View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Diposit Digital de Documents de la UAB

Desenvolupament del software de control per al projecte Gravitational Wave Low Frequency Technologies test-bed de l'IEEC/ICE

Víctor Segura Tirado

Resum — Les ones gravitatòries, enteses com ondulacions de l'espai-temps produïdes per el moviment de cossos massius accelerats, van ser predites per Albert Einstein al 1916, conseqüència de la seva teoria de la relativitat general. Degut que la interacció d'aquest tipus d'ones gravitacionals és de baixa intensitat, resulten molt difícils de detectar. El projecte Gravitational Wave Low Frequency Technologies test-bed (GRLOW), emmarcat dins dels projectes científics de l'Institut d'Estudis Espacials de Catalunya i l'Institut de Ciències de l'Espai (IEEC/ICE), pretén investigar tecnològicament dins el camp dels anomenats detectors de segona generació d'ones gravitacionals. En aquest projecte d'enginyeria del software, és realitza un anàlisi, disseny i prototipatge del software de control necessari per al projecte GRLOW per a poder realitzar l'experimentació desitjada de laboratori.

Paraules clau — Enginyeria del Software, Programes de control, Configuració d'entorns experimentals, Gestió d'informació

Abstract — Gravitational waves are ripples of space-time produced by movement of massive bodies accelerated, were predicted by Albert Einstein in 1916, as result of his theory of general relativity. Because the interaction of these type of gravitational waves is of low intensity, they are difficult to detect. Gravitational Wave Low Frequency Technologies test-bed (GRLOW) project, wich belongs to scientific projects of Institut d'Estudis Espacials de Catalunya and Institut de Ciències de l'Espai (IEEC/ICE), investigates technologically the second generation detectors of gravitational waves. This software engineering project is about how to perform analysis, design and prototype software control for GRLOW project to perform experimentation of testing laboratory.

Index terms — Software Engineering, Control programs, Setting experimental environments, Information management

1 INTRODUCCIÓ

La predicció d'ones gravitacionals realitzada per Albert Einsetein en la seva teoria de la relativitat general al 1916, suggereix que la gravetat és el producte de la curvatura de l'espai-temps, sent modificada per determinats objectes que produeixen alhora ones gravitacionals. Aquest tipus d'ones, transporten informació sobre els objectes que les produeixen i la naturalesa de la gravetat, sent una de les principals proves de la inflació còsmica, argumentant la teoria del Big Bang alhora d'entendre l'origen del cosmos. A més, aquesta teoria d'Einstein té implicacions astrofísiques importants, com l'existència dels forats negres com a estat final dels estels massius.

Al 1973, és va obtenir la primera evidencia indirecta[1] que donava suport a la predicció d'Einstein sobre les ones gravitacionals, confirmant la seva existència. La detecció directa d'aquest tipus d'ones és complexa, degut als requeriments tecnològics que això suposa, així com la dificultat de detectar-les, ja que la seva interacció és d'una intensitat molt baixa. No obstant, el camp de la detecció d'ones gravitatòries presenta un gran interès científic, ja que el repte de detectar de forma directe l'existència de les ones gravitatòries, no només confirmaria la predicció d'ones gravitacionals de la teoria de la relativitat general, sinó que tindria un impacte directe en la comunitat científica obrint una nova porta al coneixement.

1.1 Projecte associat *GRLOW* de l'*IEEC/ICE*

L'Institut de Ciències de l'Espai (*ICE*), situat al campus de la Universitat Autònoma de Barcelona, té l'objectiu de contribuir a l'avanç i al progrés en els estudis sobre el cosmos, ajudant a millorar la capacitat científica i tecnològica del Consell Superior d'Investigacions Científiques (*CSIC*), sent un dels membres del consorci de l'Institut d'Estudis Espacials de Catalunya (*IEEC*). Un dels projectes científics de l'*IEEC/ICE*, és l'anomena't Gravitational Wave Low frequency Technologies test-bed (*GRLOW*).

En aquest projecte, és realitzen investigacions tecnològiques en el camp dels anomenats detectors de segona generació d'ones gravitacionals. A més, aspira a ser un laboratori en línia amb les investigacions sobre la detecció d'ones gravitacionals del satèl·lit *LISA Pathfinder*, llançat el 3 de desembre del 2015, i en orbita fins a l'actualitat. Aquest satèl·lit de la Agencia Espacial Europea, s'encarregarà de provar la tecnologia dels detectors de segona generació per tal d'obtenir dades empíriques de l'experiment en un entorn real.

- *Email de contacte: victor.seguratir@e-campus.uab.cat*
- *Menció realitzada: Enginyeria del Software*
- *Tutor: Lluís Gesa Boté, tutor extern: Miquel Nofrarias Serra*
- *Curs: 2015-2016*

Els progressos en el projecte *GRLOW*, s'emmarcaran en la millora de part dels setups de laboratori que es disposa en terra. Al laboratori de l'*ICE*, disposen d'un interferòmetre amb precisió picometre, en un entorn d'alta estabilitat ambiental a baixes freqüències dins del rang 100 μHz < f < 0.1 Hz, que permet realitzar les fases d'experimentació associades al projecte, contribuint a la millora dels detectors de segona generació. A la següent figura, podem veure la tecnologia desplegada al laboratori:

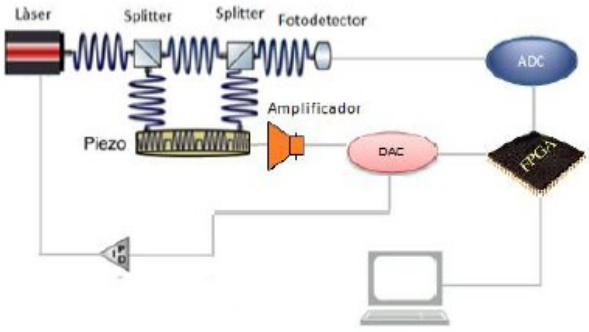


Figura 1. *Representació de l'esquema del projecte GRLOW referent a l'experiment reproduït al laboratori de l'ICE*

El seu funcionament aplica la tècnica d'interferometria; obtenint una variació en la fase d'ona del làser capturada al fotodetector. Primerament, el làser emet un feix de llum que arriba a un *splitter*, on aquest instrument òptic divideix el feix de llum en dos. Un dels feixos de llum continua el seu recorregut sobre l'actuador *Piezo*, material al qual aplicant-hi un voltatge, varien les seves dimensions físiques. Seguidament, els dos feixos de llum es tornen a agrupar en un nou *splitter*, on és produeix el fenomen d'interferència. Al final d'aquest document en la secció *Apèndix 1*, es poden observar imatges del *setup* laboratori descrit anteriorment.

Aquest experiment tecnico-científic, necessita de components i dispositius electrònics que permetin fer tot el processament de dades amb l'objectiu realitzar un post anàlisi. Per això, després que l'ona de llum passa pel fotodetector, arriba al *Analog to Digital Converter* (*ADC*), transformant la senyal analògica a digital, per ser processada a la *Field Programmable Gate Array (FPGA*), on es comunica amb un ordinador per via *TCP*[2]. A la secció *Apèndix 2*, apareix una imatge de la *FPGA* utilitzada al laboratori. La següent figura mostra la arquitectura interna d'aquest component.

