options[3] != 0) {
	return 0;
}
```

```
printf("speed: %d\nword: %d\nstop: %d\n", cfg.speed, cfg.word_size, cfg.stop_bits);

	if (!(r = (serial_config_t *)malloc(sizeof(struct serial_config)))) {
		return 0;
	}

	memcpy(r, &cfg, sizeof(struct serial_config));

	return r;
}


void serial_config_release(serial_config_t *config)
{
	free(config);
}

char *serial_device(int port_number, int usb)
{
    char buf[30];

    sprintf(buf, usb?"/dev/ttyUSB%d":"/dev/ttyS%d", port_number - 1);

    return strdup(buf);
}
```

A.1.5 Fitxer telmicom.h

```
#ifndef __TELMICOM_H__
#define __TELMICOM_H__
#include "serial.h"

#define TELMICOM_ERR_IO -1
#define TELMICOM_ERR_CHECKSUM -2
#define TELMICOM_ERR_ADDRESS -2

struct telmicom_data {
	unsigned char dst;
	unsigned char org;
	unsigned char cmd;
	unsigned char len;
	unsigned char data[256];
	unsigned char cs;
```

```
};

typedef struct telmicom_data telmicom_data_t;

int telmicom_sendcommand(serial_t * serial, telmicom_data_t* command,
telmicom_data_t* response);
void print_telmicom_data(const char * label, const telmicom_data_t* command);
int telmicom_moneda(serial_t * serial, unsigned char org, unsigned char dst);
int telmicom_temps(serial_t * serial, unsigned char org, unsigned char dst);
int telmicom_settemps(serial_t * serial, unsigned char org, unsigned char dst, int
segons);
double telmicom_get_tarifa(serial_t * serial, unsigned char org, unsigned char
dst);
int telmicom_set_tarifa(serial_t * serial, unsigned char org, unsigned char dst,
double tarifa);
int telmicom_descuento(serial_t * serial, unsigned char org, unsigned char dst, int
activar);
int telmicom_set_tarifa(serial_t * serial, unsigned char org, unsigned char dst,
double tarifa);
int telmicom_descuento(serial_t * serial, unsigned char org, unsigned char dst, int
activar);
int telmicom_setText(serial_t * serial, unsigned char org, unsigned char dst, const
char * text);
int telmicom_setFuncionesOciomatik(serial_t * serial, unsigned char org, unsigned
char dst,
            int activarReset, int tempsReset, int activarWatchdog, int
tempsWatchdog,
            int activarPower, int tempsPower);
int telmicom_setMonedes(serial_t * serial, unsigned char org, unsigned char dst,
int monedes[]);
int telmicom_getFuncionesOciomatik(serial_t * serial, unsigned char org, unsigned
char dst,
            int *activarReset, int *tempsReset, int *activarWatchdog, int
*tempsWatchdog,
            int *activarPower, int *tempsPower);
int telmicom_getMonedes(serial_t * serial, unsigned char org, unsigned char dst,
int *monedes);
int telmicom_inhabilita(serial_t * serial, unsigned char org, unsigned char dst,
int inhabilita);
int telmicom_getText(serial_t * serial, unsigned char org, unsigned char dst, char
* text);
#endif
```

A.1.6 Fitxer telmicom.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include "telmicom.h"

int telmicom_byteToInt(unsigned char *b, int length)
{
	int v = 0, i;
	for(i=length;i; i--) v = (v << 8) + (b[i-1] & 0x000000ff);
	return v;
}

void telmicom_intToByte(unsigned char *b, int v, int length)
{
	int i;
	for(i=0;i < length; i++) {
		b[i] = v & 0x000000ff;
		v >>=  8;
	}
}


unsigned char checksum(const telmicom_data_t* command)
{
	unsigned char cs=0;
	int i;
	cs += command->dst;
	cs += command->org;
	cs += command->cmd;
	cs += command->len;
	for (i = 0; i < command->len; i++)cs += command->data[i];
	return -cs;
}

void calculate_checksum(telmicom_data_t* command)
{
	command->cs = checksum(command);
}
```

```
int test_checksum(telmicom_data_t* command)
{
	return command->cs == checksum(command);
}


int _telmicom_sendcommand(serial_t * serial, telmicom_data_t* command,
telmicom_data_t* response, int retries, int tout)
{
	unsigned char buff[261];
	int numread=0;
	int numwrote=0;
	int i;
	//print_telmicom_data("Telmicom enviant: ",command);
	calculate_checksum(command);
	buff[0] = command->dst;
	buff[1] = command->len;
	buff[2] = command->org;
	buff[3] = command->cmd;
	memcpy(buff+4,command->data, command->len);
	buff[4+(command->len)] =  command->cs;
	if(serial_write_bytes(serial, buff, (command->len)+5) < (command->len)+5)
{ /*printf("error a serial_write_bytes\n");*/ goto errio;}
	memset(buff,0,261);
	memset(response,0,sizeof(telmicom_data_t));
	if((numread = serial_read_bytes_retrys(serial, buff, 2, retries, tout)) < 2)
{ /*printf("error a serial_read 1\n");*/ goto errio;}
	if((numread = serial_read_bytes_retrys(serial, buff+2,buff[1]+3, retries,
tout)) < buff[1]+3)  { /*printf("error a serial_read 2\n");*/ goto errio;}
	response->dst = buff[0];
	response->len = buff[1];
	response->org = buff[2];
	response->cmd = buff[3];
	memcpy(response->data,buff+4,buff[1]);
	response->cs = buff[buff[1]+4];
	if(!test_checksum(response)) { /*printf("error a chacksum\n");*/ return
TELMICOM_ERR_CHECKSUM;}
	if(command->org != response->dst || command->dst != response->org)
		return TELMICOM_ERR_ADDRESS;
	return 0;
```

```
	errio:
//		for (i=0; i < numread; i++) printf(" %x",buff[i]);
//		printf("\n");
		//serial_empty_data(serial);
		return TELMICOM_ERR_IO;
}

int telmicom_sendcommand_retrys(serial_t * serial, telmicom_data_t* command,
telmicom_data_t* response, int serial_retries, int serial_tout, int
command_retries)
{
	int i;
	for (i=0; i < command_retries; i++) {
		if(!_telmicom_sendcommand(serial,command,response, serial_retries,
serial_tout)) return 0;
	}
	return TELMICOM_ERR_IO;
}

int telmicom_sendcommand(serial_t * serial, telmicom_data_t* command,
telmicom_data_t* response)
{
	telmicom_sendcommand_retrys(serial,command,response, 1, 200, 5);
}

void print_telmicom_data(const char * label, const telmicom_data_t* command)
{
	int i;
	printf("%s: dst %x len %x org %x cmd %x Dades
[",label==NULL?"":label,command->dst, command->len, command->org, command->cmd);
	for (i=0; i < command->len; i++) printf("%02x ",command->data[i]);
	printf("] cs %x (%x)\n",command->cs, checksum(command) );
}

int telmicom_moneda(serial_t * serial, unsigned char org, unsigned char dst)
{
	telmicom_data_t comanda, resposta;
	int res;
	comanda.cmd = 0xff;
	comanda.len = 0x02;
	comanda.org = org;
	comanda.dst = dst;
```

```
	comanda.data[0] = 0x6a;
	comanda.data[1] = 0x00;
	res = telmicom_sendcommand_retrys(serial, &comanda, &resposta,1,200,5);
	if(res) {
/*		printf("********* Error a telmicom_moneda: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
*/		return res;
	}
	return resposta.data[1];
}

int telmicom_inhabilita(serial_t * serial, unsigned char org, unsigned char dst,
int inhabilita)
{
	telmicom_data_t comanda, resposta;
	int i,res;
	char buff[40];
	char errStr[] = " INET ERR";
	char * t;
	comanda.cmd = 0xff;
	comanda.len = 0x01;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x63;
	res = telmicom_sendcommand_retrys(serial, &comanda, &resposta,1,300,5);
	print_telmicom_data("------------------- telmicom_inhabilita_res",
&resposta);
	printf("res = %d (%d)\n",res,inhabilita);
	if(res||(resposta.len != 15))return -1;
	comanda.len=16;
	memcpy(comanda.data+1, resposta.data,15);
	comanda.data[0] = 0x65;
	if(inhabilita) {
		comanda.data[15] |= 0x08;
//		comanda.data[15] = 0x08;
	} else {
		comanda.data[15] &= ~0x08;
//		comanda.data[15] = 0x00;
	}
	print_telmicom_data("------------------- telmicom_inhabilita_cmd", &comanda);
	res = telmicom_sendcommand(serial, &comanda, &resposta),1,300,5;
```

```
	if(res) return -2;
	res = telmicom_getText(serial, org, dst, buff);
	if(res) return -3;
	printf("------------------- telmicom_inhabilita str: %s\n", buff);
	if(inhabilita) {
		printf("------------------- telmicom_inhabilita 1 str: %s", buff);
		if (((t = strstr(buff,errStr)) == NULL) && (strlen(buff) +
strlen(errStr) < 32)) {
			printf("------------------- telmicom_inhabilita 2 str: %s\n",
buff);
			strcat(buff,errStr);
		}
	} else {
		printf("------------------- telmicom_inhabilita 3 str: %s\n", buff);
		if (((t = strstr(buff,errStr)) != NULL) && (strlen(t) ==
strlen(errStr))) {
			printf("------------------- telmicom_inhabilita 4 str: %s\n",
buff);
			*t = 0;
		}
	}
