TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: SISTEMES D'ADQUISICIÓ D'IMATGES AÈRIES

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat en Sistemes de Telecomunicació

AUTOR: Josep M. Batlle Martín

DIRECTOR: Enric Pastor Llorens

DATA: 21 de gener de 2008

Títol: Sistemes d'adquisició d'imatges aèries

Autor: Josep M. Batlle Martín

Director: Enric Pastor Llorens

Data: 21 de gener de 2008

Resum

Es tracta de dissenyar un sistema per realitzar la gestió de l'adquisició d'imatges des d'una càmera embarcada en un UAV. Per un costat el sistema ha de poder identificar amb un cert grau de precisió la posició i orientació de la càmera en el moment d'adquirir la imatge; i per un altre la imatge s'ha de codificar en algun format digital sense pèrdua d'informació pel seu posterior processament.

El sistema d'adquisició es realitzarà amb una càmera PAL mentre que per identificar la posició i orientació s'utilitzarà una Unitat de Mesura Inercial (IMU). Utilitzant una placa FPGA de la Xilinx (la XUPV2P) primer s'estudien els dos elements per separat i es creen unes aplicacions individuals, llavors, finalment, s'integren per formar el sistema final.

El sistema final es crea en dues versions diferents, una utilitzant un Sistema Operatiu GNU/Linux encastat (embedded) i l'altre sense. Aquest SO encastat es construeix directament a partir de les fonts del kernel 2.6

Paraules clau: Sistemes FPGA. Embedded Systems. Linux. Càmeres PAL. Sistemes inercials. Programació en C.

Title: Aerial Image Acquisition System

Author: Josep M. Batlle Martín

Director: Enric Pastor Llorens

Date: January, 21th 2008

Overview

The project consists on designing a system to carry out the management of the acquisition of images from a camera embarked on a UAV. The system has to be able to identify the position and orientation of the camera in the moment of acquiring the image with certain accuracy, finally, the image has to be coded in some digital format without loss of information.

The acquisition system will be carried out with a PAL camera while to identify the position and orientation an Inertial Measurement Unit (IMU) will be used. Using one Xilinx FPGA board (XUPV2P) firstly we will study the two systems separately and finally they will integrate to form the final system.

The final system will be created in two different versions: one with a GNU/Linux Embedded OS and one without. The Embedded OS will be created directly from the linux 2.6 sources.

Key words: FPGA Systems. Embedded Systems. Linux. PAL cameras. Inertial System. C programming.

Per a la meva família.

ÍNDEX

ÍNDEX DE FIGURES I TAULES

CAPÍTOL 1. UNITAT DE MESURA INERCIAL

CAPÍTOL 2. ADQUISICIÓ D'IMATGES

CAPÍTOL 3. PLACA D'IMPLEMENTACIÓ

CAPÍTOL 4. MUNTATGE

ANNEXOS

INTRODUCCIÓ

Aquest projecte va néixer amb l'objectiu de dissenyar un sistema per adquirir imatges aèries. El sistema havia de poder gestionar, també, la posició i orientació del sistema perquè durant el processament posterior de la imatge es pugui fer una millor interpretació dels resultats.

El projecte estava pensat, d'es d'un inici, per a ser embarcat en un UAV (un vehicle aeri no tripulat) ja que en el departament ICARUS des d'on s'implementaria es treballa en el desenvolupament d'aquestes aeronaus.

El sistema d'adquisició s'havia de realitzar amb una càmera PAL mentre que per identificar la posició i orientació s'utilitzaria una Unitat de Mesura Inercial (IMU) o acceleròmetres i giroscopis per separat. En cas que es pogués hi havia la idea d'utilitzar una càmera d'infrarojos per utilitzar el sistema en tasques relatives a l'extinció d'incendis, no es va poder implementar.

Per desenvolupar l'aplicació inicialment es va estudiar si treballar amb circuits integrats enfocats específicament a aquest objectiu o amb una placa amb una FPGA d'un plantejament més global.

En el primer dels casos el treball derivaria més a un disseny electrònic del sistema, des del disseny esquemàtic, circuital, d'aprofitament dels recursos i fins al seu muntatge físic. En canvi, pel cas de la FPGA el disseny partia amb un enfocament més de programari però que, en principi, permetia escollir el rumb del disseny ja que es tractava d'un sistema més complet.

Al final es va decidir d'utilitzar la placa ja que oferia unes perspectives d'implementació més potents i de reutilització en d'altres projectes. Amb aquesta elecció s'afegia l'objectiu de dur a terme una correcta utilització d'un maquinari d'aquestes característiques i possibilitats, a més d'incloure la voluntat d'utilitzar d'un SO GNU/Linux encastat (embedded systems).

Les repercussions ambientals d'aquest projecte es troben a la tecnologia de fabricació dels components electrònics les quals es troben degudament regulades per les administracions i organismes competents. El que també s'ha de tenir en compte és la reutilització de tots aquests component i, un cop hagi finalitzat la seva vida útil, dipositar-los

CAPÍTOL 1. UNITAT DE MESURA INERCIAL

1.1 Descripció

Una Unitat de Mesura Inercial (IMU, de les seves sigles en anglès) és un element que s'utilitza per a determinar la posició i orientació de l'objecte que l'incorpora, especialment dins un Sistema de Navegació Inercial (INS). En aquesta unitat es mesuren les acceleracions i les velocitats angulars que permeten fixar un posicionament respecte una configuració de partida, és a dir, permeten un posicionament relatiu.

El posicionament relatiu és aquell que només utilitza mesures pròpies de la unitat, sense utilitzar informació provinent de l'exterior. En contraposició hi ha el posicionament absolut que sí utilitza mesures externes a la unitat. Són exemples de posicionament absolut el GPS o el futur Sistema Galileu, que permeten fixar una posició utilitzant la informació que prové de l'extensa xarxa de satèl·lits de què disposa el conjunt global del sistema, no de la unitat GPS, o Galileu, en concret.

El posicionament relatiu és, en general, més ràpid i lleuger a l'utilitzar sensors més senzills però té l'inconvenient d'acumular errors. Aquesta acumulació ve donada pel fet que els canvis es van afegint als càlculs anteriors i els errors, per petits que siguin, es van arrossegant. Llavors l'error comès és més gran en el transcurs del seu ús. En canvi, aquest tipus de desviament no es dóna en els sistemes absoluts ja que al utilitzar referències externes s'evita aquesta acumulació al tractar-se de dades conegudes, això sí, es perd temps de resposta pel fet d'utilitzar algorismes de càlcul més complexos.

Per minimitzar aquest error acumulatiu una de les solucions bàsiques consisteix en controlar la temperatura dels sensors. Aquesta mesura de la temperatura passa a ser un element més de la informació que subministra l'IMU i que és utilitzada per corregir les desviacions en el càlcul degut a aquest fet. Una altra possibilitat, no excloent, és la utilització simultània del posicionament relatiu i l'absolut, essent l'absolut una rectificació periòdica del relatiu.

Un cop s'han aconseguit les dades del dispositiu es pot procedir al càlcul de la posició i la orientació utilitzant un filtre de Kalman. Amb aquest filtre s'aconsegueix augmentar la precisió dels giroscopis ja que té en compte els errors associats a cada sensor, no és així pel cas dels acceleròmetres ja que no hi ha millores substancials.

Si s'uneix aquesta informació processada amb la configuració inicial de l'objecte es pot calcular la posició en un determinat instant. És per això que la principal aplicació d'aquesta Unitat de Mesura és la de proveir a un Sistema de Navegació Inercial (INS), tant si és embarcat en un avió, un vaixell o una nau espacial. També hi ha altres aplicacions com, per exemple, el camp de l'animació digital.

1.2 Parts d'un IMU

Un IMU bàsic consta de tres acceleròmetres i tres giroscopis, els quals proporcionen l'acceleració i la velocitat angular, respectivament, en els tres eixos de coordenades. Per a tenir aquestes magnituds en els tres eixos els acceleròmetres es munten en angles rectes entre sí, igual pel cas dels giroscopis.

1.2.1 Acceleròmetre

Un acceleròmetre és un dispositiu electromecànic que permet mesurar les forces d'acceleració. Aquestes forces poden ser estàtiques, com la força constant de la gravetat terrestre, o poden ser dinàmiques, com les provocades pel moviment o la vibració de l'acceleròmetre.

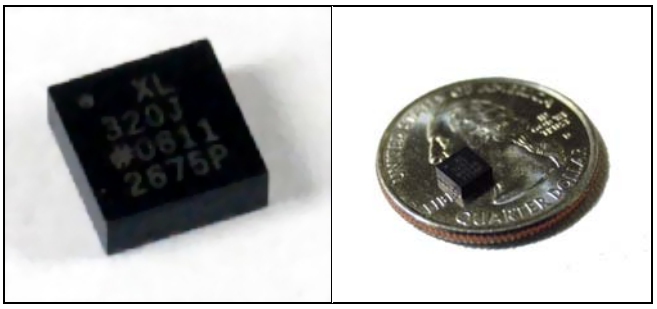


Fig. 1.1 Acceleròmetre ADXL320

Mesurant la quantitat d'acceleració estàtica respecte la gravetat es pot trobar l'angle d'inclinació del dispositiu respecte del terra, mentre que calculant la quantitat d'acceleració dinàmica es pot calcular la direcció i el sentit del moviment del dispositiu.

Les diferents tecnologies amb què es pot construir un acceleròmetre fan que aquest sensor es pugui subdividir en més d'una vintena de tipus diferents. Per exemple es pot crear a partir de l'efecte piezoelèctric, en aquest cas uns cristalls microscòpics es deformaran mecànicament sota les forces d'acceleració, cosa que provocarà la generació d'una tensió. Una altra tècnica és mitjançant la capacitància. Per fer-ho s'hauran de col·locar dues microestructures molt a prop, fet que provocarà una capacitat entre elles. Si s'hi apliquen unes forces d'acceleració una es mourà i la capacitat entre elles variarà, llavors només falta convertir aquesta càrrega elèctrica en diferència de potencial.

Com en tot element, a l'hora d'escollir entre acceleròmetres s'han de comprovar les característiques i escollir en funció de les nostres necessitats. Uns quants factors a tenir en compte són si es vol tenir sortida analògica o

digital, quants eixos es requereixen, la màxima acceleració, la sensibilitat, quin ample de banda es vol i vigilar amb el tema d'impedància/buffering..

Les aplicacions no aeronàutiques d'aquest element són moltes i variades. Des de la protecció de discs durs en portàtils IBM i Apple, desconnectant-los si cauen a terra per accident, fins a l'activació dels airbags d'un cotxe si detecten un accident imminent. També té aplicacions més recreatives com al comandament de la consola Wii de la marca Nintendo, que incorpora un acceleròmetre per permetre un joc diferent del tradicional, a on es pot utilitzar el comandament com si es tractés d'un pal de golf o una espasa.

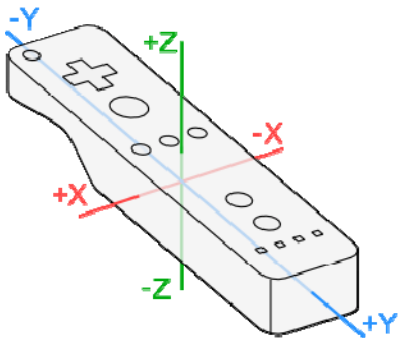


Fig. 1.2 Recreació del comandament de la consola Wii

1.2.2 Giroscopi

El giroscopi és un component que permet mesurar i definir una direcció a l'espai i que s'utilitza per guiar, navegar i estabilitzar un objecte en moviment.

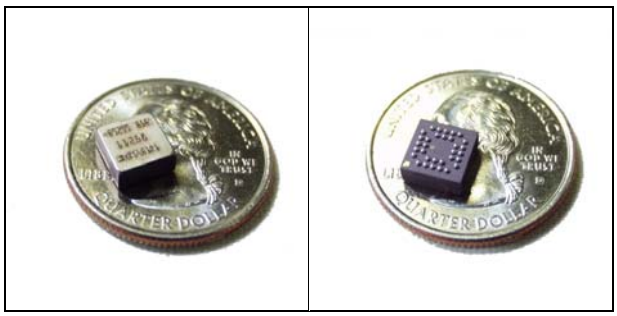


Fig. 1.3 Giroscopi ADXRS300

Amb el giroscopi es mesura la variació de velocitat angular al voltant d'un eix i, si aquest eix és el de canvi d'orientació de l'objecte, la integració de la seva magnitud determina l'increment de l'angle d'orientació.

Els giroscopis poden operar en llaç tancat o en llaç obert. En llaç tancat significa que hi ha una realimentació (de la sortida) que restaura l'estat intern

del giroscopi, provocant que torni a un estat fixat o al nul. En canvi, pel llaç obert el giroscopi opera fora de la posició fixa mentre respon als canvis de la velocitat angular.

Els giroscopis es poden basar en diferents fenòmens físics. Per exemple, els que consisteixen en una massa giratòria detecten canvis en el moment angular per l'Efecte o acceleració de Coriolis, els giroscopis ressonadors ho fan per les desviacions d'aquesta acceleració i els òptics detecten desfasaments entre raigs de llum. Malgrat que els instruments que no tenen una massa giratòria tècnicament no es poden considerar giroscopis, a la pràctica s'anomenen giroscopis aquells dispositius que mesuren l'efecte descrit.

Es poden trobar giroscopis en multitud d'aparells. Un d'ells és el d'un apuntador per a ordinadors, aquí el giroscopi s'utilitza per a permetre un ús espacial de l'apuntador que veu com el cursor segueix el moviment d'aquest. Altres casos serien en entorns de realitat virtual per controlar el moviment o en cotxes i motos per ajudar en la conducció i equilibri del vehicle.

1.3 El nIMU

Com a Unitat de Mesura Inercial es va escollir el nano IMU (nIMU) de l'empresa MEMSense. Una de les alternatives era utilitzar dos dels components més comuns per aquests casos com o són l'acceleròmetre ADXL320 i el giroscopi ADXRS300 d'Analog Devices. Es va optar pel nIMU ja que comporta una simplificació en el tractament de les dades i redueix tant les mides del sistema com el seu disseny.

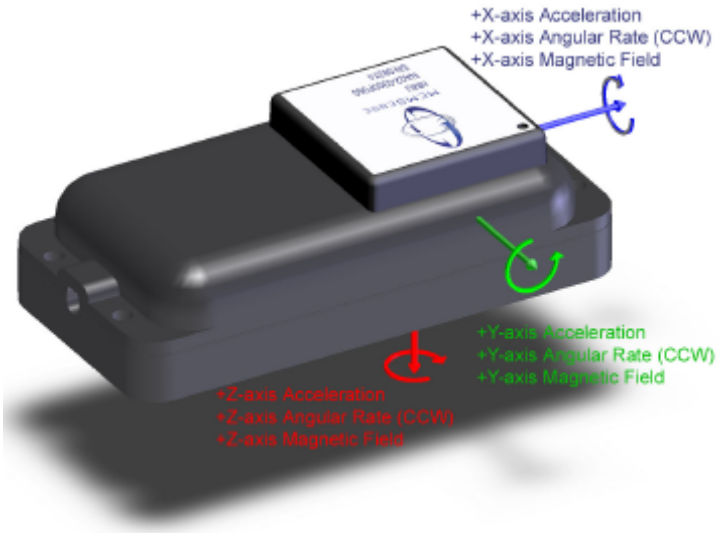


Fig. 1.4 El nIMU

Segons afirma el fabricant, quan el nIMU es va començar a comercialitzar va ser l'IMU basat en microprocessador més petit disponible al mercat. Segurament a dia d'avui, quasi tres anys després, aquesta afirmació pot ser falsa però ens va interessar la integració aconseguida. I és que el nIMU està format pels acceleròmetres i giroscopis requerits però també per un

microcontrolador i tots els circuits de condicionament de senyal i alimentació, que, d'altra manera, s'haurien de dissenyar.