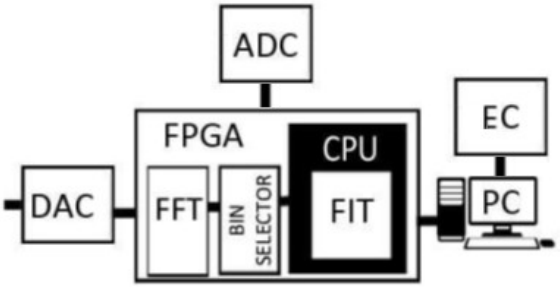


Figura 2. *Arquitectura interna de la FPGA del laboratori*

Dins de la *FPGA*, les dades es processen a través de diversos components. EL *FFT*[3], mòdul programant amb llenguatge *VHDL*[4], realitza l'extracció freqüencial de la senyal del fotodíode. A continuació, s'envien les dades al *Bin Selector* que s'encarrega de discriminar les freqüències escollides de la senyal rebuda. Seguidament, la informació es processa a la *CPU*, on un mòdul d'ajustament paramètric (*FIT* en anglès), desmodula la senyal utilitzant l'algoritme *Levenberg-Marquardt*. Aquest algoritme, utilitzat per resoldre problemes no lineals amb mínims quadràtics, permet extraure dades com la diferencia de fase, on tenint en compte la longitud de l'ona del laser, permet extraure la distancia real que un feix de llum ha recorregut respecte l'altre. Finalment, tota la informació processada es enviada al *PC* on s'hi executa l'Experiment Controller (*EC*), un programa que forma part del software de control que permet analitzar les dades rebudes.

1.2 Objectius

L'objectiu principal del projecte, és permetre el control via software de l'experiment que es realitza en el projecte *GRLOW*, mitjançant un software de control. Aplicant un conjunt d'activitats pròpies de l'àmbit de l'Enginyeria del Software, el software de control permet:

- Configurar l'entorn de l'experiment
- Gestionar i emmagatzemar les dades generades
- Realitzar l'experiment a través del software

1.3 Estructura de l'article

En les següents seccions de l'article, és detallarà primerament a la secció 2, l'estat de l'art del projecte. En la secció 3, s'especificarà la metodologia de treball executada en el transcurs del projecte. A la secció 4, és detallarà la fase d'anàlisi del sistema. Respecte la secció 5, s'aprofundirà en el disseny del programa de control. La secció 6 permetrà obtenir la primera versió del software de control amb la fase de prototipatge. Finalment la secció 7, finalitzarà amb unes conclusions del projecte.

2 ESTAT DE L'ART

Actualment en el laboratori de l'*ICE* on trobem el projecte *GRLOW*, son múltiples les eines de control encarregades de processar i automatitzar fases experimentals en l'execució d'experiments amb caràcter investigador. El projecte *GRLOW* de l'*IEEC/ICE*, conté una infraestructura bàsica en software, desplegada per poder realitzar operacions envers a l'experiment. No obstant, és necessari aplicar un enfocament sistemàtic, disciplinat i quantificable, per poder garantir que el projecte assoleixi tecnològicament, totes les fites envers als reptes científics plantejats associats als progressos de les investigacions.

L'objectiu de l'enfocament aplicat, anomenat Enginyeria del Software, és garantir l'èxit del projecte, generant un software de qualitat. Així doncs, es pretén cobrir la necessitat de generar una infraestructura que permeti el control de l'experiment, cobrint en aquest

projecte, les necessitats d'anàlisi, disseny i prototipatge per al projecte *GRLOW*.

3 METODOLOGIA DE TREBALL

Dins de l'àmbit de l'Enginyeria del Software, existeixen diferents models de desenvolupament de software. En aquest cas, la metodologia ha de ser adaptable a canvis al llarg del seu desenvolupament, ja que encara que els requeriments del projecte seran definits des de l'inici, és possible que els progressos en les línies d'investigació del projecte associat *GRLOW*, tinguin una implicació directe en el software de control, alterant d'aquesta manera possibles requeriments o funcionalitats del mateix.

Per tant, la metodologia de treball ha de contemplar escenaris adaptables a canvis tant en els requeriments, com en el disseny del software, aplicant una metodologia àgil, amb un model de desenvolupament evolutiu. Tanmateix, donada la necessitat de gestionar els diferents documents del projecte, és necessari definir una estructura jeràrquica de carpetes i documents per poder gestionar tota la informació, ja que es crearan un seguit de documents, de diferents tipologies, a més de codis i arxius relacionats amb el software.

Una part essencial en qualsevol projecte d'enginyeria, és la gestió de la documentació generada, fruit del projecte desenvolupat. Per això, s'ha establert un control d'identificació de documents, definint una nomenclatura, com a eina de gestió de versions. La nomenclatura que s'ha utilitzat, ha estat la següent: GRLOW - SW - Tipologia del document - Versió associada. La taula que hi ha a continuació, discrimina les diferents tipologies:

Tipologia	Nom per nomenclatura	Contingut
Reports	RPT	Informe per proveir informació
Design Document	DD	Disseny modular de l'arquitectura del software
Requirement Document	RD	Descripció del comportament del sistema
Formal Technical Review	FTR	Revisions i inspeccions de validació del software
Tecnical notes	TN	Notes de caràcter tècnic associades al projecte

Figura 3 *Taula per diferenciar els tipus de documents generats al llarg del desenvolupament del projecte*

Tots aquest documents, han estat emmagatzemats al repositori *BitBucket*[5], espai compartit amb el tutor del projecte Lluís Gesa Boté, per tal d'establir un canal de comunicació on guardar tots els documents mitjançant el sistema de control de revisions *Git*[6]. El programa *GitEye*[7], ha permès pujar al repositori tots els fitxers associats al projecte.

3.1 Model evolutiu en espiral

Aquest model de cicle de vida del software, utilitzat en el desenvolupament del projecte, és utilitzat generalment en el camp de l'Enginyeria del Software. Les activitats es desenvolupen repetides vegades dins d'un bucle, on cada iteració representa un conjunt d'activitats. Per aquest projecte, el model evolutiu en espiral, està format per tres activitats: anàlisi, disseny i prototipatge.

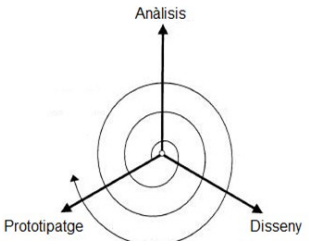


Figura 4. *Representació de la metodologia àgil de treball, basat en el model evolutiu en espiral*

Cada activitat realitza a la vegada un conjunt de tasques associades, incorporant millores i requeriments al projecte sense trencar amb la metodologia establerta. Cal destacar, que la metodologia de treball representada a la figura anterior, no te un cicle de vida seqüencial. Aquest fet, implica que el final d'una activitat, pot realitzar-se en paral·lel amb el principi de la següent activitat, obtenint un cicle de vida dinàmic.

3.2 Planificació

La planificació del projecte, ha estat realitzada envers a les activitats que s'han dut a terme en la iteració del model evolutiu en espiral. En aquest apartat s'especifica el calendari del projecte. Als següents diagrames de Gantt, a tall d'exemple, podem observar la planificació que s'ha anat realitzant al llarg del projecte, representades de manera temporal, amb una estimació sobre la durada de cadascuna d'elles.