	res = telmicom_setText(serial, org, dst, buff);
	printf("------------------- telmicom_inhabilita 5 str: %s res: %d\n", buff,
res);
	if(res) return -4;
	return 0;
}

int telmicom_temps(serial_t * serial, unsigned char org, unsigned char dst)
{
	telmicom_data_t comanda, resposta;
	int t1,t2,res;
	comanda.cmd = 0xff;
	comanda.len = 0x01;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x63;
	res = telmicom_sendcommand_retrys(serial, &comanda, &resposta,1,200,5);
	if(res) {
/*		printf("********* Error a telmicom_temps: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
```

```
*/            return res;
        } else {
/*            printf("********* Got telmicom_temps: %d\n",res);
              print_telmicom_data("\tCommanda:", &comanda);
              print_telmicom_data("\tResposta:", &resposta);
*/      }
        if(resposta.len != 15) {
//            printf("********* Error a telmicom_temps: len %d\n",resposta.len);
              return -99;
        }
        t1 = telmicom_byteToInt(resposta.data,4);
        t2 = telmicom_byteToInt(resposta.data+8,4);
//      printf("t1:%d,t2:%d, t: %d \n", t1, t2, t1+t2);
        return t1+t2;
}

int telmicom_settemps(serial_t * serial, unsigned char org, unsigned char dst, int
segons)
{
        printf("telmicom_settemps %d\n",segons);
        telmicom_data_t comanda, resposta;
        int t1,t2,res;
        comanda.cmd = 0xff;
        comanda.len = 0x10;
        comanda.org = org;
        comanda.dst = dst;
        comanda.data[0] = 0x65;
        memset(comanda.data+1,0,15);
        telmicom_intToByte(comanda.data+9,segons,4);
        res = telmicom_sendcommand(serial, &comanda, &resposta);
        if(res) {
              printf("********* Error a telmicom_settemps: %d\n",res);
              print_telmicom_data("\tCommanda:", &comanda);
              print_telmicom_data("\tResposta:", &resposta);
              return res;
        }
        return 0;
}

double telmicom_get_tarifa(serial_t * serial, unsigned char org, unsigned char dst)
{
        telmicom_data_t comanda, resposta;
```

```
	int res;
	comanda.cmd = 0xff;
	comanda.len = 0x01;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x68;
	res = telmicom_sendcommand(serial, &comanda, &resposta);
	if(res) {
		printf("********* Error a telmicom_settemps: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
	}
	return ((double)telmicom_byteToInt(resposta.data,2))/100;
}

int telmicom_set_tarifa(serial_t * serial, unsigned char org, unsigned char dst,
double tarifa)
{
	telmicom_data_t comanda, resposta;
	int t1,t2,res;
	comanda.cmd = 0xff;
	comanda.len = 0x03;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x69;
	memset(comanda.data+1,0,15);
	telmicom_intToByte(comanda.data+1,(int)(tarifa*100),2);
	print_telmicom_data("\ttelmicom_set_tarifa Commanda:", &comanda);
	res = telmicom_sendcommand(serial, &comanda, &resposta);
	if(res) {
		printf("********* Error a telmicom_set_tarifa: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
	}
	return 0;
}


int telmicom_descuento(serial_t * serial, unsigned char org, unsigned char dst, int
activar)
```

```
{
	telmicom_data_t comanda, resposta;
	int t1,t2,res;
	comanda.cmd = 0xff;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x67;
	comanda.len = 2;
	if (activar) comanda.data[1] = 0x04;
	else comanda.data[1] = 0x00;
	res = telmicom_sendcommand(serial, &comanda, &resposta);
	if(res) {
		printf("********* Error a telmicom_descuento escriure: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
	}

	return 0;
}

int telmicom_setText(serial_t * serial, unsigned char org, unsigned char dst, const
char * text)
{
	telmicom_data_t comanda, resposta;
	int t1,t2,res;
	comanda.cmd = 0x15;
//	comanda.len = strlen(text) +1;
//	if (comanda.len > 32) comanda.len = 32;
	comanda.len = 32;
	comanda.org = org;
	comanda.dst = dst;
	strncpy(comanda.data,text, 31);
	comanda.data[31]=0;

	res = telmicom_sendcommand_retrys(serial, &comanda, &resposta, 1, 500, 10);
	if(res) {
		printf("********* Error a telmicom_setText escriure: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
	}
```

```
	return 0;
}

int telmicom_setFuncionesOciomatik(serial_t * serial, unsigned char org, unsigned
char dst, int activarReset, int tempsReset, int activarWatchdog, int tempsWatchdog,
int activarPower, int tempsPower)
{
	telmicom_data_t comanda, resposta;
	int t1,t2,res;
	comanda.cmd = 0xff;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x6e;
	comanda.len = 8;
	comanda.data[1] = 0;
	if(activarReset) comanda.data[1] |= 1;
	if(activarWatchdog) comanda.data[1] |= 2;
	if(activarPower) comanda.data[1] |= 4;

	telmicom_intToByte(comanda.data + 2,tempsReset,2);
	telmicom_intToByte(comanda.data + 4,tempsPower,2);
	telmicom_intToByte(comanda.data + 6,tempsWatchdog,2);

	res = telmicom_sendcommand(serial, &comanda, &resposta);
	if(res) {
		printf("********* Error a telmicom_setFuncionesOciomatik escriure:
%d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
	}

	return 0;
}

int telmicom_setMonedes(serial_t * serial, unsigned char org, unsigned char dst,
int monedes[])
{
	telmicom_data_t comanda, resposta;
	int i,res,m=0;
	int monedesMap[] = {0x0010, 0x0040, 0x0200, 0x0400, 0x2000};
```

```
	comanda.cmd = 0xe7;
	comanda.len = 0x02;
	comanda.org = org;
	comanda.dst = dst;
	for (i=0; i < 5; i++) if (monedes[i]) m |= monedesMap[i];
	telmicom_intToByte(comanda.data,m,2);
	res = telmicom_sendcommand(serial, &comanda, &resposta);
	if(res) {
		printf("********* Error a telmicom_descuento escriure: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
	}

	return 0;
}

int telmicom_getFuncionesOciomatik(serial_t * serial, unsigned char org, unsigned
char dst,
		int *activarReset, int *tempsReset, int *activarWatchdog, int
*tempsWatchdog,
		int *activarPower, int *tempsPower)
{
	telmicom_data_t comanda, resposta;
	int t1,t2,res;
	comanda.cmd = 0xff;
	comanda.org = org;
	comanda.dst = dst;
	comanda.data[0] = 0x6d;
	comanda.len = 1;

	res = telmicom_sendcommand(serial, &comanda, &resposta);
	if(res) {
		printf("********* Error a telmicom_getFuncionesOciomatik escriure:
%d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		return res;
/*	} else {
		printf("------------ Get Funciones Ociomatik: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
```

```
*/     }

       *activarReset = (resposta.data[0] & 1)? 1:0;
       *activarWatchdog = (resposta.data[0] & 2)? 1:0;
       *activarPower = (resposta.data[0] & 4)? 1:0;

       *tempsReset = telmicom_byteToInt(resposta.data+1,2);
       *tempsPower = telmicom_byteToInt(resposta.data+3,2);
       *tempsWatchdog = telmicom_byteToInt(resposta.data+5,2);

       return 0;
}

int telmicom_getMonedes(serial_t * serial, unsigned char org, unsigned char dst,
int *monedes)
{
       telmicom_data_t comanda, resposta;
       int i,res,m=0;
       int monedesMap[] = {0x0010, 0x0040, 0x0200, 0x0400, 0x2000};
       comanda.cmd = 0xe6;
       comanda.len = 0x0;
       comanda.org = org;
       comanda.dst = dst;

       res = telmicom_sendcommand(serial, &comanda, &resposta);
       if(res) {
              printf("********* Error a telmicom_descuento escriure: %d\n",res);
              print_telmicom_data("\tCommanda:", &comanda);
              print_telmicom_data("\tResposta:", &resposta);
              return res;
       }

       m = telmicom_byteToInt(resposta.data,2);
       for (i=0; i < 5; i++) monedes[i] = (m & monedesMap[i])? 1:0;
       return 0;
}


int telmicom_getText(serial_t * serial, unsigned char org, unsigned char dst, char
* text)
{
       telmicom_data_t comanda, resposta;
```

```
	int t1,t2,res;
	comanda.cmd = 0x14;
	comanda.len = 0;
	comanda.org = org;
	comanda.dst = dst;

	res = telmicom_sendcommand_retrys(serial, &comanda, &resposta, 1, 500, 10);
	if(res) {
		printf("********* Error a telmicom_getText escriure: %d\n",res);
		print_telmicom_data("\tCommanda:", &comanda);
		print_telmicom_data("\tResposta:", &resposta);
		if(res != TELMICOM_ERR_CHECKSUM) return res;
	}

	strncpy(text,resposta.data, 31);
	text[31]=0;

	return 0;
}
```