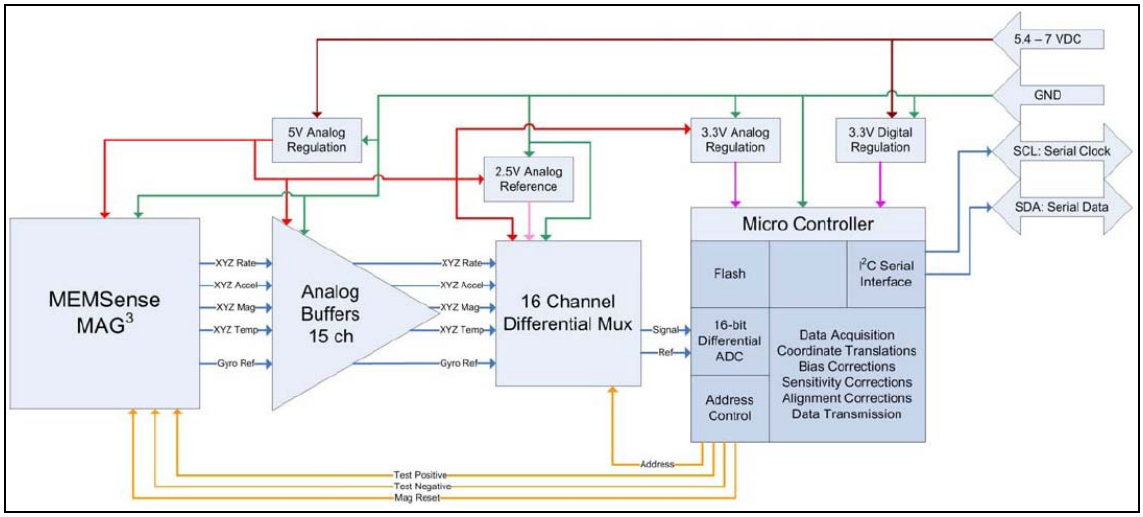


Fig. 1.5 Diagrama de blocs del nIMU

La unitat proporciona les dades d'acceleració, velocitat angular i camp magnètic en les tres dimensions de l'espai i compensades per temperatura, cosa que suavitza l'error acumulatiu de què s'ha parlat al principi. Tot i això, el més recomanable seria disposar d'un sistema de posicionament GPS per corregir periòdicament l'error de l'IMU.

La senyal de sortida del component és en format digital I^2C/SMBus, característica que ens elimina la problemàtica electrònica amb les impedàncies d'entrada i sortida deixant el bon funcionament de la comunicació a la programació posterior. A l'Annex A hi ha disponible un primer contacte amb el I^2C/SMBus des del S.O. GNU/Linux, exemple que permet conèixer les opcions disponibles al kernel per a aquest protocol.

Fig. 1.6 Detall del nIMU

CAPÍTOL 2. ADQUISICIÓ D'IMATGES

2.1 Descripció

Per a l'adquisició de les imatges s'utilitzarà una càmera CCD. L'element CCD (Charge-Coupled Device) és un dels principals components d'una càmera d'aquest tipus.

El CCD és una matriu de fotosensors que converteixen radiació (dins un rang limitat de longituds d'ona) en voltatge. La quantitat de fotosensors configura la quantitat de píxels de la imatge resultant.

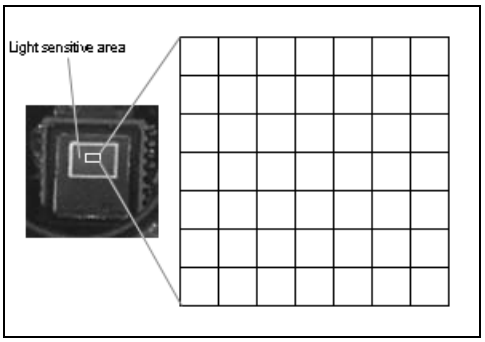


Fig. 2.1 CCD amb una zona de més sensibilitat

La càmera és de format PAL i té sortida de vídeo compost, a través d'un connector BNC.

Fig. 2.2 Càmera PAL utilitzada

La taula següent mostra els tipus de senyals i els connectors associats a aquests, mentre que la figura fa un esbós dels diferents connectors:

Taula 2.1 Tipus de senyals i connectors

Tipus	Connector
Vídeo compost	BNC, RCA
Y-C	Pin-DIN (4, 12 pins)
RGB	4 connectors BNC

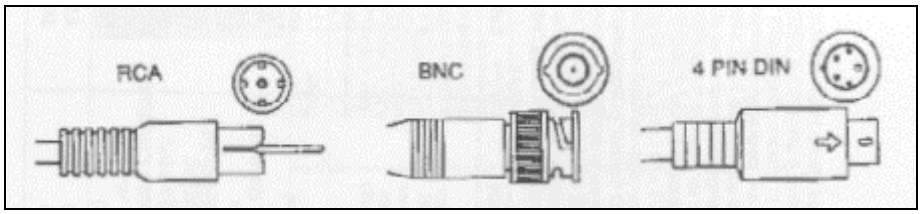


Fig. 2.3 Detall dels diferents connectors

2.2 Emmagatzematge

L'objectiu d'aquest projecte és la captura i emmagatzematge d'imatges que, posteriorment, han de ser tractades. Per tant, la codificació de la imatge ha de fer-se en algun format que produeixi la mínima pèrdua d'informació i s'adeqüi al tipus d'aplicació que se li donarà.

En termes generals hi ha dues formes bàsiques d'emmagatzemar la informació que conté una imatge digital: les imatges de mapa de bits (bitmap) i les imatges vectorials. Tot i això, les imatges fotogràfiques són un arxiu codificat en un format gràfic de mapa de bits, ja que el format vectorial no permet transmetre la sensació d'imatge real.

Les imatges de mapa de bits consisteixen en una matriu de gran dimensió sobre la qual es disposen una sèrie de bits d'informació. Aquests bits d'informació determinen el color i la posició de cada píxel i el total de píxels forma la imatge de mapa de bits. La definició dels principals tipus d'imatges són:

- ***TIFF (Tagges Image File Format)***. És un format de compressió per àrees que permet guardar les imatges amb la màxima qualitat, a més

d'especificar paràmetres propis per a la impressió. És un dels formats que més espai ocupen, però també el millor format per a ser imprès.

- ***BMP (Windows Bitmap Format)***. Format creat per Microsoft que permet tanta qualitat com les imatges TIFF, però que se'n diferencia pel fet que no aporta cap informació per a la seva impressió, cosa que comporta que les imatges ocupin menys espai de memòria. S'utilitza per a imatges que s'imprimeixen amb qualitat normal, o bé per a les imatges de qualitat que només han d'aparèixer en pantalla.

- ***GIF (Graphics Interchange Format)***. El format GIF és un dels més emprats al Web pel baix pes que implica generalment. La seva fórmula de compressió és secreta i pertany a Unisys. La principal característica d'aquest algoritme és que no perd informació, és a dir, que, un cop descomprimida la imatge, conserva la mateixa informació que quan es va comprimir. El format GIF se sol utilitzar per a les imatges de mode de color indexat, sempre que no tinguin massa tonalitats diferents de color.

- **PNG (*Portable Network Graphics)***. El PNG és un format que es va desenvolupar per millorar i reemplaçar el format GIF, l'algorisme del qual està patentat. El PNG supera al GIF en una encara millor compressió sense pèrdua de qualitat i una profunditat de color superior. Igual que el GIF se sol utilitzar amb imatges que no tinguin massa tonalitats diferents de color.

- ***JPEG (Joints Photographic Experts Group)***. Format de compressió per síntesi. Es fa servir per a les imatges digitals o web que contenen moltes tonalitats, com por exemple fotografies digitals i imatges amb degradats. Es considera el millor format de compressió per a les imatges fotogràfiques, ja que, a causa del seu algoritme de compressió, no dóna bons resultats amb imatges planes o de pes molt baix. Es tracta d'un format de compressió que comporta pèrdues, per molt poc que es vulgui comprimir, sempre es perd informació. El format JPEG disposa d'una profunditat de color de 24 bits, és a dir, de milions de colors, capaç, per tant, de suportar i representar imatges de qualitat fotogràfica.

CAPÍTOL 3. PLACA D'IMPLEMENTACIÓ

3.1 Descripció

La placa d'implementació d'aquest projecte és una XUPV2P (Xilinx University Program Virtex-II Pro). Aquesta placa serà l'element central per al desenvolupament dels sistemes d'adquisició d'imatges àrees. L'utilitzarem com el banc de proves entorn i a partir del qual s'anirà construint el sistema.

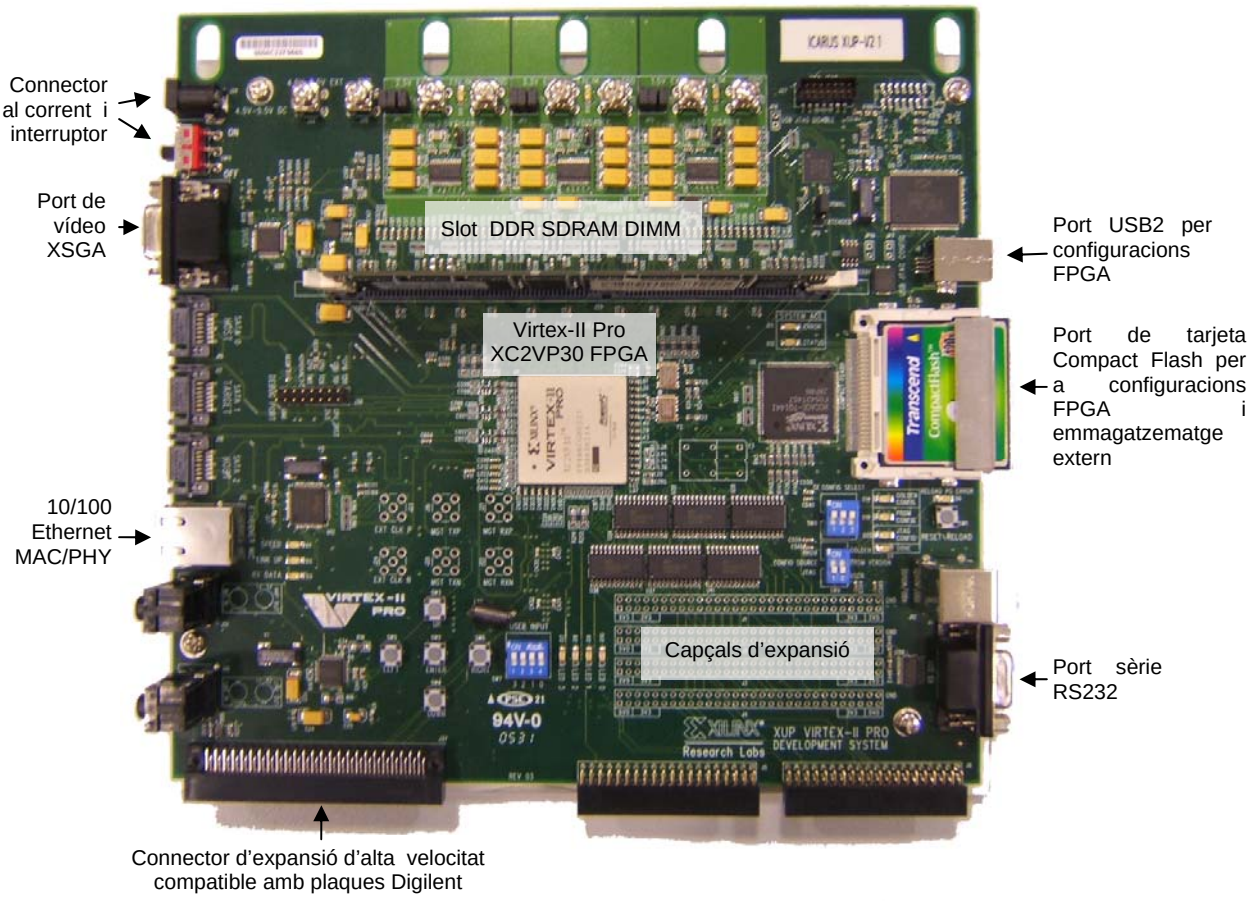


Fig. 3.1 Fotografia de la XUP Virtex-II Pro Development System Board

La XUPV2P ha estat dissenyada per la Xilinx Inc. com una eina d'aprenentatge tant per a estudiants com per a enginyers. És una eina de desenvolupament molt potent, de les més completes i versàtils del mercat. Degut al seu gran nombre de dispositius integrats i al potent entorn de disseny de què disposa és una plataforma útil per a un gran ventall de dissenys.

Aquesta placa tant pot servir com a sistema per al desenvolupament de microprocessadors i de dissenys digitals com per a ser utilitzada com a

receptacle per a nuclis de processadors encastats i sistemes digitals complexos. El cas del present projecte s'ubica en el segon bloc, com ja s'ha comentat a l'inici.

La placa es basa en una FPGA Virtex-II Pro. La FPGA, que és un dispositiu lògic programable, porta encastats dos processadors PowerPC de IBM. Si aquests processadors s'utilitzen en un disseny permeten l'execució de SO encastats com el GNU/Linux o el vxWorks.

Les eines de disseny disponibles són les estàndard que proporciona Xilinx Inc. per als seus productes de lògica programable. L'entorn es basa en el paquet ISE Fundation amb el qual es pot treballar en solitari si no es vol utilitzar cap processador encastat. Per al cas dels sistemes encastats, el d'aquest projecte, es requerirà també el paquet EDK (Embedded Development Kit), que passarà a ser la nostra eina de disseny. També hi ha disponible més programari com el ChipScope o el ModelSim.

Segurament l'inconvenient més important a l'hora d'escollir aquesta eina és el seu elevat preu, tant del maquinari com del programari associat. Per afavorir que s'utilitzin els seus productes, el venedor ofereix un descompte educatiu als centres universitaris. Per a més informació consultar la seva pàgina web [1].

3.1.1 Processadors

Un dels aspectes més atractius de la XUPV2P és el fet de disposar de dos hard-processors i de poder implementar diferents soft-processors dins la FPGA.

Fig. 3.2 Virtex-II Pro FPGA

Els hard-processors són microprocessadors IBM PowerPC 405. Aquest processador és una CPU RISC de 32 bits incrustada directament a la fàbrica

FPGA de la Xilinx per implementar aplicacions incrustades d'alt rendiment. La combinació de sistemes de doble processador PowerPC integrat amb la capacitat de processament simultani permet un ample ventall d'opcions d'optimització del rendiment.

El soft-processor és un microprocessador que es pot implementar completament utilitzant síntesi lògica. Els disponibles amb la Xilinx són el MicroBlaze i el PicoBlaze.

El processador MicroBlaze té arquitectura Harvard RISC a 32 bits amb un conjunt d'instruccions optimitzades per aplicacions incrustades a les FPGAs de la Xilinx. Aquesta arquitectura permet el control del tamany de la cache, les interfícies i l'execució d'unitats. A més és configurable: l'usuari pot canviar propietats per estalviar espai i així fer confluir preu i rendiment. El problema és que no té MMU i, per tant, no s'hi pot incrustar un GNU/Linux, en canvi sí que s'hi pot incrustar un µclinux o un FreeRTOS

El PicoBlaze està basat en arquitectura RISC de 8 bits i és d'ús lliure, el qual es distribueix amb els fitxers font. El fet de disposar del codi font VHDL fa que sigui més immune a la obsolescència dels productes, ja que permet una reutilització en futures FPGA de la Xilinx que implicarien una explotació de les aplicacions i reduirien el cost en el futur.

3.1.2 Perifèrics

System RAM: La placa XUPV2P disposa d'un slot per posar-hi una memòria RAM DDR. Es pot suportar fins a una capacitat de 2 GB. No es poden utilitzar totes les memòries, aquestes es restringeixen a les que són compatibles amb alguna de la taula següent. És molt important adquirir una RAM adequada ja que si no fos el cas el sistema no la reconeixeria, per saber si funciona és molt recomanable realitzar el test de memòria.

Taula 3.1 RAMs de Crucial Technology

Crucial Technology Memory Module	Mem. Organization	Number of Ranks	Registered or Unbuffered	CAS Latency
CT6472Z265.18T*	512MB64MX72	Dual	Unbuffered	2.5
CT6464Z265.16T*	512MB64MX64	Dual	Unbuffered	2.5
CT6472Z265.9T*	512MB64MX72	Single	Unbuffered	2.5
CT6464Z265.8T*	512MB64MX64	Single	Unbuffered	2.5
CT1664Z265.4T*	128MB16MX64	Single	Unbuffered	2.5

Taula 3.2 RAMs de Kingson Technology

Kingston Techonology Memory Module	Mem. Organization	Number of Ranks	Registered or Unbuffered	CAS Latency
KVR266X64C25/256	256MB32M X 64	Single	Unbuffered	2.5
KVR266X64C25/512	512MB64MX64	Dual	Unbuffered	2.5

System ACE Compact Flash Controller: El controlador SystemACE (Advanced Configuration Environment) controla la configuració de la FPGA. La configuració es pot fer des de diferents fons: de d'una targeta Compact Flash, a través de port JTAG, amb unes configuracions predefinides de test, enviament per USB2... La placa disposa d'un controlador SystemACE .