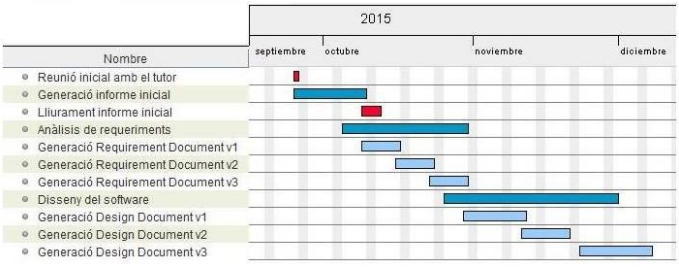


Figura 5. *Diagrama de Gantt on es presenta la planificació referent a la primera meitat de desenvolupament del projecte*

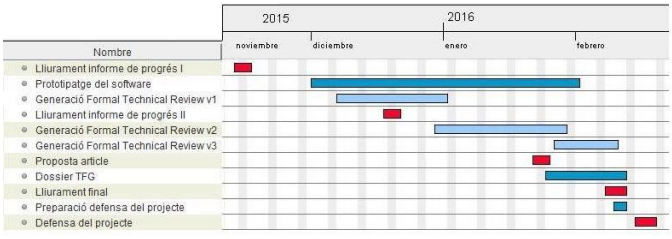


Figura 6. *Diagrama de Gantt on es presenta la planificació referent a la segona meitat de desenvolupament del projecte.*

Inicialment, es va realitzar un estudi de la infraestructura existent de l'experiment, que ha permès conèixer amb profunditat la infraestructura del projecte, avaluant les necessitats a nivell software del programa de control y generant el corresponent *Requeriment Document*. Al mateix temps, també hi ha hagut un procés formació alhora de configurar i utilitzar l'entorn *RTEMS*, sistema operatiu executat a la *CPU* integrada de la *FPGA*. Degut que aquest entorn s'executa a temps real, ha sigut necessari aprofundir i ampliar coneixements sobre gestió de tasques, execucions multi-threads, així com gestió de temps d'execució. El llenguatge de programació dels fitxers amb l'entorn *RTEMS*, ha sigut en llenguatge *C/C++*. Amb l'objectiu d'analitzar les implementacions del software actual, l'*IEEC/ICE* va proporciona el projecte amb l'aplicació que s'executa al laboratori.

Seguidament, es va realitzar la segona fase de disseny, amb l'objectiu de realitzar el disseny de l'arquitectura tan física com lògica, del software de control del projecte *GRLOW*. Per això, es va generar el document tècnic *Desing Document*, on es mostra la implementació dels requeriments especificats al document *Requeriment Document*, modelant el sistema amb diagrames, per definir l'arquitectura del projecte. El modelatge dels diagrames *UML*[8], s'ha dut a terme a través de l'eina online *Cacoo*[9]. De manera paral·lela, l'*ICEE/ICE* va facilitar el projecte per executar la infraestructura del *EC* al *PC*, on va començar la fase de prototipatge del software de control. Ha sigut necessari aprendre el funcionament del protocol de comunicacions *Packet Utilisation Standard (PUS)[10]*, per entendre l'arquitectura de comunicacions del sistema, a més de adquirir coneixements sobre la llibreria gràfica *wxPython*[11]. El disseny i edició d'imatges, així com el logotip del software de control, s'ha realitzat amb el programa *Adobe Photoshop*[12]. Durant el transcurs del projecte, ha sigut necessari tornar a fer planificacions del calendari de treball, ja que la planificació inicial no ha contemplat els escenaris d'aprenentatge sobre les tecnologies utilitzades alhora de codificar el software de control, fet que ha suposat no assolir algunes fites de treball, com la creació del manual d'usuari del programa de control.

4 ANÀLISI DEL SISTEMA

Realitzar l'anàlisi del sistema, ha permès avaluar les necessitats tecnològiques del projecte envers al software de control desenvolupat. Concretar i definir amb precisió, els requeriments del projecte, a contribuït a realitzar des de l'anàlisi de les necessitats d'interfície d'usuari, fins a l'avaluació de tot el conjunt de dades i volum d'informació que es gestiona des de el software. Tots els requeriments del projecte estan llistats, identificats, descrits i especificats en el *Requirement Document*, oferint una descripció del comportament del sistema que s'ha prototipat.

4.1 Descripció del sistema

El sistema necessita d'una infraestructura que permeti controlar i gestionar el volum de dades generades, així com el seu processament a través dels components i dispositius electrònics. Els dos elements més importants del sistema son la *FPGA* i l'EC. La *FPGA* és un dispositiu semiconductor, el qual conté blocs de lògica programable, que s'encarrega de realitzar tot processament de dades. A la *FPGA* del laboratori, hi ha sintetitzada una *CPU Leon3*[13], generant d'aquesta manera l'anomena't S*ystem on chip* (*SoC*)[14], en el qual podem executar el sistema operatiu *RTEMS*[15]. La *FPGA* utilitzada, desenvolupada per l'empresa *Xilinx*, correspon al model *Virtex-7*. Cal destacar que les implementacions realitzades sobre la *FPGA*, han estat aquelles relacionades amb la configuració de l'entorn de l'experiment a nivell d'*RTEMS*. L'*EC*, és un controlador que disposa d'una interfície gràfica, amb la que es pot interactuar amb la resta d'elements del sistema. A través del *PC* de característiques convencionals que hi ha al laboratori, i que s'executa amb el sistema operatiu *Linux*, l'usuari pot interactuar amb el sistema.

4.2 Requisits funcionals

Els requeriments funcionals del sistema, defineixen les funcions del software de control o un component del mateix, descrits com un conjunt d'entrades i sortides, amb un comportament determinat. A continuació es detallaran els principals requisits funcionals dels dos elements més importants del sistema.

4.2.1 Requisits funcionals *FPGA*

4.2.1.1 Executar el sistema amb dos modalitats

Una modalitat anomenada experimental serà utilitzada com la modalitat per defecte, amb la que es pretén controlar habitualment l'experiment. L'altre modalitat anomenada debug, permetrà obtenir informació més detallada sobre l'experiment. Aquesta última modalitat necessita processar més informació que la modalitat experimental, per tant serà utilitzada en el cas de realitzar un anàlisi més detallat del sistema.

4.2.1.2 Controlar paràmetres

Els paràmetres de freqüència i voltatge del *Piezo*, juntament amb la intensitat i temperatura del làser, son paràmetres de control per configurar la modulació.

4.2.1.3 Escollir algoritme FIT

Executar el sistema seleccionant un algoritme de desmodulació, configurant cada algoritme i modificant els paràmetres inicials abans de realitzar l'execució del sistema.

4.2.1.4 Monitoritzar dades

Les dades a monitoritzar per pantalla, per tal d'obtenir informació sobre d'anàlisi de l'experiment, son la pressió de la cambra de buit i els sensors de temperatura.

4.2.1.5 Configurar l'experiment

Configurar els bins del *Bin Selector* que farà servir la desmodulació, samples a executar al *FFT*, la freqüència de mostreig dels sensors i del fotodíode.

4.2.2 Requisits funcionals *EC*

4.2.2.1 Mostrar informació

Mostrar el resultat dels càlculs numèrics de les dades associades a l'execució del algoritme *FIT*, la monitorització dels sensors de temperatura, el gràfic de la senyal del fotodíode i els bins resultants del *Bin Selector*.

4.2.2.2 Configurar elements

Els elements del sistema descrits als requisits funcionals de la *FPGA*, i els modes de funcionament han de ser configurables.

4.2.2.3 Emmagatzematge dades

Guardar les dades del sistema un cop l'usuari finalitzi la sessió del software de control, en un format i ordre planificat.