A.1.7 Fitxer sign.c

```
#include <string.h>
#include <glib.h>
#include "../../md5lib/md5.h"

#define KEY "ivanlorcaarago"

char *printer_sign(char *key, int id, int pagines, int color)
{
	char md5_f1[16];
	char bufo[33];
	gchar *buf;
	md5_t md5;

	buf = g_strdup_printf("%s:%s:%d:%d:%d", KEY, key, id, pagines, color);
	md5_init(&md5);
	md5_process(&md5, buf, strlen(buf));
	md5_finish(&md5, md5_f1);
	md5_sig_to_string(md5_f1, bufo, 33);

	return strdup(bufo);
}
```

A.1.8 Fitxer main-ui.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <gtk/gtk.h>
#include <glade/glade.h>
#include <signal.h>
#include <sys/types.h>
#include <glib.h>
#include <glib/gstdio.h>
#include "config.h"
#include "serial.h"
#include "splash.h"
#include "session.h"
#include "mouse.h"
#include "telmicom.h"

#ifdef __linux__
#      define PATH_PROGRAM "/cyber"
#      define SCREENSAVER "kdesktop_lock"
#      define EVENT
#      define UNUSED  __attribute__((__unused__))
#     include <unistd.h>
#elif __WIN32__
#      define PATH_PROGRAM "C:/WINDOWS/system32/config/cyber"
#      define EVENT __declspec(dllexport)
#      define UNUSED
#      include <windows.h>
#      include <ws2tcpip.h>
#endif
#ifdef __linux__
       #define msleep(x) usleep(1000*(x))
       #include <sys/socket.h>
       #include <netinet/in.h>
       #include <netdb.h>
       #include <arpa/inet.h>
       #include <sys/wait.h>
#elif __WIN32__
       #define msleep(x) Sleep(x)
       #include <ws2tcpip.h>
```

```
	void WSAAPI freeaddrinfo (struct addrinfo*);
	int WSAAPI getaddrinfo (const char*,const char*,const struct addrinfo*,
struct addrinfo**);
	int WSAAPI getnameinfo(const struct sockaddr*,socklen_t,char*,DWORD,
char*,DWORD,int);
	#define close(x) closesocket(x)
	typedef int socklen_t;
#endif

#define FILE_CONFIG     (PATH_PROGRAM "/cyber.conf")
#define FILE_GLADE   PATH_PROGRAM "/data/resta_credit.glade"
#define MESSAGE_MAX 2000
#define LISTEN_PORT "7777"

typedef enum
{
	esperant, iniciat, cancelat, acceptat, cobrat
} job_status_t;

typedef struct
{
	serial_t *serial;
	GArray *missatges;
	unsigned char org;
	unsigned char dst;
	double preu;
	int temps;
	int idiomes;
	int lang;
	int print_pages;
	double preu_pagina;
	int printer_timeout;
	job_status_t job_status;
	int job_init;
} status_t;

status_t status = {
	NULL,
	NULL,
	1,
	2,
	3.0,
```

```
	0,
	4,
	0,
	0,
	.3,
	30,
	esperant,
	0,
};

GKeyFile *config = NULL;

char text_timer[10];
char *messages[4];
static GtkWidget *window;
char ** messages_ok;
char ** messages_ko;
int idiomes;
int lang=0;
GKeyFile *f;
char message[MESSAGE_MAX + 1];
GladeXML *mainXML = NULL;
int sockfd= -1;
FILE *loggerfile = NULL;

void logger(const char *format, ...)

{
	va_list ap;
	GTimeVal t;

	va_start(ap, format);
	if (loggerfile) {
		g_get_current_time(&t);
		fprintf(loggerfile, "%s: ", g_time_val_to_iso8601(&t));
		vfprintf(loggerfile, format, ap);
		fprintf(loggerfile, "\n");
		fflush(loggerfile);

		g_printf("%s: ", g_time_val_to_iso8601(&t));
		g_vprintf(format, ap);
		g_printf("\n");
```

```
	}

	va_end(ap);
}

void subst_label(char* src, char *label, char* value)
{
//	char format[16];
	char buff[MESSAGE_MAX + 1];
	char *p, *q;
	int i = 0,len = strlen(label), len_val = strlen(value);
	p = src;
	while ((q = strstr(p,label)) && i < MESSAGE_MAX) {
		*q =  0;
		strncpy(buff + i, p, MESSAGE_MAX - i);
		i += q - p;
		strncpy(buff + i, value, MESSAGE_MAX - i);
		i += len_val;
		p = q + len;
	}
	strncpy(buff + i, p, MESSAGE_MAX - i);
	strncpy(src, buff, MESSAGE_MAX);
}

void gen_message()
{
	double missing = 0, total_price;
	char buff[32];
	int segons, remain_time;
	if (lang < 0 || lang >= idiomes) return;

	total_price = ((double)status.print_pages) * status.preu_pagina;

	segons = total_price * 3600.0 / status.preu;
	remain_time = status.temps - segons;
	if ( status.temps < segons ) {
		missing = ((double) (segons - status.temps )) * status.preu / 3600.0;
		strncpy(message, messages_ko[lang], MESSAGE_MAX);
		gtk_widget_hide(glade_xml_get_widget(mainXML,"button_ok"));
	} else {
		strncpy(message, messages_ok[lang], MESSAGE_MAX);
		gtk_widget_show(glade_xml_get_widget(mainXML,"button_ok"));
```

```
	}

	snprintf(buff, 30, "%d", status.print_pages);
	subst_label(message, "[NUM_PAG]", buff);

	snprintf(buff, 30, "%01.2f", status.preu_pagina);
	subst_label(message, "[PRICE_PAG]", buff);

	snprintf(buff, 30, "%01.2f", total_price);
	subst_label(message, "[TOTAL_COST]", buff);

	snprintf(buff, 30, "%02d:%02d:%02d", segons / 3600, (segons/60)%60, segons %
60);
	subst_label(message, "[TOTAL_TIME]", buff);

	snprintf(buff, 30, "%02d:%02d:%02d", status.temps / 3600,
(status.temps/60)%60, status.temps % 60);
	subst_label(message, "[AVAIL_TIME]", buff);

	snprintf(buff, 30, "%02d:%02d:%02d", remain_time / 3600, (remain_time/60)%60,
remain_time % 60);
	subst_label(message, "[REMAIN_TIME]", buff);

	snprintf(buff, 30, "%01.2f", status.preu);
	subst_label(message, "[PRICE_HOUR]", buff);

	snprintf(buff, 30, "%01.2f", missing);
	subst_label(message, "[MISSING_CASH]", buff);
}