Fast Ethernet Interface: La XUPV2P disposa d'un sistema Fast Etherne que compleix els estàndards internacionals. Permet connexió dual de fins a 100Mb/s i incorpora una interfície que permet *modificar* la MAC de l'element.

Serial Ports: La placa disposa de tres ports sèrie: un port RS232 i dos ports PS/2. El primer d'aquests està pensat per permetre la comunicació amb un ordinador en consola remota, mentre que els dos següents estan dissenyats per connectar-hi un teclat i un ratolí.

Expansion Connectors: Una de les característiques que permeten entreveure una vida més llarga per a la placa són els connectors d'expansió. La placa té fins a 80 pins de la FPGA destinats a aquests connectors els quals es divideixen en un connector d'alta velocitat, dos connectors més muntats i fins a quatre capçals per a soldar sobre la placa. Aquí es poden incorporar moltes aplicacions i amb connexió directa als pins de la FPGA.

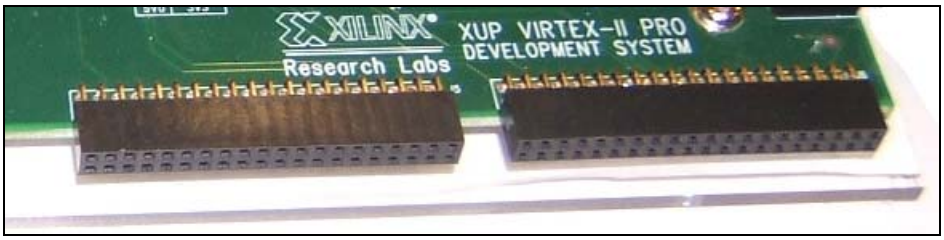


Fig. 3.3 Connectors d'expansió Digilent dret i esquerra

XSGA Output: La placa disposa d'un vídeo DAC per a sortida XSGA, compatible amb l'estàndard VESA 1280 x 1024 a 75Hz i un màxim de 1600x 1200 a 70Hz.

USB 2 Programming Interface: La XUPV2P inclou un sistema USB 2.0. Aquesta connexió s'utilitza com a eina de configuració i programació de la placa

CAPÍTOL 4. MUNTATGE

4.1 Descripció general

En aquest capítol es descriu tot el procés de disseny del projecte. Disseny que es divideix en dos mòduls, el de l'adquisició d'imatges i el de l'IMU, i que finalment s'integren fer formar el sistema final.

En tot aquest procés l'eina bàsica va ser el paquet *Embedded Development Kit* 7.1i [1], que, com indica el seu nom, inclou totes les eines de la Xilinx necessàries en cas que es vulgui treballar amb sistemes encastats, com en aquest cas.

Els programes utilitzats van ser el *Xilinx Platform Studio* i el XMD. En el primer programa és a on es va fer tot el disseny en sí, mentre que en el segon majoritàriament s'hi dugué a terme la creació final del fitxer per a la placa.

Amb el *Platform Studio* es configurava el sistema, tant de maquinari com de programari, i del qual se n'obtenien dos fitxers: el *.bit* (pel hardware) i el *.elf* (pel software). Llavors amb el programa XMD aquests dos fitxers eren utilitzats per crear el fitxer final, el *.ace*, que permetia executar el disseny a la placa.

Per a la prova empíricament dels dissenys, els fitxers *.ace* resultants es copiaven a una targeta Compact Flash que després s'introduïa a la placa i bolcava la configuració correcta. Perquè això fos possible, però, la targeta havia d'estar particionada correctament tal i com es detalla al punt "Using System ACE Controllers for Non-Volatile Storage" del *Hardware Reference Manual* de la placa.

4.1.1 Sistema Base

Per el desenvolupament de cada part del projecte es requerien unes prestacions, tant inicials com finals, diferents. Però, tot i així, sempre hi havia en comú una base sobre la qual s'anava construint.

Aquesta base era molt importat ja que fixava les característiques generals del processador, memòria, sistemes d'emmagatzematge, etc. El *Platform Studio* incorpora una eina que permet configurar aquest punt de partida de forma ràpida i entenedora: el *Base System Builder*.

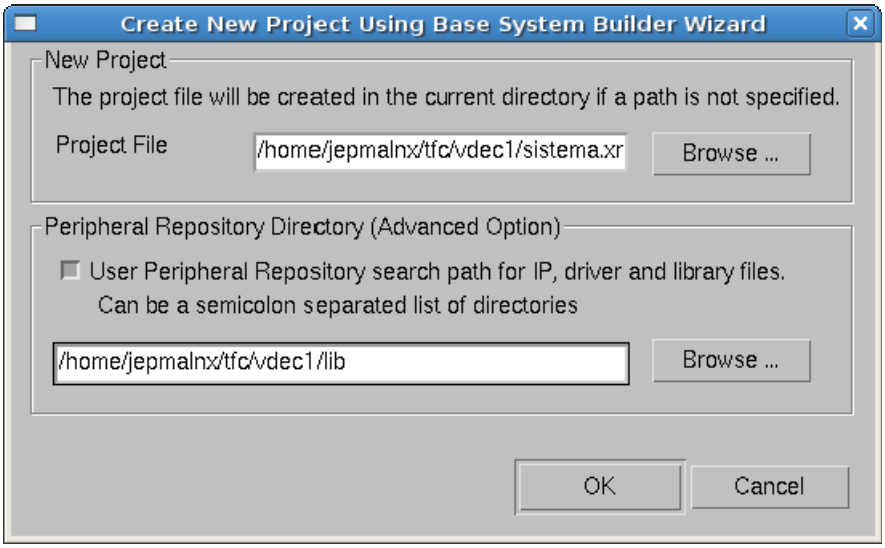


Fig. 4.1 Inici del Base System Builder

A la figura anterior es mostra la primera pantalla d'aquest constructor. Aquí és important remarcar que cal introduir la llibreria correcta al "Peripheral Repository Directory", ja que si no s'introdueix correctament no es podria utilitzar la nostra placa d'implementació. Aquesta llibreria es troba en la documentació de la placa i també està disponible a la seva pàgina web [2] [3].

Un cop seleccionada la placa "XUP Virtex-II Pro Development System", es seleccionava el PowerPC amb una freqüència de rellotge de 300MHz i els perifèrics que fessin falta. En general especificar que s'utilitzava el "RS232Uart_1" en la versió UARTLite, el "SysACE_CompactFlash" i la memòria RAM de què es disposava (256MB).

Per acabar es donava la màxima capacitat a la memòria del "PLB_BRAM" i s'inhabilitaven tots els testos que proporciona el programa per defecte. El Sistema Base ja estava creat.

Per a més informació es pot consultar el "EDK Base System Builder" [4] o el "Creating a Basic Hardware System in XPS" del *Platform Studio User Guide*. A més, a Internet hi ha disponibles un bon nombre de documents i tutorials que poden ser de gran ajuda tant pels primers passos com per aprofundir una mica més.

4.1.2 Creació del fitxer ACE

Per veure els resultats a la placa d'un disseny s'havia d'unir el fitxer de maquinari amb el de programari. El procés era senzill i es va fer utilitzant el programa XMD del mateix paquet EDK.

El més pràctic era crear un nou directori, copiar-hi els fitxers BIT i ELF i crear un script que permetés configurar el sistema tal i com era. Un cop copiats els fitxers s'havia de crear un nou fitxer amb nom "xupGenace.opt". El contingut d'aquest fitxer és el de la figura següent:

```
-jprog
-board user
-target ppc_hw
-hw <nom_projecte>.bit
-elf executable.elf
-configdevice devicenr 1 idcode 0x1127e093 irlength 14
partname xc2vp30
-debugdevice devicenr 1 cpunr 1
-ace sistema.ace
```

Fig. 4.2 Contingut del guió *xupGenace.opt*

Llavors s'executava el XMD i, a través de la línia de comandes, es situava el programa a la ruta on estaven guardats els tres fitxers anteriors, el *.bit*, el *.elf* i el *.opt*. Un cop situat s'executava l'ordre: "`xmd -tcl genace.tcl -opt xupGenace.opt`" i el fitxer ACE estava creat.

4.2 Implementació de l'adquisició d'imatges

4.2.1 VDEC1

Per a la utilització de la càmera PAL es requeria la utilització d'un complement per connectar-la a la placa i poder adquirir el vídeo, ja que la placa no disposa del connector necessari. La solució va ser utilitzar la VDEC1, que posa a disposició el mateix distribuïdor de la placa.

La VDEC1 (Video DECoder) [6] és una petita placa de Digilent [5] que es connecta a través del port d'expansió ràpid de la XUPV2P. Aquesta placa digitalitza els senyals analògics de vídeo que poden ser utilitzats en aplicacions de processat d'imatge i vídeo. Suporta els sistemes d'entrada PAL, NTSC i SECAM i disposa de tres connexions diferents: senyal composta (connector RCA), RGB (tres RCA) i S-Video (mini-DIN de 4 pins).

Fig. 4.3 Placa d'adquisició de vídeo VDEC1

El sistema és, bàsicament, una plataforma de suport per a treballar amb el xip d'Analog Devices ADV7183B [7], tal i com mostra la següent figura. Aquest circuit integrat és un descodificador de vídeo de 10 bits, que detecta i converteix automàticament les senyals analògiques de vídeo al format YCrCb 4:2:2.

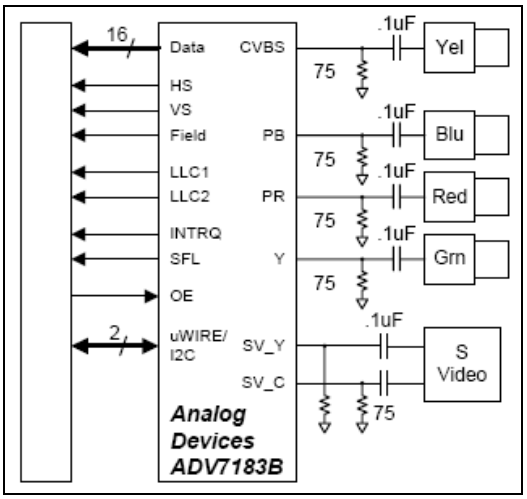


Fig. 4.4 Diagrama del circuit de la VDEC1

La documentació que es subministra amb la VDEC1 no és gaire completa. El motiu és que el fabricant pretén fer una introducció al sistema i en cas que es vulgui ampliar el disseny remet a la documentació del citat ADV7183B. Tot i això, pel muntatge que es volia dissenyar en un primer moment, la documentació havia de ser suficient.

La documentació es troba a la pàgina web del dispositiu amb el nom de "video_capture_rev_1_1.zip". Aquest comprimit compta amb uns quants exemples de configuració i execució i va ser utilitzat per la resta del projecte.

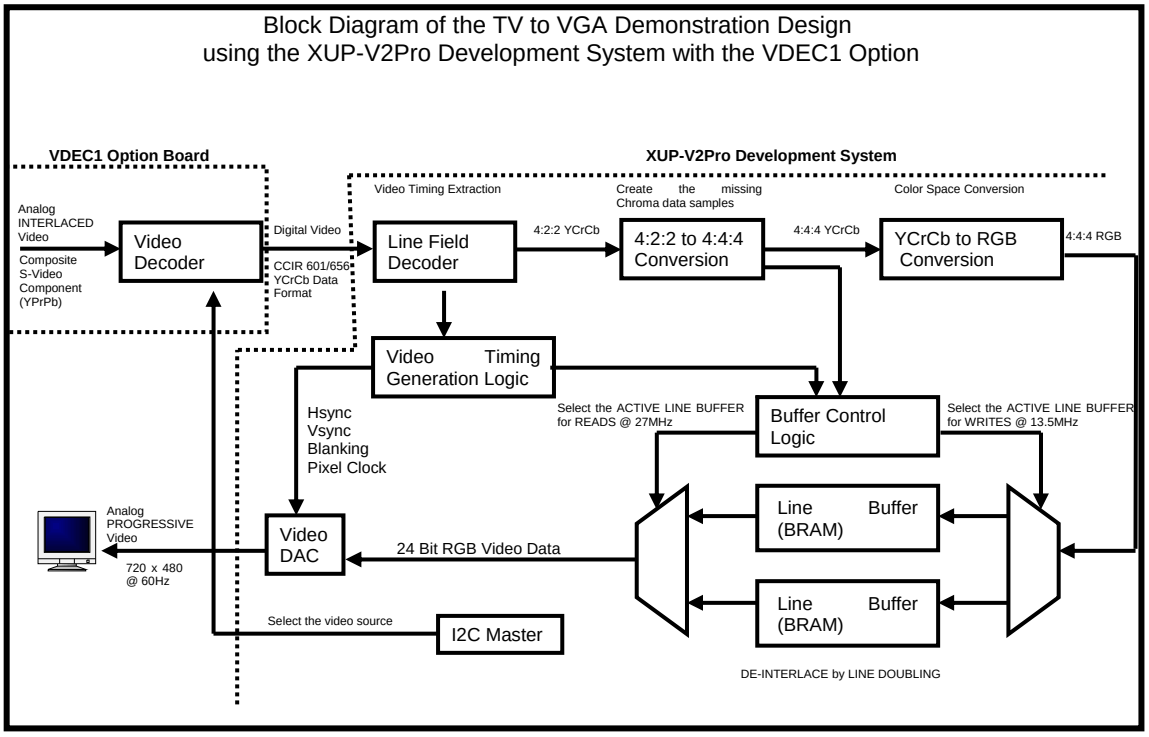


Fig. 4.5 Diagrama de blocs del disseny exemple

4.2.2 Muntatge

Per al muntatge de l'adquisició de vídeo es requeria el següent maquinari, a part dels connectors adients: la placa principal XUPV2P, una targeta Compact Flash, la VDEC1, la càmera PAL, un monitor VGA i un ordinador amb lector de CF. Es va muntar tot i, utilitzant el programa *Platform Studio* es va configurar el sistema hardware i el sistema software i es van provar els resultats a la placa.

Fig. 4.6 Connexió de la càmera a la VDEC1

4.2.2.1 Sistema hardware

Primer de tot s'havia de crear l'entorn de treball per a tot el muntatge. Aquest consisteix en la creació del directori des d'on es voldrà treballar i la còpia dels fitxers necessaris.

Com s'ha explicat abans, en el cas de la VDEC1 es disposa d'un comprimit amb tots els fitxers de configuració. Per aquest muntatge es requeriran els directoris "pcores" i "config_decoder_revb" tot i que, per començar, només es va copiar el directori "pcores" a l'entorn de treball. A més també es va copiar la llibreria necessària per poder utilitzar la placa de la Xilinx en la qual es basa tot el muntatge.

Un cop copiats els fitxers s'havia de fer una modificació en un d'ells, el "pcores/video_capture_v1_01_b/hdl/verilog/special_svaga_timing_generation.v" Aquest fitxer està configurat pel sistema nord-americà NTSC i, per tant, s'havia de configurar al de la càmera, que és del sistema PAL. Per fer-ho es va modificar la configuració comentant la disponible i afegint la que correspon, que es mostra a la següent figura:

```
// 720  X 576 @ 50Hz with a 27MHz pixel clock for PAL video capture
`define H_ACTIVE         720  // pixels
`define H_FRONT_PORCH    22   // pixels
`define H_SYNCH          64   // pixels
`define H_BACK_PORCH     58   // pixels
`define H_TOTAL          864  // pixels

`define V_ACTIVE         576  // lines
`define V_FRONT_PORCH  6   // lines
`define V_SYNCH        6   // lines
`define V_BACK_PORCH   37  // lines
`define V_TOTAL       625 // lines
```

Fig. 4.7 Configuració PAL

A continuació es va configurar el sistema hardware emprant el *Platform Studio*. Per al primer pas de crear el sistema base s'utilitzà l'eina *Base System Builder* i les opcions ja explicades.

Amb el maquinari mínim creat es va procedir a la configuració completa d'aquest primer muntatge. Per fer-ho es va utilitzar l'opció "Add/Edit Cores" del menú *Project*, eina que permet modificar el maquinari de manera ordenada i entenedora.

Degut a la mínima documentació de què es disposava, per realitzar aquesta configuració es va utilitzar de model el muntatge que inclou el comprimit de la VDEC1. Estudiant aquest exemple es dedueix que es basa en cinc elements: el

video_capture_0, l'*i2c*, l'*i2c_rst_or*, el *reset_split* i *l'and_gate_0*. A continuació es van incloure tots els perifèrics utilitzant les mateixes configuracions que les estudiades.