4.3 Requisits no funcionals

Tots aquells requeriments del software de control que atribueixen criteris que no impliquen funcionalitats directes del software, sent utilitzats per avaluar el sistema, formen part dels requisits no funcionals.

4.3.1 Requisits no funcionals *FPGA*

4.3.1.1 Comunicacions amb *PC*

Implementades utilitzant el protocol de comunicacions *PUS* sobre ethernet *TCP/IP*.

4.3.1.2 Software encastat

Ha de funcionar en el *SoC* sintetitzat, programat en llenguatge *VHDL*.

4.3.1.3 Sistema operatiu *RTEMS*

El sistema operatiu que s'executarà de forma encastada en el *SoC* serà l'anomena't *RTEMS*.

4.3.1.4 Llenguatge codificació algoritmes *FIT*

Es realitzarà en llenguatge de programació *C/C++*.

4.3.2 Requisits no funcionals *EC*

4.3.2.1 Manual d'usuari

Manual detallant com executar el software de control, aplicant diverses configuracions, i servint de guia a l'usuari alhora d'utilitzar qualsevol funcionalitat del sistema.

4.3.2.2 Sistema operatiu *UNIX*

El sistema operatiu que s'utilitza a l'ordinador del laboratori és un sistema operatiu *UNIX*.

4.3.2.3 Rendiment

Executar el sistema de manera que permeti una navegació fluida en el software de control.

4.3.2.4 Usabilitat

El software haurà de permetre que la interacció entre el programa i l'usuari sigui efectiva i eficient.

4.3.2.5 Escalabilitat

El sistema haurà de reaccionar i adaptar-se a canvis sense perdre qualitat.

4.3.2.6 Estabilitat

Executar el sistema sense que es produeixi cap error o fallada en el procés d'execució.

5 DISSENY DEL PROGRAMA DE CONTROL

El software de control ha estat dissenyat per adaptar-se a les necessitats que presenta l'experiment, de manera que permet la configuració i usabilitat del mateix. Tanmateix, està orientat a l'usuari perquè el programa es comporti exactament com aquest espera. Totes les dades emmagatzemades segueixen un conjunt de principis d' usabilitat, fiabilitat, rendiment, escalabilitat i estabilitat.

5.1 Principis i patrons

En el procés de dissenyar l'arquitectura lògica d'un software, no existeix una metodologia definida per obtenir el millor disseny orientat a objectes. No obstant, existeixen un conjunt de principis anomenats *GRASP*[16], format per patrons i heurístiques, que ens permeten assignar responsabilitats a classes i objectes. Donada la necessitat del disseny del software de control, s'han seguit els següents principis:

1) Information Expert: s'encarrega d'assignar un conjunt de responsabilitats als objectes, de manera que cada classe tingui exactament la informació necessària per dur a terme les funcionalitats de la mateixa. És important que les classes assumeixin la mínima dependència entre elles.

2) High Cohesion: per tal de mantenir les classes enfocades i comprensibles als programadors, la assignació de responsabilitats s'ha de fer amb una cohesió que es mantingui alta. S'han d'evitar classe molt extenses, que realitzin moltes funcionalitats.

3) Low Coupling: l'acoblament ens indica el nivell de dependència entre les classes. Un acoblament relativament alt entre classes, implica augmentar el número de modificacions necessàries per realitzar un canvi en el software, incrementant alhora la dificultat per reutilitzar el codi. Aquest principi es basa en el fet d'evitar que les classes tinguin excessius accessos a altres classes.

El disseny del software minimitza l'accessibilitat de classes i membres, utilitzant mètodes d'abstracció, encapsulant les classes i ocultant la seva informació accedint-hi tan sols amb mètodes definits *getters*[17] o *setters*[18]. *A* més, permet crear l'estructura del sistema, aplicant una descomposició en subsistemes, per tal d'identificar l'arquitectura, determinant les relacions entre les classes, i descrivint les funcionalitats de la mateixa. Aquest procés d'anàlisi, permet orientar el disseny per que aquest sigui modulable i escalable. El software facilitat per l'*IEEC/ICE*, segueix un estàndard de codificació basat en l'estàndard de l'Agencia Espacial Europea anomenat *C and C++ Coding Standards (BSSC(2000)1)*, ja que és una herència del codi del projecte *LISA Pathfinder*. Per tant, l'estil de codificació del software de control està basat en la codificació esmentada anteriorment.

5.2 Arquitectura física

Dins de l'arquitectura física del sistema, existeixen molts elements que formen part de l'experiment del projecte *GRLOW*. Com a elements més importants que permeten el control de l'experiment del projecte *GRLOW*, destacarem la *FPGA* i el *PC*. Abans de exposar l'arquitectura física dels dos components principals, mostrarem en la següent imatge, un diagrama amb els components físics de l'entorn de l'experiment.

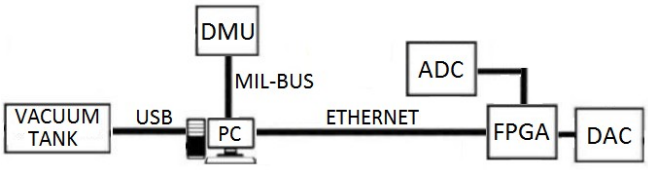


Figura 7. *Esquema de l'arquitectura física del software de control, especificant les connexions entre els elements principals*

El *PC* del laboratori s'encarrega de rebre les dades de l'entorn de l'experiment, a través del software de control *EC*. Aquest ordinador proporcionat per l'*IEEC/ICE*, té unes prestacions i característiques convencionals que permeten executar el programa de control en un sistema operatiu *LINUX*. A través de tres tipus de connexions diferents, el *PC* es comunica amb els següents elements: la Data Managment Unit (*DMU)*[19], la cambra de buit i la *FPGA*. A través d'una connexió *Mil-BUS*, l'ordinador es comunica amb el primer element, una unitat de gestió de dades, encarregada de processar i gestionar la informació, alhora que actua com un sistema de sensors d'alta precisió, format també per diversos actuadors. La cambra de buit, on hi han instal·lats un sensors de pressió que monitoritzen el buit de la cambra, es comunica amb l'ordinador mitjançant una connexió *USB*. I per últim una connexió *ethernet* permet comunicar l'ordinador amb la *FPGA*, encarregada de rebre, processar i enviar dades als diferents elements del sistema, ja sigui el *ADC*, el *DAC* o el propi ordinador.

5.3 Arquitectura lògica

Per definir la estructura i el funcionament del software, s'exposarà en primer lloc l'arquitectura lògica general, per després aprofundir en el disseny de les classes, seguint un model de descomposició en subsistemes. Tota l'arquitectura lògica que es descriu, proporciona, una visió general de l'arquitectura del sistema. Al següent diagrama *UML*, podem veure l'arquitectura bàsica dels dos components.

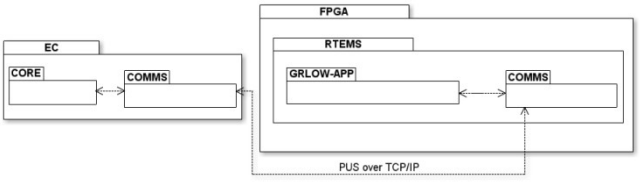


Figura 8. *Esquema de l'arquitectura física del sistema*

El mòdul de l'EC conté principalment el submòdul anomenat *Core* on resideix l'arquitectura lògica del sistema que s'encarrega d'executar el programa de control al *PC*. L'altre mòdul de la *FPGA*, està format pel submòdul *RTEMS*, capa on s'executa el sistema a temps real. Aquest últim,conté alhora el submòdul anomenat *GRLOW-APP*, on resideix l'arquitectura lògica del sistema que s'executa en la *FPGA*. Els dos mòduls mencionats anteriorment contenen cadascun un mòdul anomenat *COMMS*, que s'encarrega de gestionar les comunicacions.