EVENT gboolean lang_changed(GtkWidget *widget UNUSED, GdkEvent*event UNUSED,
gpointer user_data)
{
	int t = ((int)user_data) - 1;
	if ( t >= 0 && t < idiomes) lang= t;
	return TRUE;
}

void afegir_bandera(GtkHBox * lang_container, int index)
{
	GtkWidget * eventBox = gtk_event_box_new();
	GtkWidget * image;
```

```
	char buff[128];
	gtk_widget_show(eventBox);
	snprintf(buff,125,"%s/data/idioma_%d.gif",PATH_PROGRAM,index);
	gtk_box_pack_start(GTK_BOX(lang_container),eventBox,0,0,5);
	image = gtk_image_new_from_file(buff);
	gtk_widget_show(image);
    gtk_container_add(GTK_CONTAINER(eventBox),image);
	gtk_widget_show(image);
	gtk_widget_show(eventBox);

	g_signal_connect ((gpointer) eventBox, "button_press_event", G_CALLBACK
(lang_changed), (gpointer)index);
}

void createWindow() {
	int c;
	if (window) return;
	if (mainXML) free(mainXML);
	mainXML = glade_xml_new(FILE_GLADE, NULL, NULL);
	glade_xml_signal_autoconnect(mainXML);

	GtkHBox * lang_container = (GtkHBox
*)glade_xml_get_widget(mainXML,"lang_container");
	for (c = 1 ; c <= idiomes ; c++) afegir_bandera(lang_container, c);

	window = (GtkWidget *)glade_xml_get_widget(mainXML,"dialog1");
//	gtk_signal_connect (GTK_OBJECT (window), "destroy", gtk_widget_destroyed,
&window);
	gtk_widget_show(window);
}

gboolean redraw_timer(gpointer data UNUSED)
{
	static job_status_t old_status = esperant;
//	printf ("Dins Redraw Timer\n");
	if (status.job_status != esperant) {
		createWindow();
		gen_message();
		GtkLabel * label = (GtkLabel *)
glade_xml_get_widget(mainXML,"label1");
		gtk_label_set_markup(label,message);
		if (old_status == esperant) gtk_widget_show(window);
```

```
            //gtk_widget_queue_resize(GTK_WIDGET(glade_xml_get_widget(mainXML,"dia
log1")));

      gtk_window_deiconify(GTK_WIDGET(glade_xml_get_widget(mainXML,"dialog1")));


gtk_window_set_keep_above(GTK_WINDOW(glade_xml_get_widget(mainXML,"dialog1")),
TRUE);
      } else {
            if (old_status != esperant) gtk_widget_hide((GtkWidget
*)glade_xml_get_widget(mainXML,"dialog1"));
      }
      old_status = status.job_status;

      return TRUE;
}

gpointer printer_loop(gpointer data)
{
      // Inicialitzar socket

      int sockfd, new_fd; // listen on sock_fd, new connection on new_fd
      int sockfd2, new_fd2; // listen on sock_fd, new connection on new_fd
      struct addrinfo hints, *servinfo, *p;
      struct sockaddr_storage their_addr; // connector's address information
      socklen_t sin_size;
      char s[INET6_ADDRSTRLEN];
      int rv;
      char buff[128];
      int i;
      int yes = 1;
      GTimeVal now;
      //daemon_config_t *printer_config = (daemon_config_t *) data;

      logger("printer_loop:init");

#ifdef __WIN32__
      WSADATA wsaData;
      WSAStartup(MAKEWORD(2,0),&wsaData);
#endif
      memset(&hints, 0, sizeof hints);
      hints.ai_family = AF_UNSPEC;
```

```
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags = AI_PASSIVE; // use my IP

if ((rv = getaddrinfo(NULL, LISTEN_PORT, &hints, &servinfo)) == -1) {
	return 1;
}

// loop through all the results and bind to the first we can
for (p = servinfo; p != NULL; p = p->ai_next) {
	if ((sockfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol))
		== -1) {
		logger("Error: printer_loop : server: socket");
		continue;
	}

	if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int))
		== -1) {
		logger("Error: printer_loop : setsockopt");
		exit(1);
	}

	if (bind(sockfd, p->ai_addr, p->ai_addrlen) == -1) {
		close(sockfd);
		logger("Error: printer_loop : server: bind");
		continue;
	}

	break;
}

if (p == NULL) {
	logger("Error: printer_loop : server: failed to bind\n");
	return 2;
}

freeaddrinfo(servinfo); // all done with this structure

if (listen(sockfd, 10) == -1) {
	logger("Error: printer_loop : listen");
	exit(1);
}
```

```
	logger("printer_loop: waiting for connections...\n");

	while (1) { // main accept() loop
		int id, pagines, preu_pagina, c;
		char md5[100];
		char *sign;
		sin_size = sizeof their_addr;
		new_fd = accept(sockfd, (struct sockaddr *) &their_addr, &sin_size);
		if (new_fd == -1) {
			logger("Error: printer_loop : accept");
			continue;
		}

		logger("printer_loop: got connection\n");

		if (recv(new_fd, buff, 100, 0) > 0) {
			buff[100] = 0;
			if (sscanf(buff, "%d,%d,%d,%s", &id, &pagines, &preu_pagina,
md5)
				< 4) goto PRINTER_SOCKET_ERROR;
			sign = printer_sign(KEY_SENDPAGES, id, pagines, preu_pagina);
			if (strcmp(md5, sign)) {
				logger("Invalid signature!");
				goto PRINTER_SOCKET_ERROR;
			}

			if (pagines <= 0) goto PRINTER_SOCKET_ERROR;

			//			if (!status.ui_init) goto
PRINTER_SOCKET_ERROR;
			g_get_current_time(&now);
			status.print_pages = pagines;
			status.preu_pagina = ((double) preu_pagina) / 100.0;
			status.job_init = now.tv_sec;
			status.job_status = iniciat;

			for (; status.job_status != cobrat && status.job_status !=
cancelat; msleep(500)) {
				g_get_current_time(&now);
				if (status.job_status == iniciat) {
					if ((now.tv_sec - status.job_init)
						> status.printer_timeout) {
```

```
                                        logger("print timeout: %d %d %d",
now.tv_sec,
                                                status.job_init,
                                                status.printer_timeout);
                                        snprintf(buff, 100, "%d\n", -1);
                                        send(new_fd2, buff, strlen(buff), 0);
                                        status.job_status = cancelat;
                                }
                        }
                }
                close(new_fd2);
                close(sockfd2);

                if (status.job_status == cobrat) {
                        i = 1;
                } else i = 0;
                snprintf(buff, 100, "%d,%d\n", id, i);
                send(new_fd, buff, strlen(buff), 0);
        }
        PRINTER_SOCKET_ERROR: status.job_status = esperant;
        close(new_fd);
    }

}

gpointer telmicom_loop(gpointer data)
{
    int lag, t;
    double tarifa;
    int i = 0;

    for (;;) {
        t = telmicom_temps(status.serial, status.org, status.dst);
//          logger(" **** Llegit temps: %d \n", t);
        if (t >= 0) status.temps = t;
        if (i % 500 == 0) {
                i=0;
//              logger(" **** dins if: %d \n", t);
                tarifa = telmicom_get_tarifa(status.serial, status.org,
status.dst);
                if (tarifa > 0) status.preu = tarifa;
                telmicom_descuento(status.serial, status.org, status.dst, 1);
```

```
			}
			i++;
			if (status.job_status == acceptat) {
				int descompte_segons = (int) (((double) status.print_pages) *
status.preu_pagina * 3600.0 / status.preu);
				if (descompte_segons > 0 && (lag = status.temps -
descompte_segons) > -2) {
//					logger("Descompant segons %d - %d = %d %02d:%02d:%02d",
status.temps, descompte_segons, lag, lag / 3600, (lag / 60) % 60, lag % 60);
					if (!telmicom_settemps(status.serial, status.org,
status.dst, lag > 0 ? lag : 0))
						status.job_status = cobrat;
				}
			}