Aquesta eina es composa de cinc pestanyes. A la primera pestanya, la *Peripherals*, s'hi mostren tots els components del disseny que no són busos així com tots els possibles elements disponibles d'utilitzar en el sistema. S'ha de seleccionar de la columna de l'esquerra el perifèric en qüestió i assignar-li el nom d'instància que es vulgui. En el nostre cas va ser el que mostra la següent taula:

Taula 4.1 Pestanya *Peripherals*

Perifèrics	Instància
video_capture	video_capture_0
opb_iic	i2c
util_reduced_logic	i2c_rst_or
util_bus_split	reset_split)
util_reduced_logic	and_gate_0

A la pestanya "Bus Connections" s'hi mostra una matriu amb els busos del sistema i alguns dels perifèrics que hi estan connectats. Pel nostre disseny només es va haver d'afegir la connexió de l'*i2c sopb* al bus *opb*.

El mapa d'adreces del sistema hardware es determina, tal i com indica el seu nom, a la tercera pestanya, l'*Addresses*. Aquí es pot assignar, als components que ho requereixen, una posició de memòria automàtica o una de manual. A l'únic element afegit que requereix aquest pas, el *i2c*, se li va assignar una adreça automàtica i un espai de 512 bytes.

La pestanya *Ports* va ser la que més feina va implicar. És aquí a on es configuren els diferents ports de cada instància del disseny, com es connecten entre ells, si són exclusivament d'àmbit intern de la FPGA o si hi ha connexió amb els pins d'aquesta. Per a la configuració dels ports interns es va seguir la següent taula:

Taula 4.2 Connexió dels ports interns

Instància	Port Name	Net Name
opb	OPB_Rst	OPB_Rst
i2c	OPB_Clk	sys_clk_s
	OPB_Rst	i2c_rst

	Freeze	net_gnd
	Scl	Scl_decoder
	Sda	Sda_decoder
	Gpo	gpo_data
i2c_rst_or	Op1	OPB_Rst & i2c_rst_i
	Res	i2c_rst
video_capture	system_dcm_lockked	dcm_0_lock
	VDEC1_PWRDN_Z	VDEC1_PWRDN_Z
	VDEC1_OE_Z	VDEC1_OE_Z
	RESET_VDEC1_Z	RESET_VDEC1_Z
	COMP_SYNC	COMP_SYNC
	BLANK_Z	BLANK_Z
	V_SYNC_Z	V_SYNC_Z
	H_SYNC_Z	H_SYNC_Z
	PIXEL_CLOCK	PIXEL_CLOCK
	B	B
	G	G
	R	R
	LLC_CLOCK	LLC_CLOCK
	YCrCb_in	YCrCb_in
reset_split	Out2	vid_dec_rst_i
	Out1	i2c_rst_i
	Sig	gpo_data
and_gate_0	Res	vid_dec_rst
	Op1	vid_dec_reset_in & vid_dec_rst_i

Un cop creats els ports interns, es va passar als externs els quals, tot especificant paràmetres, eren els següents:

Taula 4.3 Connexió dels ports externs

Instància	Net Name	Range
i2c	Sda_decoder	
	Scl_decoder	
video_capture	VDEC1_PWRDN_Z	
	VDEC1_OE_Z	
	RESET_VDEC1_Z	
	COMP_SYNC	
	BLANK_Z	
	V_SYNC_Z	
	H_SYNC_Z	

	PIXEL_CLOCK	
	B	[7:0]
	G	[7:0]
	R	[7:0]
	LLC_CLOCK	
	YCrCb_in	[9:2]

A més d'aquests també es va crear un nou port extern des de zero. Tot i que aquesta connexió no s'utilitza es va introduir al disseny per si en futures modificacions es requereix, ja que el model seguit l'enllaça amb l'element intern *and_gate_0*. Les dades d'aquesta connexió són:

Taula 4.4 Port extern nou

Nom	Tipus	Xarxa
vid_dec_reset_in	IN	vid_dec_reset_in

Per a l'última de les pestanyes, la *Parameters*, es van utilitzar i modificar les següents propietats:

Taula 4.5 Pestanya *Parameters*

Instància	Paràmetre	Valor
i2c	C_IIC_FREQ	10000
	C_GPO_WIDTH	2
reset_split	C_SIZE_IN	2
	C_LEFT_POS	0
	C_SPLIT	1
and_gate_0	C_OPERATION	and
	C_SIZE	2
i2c_rst_or	C_OPERATION	or
	C_SIZE	2

Arribats a aquest punt l'estructura interna del nostre sistema hardware estava creada, només faltava especificar la localització física dels nous ports externs. Aquest pas no es va fer a través del programa *Platform Studio* (tot i que també hagués estat possible) sinó que es va utilitzar un editor de text del nostre Sistema Operatiu.

La tasca a dur a terme consisteix en enllaçar cada un dels ports externs afegits amb un pin de la FPGA, a més de definir el funcionament del pin en qüestió. Es pot fer a partir de la documentació de la placa Xilinx i de la VDEC1, tot buscant cada port extern a la placa VDEC1 i veient per quin pin enllaça a la FPGA.

Tot i això, el més senzill i ràpid és utilitzar l'exemple que inclou el comprimit de la VDEC1, ja que el resultat serà exactament igual respecte a la localització dels ports. Per realitzar la tasca es va obrir el fitxer de l'entorn de treball "data/<nom_projecte>.ucf" i es van copiar al final d'aquest les línies de la 133 a la 180 (*inclusive*) del fitxer "data/system.ucf" del comprimit .

Tornant al *Platform Sudio*, a la pestanya *Applications* s'escollí el "Default: ppc405_0_bootloop" per a que inicialitzi el BRAMs. Aquesta opció evita que el processador entri en estats desconeguts durant l'inicialització de la placa. És important remarcar que només hi pot haver una opció seleccionada.

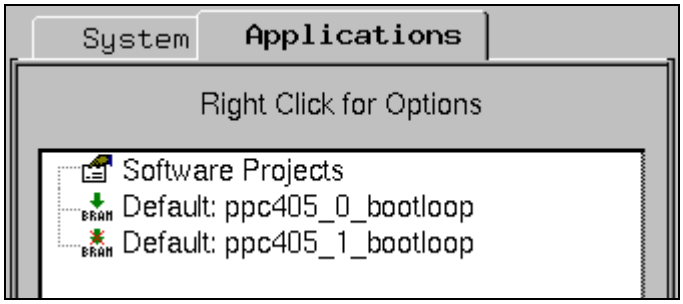


Fig. 4.8 Selecció del *bootloop*

Per finalitzar només mancava generar el fitxer de hardware, el que té extensió *bit*. S'utilitzà l'eina *Generate Bitstream* del menú *Tools*. Després d'un dotze minuts de compilació el programa generà una sèrie de fitxers, entre ells el "implementation/<nom_projecte>.bit" que ens interessava.

4.2.2.2 Sistema software

Un cop creat el fitxer del maquinari es va procedir a la creació del fitxer de software. Utilitzant el programa *Platform Studio* es va anar a la pestanya *Applications* i es va crear un nou projecte de programari tot clicant amb el botó dret sobre "Software Projects".

Creàrem un directori al nostre entorn de treball amb el mateix nom que el del programa creat. Llavors del comprimit de la VDEC1 es va copiar el directori "config_decoder_revb/src" al directori del nostre entorn de treball "<nom_programa>/src".

D'entre els fitxers copiats es van esborrar el "i2c_master.c" i el "xup_config_decoder.c" i es va crear un nou fitxer anomenat "main.c". Aquest fitxer és el que portava el programa principal la funció del qual era obrir una terminal UART i inicialitzar el xip de la VDEC1 per a què captés a través del port de la senyal composta. El contingut d'aquest fitxer es pot consultar a l'Annex B.1.

Al tornar al *Platform Studio*, a la pestanya *Applications*, es van afegir els fitxers ".c" disponibles a l'apartat *Sources* i els ".h" al de *Headers*.

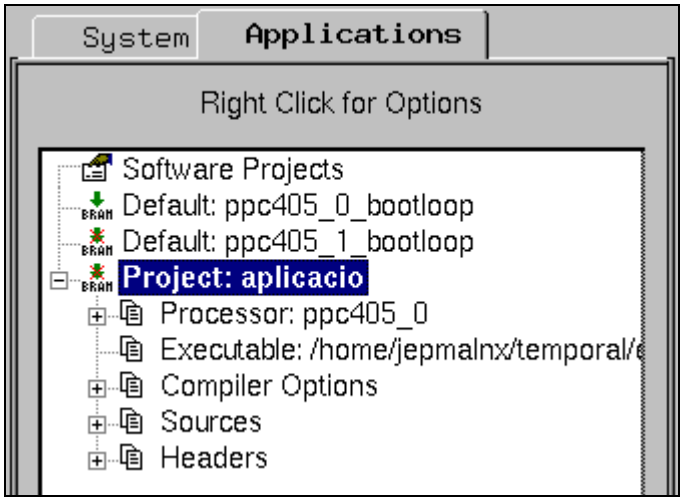


Fig. 4.9 Pestanya *Applications*

Quan tot va estar especificat es va procedir a compilar el programari clicant amb el botó dret i seleccionant "Build Project". En menys de cinc minuts es va obtenir el fitxer "<nom_programa>/executable.elf".

4.2.2.3 Resultat

Un cop creats els dos fitxers es van fusionar amb el XMD tal i com s'ha explicat en l'apartat anterior. El fitxer ACE resultant es va provar a la placa i el resultat va ser satisfactori.

Per una banda hi havia la prova gràfica que el monitor VGA mostrava el que la càmera PAL estava filmant i, implícitament, demostrava la correcta programació de la placa VDEC1. Per altre banda l'emulador de terminal Minicom mostrava el text de configuració correcta de la VDEC1.

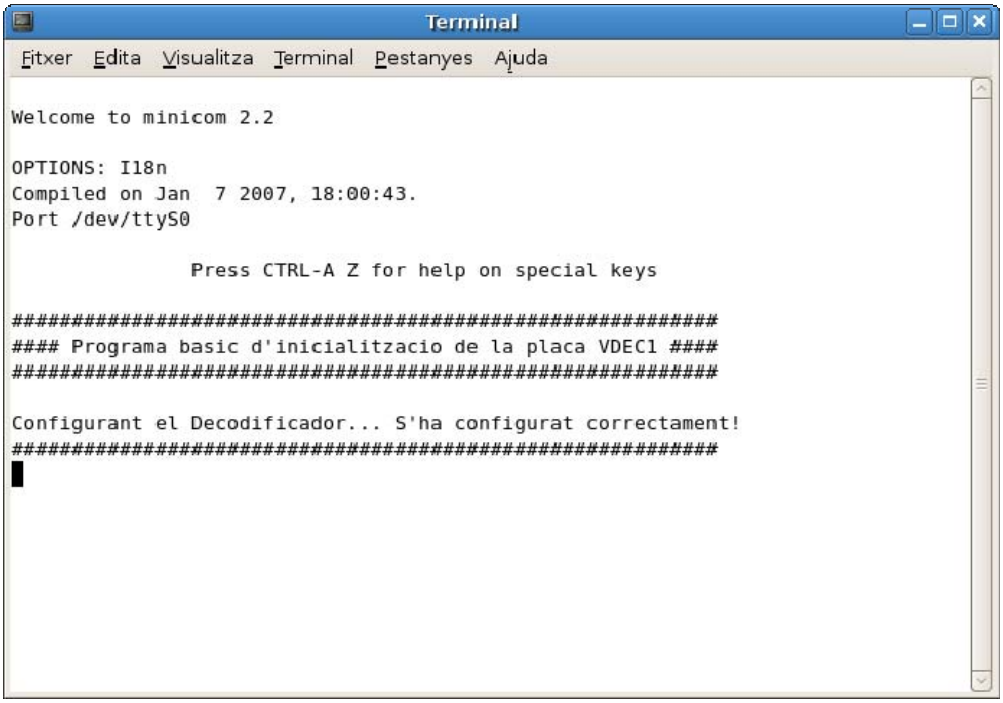


Fig. 4.10 Captura del programa Minicom

Arribats aquí es comprovava que aquest muntatge bàsic era correcte i s'oferia, així, una base sobre la qual poder ampliar el sistema per a l'objectiu d'emmagatzemar les imatges.

4.3 Implementació de l'Unitat de Mesura Inercial

4.3.1 Disseny

Per a la utilització de la Unitat de Mesura Inercial, com en el cas per a l'adquisició de vídeo, es requeria la utilització d'un complement per connectar el dispositiu a la placa. Però, a diferència del cas anterior, cap dels distribuïdors consultats disposava d'un accessori adequat i, per aquest motiu, se'n va dissenyar un de propi.

El mòdul que es volia dissenyar havia de permetre connectar la nostra Unitat de Mesura Inercial, el nIMU, a la placa i utilitzar-lo correctament. A més a més, es va decidir que també incorporaria un port sèrie RS232 que serviria per augmentar les possibilitats de la placa de manera interessant.

Primer de tot, i per saber com enfocar el disseny, es van buscar les possibles zones de la placa preparades per connectar-hi l'element i, tal i com s'han mostrat al Capítol 3, es van localitzar els ports d'expansió. En aquest cas es van utilitzar els "Expansion Headers" ja que permeten treballar amb tires de connectors de pas estàndard (2.54mm) tant si és soldant-ho directament a la

placa i al mòdul o a través d'un sòcol a la placa. A la següent figura es poden observar els quatre capçals d'expansió disponibles:

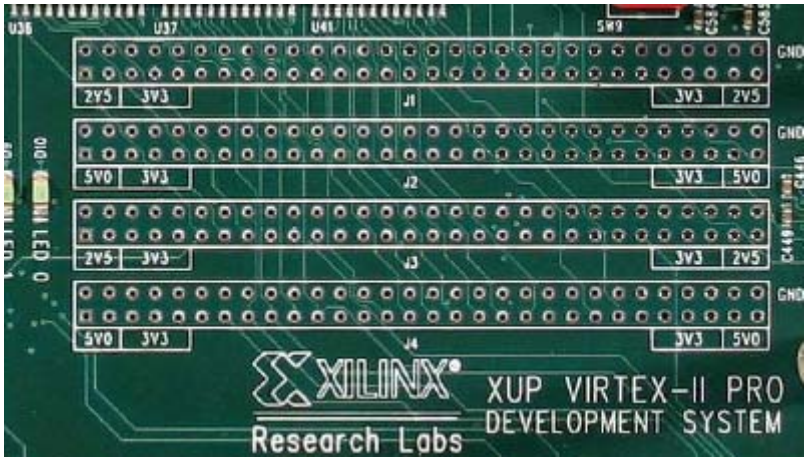
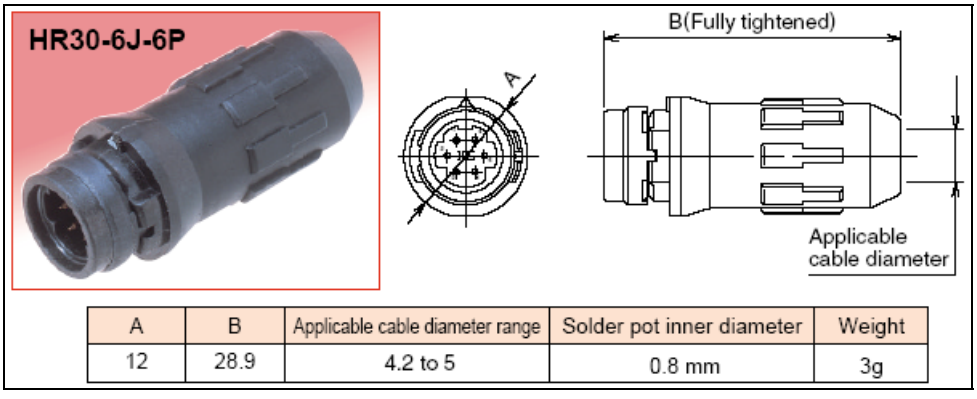


Fig. 4.11 Expansion Headers

Un cop fixada la zona de treball i les seves possibilitats es van anar especificant les característiques del mòdul, començant per la connexió amb el nIMU.