5.4 Disseny de classes

5.4.1 Disseny de classes *FPGA*

A continuació detallarem el disseny de classes de la *FPGA*, que s'executa en la capa *RTEMS*. Tot el disseny que es mostra, està basat en el document de requeriments, definint una arquitectura lògica que pugui satisfer totes les funcionalitats especificades.

5.4.1.1 *GRLOWSystem*

Primerament definirem la classe principal anomenada *GRLOWSystem*. Dins d'aquesta la classe, es poden realitzar principalment tres accions: configurar el sistema amb les dades i els components necessaris per executar el software a la *FPGA*, realitzar l'acció r*un()*, que permet iniciar l'execució del sistema, o bé aplicar una configuració determinada, emmagatzemant les dades generades, a través de l'acció *setConfiguration()* . Al mateix temps, segueix un patró Singleton, assegurant que la classe només té una instància que permet ser accedida de manera global. Per mitjà de la funció definida *static Instance()*, es pot accedir a la seva instància única. Al següent diagrama *UML* apareix el diagrama d'aquest bloc:

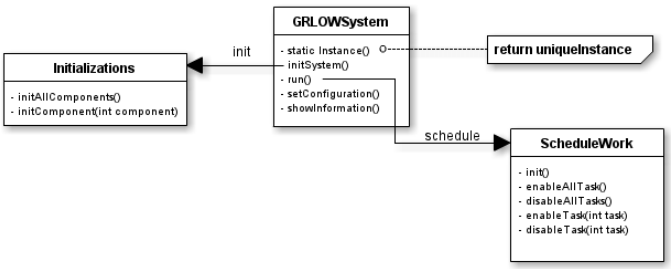


Figura 9*. Diagrama UML del bloc GRLOWSystem*

També podem observar que apareixen dues funcions *setConfiguration*() i *showInformation*(). La primera d'elles, s'encarrega de proporcionar l'opció de modificar els paràmetres de configuració del sistema. La segona funció, pretén obtenir les dades necessàries per mostrar informació addicional per pantalla, que permeti a l'usuari obtenir dades d'anàlisi del sistema.

5.4.1.2 *Initialization*

Aquesta classe conté els mètodes per inicialitzar el sistema, encarregats d'executar els mòduls *CPU, SRV* i *RTOS*. El primer mòdul s'encarrega de inicialitzar les capes d'abstracció del hardware de la *CPU*, relacionades amb els temporitzadors del sistema: un anomenat *General Purpouse Timer* (*GPT*) d'us general, i un altre *Real Time Clock (RTC)* que funciona a temps real, tots ells associats a les característiques de l'arquitectura *Leon*. El mòdul *SRV*, carrega tots els serveis disponibles que conté aquesta classe, necessaris per executar el sistema. L'últim mòdul *RTOS*, inicialitza els serveis disponibles. L'execució d'aquestes inicialitzacions, permeten preparar al sistema per executar-lo. A continuació podem veure el diagrama *UML* associat a aquesta classe:

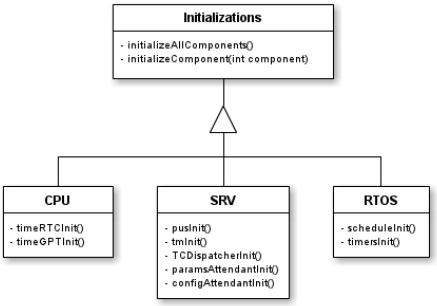


Figura 10. *Diagrama UML del bloc Initialization*

Permet organitzar els conjunt de les inicialitzacions esmentades anteriorment, en un conjunt de classe jeràrquiques, on el sistema pot inicialitzar cadascuna d'elles en funció de les necessitats del mateix. Abans de realitzar qualsevol operació a través del sistema, cal executar totes les inicialitzacions que apareixen en el diagrama esmentant anteriorment. Encara que les funcions d'inicialització seran transparents a l'usuari, es necessari que la infraestructura de la *FPGA* estigui dotada d'aquestes funcions que permeten iniciar el sistema.

5.4.1.3 *ScheduleWork*

Proporciona un conjunt de mètodes per iniciar l'execució de les tasques del sistema, que s'executen en el sistema mitjançant les llibreries *RTEMS*. Les principals característiques de les tasques son els seus temporitzadors, on cada tasca té les seves pròpies característiques per identificar quan ha expirat el seu període, i les llistes prioritàries, que permeten executar una tasca abans que una altre, en funció de la seva prioritat. No obstant, aquesta classe es centrarà en la gestió de les tasques envers al marc temporal d'execució. Al següent diagrama *UML* podem veure la representació:

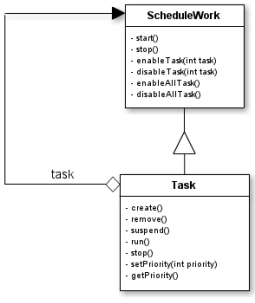


Figura 11*. Diagrama UML del bloc ScheduleWork*

Segueix un patró *Composite* per tal de crear i gestionar les diferents tasques, realitzant alhora diverses funcionalitats sobre la gestió de les mateixes. Per exemple, aplicant funcionalitats d'habilitar o deshabilitar una tasca determinada. Tota l'execució del sistema, està emmarcada en un calendari de treball gestionat per tasques.

5.4.2 Disseny de classes *EC*

Mòdul on resideix la lògica del software de control executat al *PC*. Tot el disseny del software està basat en el document de requeriments, definint una arquitectura lògica que pugui realitzar totes les funcionalitats especificades.

5.4.2.1 *ExperimentController*

En aquests bloc, podem establir el mode de funcionament de l'aplicació, mode debug o experimental, a més de inicialitzar tots els components de l'experiment, i mostrar per pantalla la informació associada. Al següent diagrama *UML*, podem observar la representació de l'arquitectura lògica:

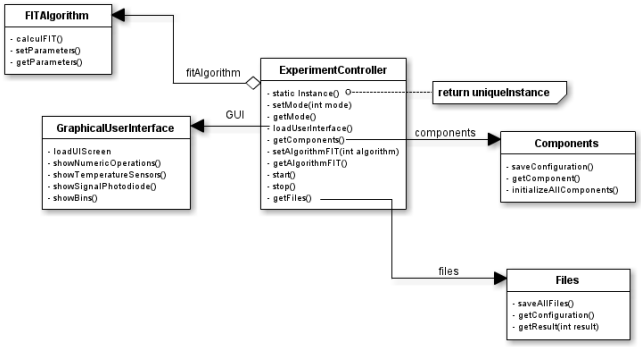


Figura 12*. Diagrama UML del bloc ExperimentController*

Conté tots els objectes que formen l'arquitectura lògica del sistema. Els objectes continguts en aquesta classe son *GraphicalUserInterface, Components, FITAlgorithm, i Files.* El primer s'encarrega d'inicialitzar la interfície gràfica del programa, a més de mostrar per pantalla informació de l'experiment. L'objecte *Components*, conté tots els components i mòduls del sistema. *FITAlgorithm p*ermet executar el sistema utilitzant un algoritme a escollir del mòdul *FIT.* L'últim objecte *Files,* permet serialitzar totes les dades en fitxers. Aquesta classe, segueix un patró *Singleton*,de manera que assegura que la classe només tingui una instància que permeti ser accedida de manera global. Per mitjà de la funció definida *static Instance()*, podem accedir a la seva instància única.