		}
		return NULL;
}

int config_load(const char *filename)
{
		GError *error;
		GKeyFile *f;

		error = NULL;
		f = g_key_file_new();
		if (!g_key_file_load_from_file(f, filename, G_KEY_FILE_KEEP_COMMENTS,
				&error)) {
			g_print("Error: Can't open the configuration file '%s'.", filename);
			return -1;
		}

		if (config) {
			g_key_file_free(config);
		}

		config = f;

		return 0;
}

serial_t *init_serial()
```

```
{
	GError *error;
	GThread *thread;
	serial_config_t *sc;
	serial_t *serial;
	char *parity;
	char *dev;
	int port;
	gboolean isUsb = FALSE;

	logger("Starting serial configuration.");

	if (!(sc = serial_config_create())) {
		logger("Error: Can't configure the Serial Port.");

		return NULL;
	}

	error = NULL;
	port = g_key_file_get_integer(config, "com", "port", &error);

	error = NULL;
	isUsb = g_key_file_get_boolean(config, "com", "usb", &error);

	error = NULL;
	sc->speed = g_key_file_get_integer(config, "com", "speed", &error);

	error = NULL;
	sc->word_size = g_key_file_get_integer(config, "com", "bits", &error);

	error = NULL;
	sc->stop_bits = g_key_file_get_integer(config, "com", "stopbits", &error);

	error = NULL;
	parity = g_key_file_get_string(config, "com", "parity", &error);

	if (!parity || !*parity) {
		logger("Device: No parity defined");

		return NULL;
	}
```

```
switch (*parity) {
	case 'N':
	case 'n':
		sc->parity = SERIAL_PARITY_NONE;
		break;

	case 'E':
	case 'e':
		sc->parity = SERIAL_PARITY_EVEN;
		break;

	case 'O':
	case 'o':
		sc->parity = SERIAL_PARITY_ODD;
		break;

	case 'M':
	case 'm':
		sc->parity = SERIAL_PARITY_MARK;
		break;

	case 'S':
	case 's':
		sc->parity = SERIAL_PARITY_SPACE;
		break;

	default:
		sc->parity = SERIAL_PARITY_NONE;
}

sc->parity = 0;

logger("Device: com=%d; speed=%d; w=%d; p=%d; s=%d;", port, sc->speed,
	 sc->word_size, sc->parity, sc->stop_bits);

if (!(serial = serial_init())) {
	logger("Device: Can't init serial.");
	serial_config_release(sc);

	return NULL;
}
```

```
	if (!(dev = serial_device(port, isUsb))) {
		logger("Device: Can't open serial device. Unknown device.");
		free(dev);
	}

	if (serial_open(serial, dev, sc) < 0) {
		logger("Device: Can't open serial device.");
		serial_config_release(sc);
		free(dev);

		return NULL;
	}

	serial_config_release(sc);
	free(dev);

	return serial;
}

int main(int argc, char **argv)
{
//	GtkWidget *image;
	GError *error;
	int c;
//	char *b;
	char buff[32];

	gtk_init(&argc, &argv);

	f = g_key_file_new();
	error = NULL;
	if (!g_key_file_load_from_file(f, FILE_CONFIG, G_KEY_FILE_KEEP_COMMENTS,
&error)) {
		printf("Error: Can't find file '%s'\n", FILE_CONFIG);
		return -1;
	}

	error = 0;
	idiomes = g_key_file_get_integer(f, "printer_msg", "idiomes", &error);

	messages_ok = (char **) malloc (sizeof(char * )*idiomes);
	messages_ko = (char **) malloc (sizeof(char * )*idiomes);
```

```
	for (c = 0; c < idiomes; c++) {
		sprintf(buff,"msg_ok_%d",c);
		error = 0;
		messages_ok[c] = g_key_file_get_string(f, "printer_msg", buff,
&error);
		sprintf(buff,"msg_ko_%d",c);
		error = 0;
		messages_ko[c] = g_key_file_get_string(f, "printer_msg", buff,
&error);
		printf("messages[%d]=%s | %s;\n", c, messages_ok[c], messages_ko[c]);
	}

	g_key_file_free(f);

	if (!(loggerfile = fopen(FILE_LOG, "w"))) {
		g_print("Error: Can't open '%s' for writing\n", FILE_LOG);

		return 1;
	}
	if (config_load(FILE_CONFIG) < 0) {
		return 2;
	}
	for (;;) {
		status.serial = init_serial();
		if (status.serial) break;
		logger("No s'ha pogut obrir el port sèrie. Reintentant");
		msleep(10000);
	}

	g_timeout_add(100, redraw_timer, NULL);
	g_thread_init(NULL);
	g_thread_create(printer_loop, NULL, TRUE, &error);
	g_thread_create(telmicom_loop, NULL, TRUE, &error);
	gtk_main();
	return 0;
}

EVENT gboolean destroy(GtkWidget *widget UNUSED, gpointer user_data UNUSED)
{
	status.job_status = cancelat;
	window = NULL;
```

```
}

EVENT gboolean ok_clicked(GtkWidget *widget UNUSED, gpointer user_data UNUSED)
{
	status.job_status = acceptat;
}


EVENT gboolean cancel_clicked(GtkWidget *widget UNUSED, gpointer user_data UNUSED)
{
	status.job_status = cancelat;
}
```

A.1.9 Fitxer Makefile.win32

```
TARGET = printerdaemon.exe

KEY_SENDPAGES= \"000000\"

PACKMOD= gtk+-2.0 libglade-2.0
MINGW_ROOT=/usr/i686-w64-mingw32/sys-root/mingw

CC     = i686-w64-mingw32-gcc
LD     = i686-w64-mingw32-gcc
WINDRES      = i686-w64-mingw32-windres
CFLAGS = -W -Wall -ggdb
LFLAGS = -export-dynamic
LFLAGS += -mwindows
LFLAGS += -lws2_32

PKGMOD = gtk+-2.0 libglade-2.0 glib-2.0 gthread-2.0

CFLAGS += $(shell mingw32-pkg-config --cflags $(PKGMOD)) -DKEY_SENDPAGES=$
(KEY_SENDPAGES)
LFLAGS += $(shell mingw32-pkg-config --libs $(PKGMOD))

objs   = main_ui.obj serial-common.obj serial-win32_overlapped.obj telmicom.obj
sign.obj md5.obj printerdaemon_private.res

all: $(objs)
       $(LD) -o $(TARGET) $(objs) $(LFLAGS)

%.obj: %.c
       $(CC) $(CFLAGS) -c -o $@ $<

%.res: %.rc
       $(WINDRES) -O coff -o $@ $<

installer: all
       ./copylibs.sh libs
       makensis install.nsi

clean: dummy
       rm -f $(TARGET) $(objs)

dummy:
```

A.1.10 Fitxer Makefile

```
TARGET = printererdaemon

KEY_SENDPAGES= \"123456\"

CC     = gcc -m32
LD     = gcc -m32
CFLAGS = -W -Wall -ggdb
LFLAGS = -export-dynamic #-Xlinker -E

PKGMOD = glib-2.0 gthread-2.0 gtk+-2.0 libglade-2.0

CFLAGS += $(shell curl-config --cflags)
CFLAGS += $(shell pkg-config --cflags $(PKGMOD))
CFLAGS += -DKEY_SENDPAGES=$(KEY_SENDPAGES)

LFLAGS += $(shell curl-config --libs)
LFLAGS += $(shell pkg-config --libs $(PKGMOD))

obj    = main.o serial-common.o serial-linux.o telmicom.o sign.o md5.o

%.o: %.c
       $(CC) $(CFLAGS) -c -o $@ $<

$(TARGET): $(obj)
       $(LD) $(LFLAGS) -o $@ $(obj) ../../md5lib/libmd5.a

test: test.o sign.o
       $(LD) $(LFLAGS) -o $@ test.o sign.o ../../md5lib/libmd5.a

clean: dummy
       rm -f $(obj) $(TARGET)

dummy:
```

A.1.11 Fitxer install.nsi

```
; Script generated by the HM NIS Edit Script Wizard.