Segons la documentació del producte [8] el nIMU disposa d'un connector Hiroshe HR30-6P-6S [9] que consta de sis pins i dels quals se n'utilitzen quatre, dos per alimentació i dos més per la transmissió de dades. El que requeria el mòdul era un connector de 90º que permetés endollar el nIMU horitzontalment a la placa, al no existir es va optar pel connector Hiroshe "HR30-6J-6P" [9]:



A	B	Applicable cable diameter range	Solder pot inner diameter	Weight
12	28.9	4.2 to 5	0.8 mm	3g

Fig. 4.12 Connector Hiroshe HR30-6J-6P

Amb aquest model s'obliga a que el muntatge del connector es faci externament utilitzant quatre cables (agrupats o no) i que després es soldi cada

cable al pin correcte del mòdul que estem dissenyant. La disposició dels pins en el connector ha de ser la que marca la següent figura:

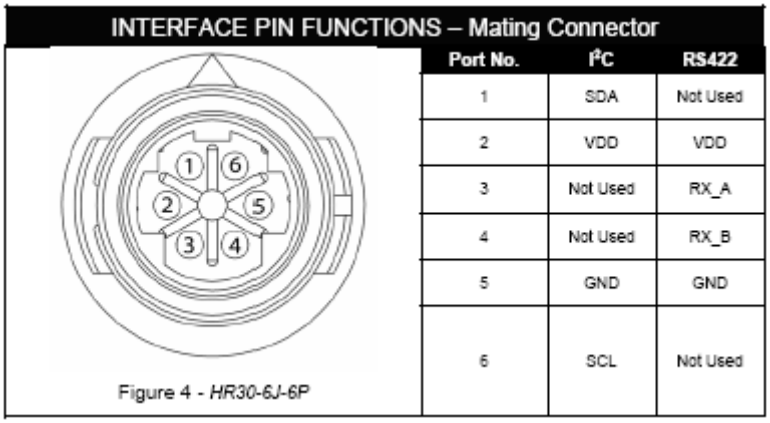


Fig. 4.13 Descripció dels pins del connector

Quan es va fer el disseny de la placa del mòdul els dos pins de transmissió de dades (1 i 6) es van connectar a dos pins de la FPGA de la placa, mentre que pels dos pins d'alimentació (2 i 5) es va afegir un nou connector al mòdul que permetrà endollar-hi una font d'alimentació externa. Aquesta font externa és necessària ja que la placa Xilinx pot proveir un voltatge màxim de 5V i, en canvi, el nIMU necessita una alimentació d'entre 5,4V i 9V.

Per al disseny del port sèrie es va partir d'un circuit estàndard per aquest tipus de ports, tal i com mostra la següent figura. Simplement consta del xip SN65C3221 [10], d'uns condensadors de desacoblament i del connector final.

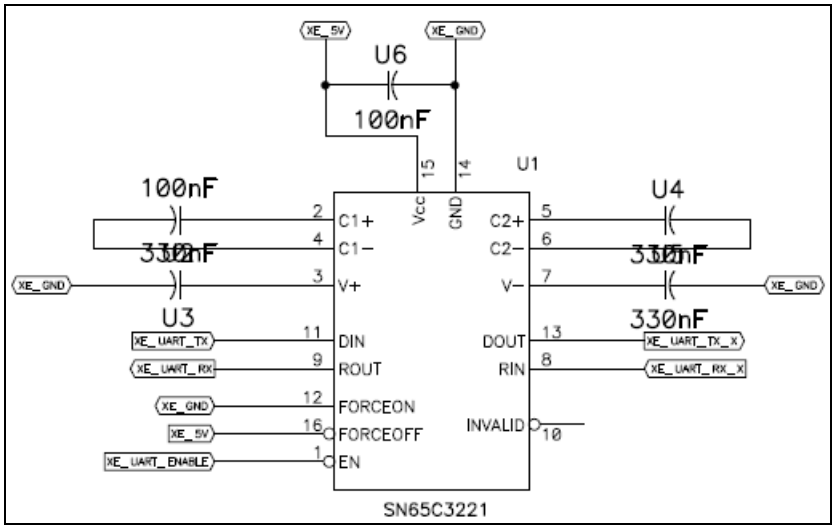


Fig. 4.14 Port sèrie RS232

En aquest punt es va especificar que el segon i el quart capçal serien els utilitzats per sustentar i connectar el mòdul. Això és així ja que el circuit del port sèrie es pot alimentar dels 5V que proveeixen aquests capçals.

Respecte al connector de 90º s'utilitzen els pins estàndards d'entrada (2) i sortida (3), els quals estan lligats als senyals que van a la FPGA després o abans de passar pel xip SN65C3221, que adequa la senyal. A més, el disseny incorpora el control directe de la senyal d'habilitació o no habilitació del circuit a través de la senyal "XE_UART_Enable".

El port sèrie resultant no permet control de fluxos ni altres aplicacions avançades. Tot i això aquest element pot ser de gran utilitat i dóna noves possibilitats per a la majoria d'aplicacions. Un exemple molt clar és el fet de poder tenir dues terminals remotes, les quals es podrien utilitzar independentment si de manera simultània volem executar els dos hard-processors PowerPC que incorpora la placa, o d'un d'aquests i el soft-processor Microblaze. D'aquesta manera es podrien explotar molt més les grans prestacions d'aquesta placa de desenvolupament de la Xilinx.

A l'Annex D es disposa de l'esquemàtic complet del disseny, mentre que el *layout* de la placa és el següent:

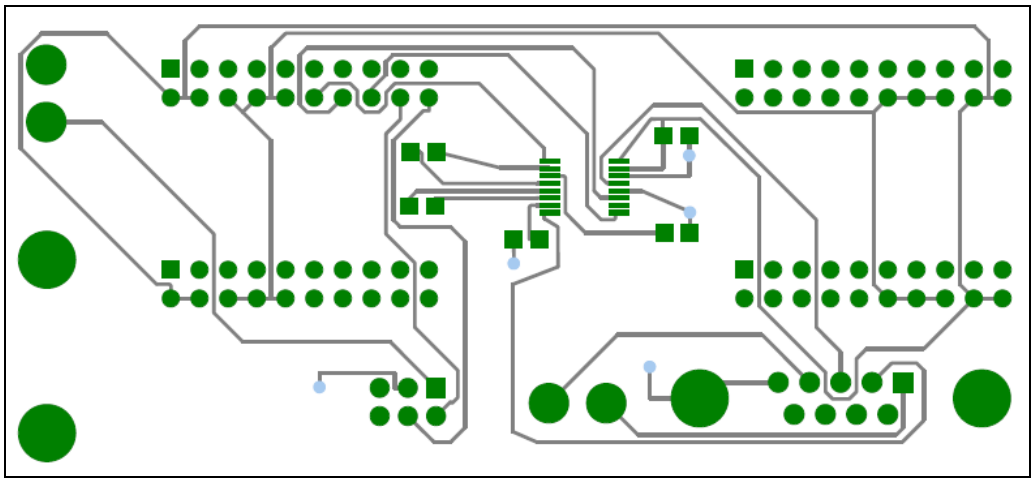


Fig. 4.15 *Layout* de la placa XilExpa v1.0

4.3.2 Característiques del XilExpa

El mòdul final s'ha anomenant XilExpa v.1.0.

Aquest mòdul requereix estar connectat a la placa XUPV2P pels connectors inferiors perquè funcioni el port sèrie. Si també es vol que el connector del nIMU sigui operatiu s'ha de disposar d'una font d'alimentació externa que es

connecti correctament a l'element *Wire_Switch* número 1 i que subministri un voltatge d'entre 5,4V i 9V amb una intensitat mínima de 113mA.

La següent taula identifica les connexions entre el connector del nIMU, el port sèrie i la FPGA.

Taula 4.6 Pinout de XilExpa v.1

Dispositiu	Senyal	Nom intern	Entrada FPGA	Tipus I/O
nIMU (I^2C)	SDA	EXP_IO_20	P8	LVTTL
	SCL	EXP_IO_21	P7	LVTTL
UART	RX	EXP_IO_22	N4	LVTTL
	TX	EXP_IO_23	N3	LVTTL
	ENABLE	EXP_IO_24	P3	LVTTL

4.3.3 Muntatge

Per al muntatge d'aquest mòdul es requeria el següent maquinari: la placa principal XUPV2P, una targeta Compact Flash, la XilExpa, una font d'alimentació, el nIMU i un PC amb lector de CF. En aquest mòdul el procés de disseny es va quedar en el camp teòric ja que no es disposava de la placa XilExpa.

4.3.3.1 Sistema hardware

Primer de tot es va crear un nou projecte partint del *Base System Builder* però amb una modificació: no es va incloure cap "RS232 Uart 1". Un cop fet es va procedir a crear dos nous elements: un i2c i un port sèrie. El port sèrie que es volia crear era per utilitzar el que incorpora la XilExpa, per això no se'n va afegir cap en la creació del Sistema Base.

Seguint el passos que s'han descrit en l'apartat anterior es va configurar el sistema hardware tal i com mostren les taules següents.

Taula 4.7 Instàncies i la seva connexió interna

Perifèrics	Instància	Port Name	Net Name
opb_iic	i2c	OPB_Clk	sys_clk_s

		OPB_Rst	i2c_rst
		Freeze	net_gnd
		Scl	Scl
		Sda	Sda
		Gpo	gpo_data
uartlite	XilExpa_UART	OPB_Clk	Sys_clk_s
		Interrupt	XilExpa_UART_Interrupt
		RX	XilExpa_UART_RX
		TX	XilExpa_UART_TX

Taula 4.8 Connexió dels ports externs

Instància	Net Name	Range
i2c	Sda	
	Scl	
XilExpa_UART	XilExpa_UART_RX	
	XilExpa_UART_RX	

Taula 4.9 Paràmetres de les instàncies

Instància	Paràmetre	Valor
i2c	C_IIC_FREQ	10000
	C_GPO_WIDTH	2
XilExpa_UART	C_BAUDRATE	9600
	C_DATA_BITS	8
	C_ODD_PARITY	0
	C_USE_PARITY	0
	C_CLK_FREQ	100000000

Per acabar de definir la configuració, pel cas de la instància i2c es va haver de connectar el *i2c sopb* al bus *opb* i se li va assignar una posició de memòria amb una ocupació de 512 bytes. Pel cas de la instància XilExpa_UART també es va connectar el seu *XilExpa_UART sopb* al bus *opb* i se li va assignar una posició de memòria, però amb un espai de 64KB.

El pas següent va ser especificar a quin pin enllaçava cada port extern per després afegir-ho al fitxer UCF del sistema. La relació amb la FPGA va quedar fixada com segueix:

Taula 4.10 Configuració dels pins de la FPGA

Instància	Nom del port extern	PIN	Informació
i2c	Sda	AE5	IOSTANDARD = LVTTL
			DRIVE = 8
			PULLUP = TRUE
	Scl	AB8	IOSTANDARD = LVTTL
			DRIVE = 8
			PULLUP = TRUE
XilExpa	XilExpa_UART_RX	AJ8	IOSTANDARD = LVCMOS25
	XilExpa_UART_TX	AE7	IOSTANDARD = LVCMOS25
			SLEW = SLOW
			DRIVE = 12

Finalment es va escollir el *bootloop* pel primer processador i es va compilar el disseny.

4.3.3.2 Sistema software

Es va programar una petita aplicació per treballar amb el nIMU que es pot consultar. El programa havia de seguir el protocol I^2C/SMBus [11] [12], per tant el primer que feia era inicialitzar la comunicació i tot seguit llegia els registres de l'aparell. La informació enviada pel nIMU havia de seguir l'ordre programat pel fabricant:

BYTE	ELEMENT	BYTE	ELEMENT
0	Synchronization byte (FF)	21	Accelerometer Y (2/5g) (MSB)
1	Synchronization byte (FF)	22	Accelerometer Y (2/5g) (LSB)
2	Synchronization byte (FF)	23	Accelerometer Z (2/5g) (MSB)
3	Synchronization byte (FF)	24	Accelerometer Z (2/5g) (LSB)
4	Message size	25	Magnetometer X (MSB)
5	Device ID	26	Magnetometer X (LSB)
6	Message ID	27	Magnetometer Y (MSB)
7	Sample Timer (MSB)	28	Magnetometer Y (LSB)
8	Sample Timer (LSB)	29	Magnetometer Z (MSB)
9-12	Reserved	30	Magnetometer Z (LSB)
13	Gyro X (MSB)	31	Temperature Gyro X (MSB)
14	Gyro X (LSB)	32	Temperature Gyro X (LSB)
15	Gyro Y (MSB)	33	Temperature Gyro Y (MSB)
16	Gyro Y (LSB)	34	Temperature Gyro Y (LSB)
17	Gyro Z (MSB)	35	Temperature Gyro Z (MSB)
18	Gyro Z (LSB)	36	Temperature Gyro Z (LSB)
19	Accelerometer X (2/5g) (MSB)	37	8-bit Checksum
20	Accelerometer X (2/5g) (LSB)		

Fig. 4.16 Ordre de les dades del nIMU

4.3.3.3 Resultat

No es va poder finalitzar el muntatge i fer la prova amb la placa ja que no es disposava del mòdul XilExpa.

Fig. 4.17 Kit bàsic i connector del nIMU

4.4 Muntatge final

4.4.1 Descripció

Un cop dissenyades les diferents parts del sistema, el pas final era la unió dels dos mòduls creats. Aquests, a priori, haurien d'haver estat plenament operatius, però no va ser així. El mòdul d'adquisició estava configurat i testat però el mòdul de l'IMU només permetia una integració a nivell de configuració del sistema.

Es van fer dos muntatges: el primer es va realitzar en *standalone* mentre que el segon consistia en un S.O. GNU/Linux Encastat. La diferència entre els dos muntatges era purament en el sistema del programari, és a dir, els dos sistemes van utilitzar la mateixa configuració hardware només que pel cas del sistema encastat es generaren uns fitxers de més.

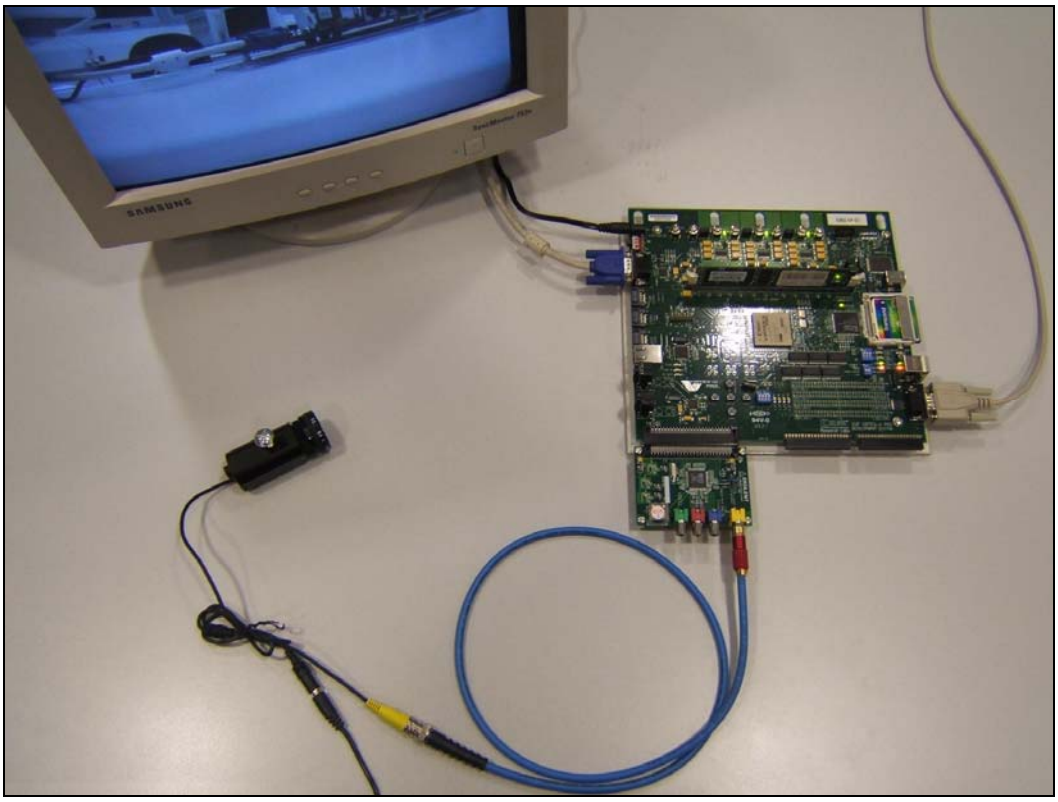


Fig. 4.18 El muntatge final

El material necessari per a la implementació del sistema d'adquisició d'imatges aèries consistiria en la placa principal XUPV2P, la VDEC1, la placa XilExpa, una font d'alimentació, una targeta Compact Flash, el nIMU, la càmera PAL, un monitor VGA i un ordinador amb lector de CF.