5.4.2.2 *GraphicalUserInterface*

Proporciona un conjunt de mètodes i funcionalitats per mostrar tota la informació necessària per pantalla, a més de carregar tota la interfície gràfica amb la que interactuarà l'usuari. Totes les dades que s'han de mostrar per pantalla, formen part del requisits funcionals del *EC.* Les dades que s'han de mostrar per pantalla son les següents: els resultats del càlculs numèrics, la monitorització dels sensors de temperatura, el gràfic de la senyal de fotodíode i els bins resultants del *Bin Selector*.

Per mostrar totes aquestes dades hi han definides un conjunt de funcions per visualitzar la informació per pantalla. Tanmateix, el mode de funcionament del *EC*, implicarà visualitzar un número determinat de dades, que dependrà de la modalitat executada en el software de control. Amb la modalitat experimental, es poden visualitzar totes les dades esmentades anteriorment. En el cas de executar el sistema amb la modalitat debug, a part de visualitzar totes les dades anteriors, es mostraran els resultats numèrics del mòdul *FFT*, abans de ser tractats

pel *Bin Selector*. Aquesta dualitat de modes, permet carregar el sistema amb una configuració determinada, visualitzant en cada cas la informació descrita prèviament.

La classe *GraphicalUserInterface* s'encarrega exclusivament de mostrar informació per pantalla, seguint l'arquitectura *Model-Vista-Controlador* (*MVC*), actuant aquesta classe de *Vista*. La lògica del software de control on es processen les dades, recau en les altres classes contingudes a la classe principal *ExperimentController*, sent alhora el *Controlador* del model *MVC*. Les dades del sistema del software de control corresponents al *Model* del *MVC*, les gestiona la classe *Files*, on s'explicarà posteriorment en la secció 5.4.2.5. Alhora de dissenyar l'aspecte gràfic del software de control, que carregarà la funció *loadUIScreen*, s'han tingut en compte els següents principis:

- *Simplicitat*. Facilitat de l'usuari alhora de executar el software de control. La informació i les accions més importants han de ser fàcilment accessibles i controlables per l'usuari.

- *Visualització*. S'han de definir les dimensions i els graus de contrast que permetin que l'usuari pugui observar ràpidament aquelles seccions o tasques més importants.

5.4.2.3 *FITAlgorithm*

Conté les mètodes necessaris per executar un tipus d'algoritme del mòdul *FIT*. Encara que la implementació d'aquests algoritmes no correspon al desenvolupament d'aquest *TFG*, donat que ja estan implementats, cadascun d'ells necessita d'una configuració determinada. Per tant, tots els algoritmes necessiten d'una configuració específica per realitzar la seva inicialització, determinant els seus paràmetres inicials per la seva posterior execució.

Així doncs, disposarem d'un conjunt de classes associades als algorismes, que difereixen en el seu comportament. S'ha aplicat un patró *Strategy*, per definir la família d'algorismes, i un patró *Composite*, per crear un tipus d'algorisme que s'encarregui de generar els altres. Al següent diagrama podem veure la representació de la classe:

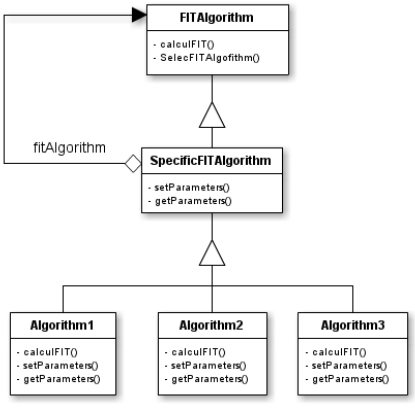


***Figura 13**. Diagrama UML del bloc FITAAlgorithm*

Es pot escollir la implementació de l'algorisme que es vulgui utilitzar, segons les necessitats d'executar un tipus determinat. Al mateix temps, també s'aconsegueix aïllar els objectes dels tipus concrets d'algorismes de la classe *FITAlgorithm* que executa el càlcul de l'algorisme seleccionat, disminuït la dependència i l'acoblament entre les classes definides.

5.4.2.4 *Components*

Conté els components de l'experiment classificats en dos tipus: actuadors i sensors. Els actuadors son aquells components que formen part de l'experiment, amb els quals es pot interactuar modificant-hi els seus paràmetres i configuracions per realitzar l'experiment d'una manera determinada. En canvi, els sensors, mesuren magnituds físiques de l'entorn de l'experiment, en aquest cas, la temperatura amb el sensor *TemperatureSensor*, i la pressió en la cambra de buit amb el sensor *PreassureVaccumTank*. Dins dels actuadors, existeixen tres tipus: *Laser*, *Photodiode* i *Piezo*, cadascun d'ells amb els seus paràmetres de configuració. Al següent diagrama UML podem observar l'arquitectura del bloc:

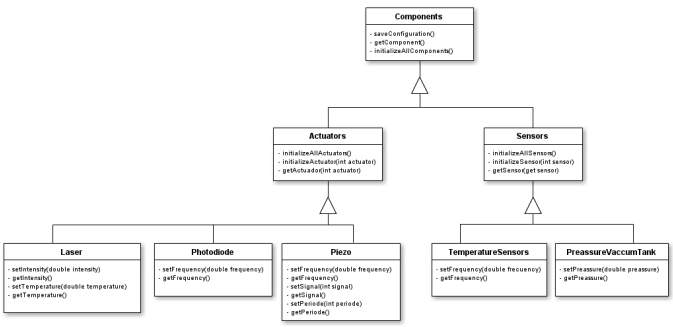


***Figura 14**. Diagrama UML del bloc Components*

Aquesta organització jeràrquica del bloc anterior, permet gestionar d'una forma ordenada, els diferents tipus de classes. Tanmateix, augmenta el grau d'escalabilitat de l'arquitectura, ja que concedeix llibertat per crear nous tipus de components.

5.4.2.5 *Files*

S'encarrega del emmagatzematge de dades del sistema. Donada la tipologia de fitxers definida en el *Requirement Document*, existeixen dos tipus de fitxers: fitxers de configuració i fitxers de resultats. Els fitxers de configuració permeten guardar les dades de configuració que s'aplicaran inicialment per dur a terme l'experiment, facilitant i garantit al mateix temps, la repetibilitat dels experiments. En canvi, els fitxers de resultats, son aquells que contenen únicament els resultats i les dades generades pel sistema. A més, cal destacar que cada fitxer de configuració pot estar associat a un o més fitxers de resultats, ja que per una certa configuració del sistema, es pot repetir l'experiment tantes vegades com l'usuari desitgi. Per tant, la representació amb diagrama d'*UML* d'aquests bloc és la següent:

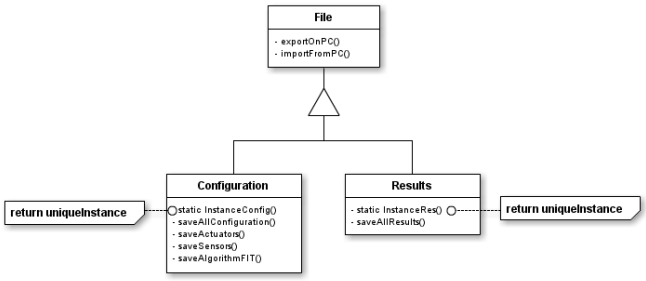


***Figura 15**. Diagrama UML del bloc Files*

Permet generar els tipus de fitxers per emmagatzemar les dades, mitjançant una jeràrquica de fitxers de tipus *Configuration* i de tipus *Results.* També es fa d'ús del patró *Singleton*, que permet assegurar que les classe tan sols tinguin una instància, definida per un identificador únic. És important que els dos tipus de fitxers tinguin associat un identificador únic, ja que permet gestionar els fitxers d'una manera més organitzada. El format dels tipus de fitxers, segueix el model definit en el *Requirement Document*, aplicant un format amb extensió *.config* per als fitxers de configuració, i un format amb extensió *.res* per als fitxers de resultats. El format intern dels dos fitxers tenen una representació binaria que facilita la seva gestió en el software de control. No obstant, en relació als fitxers de configuració, representarem a la següent figura el format en text seguint la estructura *hashmap*:

```
<User_name>_<date>_<hour>
<key_string01>:<value>
<key_string02>:<value>
<key_string03>:<value>
.
.
.
<key_stringn>:<value>
```

***Figura 16**. Estructura hasmap que representa el format intern dels fitxers de configuració*

6 PRIMER PROTOTIPATGE

La generació de la primera versió del software de control ha permès tancar la primera iteració de la metodologia de treball referent al model evolutiu en espiral. Els documents generats *Requeriment Document* i *Design Document*, han estat la base per realitzar la codificació del programa. Abans de descriure les implementacions realitzades, és important entendre el protocol de comunicacions *PUS*, que permet la transferència de dades entre les infraestructures. Per això, s'exposarà en primer lloc el funcionament del protocol de comunicacions, per després descriure la programació de les infraestructures.

6.1 Protocol de comunicacions *PUS*

L'arquitectura de comunicacions del sistema, fent ús del protocol de comunicacions *PUS*, genera un servei de transferència de dades, amb el que s'implementa en el projecte *GRLOW* de manera personalitzada, alhora de realitzar la transmissió de dades, empaquetant i desempaquetant la informació a processar. Principalment, aquest protocol de comunicacions proporciona un procés d'estandardització per la capa d'aplicació que es troba en la *ESA Packet Telecomand Standard* (*TC*) i *ESA Packet Telemetry Standar* (*TM*). L'objectiu d'aquest estàndard, és definir un model operacional, basat en *service providers*, definits com processos de l'aplicació, on cadascun proporciona una col·lecció de serveis *PUS*, i *service users*, entesos com centres on es gestionen les peticions de comunicacions. El programa executat a la *FPGA*, actua com *services providers*, actuant alhora com processos del software executats en *RTEMS*, que s'encarreguen de processar les peticions *TC* que rep. El programa *EC* executat al *PC*, actua com *service users*, controlat per l'usuari, que genera alhora els paquets *TM*. Al següent diagrama podem observar la representació de les comunicacions, juntament amb la representació del paquet de comunicacions PUS:

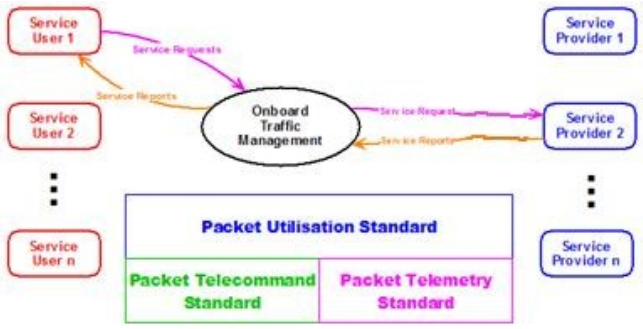


***Figura 17**. Representació de les comunicacions i el paquet de dades PUS*

Cada paquet te associat un servei i un subservei, de manera que podem definir cada paquet com:

$$PA = (S, SS) \quad \text{on} \quad \begin{array}{l} PA = \text{paquet TC o TM} \\ S = \text{servei} \\ SS = \text{subservei} \end{array}$$

Cal destacar que la implementació del protocols de comunicació *PUS*, juntament amb les definicions dels missatges de comunicació, incloent les definicions dels paquets *TC* i *TM*, ja estan implementades en el software proporcionat per l'*IEEC/ICE*.

6.2 Programació de la *FPGA*

Per tal de poder assolir els objectius d'un hipotètic experiment del projecte *GRLOW*, corresponents a la configuració de l'entorn de l'experiment i la gestió del volum d'informació, cal determinar les dades de configuració que permeten inicialitzar el sistema. Les implementacions relacionades amb les estructures de dades de configuració, s'han dut a terme aprofitat estructures de dades ja definides en el software facilitat per l'*IEEC/ICE*. D'aquesta manera, s'ha pogut assegurar la compatibilitat de les dades de configuració, ja que han estat declarades amb un conjunt de tipus de dades compatibles amb l'arquitectura de la *CPU* del laboratori. En relació al protocol de comunicacions *PUS*, ha estat necessari programar les funcions encarregades de registrar les peticions dels paquets *TC*, així com d'altres per tractar les dades rebudes, desempaquetant la

informació i emmagatzemant les dades de configuració. Els paràmetres de configuració, apareixen en la següent taula, amb la informació associada a cada dada, el seu rang de valors, les unitats amb que es mesuren, i el tipus de dada a emmagatzemar.

DADA DE CONFIGURACIÓ	RANG DE VALORS			UNITATS	TIPUS
	MIN	MAX	DEFAULT		
BINS A EXECUTAR AL MODUL BIN SELECTOR	0	100	10	BINS	INTEGER
SAMPLES A EXECUTAR AL MÒDUL FFT	0	10240	1024	SAMPLES	INTEGER
FREQÜENCIA DE MODULACIÓ DEL PIEZO	0	1000	200	HERTS (hz)	INTEGER
TIPUS DE SENYAL DEL PIEZO	1	3	1	N/A	INTEGER
VOLTATGE DEL PIEZO	0	600	200	VOLTS (v)	DOUBLE
FREQÜENCIA DE MOSTREIG DELS SENSORS DE TEMPERATURA	1	5	1	HERTS (hz)	DOUBLE
INTENSITAT DEL LÀSER	TBC	TBC	TBC	TBC	-
TEMPERATURA DEL LÀSER	20	25	22	CELSIUS (c)	DOUBLE
FREQÜENCIA DE MOSTREIG DEL FOTODÍODE	10	50	20	KILOHERTS (khz)	DOUBLE
PRESSIÓ CAMBRA DE BUIT	1	5	1	HERTS (hz)	DOUBLE
ALGORITME A ESCOLLIR AL MÒDUL FIT	1	3	1	N/A	INTEGER

***Figura 18**. Taula amb les dades de configuració del sistema*

Totes les dades que apareixen en la taula anterior, han de permetre configurar el software de control, per poder realitzar l'experiment del laboratori, a través d'aquest programa de control.

6.3 Programació de l'*EC*

Amb l'objectiu de controlar l'experiment del projecte *GRLOW* via software, s'ha desenvolupat una interfície gràfica d'usuari durant el període de prototipatge. Amb un total de quatre pantalles gràfiques, mostrades en la secció *Apèndix 3*, l'usuari pot interaccionar amb el sistema de control mitjançant un conjunt d'elements gràfics. Seguidament, detallarem amb més detall el funcionament de cada pantalla desenvolupada, així com la interacció de l'usuari amb la mateixa.