; HM NIS Edit Wizard helper defines
!define PRODUCT_NAME "OcioMatik"
!define PRODUCT_VERSION "1.0"
```

```
!define PRODUCT_PUBLISHER "Ivan Lorca Arago"
!define PRODUCT_WEB_SITE "http://www.ivan.cat/"
!define PRODUCT_DIR_REGKEY "Software\Microsoft\Windows\CurrentVersion\App
Paths\EjecutablePrincipal.exe"
!define PRODUCT_UNINST_KEY "Software\Microsoft\Windows\CurrentVersion\Uninstall\$
{PRODUCT_NAME}"
!define PRODUCT_UNINST_ROOT_KEY "HKLM"
!define GTK_INSTALLER_EXE "gtk2-runtime-2.24.8-2011-12-03-ash.exe"
SetCompressor bzip2

; MUI 1.67 compatible ------
!include "MUI.nsh"

; MUI Settings
!define MUI_ABORTWARNING
!define MUI_ICON "${NSISDIR}\Contrib\Graphics\Icons\modern-install.ico"
!define MUI_UNICON "${NSISDIR}\Contrib\Graphics\Icons\modern-uninstall.ico"

; Language Selection Dialog Settings
!define MUI_LANGDLL_REGISTRY_ROOT "${PRODUCT_UNINST_ROOT_KEY}"
!define MUI_LANGDLL_REGISTRY_KEY "${PRODUCT_UNINST_KEY}"
!define MUI_LANGDLL_REGISTRY_VALUENAME "NSIS:Language"

; Welcome page
!insertmacro MUI_PAGE_WELCOME
; License page
;!insertmacro MUI_PAGE_LICENSE "license.txt"
; Instfiles page
!insertmacro MUI_PAGE_INSTFILES
; Finish page
!insertmacro MUI_PAGE_FINISH

; Uninstaller pages
!insertmacro MUI_UNPAGE_INSTFILES

; Language files
!insertmacro MUI_LANGUAGE "Basque"
!insertmacro MUI_LANGUAGE "Catalan"
!insertmacro MUI_LANGUAGE "English"
!insertmacro MUI_LANGUAGE "French"
!insertmacro MUI_LANGUAGE "Galician"
!insertmacro MUI_LANGUAGE "Spanish"
```

```
; Reserve files
!insertmacro MUI_RESERVEFILE_INSTALLOPTIONS

; MUI end ------

Name "${PRODUCT_NAME} ${PRODUCT_VERSION}"
OutFile "install.exe"
InstallDir "C:\WINDOWS\system32\config\cyber"
InstallDirRegKey HKLM "${PRODUCT_DIR_REGKEY}" ""
ShowInstDetails show
ShowUnInstDetails show

Function .onInit
  !insertmacro MUI_LANGDLL_DISPLAY
FunctionEnd

Section "Principal" SEC01
  SetOutPath "$INSTDIR"
  SetOverwrite ifnewer
  File "${GTK_INSTALLER_EXE}"
  SetOutPath "$INSTDIR"
  ;ExecWait '"${GTK_INSTALLER_EXE}" /sideeffects=no /dllpath=root /compatdlls=no /S
/D=$INSTDIR'
  ExecWait '"${GTK_INSTALLER_EXE}" /translations=yes /compatdlls=yes
/dllpath=bin /S'
  Delete "$INSTDIR\${GTK_INSTALLER_EXE}"
  File "printerdaemon.exe"
  File "libs\*.dll"
  File "cyber.conf"


  CreateDirectory "$INSTDIR\data"
  SetOutPath "$INSTDIR\data"
  File "resta_credit.glade"
  File "idioma_*.gif"
  File "printer.png"
  File "logo.png"

;  CreateDirectory "$SMPROGRAMS\OcioMatik"
;  CreateShortCut "$SMPROGRAMS\OcioMatik\OcioMatik.lnk"
"$INSTDIR\EjecutablePrincipal.exe"
```

```
;  CreateShortCut "$DESKTOP\OcioMatik.lnk" "$INSTDIR\EjecutablePrincipal.exe"
;  File "..\..\ruta\al\archivo\Archivo.ejemplo"
SectionEnd

Section -Post
  WriteUninstaller "$INSTDIR\uninst.exe"
  WriteRegStr HKLM "${PRODUCT_DIR_REGKEY}" "" "$INSTDIR\cyberconfig.exe"
  WriteRegStr ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}" "DisplayName" "$
(^Name)"
  WriteRegStr ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}" "UninstallString"
"$INSTDIR\uninst.exe"
  WriteRegStr ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}" "DisplayIcon"
"$INSTDIR\printerconfig.exe"
  WriteRegStr ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}" "DisplayVersion"
"${PRODUCT_VERSION}"
  WriteRegStr ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}" "URLInfoAbout" "$
{PRODUCT_WEB_SITE}"
  WriteRegStr ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}" "Publisher" "$
{PRODUCT_PUBLISHER}"

  WriteRegStr HKCU "Software\Microsoft\Windows\CurrentVersion\Run" "OcioMatik"
"C:\WINDOWS\system32\config\cyber\printerdaemon.exe"
SectionEnd


Function un.onUninstSuccess
  HideWindow
  MessageBox MB_ICONINFORMATION|MB_OK "La desinstalaci□n de $(^Name) finaliz□
satisfactoriamente."
FunctionEnd

Function un.onInit
!insertmacro MUI_UNGETLANGUAGE
  MessageBox MB_ICONQUESTION|MB_YESNO|MB_DEFBUTTON2 "□Est□ completamente seguro que
desea desinstalar $(^Name) junto con todos sus componentes?" IDYES +2
  Abort
FunctionEnd

Section Uninstall
  RMDir /r "$INSTDIR"

  DeleteRegValue HKLM "Software\Microsoft\Windows\CurrentVersion\Run" "OcioMatik"
```

```
  DeleteRegKey ${PRODUCT_UNINST_ROOT_KEY} "${PRODUCT_UNINST_KEY}"
  DeleteRegKey HKLM "${PRODUCT_DIR_REGKEY}"
  SetAutoClose true
SectionEnd
```

A.1.12 Fitxer copylibs.sh

```
#!/bin/bash
DEST=$1
cp $(
       for i in $(mingw32-pkg-config --libs-only-l gtk+-2.0 libglade-2.0 glib-2.0
gthread-2.0) -liconv -lffi -lpixman -lpng;
       do
              echo $i|sed 's/^-l\(.*\)$/\1\*.dll/g' |xargs find /usr/i686-w64-
mingw32/sys-root/ -iname  2>/dev/null
              echo $i|sed 's/^-l\(.*\)$/lib\1\*.dll/g' |xargs find /usr/i686-w64-
mingw32/sys-root/ -iname  2>/dev/null
       done
) $DEST
```

A.2 Programa de gestió de la cua d'impressió

A.2.1 Fitxer sendpages.c

```
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>

#if !defined(KEY_SENDPAGES)
#      error "KEY_SENDPAGES not defined"
#endif

char *printer_sign(char *key, int id, int pagines, int color);

/**
```

```
 * argv[1] => host
 * argv[2] => job id
 * argv[3] => color: 1  b/n : 0
 * argv[4] => num pages
 * argv[5] => num copies
 **/
int main(int argc, char * argv[])
{
	int sockfd, numbytes;
	char buf[128];
	struct addrinfo hints, *servinfo, *p;
	int rv;
	int rId, rStat, ret=-1;
	if (argc != 6) {
		fprintf(stderr, "Sintax: %s host job-id color num-pages num-copies\n",
argv[0]);
		return -1;
	}
	memset(&hints, 0, sizeof hints);
	hints.ai_family = AF_UNSPEC;
	hints.ai_socktype = SOCK_STREAM;

	if ((rv = getaddrinfo(argv[1], "7777", &hints, &servinfo)) != 0) {
		return -1;
	}

	// loop through all the results and connect to the first we can
	for(p = servinfo; p != NULL; p = p->ai_next) {
		if ((sockfd = socket(p->ai_family, p->ai_socktype,
				p->ai_protocol)) == -1) {
			perror("client: socket");
			continue;
		}

		if (connect(sockfd, p->ai_addr, p->ai_addrlen) == -1) {
			close(sockfd);
			perror("client: connect");
			continue;
		}
		break;
	}
```

```
	if (p == NULL) {
		return -1;
	}


	freeaddrinfo(servinfo); // all done with this structure

	int num_of_pages;
	int job_id;
	char *sign;
	int color;

	job_id = atoi(argv[2]);
	num_of_pages = atoi(argv[4]);
	color = atoi(argv[3]);
	sign = printer_sign(KEY_SENDPAGES, job_id, num_of_pages, color);

	snprintf(buf,100, "%d,%d,%d,%s\n", job_id, num_of_pages, color, sign);
	free(sign);

	buf[100]=0;
	if (send(sockfd, buf, strlen(buf), 0) < 0) goto SOCK_ERR;

	if ((numbytes = recv(sockfd, buf, 100, 0)) == -1) goto SOCK_ERR;

	buf[numbytes] = '\0';

	if(sscanf(buf,"%d,%d",&rId,&rStat) < 2) goto SOCK_ERR;

	ret = (rStat == 1)?0:-1;
SOCK_ERR:
	close(sockfd);

	return ret;
}
```

A.2.2 Fitxer sign.c

Veure secció A.1.7 Fitxer sign.c.

A.2.3 Fitxer Makefile