El muntatge final, però, no compta amb la placa XilExpa. Llavors es necessita la placa principal XUPV2P, una targeta Compact Flash, la VDEC1, la càmera PAL, un monitor VGA i un ordinador amb lector de CF.

4.4.2 Sistema hardware

Com s'ha dit al principi, el muntatge final va consistir en la realització de dos projectes separats però que compartien la mateixa configuració del maquinari de la placa. Aquesta configuració consistia en la fusió dels dos sistemes hardware de cada mòdul.

Hi havia diferents maneres d'integrar els dos mòduls. Per una banda es podia utilitzar un dels dos dissenys realitzats i incorporar-hi la configuració de l'altre o es podia crear un nou disseny des de zero i incorporar-hi les dues configuracions.

Es va optar per partir de zero ja que s'havien anat fent proves en tots els projectes disponibles. Un cop tot va estar configurat es compilà.

Per saber la configuració del sistema final consultar les configuracions parcials dels dos mòduls i ajuntar-les.

4.4.3 Sistema software

4.4.3.1 Processador en standalone

Un sistema en standalone significa que no té un Sistema Operatiu. Aquest fet implica que l'aplicació software opera dins la plataforma hardware a través de crides directes al sistema

Aquesta és l'opció per defecte del *Platform Studio* i és el sistema utilitzat en el disseny dels dos mòduls. La zona a on es defineix quin sistema es vol utilitzar es troba a "Software Platform Settings" dins el menú *Project*. En aquesta eina hi ha el "Kernel and Operating Systems" i, pel nostre projecte d'un sol processador, el *ppc405_0* ha d'estar en aquesta opció.

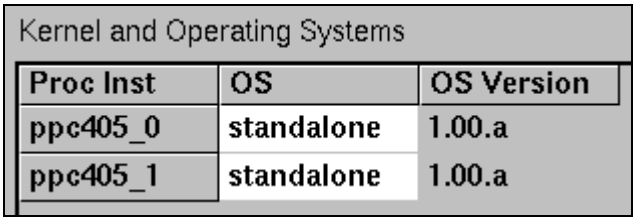


Fig. 4.19 Pestanya *Software Platform*

És en aquesta eina a on també s'especifica una opció que s'hauria d'haver utilitzat en cas de disposar del XilExpa. Aquesta opció és la definició del *stdin* i *stdout* que haurien permès d'utilitzar el port sèrie dissenyat per establir una connexió de la placa amb l'ordinador, comprovant, així, el funcionament de l'element.

Instance	Current Value	Default Value	Type
ppc405_0 : standal...			
enable_sw_intru...			
enable_sw_in...	false	false	bool
profile_timer	none	none	periphe
stdin	RS232_Uart_1	none	periphe
stdout	RS232_Uart_1	none	periphe
need_xil_malloc	false	false	bool
microblaze_exce...	((XEXC ...	((XEXC_UNAI	array

Fig. 4.20 Pestanya Library/OS Parameters

Per definir el sistema software es va utilitzar la mateixa configuració que la del mòdul d'adquisició d'imatges. El cas ideal hagués estat la programació d'una aplicació que s'encarregués de tot el sistema, és a dir, que inicialitzés el xip de la VDEC1 permetent el flux de vídeo i que també inicialitzés el nIMU per a poder consultar les seves mesures.

Resultat

El resultat d'aquesta configuració, com era d'esperar, va ser el mateix que el resultat per al mòdul d'adquisició d'imatges, en el monitor es mostrava el que la càmera estava filmant (següent figura). Va servir, en definitiva, per constatar que no hi havia hagut conflictes importants en la integració del sistema hardware.

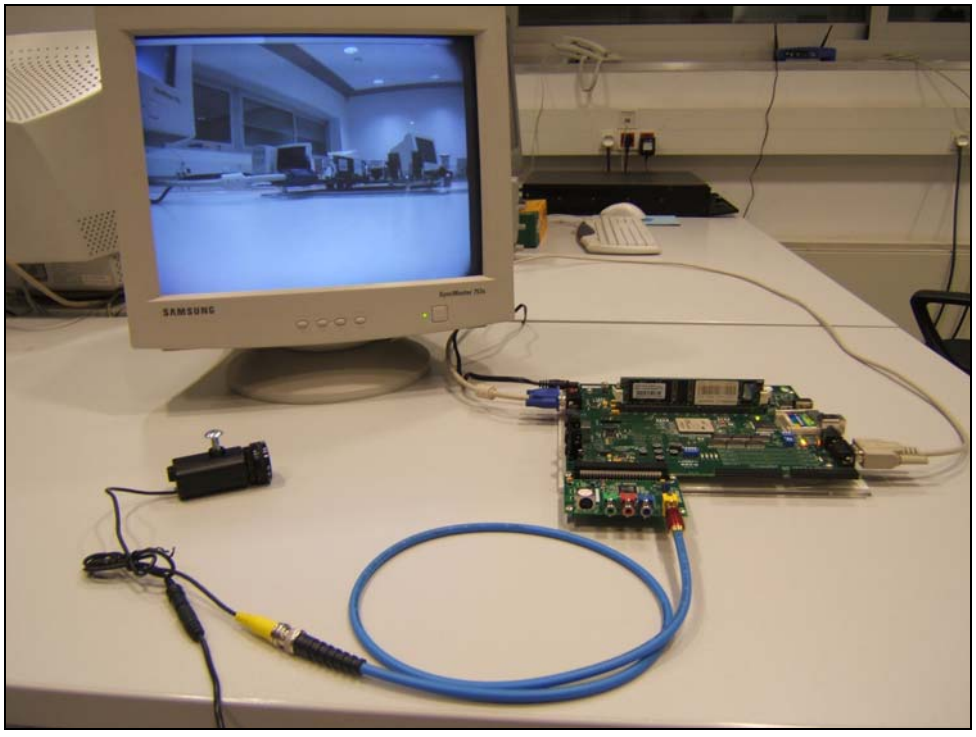


Fig. 4.21 Detall del monitor mostrant el que envia la càmera

4.4.3.2 S.O. GNU/Linux encastat

Configurant el sistema perquè usi un S.O. encastat permet d'establir una plataforma sobre la qual implementar aplicacions. D'aquesta manera s'obtenen alguns beneficis com que es pot aplicar el coneixement que es tingui en aquestes plataformes per a un desenvolupament més ràpid de noves aplicacions, sense necessitar un període d'adaptació més llarg.

El programari utilitzat per realitzar aquest sistema software consta de: nucli del Linux 2.6 [13], Busybox [14] i Crosstool [15]. Les versions exactes utilitzades van ser: per al kernel la versió 2.6.23-rc9 amb suport per als Virtex de la Xilinx [16], per al Busybox la 1.7.2 i pel compilador creuat la versió 0.43.

Aquest sistema també requeria que la targeta CF estigués particionada de la següent manera: la primera partició havia de tenir el mateix format que fins llavors, la segona havia de ser una Swap de Linux (82) i la última una Linux (83). És en aquesta última a on es muntarà el sistema de fitxers. Per a més informació consultar la documentació que està especificada al final del següent apartat..

Creació del sistema

El primer que es va fer va ser configurar el nostre disseny perquè permetés utilitzar el GNU/Linux com a Sistema Operatiu. Per a fer-ho es va seleccionar l'opció *linux_mvl31* a l'eina *Software Platform Settings* del menú Project, tal i com mostra la següent figura.

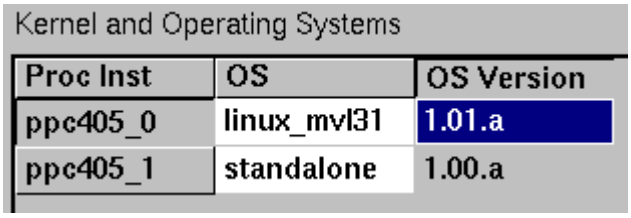


Fig. 4.22 Configuració de kernel

L'opció escollida fa referència a una distribució de Linux concreta dedicada als sistemes encastats: la MontaVista Linux 3.1. Tot i això, en principi es pot utilitzar qualsevol altre versió. El problema que hi va haver és que la versió 7.1i de l'EDK tenia compatibilitat amb el kernel 2.4, quan la versió estable era la 2.6.

Aquest fet va comportar errors en la posterior compilació del nucli 2.6 utilitzant els fitxers que generats amb el *Platform Studio*. Aquests eren errors per variables i posicions de memòria desconegudes. Al final va resultar que el problema era degut a que el fitxer que guarda les adreces de memòria i les

passa al nucli durant la compilació havia canviat de directori, la solució es detalla més endavant.

Amb aquest primer pas fet es va passar a la pestanya *Library/OS Parameters* per configurar els paràmetres del S.O. Les dades introduïdes són les que mostra la següent taula:

Taula 4.11 Pestanya *Library/OS Parameters*

Variable	Valor
MEM_SIZE	0x10000000
PLB_CLOCK_FREQ_HZ	100000000
TARGET_DIR	(carpeta a on es vol guardar els BSP)
Connected_peripherals	RS232_Uart_1, SysACE_CompactFlash, opb_intc_0

Acte seguit es va seleccionar el *bootloop* perquè inicialitzés el primer processador PowerPC i es van generar tots els fitxers implicats. Primer es va tornar a generar el sistema hardware clicant a “Update Bitstream” del menú Tools. Quan va acabar es va procedir a crear els fitxers pròpiament dits del sistema software.

Per a fer-ho es va seleccionar “Generate Libraries and BSPs” del mateix menú Tools. Amb aquesta acció es generaven uns fitxers anomenats BSP (de Board Support Package) que després s’havien d’utilitzar per la creació del nucli i que es guardaven al TARGET_DIR especificat.

Els BSP permeten a una aplicació de programari encastat una correcta inicialització i comunicació amb els recursos de hardware connectats al processador. En general els BSP contenen codis d’arrencada, d’inicialització i de drivers de dispositius.

van haver generat els fitxers aquests es van afegir a l’estructura del nucli que es volia incrustar. Si el fitxer ja existia s’havia de sobreescriure. A més, per corregir l’error entre versions del kernel es va moure de lloc el fitxer “arch/ppc/platforms/xilinx_ocp/xparameters_ml300.h” i es va desar a “arch/ppc/platforms/4xx/xparameters” amb el mateix nom.

Un cop es va tenir tot preparat s’havia de generar el nucli i el seu sistema de fitxers (RFS, de les sigles en anglès Root File System), es van utilitzar les versions del programari especificat al principi.

El primer que es va fer va ser crear el nucli. Es va configurar per a processadors PowerPC (ARCH:= ppc) i pel compilador creuat. També es copià el fitxer de configuració “arch/ppc/configs/ml300_defconfig” al root del nucli amb el nom “.config”. Amb aquest últim pas s’assegurava la compatibilitat amb la

nostra placa predefinint unes configuracions bàsiques. Llavors es va acabar de configurar el nucli mitjançant l'ordre "`make menuconfig`" i modificant les següents seleccions:

- Platform option
  - Default bootloader kernel argument
    - Console=ttyUL0,9600 root=/dev/xsysace3 rw
- Networking
  - [ ] Networking support
- Device Drivers
  - Block devices
    - < * > Xilinx SystemACE support
  - Character devices
    - < * > Hardware Random Number Generator
    - Serial drivers
      - < > 8250/16550 and compatible serial support
      - < * > Xilinx uartlite serial port support
        - [*] Support for console on Xilinx uartlite ...
  - Graphics Support
  - < * > Lowlevel video output switch controls
- Kernel hacking
  - [ ] Kernel debugging

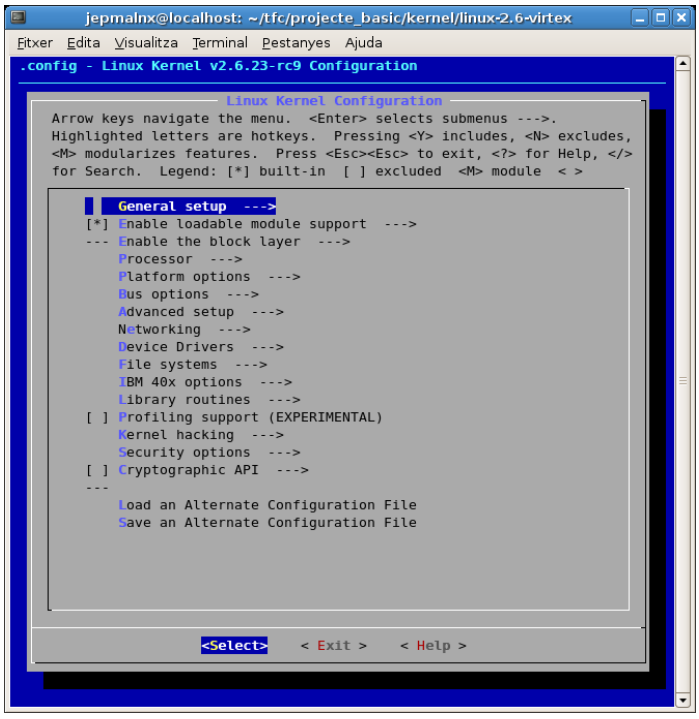


Fig. 4.23 Menú de configuració del kernel

Només faltava exportar el compilador creuat i compilar el kernel amb un "`make`". Quan va acabar la compilació el fitxer de sistema software es trobava a "arch/ppc/boot/images/zImage.elf". Finalment es va crear el fitxer ACE. En aquest cas l'ACE no permetia iniciar el disseny, no permetria iniciar el SO. Això era així ja que per a fer-ho necessitava un sistema de fitxers (RFS).

Per a crear-lo es va utilitzar el Busybox 1.7.2 [14]. Es va descarregar de la seva pàgina web, es va configurar i utilitzant el script MKROOTFS d'en J. Donaldson [18] es va crear el RFS. Aquest sistema de fitxers es va copiar a la tercera partició de la targeta CF.

Del RFS s'en va modificar el fitxer "/etc/inittab" per a que utilitzés el ttyUL0, es va afegir la línia *d'askfisrt* i es va treure l'ordre que executava el *getty*. Per últim, es van crear els dispositius necessaris, tal i com marca la següent taula:

Taula 4.12 Dispositius en el SO encastat

Nom del dispositiu	Tipus	Major	Minor
console	c	5	1
null	c	1	3
ttyUL0	c	204	187
i2c-0	c	89	0

Per a més informació es pot consultar, per exemple, la wiki del mantenidor de la versió del nucli utilitzat "Linux on Xilinx Virtex" [19]. O també el "Porting MontaVista Linux to the XUPV2P" [17], el "Configuring linux for the XUPV2P" [20], el "Configuring and Installing Linux on Xilinx FPGA Boards" [21] o el "Porting Linux to Xilinx ML300, ML310 and XUP" [22]. Tot i que totes les pàgines esmentades són correctes, la més recomanable és la primera al tractar-se de la més actualitzada i activa, a part de ser del mateix desenvolupador que el nucli utilitzat.

Resultat

Amb el fitxer ACE a la primera partició de la targeta CF, amb una segona partició Swap i amb el sistema de fitxers a la tercera partició, la Compact Flash contenia tot el que era necessari per a crear el nostre S.O. GNU/Linux encastat.

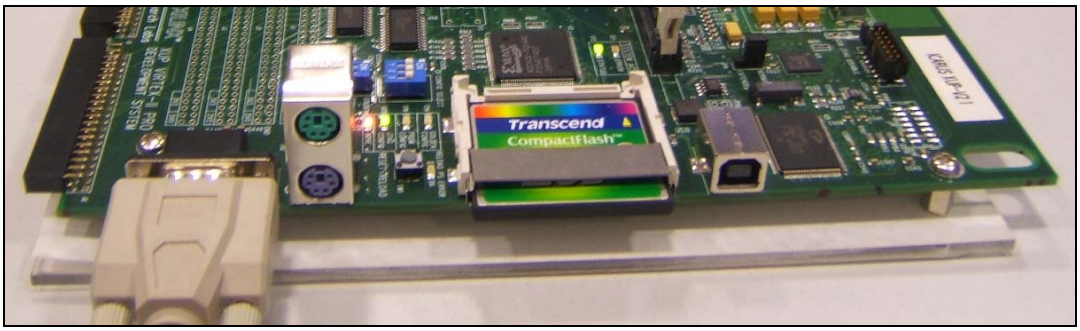


Fig. 4.24 Detall de la placa amb la CF i el UART

Es va introduir per enèsima vegada la CF dins el seu adaptador i es va connectar la placa. Primer es va encendre el LED D12 per mostrar que el contingut de la CF era el correcte i després ho va fer el LED D4 per confirmar que el sistema ACE s'havia carregat a la memòria. El nostre S.O. GNU/Linux encastat s'activava.

Des del Minicom de l'ordinador es podia seguir tot el procés d'arrencada ja que el sistema UART funcionava correctament, igual que la resta de dispositius i configuracions (veure l'Annex E). Un cop es va haver carregat del tot, es va rebre un missatge d'activació de la terminal remota per a que es pogués treballar en aquella estació i va aparèixer el *prompt*.

El sistema funcionava correctament i per testejar-lo es va executar una senzilla aplicació (Annex B.2) per a treballar amb dispositius compatibles I^2C/SMBus. Aquesta s'havia creat a partir de les explicacions del fitxer "Documentation/i2c/dev-interface" de la documentació del kernel. L'aplicació pretenia utilitzar el dispositiu "i2c-n" per programar el xip de la VDEC1 i que el monitor mostrés el que la càmera PAL filmava, cosa que ja s'havia aconseguit en *standalone*.

L'aplicació no va funcionar donant l'error de "No such device". Pel que s'ha pogut esbrinar aquest error no semblaria causat, exclusivament si més no, per un mal disseny del sistema, sinó que estaria estretament lligat amb el fet que els drivers utilitzats són d'una versió anterior del kernel ja que per a la versió actual encara no se'n disposa.

Per a poder comunicar-se rudimentàriament amb els elements I^2C la solució podria passar per treballar a nivell de bit, sense drivers. Això seria factible degut a que l'aplicació que es vol dur a terme és molt simple,

Arribats aquí ja s'havia testat aquest muntatge final i se n'havia comprovat el seu correcte funcionament. Aquest havia estat un pas important per al projecte i, sobretot, per a futures ampliacions o aplicacions doncs permetia disposar d'una potent plataforma com ho és la GNU/Linux per al desenvolupament i test d'aplicacions.

CONCLUSIONS

Durant el desenvolupament d'aquest projecte s'han pogut comprovar les dificultats que suposa el desenvolupament real d'una aplicació. En tot el procés la dinàmica ha consistit en encarar els problemes que sorgien i fer la implementació amb el màxim rigor i rapidesa possible.

S'ha aconseguit utilitzar la càmera de vídeo des de la placa de desenvolupament deixant les bases perquè, en un futur, es pugui implementar l'objectiu final de capturar i desar instantànies. La direcció a seguir podria ser la de modificar el programari per a que desés els bits d'informació en una zona de la memòria o també es podria dissenyar una aplicació externa que es connectés als ports adients i que dugués a terme aquesta tasca d'emmagatzematge.

En el tema del posicionament espacial s'han estudiat les diferents possibilitats i s'ha deixat un sistema a punt de finalitzar la implementació bàsica. En el cas que es volgués integrar dins d'un sistema encastat, si més no pel cas del GNU/Linux, les opcions més factibles passen per deixar de banda els drivers disponibles i programar una petita aplicació de comunicació directa.

Un dels objectius més aconseguits ha estat l'aprenentatge en disseny basat en FPGA i en la constatació de les grans oportunitats que aquestes plataformes ofereixen. A més, una de les experiències personals més engrescadores, tant per la novetat com per la dificultat que va suposar, va ser la d'encastar i fer funcionar un S.O. GNU/Linux a partir d'un nucli. Com a proposta d'ampliació es podria utilitzar un core d'OpenCores.org que permet codificar i desar imatges en format JPEG.

La impressió final és la d'haver descobert i treballat en un camp molt interessant i de deixar una base de treball sobre la qual finalitzar els punts que han restat oberts i per crear altres aplicacions.

BIBLIOGRAFIA

Bibliografia Referenciada

[1] Embedded Development Kit.
http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm

[2] Placa Xilinx University Program Virtex-II Pro Development System.
http://www.xilinx.com/univ/xupv2p.html

[3] XUPV2P a la web de Digilent.
http://www.digilentinc.com/Products/Detail.cfm?Nav1=Products&Nav2=Programmable&Prod=XUPV2P

[4] EDK Base System Builder.
http://www.digilentinc.com/Data/Products/XUPV2P/XUPV2P_Base_System_Builder.pdf

[5] Digilent Inc.
http://www.digilentinc.com/

[6] VDEC1.
http://www.digilentinc.com/Products/Detail.cfm?Prod=VDEC1&Nav1=Products&Nav2=Accessory

[7] Analog Devices ADV7183B, 10-Bit NTSC/PAL/SECAM TV & Video Decoder. http://www.analog.com/en/prod/0,,760_793_ADV7183B,00.html

[8] Nano Inertial Measurement Unit.
http://www.memsense.com/downloads/datasheets/nIMU.pdf

[9] Hiroshe Miniature Waterproof Plastic Connectors.
www.hirose.co.jp/cataloge_hp/e13000041.pdf

[10] Catalog of Datasheets.
http://www.datasheetcatalog.net

[11] "Using the i2c bus"
http://www.robot-electronics.co.uk/htm/using_the_i2c_bus.htm

[12] SMBus specification. http://smbus.org/

[13] Linux Kernel with Xilinx Virtex Support.
http://git.secretlab.ca/git/gitweb.cgi?p=linux-2.6-virtex.git;a=summary

[14] Busybox. http://www.busybox.net/

[15] Crosstool. http://www.kegel.com/crosstool/

[16] Versió del nucli “Xilinx Virtex Support”
http://git.secretlab.ca/git/gitweb.cgi?p=linux-2.6-virtex.git;a=summary

[17] “Porting Monta Vista to the XUPV2P Development Board”
https://rm-rfroot.net/xupv2p/

[18] “Porting Monta Vista ...” MKROOTFS script.
https://rm-rfroot.net/files/courses/xupv2p/docs/mkrootfs.tar.gz

[19] “Linux on Xilinx Virtex”
http://wiki.secretlab.ca/index.php/Linux_on_Xilinx_Virtex

[20] “Configuring linux for the XUPV2P”
http://www.cs.washington.edu/research/lis/empart/xup_ppc_linux.shtml

[21] “Configuring and Installing Linux on Xilinx FPGA Boards”.
http://splish.ee.byu.edu/projects/LinuxFPGA/configuring.htm

[22] “Porting Linux to Xilinx ML300, ML310 and XUP”
http://www.klingauf.de/index.html

Bibliografia Consultada

[23] Xilinx Inc., *XUPV2P Development System. Hardware Reference Manual*, (2005)

[24] Xilinx Inc., *Platform Studio User Guide. EDK 7.1i*, (2005)

[25] Xilinx Inc., *Embedded System Tools Reference Manual. EDK*, (2005)

[26] Wakerly, J. F., Diseño digital principios y prácticas, México [etc.] Pearson Educación (2001)

[27] Van Den Bout, D. E., The practical XILINX designer lab book, version 1.5, Upper Saddle River Prentice-Hall (1999)

[28] Yaghmour, Karim Building embedded Linux systems, Beijing [etc.] O'Reilly (2003)

[29] Berger, A., Embedded systems design an introduction to processes, tools and techniques, Lawrence, Kansas CMP Books cop. (2002)

[30] Catsoulis, J., Designing embedded hardware 2nd ed., Beijing [etc.] O'Reilly (2005)

[31] Zurawski, R. (Editor), Embedded systems handbook, Boca Raton, Fla. CRC PRESS Taylor & Francis (2006)

ANNEXOS

A. El protocol SMBus

Per poder treballar amb el System Management Bus des del kernel Linux hem de carregar els mòduls de I^2C. Això és degut a que el SMBus es basa en els principis d'operació d'aquest protocol.

Per a fer una primera aproximació aprofitem el fet que una placa de PC comuna usa el SMBus per comunicar-se amb diversos dels *sensors* de què disposa (temperatura, voltatge...).

A l'ordinador a on farem la prova hi tenim instal·lada la distribució GNU/Linux Debian "Sarge" 3.0r0 per a AMD64, amb la versió 2.6.8-amd64 del nucli, i en el qual es preten monitoritzar els sensors disponibles.

Primer s'ha de disposar d'un nucli amb suport I^2C i després tenir instal·lat el software necessari. En el cas que el nostre nucli no tingui aquest suport s'haurà de compilar incloent-hi els mòduls necessaris.

Compilació del nucli

Per a la configuració de les opcions del nucli usem el mode gràfic amb xconfig. Al directori a on hi ha les fonts del nucli (/usr/src/kernel-headers***) executem, sempre des de superusuari:
make xconfig

Amb aquesta comanda executem el programa de configuració. Suposant que tenim les opcions mínimes necessàries seleccionades, ens desplacem a l'apartat que ens interessa de I^2C.

Les opcions per aquest protocol són:

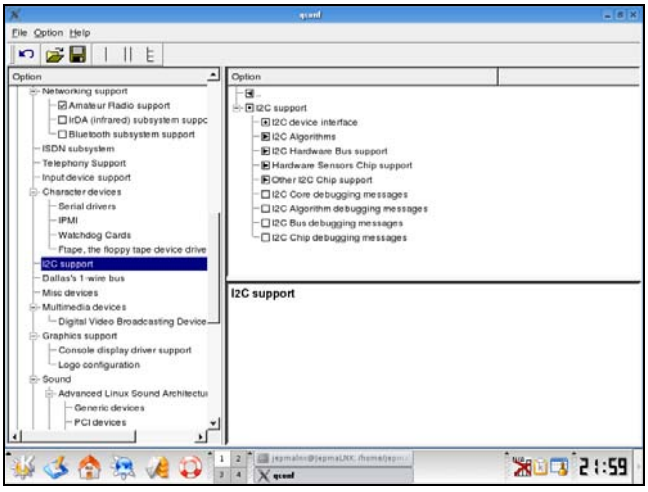


Fig. A.1 Opcions del nucli per l'I^2C

Seleccionem “I2C support” i els següents subapartats en funció de la nostra placa.

Un cop editades les configuracions compilem amb un “`make all`” i les altres opcions necessàries. Reiniciem al nou nucli i comprovem si tenim instal·lat el paquet *lm-sensors*, si no el tenim l’instal·lem.

Treballar amb els sensors

Per fer un anàlisi dels sensors del sistema executem sensors-detect. El resultat ens indica que hem d’afegir unes línies al fitxer /etc/modules i que ho pot fer automàticament (la sortida completa del programa es troba al final del document). Els dispositius que hagi trobat els guarda al fitxer /etc/sensors.conf.

```
To make the sensors modules behave correctly, add these lines
to /etc/modules:

#----cut here----
# I2C adapter drivers
i2c-viapro
i2c-isa
# I2C chip drivers
eeprom
w83627hf
#----cut here----
```

Fig. A.2 Fitxer /etc/sensors.conf

Llavors carreguem els mòduls al nucli tal i com segueix, i podem comprobar el resultat amb lsmod.

```
jepmalnx@JepmaLNX:/usr# /etc/init.d/module-init-tools
Calculating module dependencies... done.
Loading modules...
      ide-cd
      ide-disk
      ide-generic
      psmouse
      sd_mod
      i2c-viapro
      i2c-isa
      eeprom
      w83627hf
All modules loaded.
```

Fig. A.3 Càrrega dels mòduls al kernel

Amb la comanda "sensors" obtenim la informació que donen els elements que s'han detectat en el pas anterior.

```
JepmaLNX:/usr/src/kernel-headers-2.6.8-2-386# sensors
w83697hf-isa-0290
Adapter: ISA adapter
VCore:     +1.52 V  (min =  +0.08 V, max =  +0.29 V)       ALARM
+3.3V:     +3.23 V  (min =  +0.13 V, max =  +2.06 V)       ALARM
+5V:       +4.97 V  (min =  +1.18 V, max =  +0.03 V)       ALARM
+12V:     +10.88 V  (min =  +5.84 V, max =  +0.97 V)       ALARM
-12V:      +2.77 V  (min = -13.27 V, max =  +1.62 V)       ALARM
-5V:       +5.10 V  (min =  -4.09 V, max =  -0.68 V)       ALARM
V5SB:      +5.40 V  (min =  +5.43 V, max =  +4.73 V)       ALARM
VBat:      +0.32 V  (min =  +0.16 V, max =  +3.39 V)
fan1:        0 RPM  (min =  490 RPM, div = 32)              ALARM
fan2:     3629 RPM  (min = 12500 RPM, div = 2)              ALARM
temp1:       +35°C  (high =  +102°C, hyst =   +35°C)   sensor =
thermistor
temp2:     +50.5°C  (high =   +80°C, hyst =   +75°C)   sensor =
thermistor
alarms:   Chassis intrusion detection                      ALARM
beep_enable:
          Sound alarm disabled

eeprom-i2c-0-51
Adapter: SMBus Via Pro adapter at 0400
Memory type:            DDR SDRAM DIMM
Memory size (MB):       512

eeprom-i2c-0-50
Adapter: SMBus Via Pro adapter at 0400
Memory type:            DDR SDRAM DIMM
```

Fig. A.4 Captura de la sortida del sensors

Aplicació gràfica

Una utilitat gràfica que permet monitoritzar aquests sensors és la barra d'estat *gkrellm* amb la qual també podem crear events si es sobrepassen uns llindars, entre altres opcions. Les següents captures són preses des de l'escriptori Gnome ja que oferia millor resolució que al KDE.

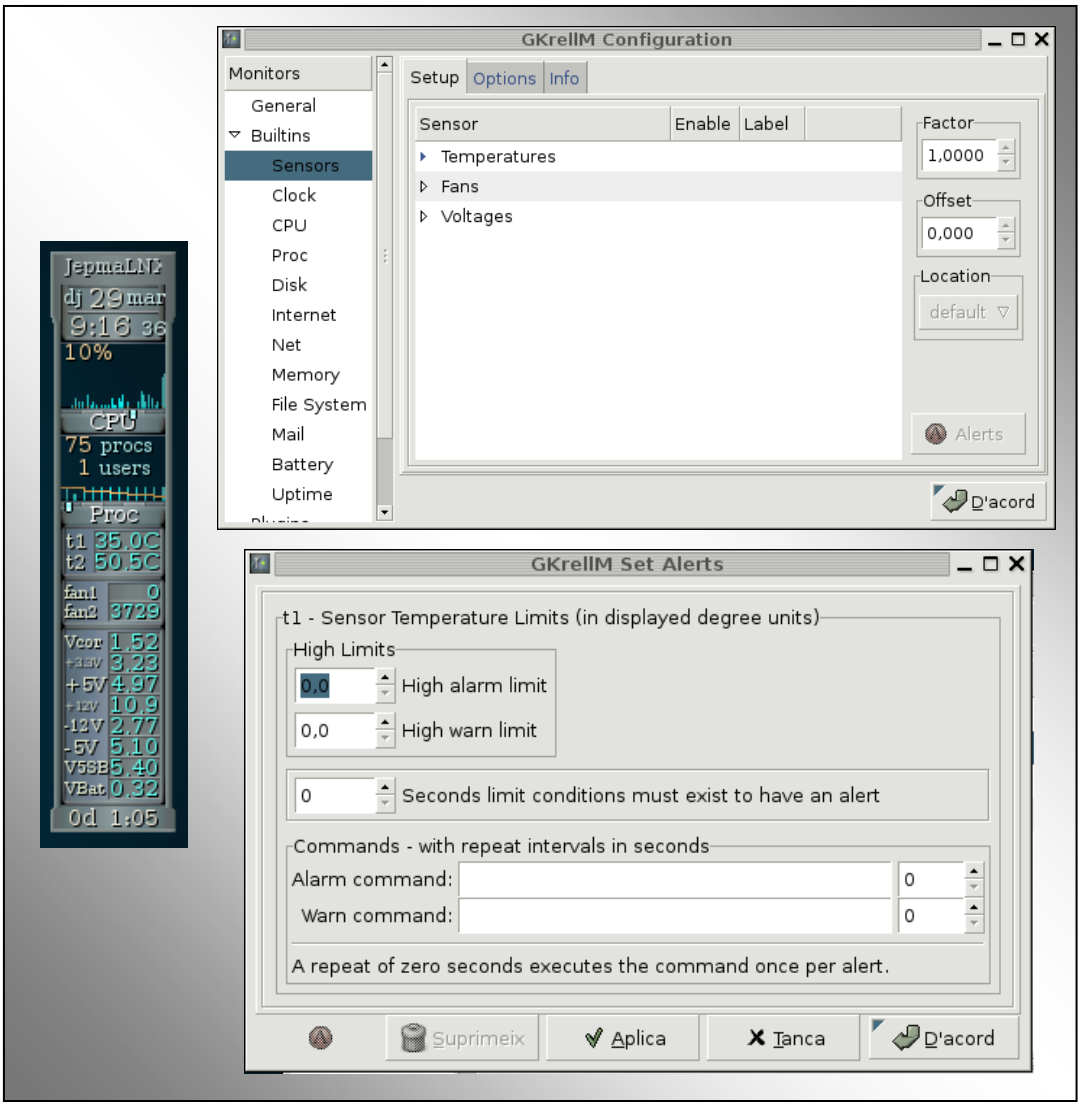


Fig. A.5 Diferents captures del gkrellm

Arribats aquí constatem el bon funcionament dels mòduls carregats, i dels programes emprats, per a utilitzar elements que utilitzen el protocol I^2C, i per extensió el SMBus.