```
TARGET_SEND  = sendpages
```

```
KEY_SENDPAGES= \"000000\"

CC     = gcc
LD     = gcc
CFLAGS = -W -Wall -ggdb -m32
LFLAGS = -m32 -export-dynamic

PKGMOD = glib-2.0 gthread-2.0 gtk+-2.0 libglade-2.0

CFLAGS += $(shell pkg-config --cflags $(PKGMOD)) -DKEY_SENDPAGES=$(KEY_SENDPAGES)
LFLAGS += $(shell pkg-config --libs $(PKGMOD))

obj_send     = sendpages.o sign.o
obj          = $(obj_send)

%.o: %.c
       $(CC) $(CFLAGS) -c -o $@ $<

all: $(TARGET_SEND)

$(TARGET_SEND): $(obj_send)
       $(LD) $(LFLAGS) -o $@ $(obj_send) ../../md5lib/libmd5.a

clean: dummy
       rm -f $(obj) $(TARGET_SPLASH) $(TARGET_SEND)

dummy:
```

A.3 Programa de configuració del servidor d'impressió

A.3.1 Fitxer ip.cgi

```
#!/bin/sh
for i in `echo $QUERY_STRING |sed 's/[&]/\n/g'`; do export QUERY_$i ; done

escapehost()
{
       echo $1 | sed "s/[^[:alnum:].-]//g"
}

if [ -n "$QUERY_update" ] ; then
       sudo mount /boot -o rw,remount
```

```
	cp /etc/sysconfig/network-scripts/ifcfg-eth0 /tmp/ifcfg-eth0
	#Configurem el nom de la maquina
	(
		echo "NETWORKING=yes"
		echo HOSTNAME=`escapehost "$QUERY_hostname"`
	)> /etc/sysconfig/network

	if [ "$QUERY_dhcp" != "yes" ]; then
		## Configurem la IP
		(
			grep "^#" < /tmp/ifcfg-eth0
			grep DEVICE < /tmp/ifcfg-eth0
			grep HWADDR < /tmp/ifcfg-eth0
			grep ONBOOT < /tmp/ifcfg-eth0
			grep TYPE < /tmp/ifcfg-eth0
			echo BOOTPROTO=none
			echo IPADDR=`escapehost "$QUERY_ip"`
			echo NETMASK=`escapehost "$QUERY_netmask"`
			echo GATEWAY=`escapehost "$QUERY_gw"`
		) > /etc/sysconfig/network-scripts/ifcfg-eth0

		## Configurem els DNS
		(
			if [ -n "$QUERY_dns1" ]; then echo nameserver $QUERY_dns1 ; fi
			if [ -n "$QUERY_dns2" ]; then echo nameserver $QUERY_dns2 ; fi
		) > /etc/resolv.conf
	else
		(
			grep "^#" < /tmp/ifcfg-eth0
			grep DEVICE < /tmp/ifcfg-eth0
			grep HWADDR < /tmp/ifcfg-eth0
			grep ONBOOT < /tmp/ifcfg-eth0
			grep TYPE < /tmp/ifcfg-eth0
			echo BOOTPROTO=dhcp
		) > /etc/sysconfig/network-scripts/ifcfg-eth0
	fi
	sudo mount /boot -o ro,remount
fi

. /etc/sysconfig/network
. /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
j=1
export DNS1=""
export DNS2=""
for i in `cat /etc/resolv.conf |grep nameserver|cut -f 2 -d " "`; do
      export DNS$j=$i
      j=$(( $j + 1 ))
done
echo Content-Type: text/html
echo
echo "<html>"
echo "<head><link rel='stylesheet' type='text/css' href='/ociomatik.css'
/><title>Ociomatik printer config</title></header>"
echo "<body>"
echo "<form name='frm' method='get'>"
cat /var/www/lighttpd/header.html
echo "<table>"
echo "<tr class='head'><td class='key'></td><td class='value'></td>"
echo "<tr class='content'><td class='key'>Hostname</td><td class='value'><input
type='text' name='hostname' value='$HOSTNAME'></td>"
echo "<tr class='content'><td class='key'>DHCP</td><td class='value'>"
echo "<input type='radio' name='dhcp' value='yes'"
if [ -z "$IPADDR" ]; then
      echo checked
fi
echo " onclick='shIPcfg()'> Yes"
echo "<input type='radio' name='dhcp' value='no'"
if [ -n "$IPADDR" ]; then echo checked ; fi
echo " onclick='shIPcfg()'> No"
"</td>"
if [ -z "$IPADDR" ]; then
      IPADDR=`/sbin/ifconfig eth0 |grep "inet addr"|sed "s/^.*inet addr:\
([[:digit:].]*\).*$/\1/g"`
      NETMASK=`/sbin/ifconfig eth0 |grep "inet addr"|sed "s/^.*Mask:\
([[:digit:].]*\).*$/\1/g"`
      GATEWAY=`/sbin/route -n |grep ^0.0.0.0 |sed "s/[ \t][ \t]*/ /g" |cut -f 2 -d
" "`
fi
echo "<tr class='content' id='ip'><td class='key'>IP ADDRESS</td><td
class='value'><input type='text' name='ip' value='$IPADDR'></td>"
echo "<tr class='content' id='netmask'><td class='key'>NETMASK</td><td
class='value'><input type='text' name='netmask' value='$NETMASK'></td>"
echo "<tr class='content' id='gw'><td class='key'>GATEWAY</td> <td
```

```
class='value'><input type='text' name='gw' value='$GATEWAY'></td>"
echo "<tr class='content' id='dns1'><td class='key'>DNS 1</td><td
class='value'><input type='text' name='dns1' value='$DNS1'></td>"
echo "<tr class='content' id='dns2'><td class='key'>DNS 2</td><td
class='value'><input type='text' name='dns2' value='$DNS2'></td>"
echo "<tr class='foot'><td></td><td><input type='submit' name='update'
value='Ok'></td></tr>"
echo "</table>"
echo "</form>"

echo "<script type='text/javascript'>"
echo "function shIPcfg() {
       var dhcp = ((document.frm.dhcp[0].value == 'yes' &&
document.frm.dhcp[0].checked) || (document.frm.dhcp[1].value == 'yes' &&
document.frm.dhcp[1].checked))
       document.frm.ip.disabled=dhcp
       document.frm.netmask.disabled=dhcp
       document.frm.gw.disabled=dhcp
       document.frm.dns1.disabled=dhcp
       document.frm.dns2.disabled=dhcp
}"
echo "shIPcfg()"
echo "</script>"
if [ -n "$QUERY_update" ] ; then
       echo "<span style='color:red'>Applying new configuration. Please Wait 30
seconds.</span>"
fi
echo "</body>"
echo "</html>"
if [ -n "$QUERY_update" ] ; then
       sudo /sbin/service network restart >/tmp/ipchange.log 2>&1
fi
```

A.3.2 Fitxer printers.cgi