B. Programes del projecte

B.1 Programa per inicialitzar la placa VDEC1

```
#include <xbasic_types.h>
```

```
#include <xparameters.h>
#include <xi2c_l.h>
#include <uart.h>


#define GPO_REG_OFFSET 0x124
#define DECODER_ADDR 0x20 //Read: 0x41, Write: 0x40
#define SEND_CNT 3
#define GPO_RESETS_OFF 1
#define GPO_RESET_IIC 3
#define GPO_RESET_DECODER 0

struct VideoModule {
  Xuint8 addr;
  Xuint8 config_val;
  Xuint8 actual_val;
};

#define DECODER_SVID_CONFIG_CNT 17
struct VideoModule decoder_svid[] = {
  { 0x00, 0x06, 0 },
  { 0x15, 0x00, 0 },
  { 0x27, 0x58, 0 },
  { 0x3a, 0x12, 0 },
  { 0x50, 0x04, 0 },
  { 0x0e, 0x80, 0 },
  { 0x50, 0x20, 0 },
  { 0x52, 0x18, 0 },
  { 0x58, 0xed, 0 },
  { 0x77, 0xc5, 0 },
  { 0x7c, 0x93, 0 },
  { 0x7d, 0x00, 0 },
  { 0xd0, 0x48, 0 },
  { 0xd5, 0xa0, 0 },
  { 0xd7, 0xea, 0 },
  { 0xe4, 0x3e, 0 },
  { 0xea, 0x0f, 0 },
  { 0x0e, 0x00, 0 } };


int main() {
  Xuint8 send_data[SEND_CNT] = {0};
  Xuint16 send_cnt, i;
  Xuint8 success = 1;
  send_cnt = 2;

  uartInit(XPAR_RS232_UART_1_BASEADDR);

print("################################################################
##\r\n");
```

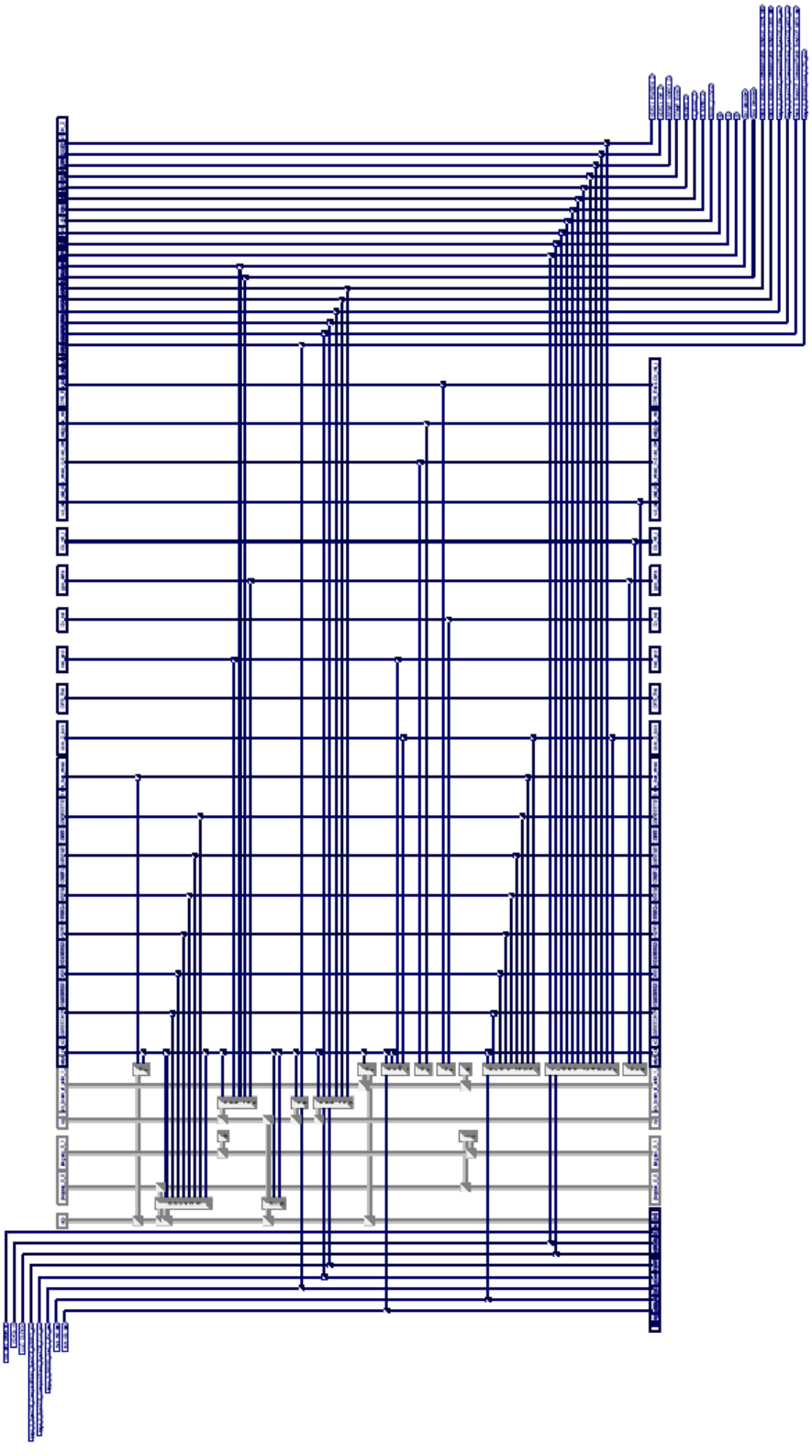
```
  print("#### Programa basic d'inicialitzacio de la placa VDEC 1 ####\r\n");

print("############################################################
##\r\n\n");
  print("Configurant el Decodificador... \r\n");
  for( i = 0; i < config_cnt; i++ )
   {

    XI2c_mWriteReg(XPAR_I2C_BASEADDR,              GPO_REG_OFFSET,
GPO_RESET_IIC);
    XI2c_mWriteReg(XPAR_I2C_BASEADDR,              GPO_REG_OFFSET,
GPO_RESETS_OFF);
    send_data[0] = decoder[i].addr;

    send_data[1] = decoder[i].config_val;

    send_cnt    =    XI2c_Send(XPAR_I2C_BASEADDR,    DECODER_ADDR,
send_data, 2);

    if( send_cnt != 2 )
        {
              xil_printf("Error escrivint a la direcció de memòria: %02x\r\n",
decoder[i].addr);
     success = 0;
     break;
        }
   }

  if( success )
        print("S'ha configurat correctament!\r\n");

print("############################################################
##");
} // end main()
```

B.2 Programa per utilitzar el “i2c-0” amb el SO encastat

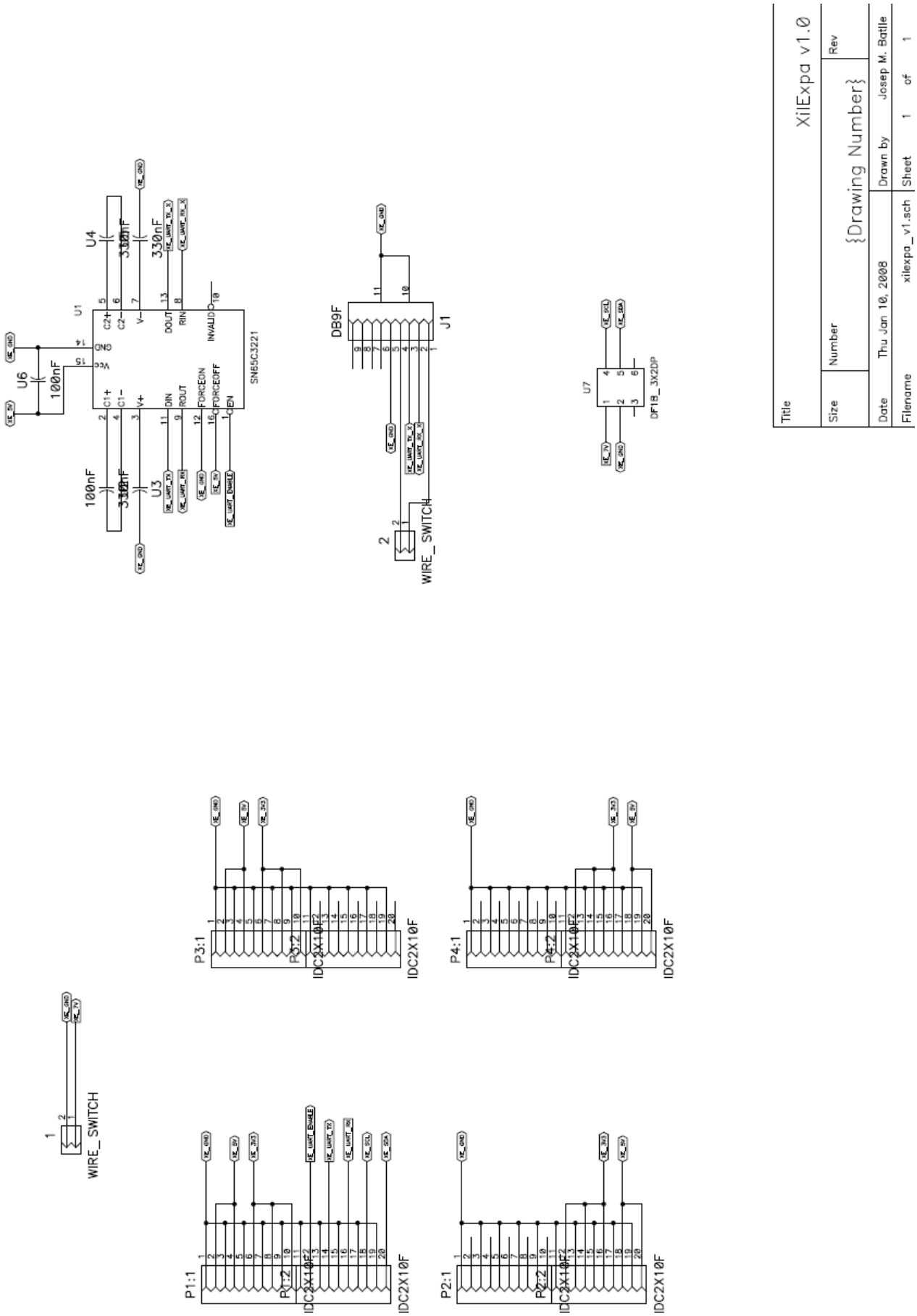
```
#include "i2c-dev.h"
#include "stdio.h"
#include "fcntl.h"

int main() {
  int file;
  int adapter_nr = 0; /* probably dynamically determined */
  char filename[20];

  adapter_nr = 0;
  sprintf(filename,"/dev/i2c-%d",adapter_nr);
  if ((file = open(filename,O_RDWR)) < 0) {
    /* ERROR HANDLING; you can check errno to see what went wrong */
    perror(filename);
```

```
  }

  adapter_nr++;
  sprintf(filename,"/dev/i2c-%d",adapter_nr);
  if ((file = open(filename,O_RDWR)) < 0) {
    /* ERROR HANDLING; you can check errno to see what went wrong */
    perror(filename);
    exit(1);
  }
}
```

C. Diagrama de blocs del mòdul d'adquisició



D. Esquemàtic del XilExpa v1.0



E. Sortida de consola del sistema GNU/Linux encastat

```
loaded at:     00400000 004A919C
board data at: 004A7120 004A719C
relocated to:  00404058 004040D4
zimage at:     00404E49 004A635E
avail ram:     004AA000 10000000

Linux/PPC load: console=ttyUL0 root=/dev/xsa3 rw
Uncompressing Linux...done.
Now booting the kernel
[      0.000000] Linux version 2.6.23-rc9 (jepmalnx@IcarusLNX) (gcc
version 3.4.4) #1 Mon Dec 24 17:38:53 CET 2007
[    0.000000] Xilinx ML300 Reference System (Virtex-II Pro)
[    0.000000] Zone PFN ranges:
[    0.000000]   DMA             0 ->    65536
[    0.000000]   Normal      65536 ->    65536
[    0.000000] Movable zone start PFN for each node
[    0.000000] early_node_map[1] active PFN ranges
[    0.000000]     0:        0 ->    65536
[    0.000000] Built 1 zonelists in Zone order.  Total pages: 65024
[    0.000000] Kernel command line: console=ttyUL0 root=/dev/xsa3 rw
[    0.000000] Xilinx INTC #0 at 0x41200000 mapped to 0xFDFFF000
[    0.000000] PID hash table entries: 1024 (order: 10, 4096 bytes)
[    0.000218] Console: colour dummy device 80x25
[      0.002686] Dentry cache hash table entries: 32768 (order: 5,
131072 bytes)
[     0.006609] Inode-cache hash table entries: 16384 (order: 4, 65536
bytes)
[      0.075366] Memory: 258432k available (1004k kernel code, 360k
data, 80k init, 0k highmem)
[    0.168530] Mount-cache hash table entries: 512
[    0.201215] io scheduler noop registered
[    0.201268] io scheduler anticipatory registered (default)
[    0.201294] io scheduler deadline registered
[    0.201454] io scheduler cfq registered
[     0.262301] uartlite.0: ttyUL0 at MMIO 0x40600003 (irq = 1) is a
uartlite
[    0.263393] console [ttyUL0] enabled
[    1.745544] RAMDISK driver initialized: 16 RAM disks of 65536K size
1024 blocksize
[    1.837854] xsysace xsysace.0: Xilinx SystemACE revision 1.0.12
[    1.910073] xsysace xsysace.0: capacity: 4061232 sectors
[    1.974062]  xsa: xsa1 xsa2 xsa3
[    2.020399] mice: PS/2 mouse device common for all mice
[    2.083150] i2c /dev entries driver
[     2.171983] EXT2-fs warning: mounting unchecked fs, running e2fsck
is recommended
[    2.262079] VFS: Mounted root (ext2 filesystem).
[    2.317893] Freeing unused kernel memory: 80k init
init started: BusyBox v1.7.2 (2007-10-25 19:43:08 CEST)
starting pid 141, tty '': '/etc/init.d/rcS'

Welcome to MontaVista PPC Embedded Linux 2.4.26 on the XUPV2P
Development Board!

Starting system...
Mounting /proc...Done!
Mounting / read-write...Done.
```

```
Mounting /dev/shm...Done!
Starting syslogd...Done!
Starting klogd...Done!
Setting the hostname...Done!
Brining up loopback interface...ifconfig: socket: Function not
implemented
Done!
Starting udhcpc...Done!
Setting static IP address...ifconfig: socket: Function not implemented
route: socket: Function not implemented
Done!
Starting inetd...Done!
Setting the system and hardware clocks...rdate: bad address
'time.mit.edu'
Done!
System started!

Please press Enter to activate this console.
starting pid 183, tty '/dev/ttyUL0': '/bin/sh'
# ls -la /dev
drwxr-xr-x    4 root     root         4096 Jan  1 00:03 .
drwxr-xr-x   17 root     root         4096 Jan 10  2008 ..
crw-r--r--    1 root     root       5,   1 Jan  1 00:00 console
crw-r--r--    1 root     root      89,   0 Jan  1 00:03 i2c-0
crw-r--r--    1 root     root       1,   3 Jan 10  2008 null
drwxr-xr-x    2 root     root         4096 Oct 25  2007 pts
drwxrwxrwt    2 root     root           40 Jan  1 00:00 shm
crw-r--r--    1 root     root     204, 187 Jan  1 00:00 ttyUL0
# cd /home
# ./i2c_main
/dev/i2c-0: No such device
/dev/i2c-1: No such file or directory
# halt
starting pid 188, tty '': '/sbin/swapoff'
starting pid 190, tty '': '/bin/umount'
The system is going down NOW!
Sending SIGTERM to all processes
Requesting system halt
[   40.896441] System halted.
[   40.928910] System Halted
```