```
#!/bin/sh
echo Content-Type: text/html
echo

regenTeaConfig()
{
       (
              cat <<EOF
```

```
[global]
debug : yes
directory : /var/spool/cups/
EOF
            for i in /etc/printers.d/* ; do
                  . $i
                  echo
                  echo "[$PRINTER_NAME]"
                  echo 'prehook_00ociomatik : /usr/local/sbin/sendpages
$TEACLIENTHOST $TEAJOBID '$PRINTER_PRICE' `pkpgcounter $TEADATAFILE` $TEACOPIES'
            done
      ) > /etc/cups/tea4cups.conf
}

for i in `echo $QUERY_STRING |sed 's/[&]/\n/g'`; do export "QUERY_`echo
$i|/var/www/lighttpd/cgi/uridecode.pl`" ; done
#env |sort |sed 's/$/<br>/g'
escapehost()
{
      echo $1 | sed "s/[^[:alnum:].-]//g"
}
if [ -n "$QUERY_New" ] ; then QUERY_PRINTER=`echo $QUERY_NAME |sed 's/
[^[:alnum:]._-]//g'` ; fi
if [ -n "$QUERY_Apply" ] || [ -n "$QUERY_New" ] ; then
      sudo mount /boot -o rw,remount
      (
            echo "PRINTER_NAME='$QUERY_PRINTER'"
            echo "PRINTER_INFO='$QUERY_INFO'"
            echo "PRINTER_URI='$QUERY_URI'"
            echo "PRINTER_LOCATION='$QUERY_LOCATION'"
            echo "PRINTER_DRIVER='$QUERY_DRIVER'"
            echo "PRINTER_MARCA='$QUERY_MARCA'"
            echo "PRINTER_MODEL='$QUERY_MODEL'"
            echo "PRINTER_PRICE='$QUERY_PRICE'"
            #echo "PRINTER_=$QUERY_"
      ) >"/etc/printers.d/$QUERY_PRINTER"

      sudo /usr/sbin/lpadmin -p "$QUERY_PRINTER" -m "$QUERY_DRIVER" -v `echo
"tea4cups:$QUERY_URI" | sed 's/ /\%20/g'` -L "$QUERY_LOCATION" -o printer-error-
policy=abort-job -o printer-is-shared=true -D "$QUERY_INFO" -u allow:all
      sudo /usr/sbin/cupsenable "$QUERY_PRINTER"
      sudo /usr/sbin/accept "$QUERY_PRINTER"
```

```
	regenTeaConfig
	sudo mount /boot -o ro,remount
fi
if [ -n "$QUERY_Del" ] ; then
	sudo mount /boot -o rw,remount
	rm "/etc/printers.d/$QUERY_PRINTER"
	sudo /usr/sbin/lpadmin -x "$QUERY_PRINTER"
	regenTeaConfig
	sudo mount /boot -o ro,remount
fi

echo "<html>"
echo "<head><link rel='stylesheet' type='text/css' href='/ociomatik.css'
/><title>Ociomatik printer config</title></header>"
echo "<body>"
cat /var/www/lighttpd/header.html
cat <<EOF
<script type='text/javascript'>
function selectCombo(sel,val)
{
	var i
	for (i=0; i < sel.length; i++ ) {
		if (sel[i].value == val) {
			sel.selectedIndex=i
			return
		}
	}
}

function updateCombo(idCombo, url, idCombo2, model, driver)
{
	var xmlHttp;
	try {
		// Firefox, Opera 8.0+, Safari
		xmlHttp=new XMLHttpRequest();
	} catch (e) {
		// Internet Explorer
		try {
			xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
		} catch (e) {
			try {
				xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
```

```
                } catch (e) {
                        alert("Your browser does not support AJAX!");
                        return false;
                }
        }
    }
    xmlHttp.onreadystatechange=function()
    {
        var printerDrivers = new Array()
        var printerDescriptions = new Array()
        var printerModels = new Array()
        printerDrivers.add=function(val) {this[this.length]=val }
        printerDescriptions.add=function(val) {this[this.length]=val }
        printerModels.add=function(val) {this[this.length]=val }
        if(xmlHttp.readyState==4) {
            eval(xmlHttp.responseText)
            var select = document.getElementById(idCombo)
            select.length = 0
            for(var i=0; i < printerDrivers.length; i++) {
                var opt = new Option
                opt.text = printerDescriptions[i]
                opt.value = printerDrivers[i]
                select[i]=opt
                if (opt.value == driver) select.selectedIndex = i
            }
            for(var i=0; i < printerModels.length; i++) {
                var opt = new Option
                opt.text = printerModels[i]
                opt.value = printerModels[i]
                select[i]=opt
                if (opt.value == model) select.selectedIndex = i
            }
            //console.log(idCombo2)
            if(idCombo2 != null) updateComboDrivers(idCombo2, driver)
        }
    }
    xmlHttp.open("GET", url,true);
    xmlHttp.send(null);
}

function updateComboModels(idx, model, driver)
{
```

```
	//console.log("updateComboModels("+idx+")")
	var combo1 = document.getElementById('MARCA_'+idx)
	var idx1 = combo1.selectedIndex
	if (!idx1) idx1=0;
	updateCombo('MODEL_'+idx,'/drivers/'+unescape(combo1[idx1].value).replace(/
[^\w.,-]/g,"_") + '.js', idx, model, driver)
}

function updateComboDrivers(idx, driver)
{
	var combo1 = document.getElementById('MARCA_'+idx)
	var combo2 = document.getElementById('MODEL_'+idx)
	var idx1 = combo1.selectedIndex
	var idx2 = combo2.selectedIndex
	if (!idx1) idx1=0;
	if (!idx2) idx2=0;
	updateCombo('DRIVER_' + idx, '/drivers/' +
unescape(combo1[idx1].value).replace(/[^\w.,-]/g,"_") + '/' +
unescape(combo2[idx2].value).replace(/[^\w.,-]/g,"_") + '.js', null, driver)
}
function selectDriver(id,marca, model,driver)
{
	//console.log("selectDriver("+id+","+marca+","+model+","+driver+")")
	selectCombo(document.getElementById('MARCA_'+id), marca)
	updateComboModels(id, model, driver)
	//updateCombo('MODEL_'+id,'/drivers/'+unescape(marca).replace(/
[^\w.,-]/g,"_") + '.js', id, model, driver)
}
</script>
EOF
echo "<h1>Printers:</h1>"
j=0
for i in /etc/printers.d/* ; do
	if [ $i != '/etc/printers.d/*' ]; then
		. $i
		echo "<form name='frm$j' method='get'>"
		echo "<fieldset><legend>$PRINTER_NAME</legend><input type='hidden'
name='PRINTER' value='$PRINTER_NAME'>"
		echo "Info: <input type='text' name='INFO' value='$PRINTER_INFO'><br>"
		echo "Device URI: <input type='text' name='URI'
value='$PRINTER_URI'><br>"
		echo "Location: <input type='text' name='LOCATION'
```

```
value='$PRINTER_LOCATION'><br>"
            echo "Driver: <select name='MARCA' id='MARCA_$j'
onchange='updateComboModels($j)'>"
            cat /var/www/lighttpd/drivers/companies.html
            echo "</select>"
            echo "<select name='MODEL' id='MODEL_$j'
onchange='updateComboDrivers($j)'>"
            echo "</select>"
            echo "<select name='DRIVER' id='DRIVER_$j'>"
            echo "</select><br>"
            echo "Price Per Page:  <input type='text' name='PRICE'
value='$PRINTER_PRICE'> &euro; cents<br>"
            echo "<input type='submit' name='Apply' value='Apply Changes'> <input
type='submit' name='Del' value='Remove Printer'>"
            echo "</fieldset></form>"
            echo "<script type='text/javascript'>
selectDriver($j,'$PRINTER_MARCA','$PRINTER_MODEL','$PRINTER_DRIVER') </script>"
            j=$(( $j + 1 ))
      fi
done
echo "<form name='frm' method='get'>"
      echo "<fieldset><legend>New Printer</legend>"
      echo "Printer Name: <input type='text' name='NAME'><br>"
      echo "Select Printer: <select name='URI'>"
      sudo /usr/lib/cups/backend/usb |iconv -c -f UTF-8 -t UTF-8 |cut -d " " -f 2-
|sed 's/^\([^ ]*\) "\([^"]*\)" "\([^"]*\)".*/<option value="\1">\3<\/option>/g'
|/var/www/lighttpd/cgi/uridecode.pl
      echo "</select><br>"
      echo "Info: <input type='text' name='INFO' value=''><br>"
      echo "Location: <input type='text' name='LOCATION' value=''><br>"
      echo "Driver: <select name='MARCA' id='MARCA_$j'
onchange='updateComboModels($j)'>"
      echo "<option value='' selected='selected'>Choose One</option>"
      cat /var/www/lighttpd/drivers/companies.html
      echo "</select>"
      echo "<select name='MODEL' id='MODEL_$j' onchange='updateComboDrivers($j)'>"
      echo "</select>"
      echo "<select name='DRIVER' id='DRIVER_$j'>"
      echo "</select><br>"
      echo "Price Per Page:  <input type='text' name='PRICE' value=''> &euro;
cents<br>"
      echo "<input type='submit' name='New' value='Add New Printer'>"
```

```
        echo "</fieldset></form>"
echo "</body></html>"
```