La pantalla inicial que apareix en executar l'*EC*, és una pantalla de login, on permet identificar la sessió amb la modalitat experimental o debug. Cap de les dues modalitats necessiten contrasenya per identificar-se, per tant, només seleccionant una modalitat, l'usuari pot iniciar sessió en el sistema. Un cop iniciada la sessió, apareix la pantalla principal de control, on l'usuari pot realitzar un conjunt d'accions de control, visualitzant alhora un display juntament amb informació sobre l'experiment. Tanmateix l'usuari pot sortir del programa amb el botó d'*èxit*, o desconnectar la sessió amb el botó de *loggout*. En la part inferior d'aquesta mateixa pantalla, hi ha implementat una secció de text que s'actualitza amb la interacció de l'usuari amb el sistema. Aquesta secció anomenada *message handler*, permet a l'usuari visualitzar les accions que va realitzant durant el transcurs del control del programa. Les accions de control que l'usuari pot realitzar des de la pantalla principal de control son les següents: iniciar o parar l'execució de l'experiment, configurar els paràmetres de configuració del sistema o visualitzar més informació sobre l'experiment. Les dues primeres accions permeten controlar l'execució de l'experiment, iniciant o parant l'execució de l'experiment, visualitzant al display informació sobre les accions de l'execució. La informació representada al display en format led, ofereix a l'usuari un mètode per verificar l'estat de l'execució de l'experiment. L'acció de configurar els paràmetres, permet visualitzar una nova pantalla de configuració del sistema, oferint a l'usuari l'opció de modificar o guardar els paràmetres del sistema. L'última acció de visualitzar més informació sobre l'experiment, permet obtenir informació addicional d'anàlisi del sistema, alhora de poder prioritzar les diferents visualitzacions de dades, especificant quines dades visualitzar des de la pantalla principal de control i des de la pantalla de visualització d'informació.

6.4 Fases de test

Aquestes fases de *testing*, son un conjunt d'activitats dins de l'àmbit de l'Enginyeria del Software que permeten obtenir informació sobre la qualitat del software. Al llarg del desenvolupament del software de control, s'han realitzat períodes de test, al mateix temps que s'ha programat la lògica del programa. Encara que existeixen diferents models per realitzar fases de test, per al desenvolupament d'aquest projecte s'han efectuat principalment proves funcionals i d'integració. Les proves funcionals han estat realitzades amb l'objectiu de verificar el comportament del sistema descrit en el *Requirement Document*. Les proves d'integració han permès assegurar el correcte funcionament del software de control envers a la comunicació entre els components del sistema. És necessari assegurar la integritat del sistema descrit en el *Design Document*, de manera que la implementació dels diferents components garanteixin l'assoliment dels requisits no funcionals del *EC* referents al rendiment, usabilitat, escalabilitat i estabilitat del sistema.

7 CONCLUSIONS

El desenvolupament del software de control per al projecte *GRLOW*, ha permès realitzar un projecte d'Enginyeria del Software, aplicant les activitats pròpies d'aquesta disciplina com a futur enginyer. Els objectius principals d'anàlisi i disseny de l'entorn de l'experiment, així com la gestió i emmagatzematge de les dades generades de l'experiment i el seu posterior prototipatge s'han assolit satisfactòriament. Cal destacar que l'únic requeriment no funcional que no s'ha arribat a assolir ha sigut el requeriment de l'*EC* envers a la creació d'un manual d'usuari.

Com a línies futures, seria necessari finalitzar el desenvolupament de la lògica de control del prototip. D'aquesta manera, es podria executar el software de control al laboratori, comprovant el control via software de l'experiment en la seva totalitat. A més, realitzar altres fases de test, com comproves de regressió, ajudaria a verificar la solidesa del programa envers a evolucions o generacions de versions noves del software de control.

AGRAÏMENTS

Als meus tutors del projecte, Lluís Gesa Boté i Miquel Nofrarias, per donar-me l'oportunitat de realitzar aquest projecte de recerca, i en especial a Lluís Gesa Boté pel suport i la confiança al llarg de tot el desenvolupament del projecte.

REFERÈNCIES

[1] Relativistic Lighthouse: The Role of the binary pulsar in proving the existence of Gravitational Waves, Daniel Kennefick , https://www.researchgate.net/publication/263773692_Relativistic_Lighthouses_The_Role_of_the_Binary_Pulsar_in_proving_the_existence_of_Gravitational_Waves

[2] What is TCP/IP, protocol communication, http://searchnetworking.techtarget.com/definition/TCP-IP

[3] Fast Fourier Transform Algorithm, ftp://ftp.udistrital.edu.co/Documentacion/Electronica/Dsp/capitulo6.PDF

[4] The VHDL language, http://web.ewu.edu/groups/technology/Claudio/ee430/Cad/AccoladeVHDLref.pdf

[5] BitBucket, web-based hosting service for projects that use either the Mercurial or Git revision control system, https://bitbucket.org/

[6] Git, widley-used source code managment system for software development, https://git-scm.com/

[7] GitEye, intuitive graphical Git client, http://www.collab.net/products/giteye

[8] Unified Modeling Language, modeling language in t he field of software engineering, http://www.uml.org/

[9] Cacoo, free online diagram tools, https://cacoo.com/diagrams/

[10] Packet Utilisation Standard: protocol de comunicacions que proporciona un procés d'estandardització per la capa d'aplicació que es troba en la ESA PacketTelecomand Standard (TC) i ESA Packet Telemetry Standar (TM).

[11] wxPython, python bindings to the wxWindwos cross-platform toolkit, http://www.wxpython.org/

[12] Adobe Photoshoop CC, raster graphics editor, http://www.adobe.com/products/photoshop.html#

[13] Leon3 Processor, synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture, http://www.gaisler.com/index.php/products/processors/leon3

[14] System on a chip: tecnologies de fabricació que integren tots o gran part dels mòduls d'un computador o qualsevol sistema informàtic o electrònic, en un únic circuit integrat o chip.

[15] Oficial website RTEMS, open source Real Time Operating System, www.rtems.org

[16] General Responsibility Assignment Software Patterns, fa referència a una sèrie de bones pràctiques i principis fonamentals a l'hora d'assignar responsabilitats als diferents objectes i classes.

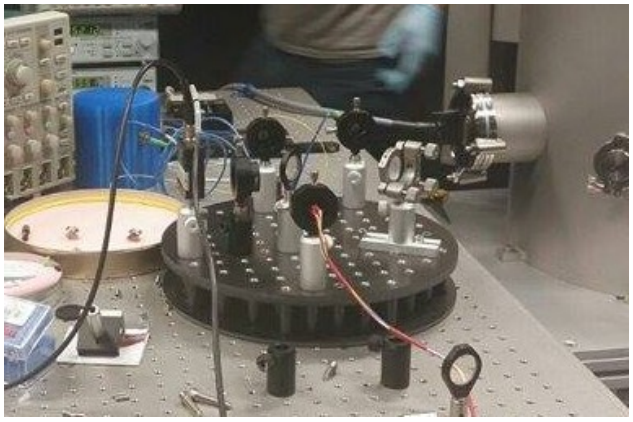
[17] Getter: mètode característic d'una classe, que permet retornar un atribut privat de la classe

[18] Setter: mètode característic d'una classe, que permet accedir a un atribut privat, modificant el seu valor.

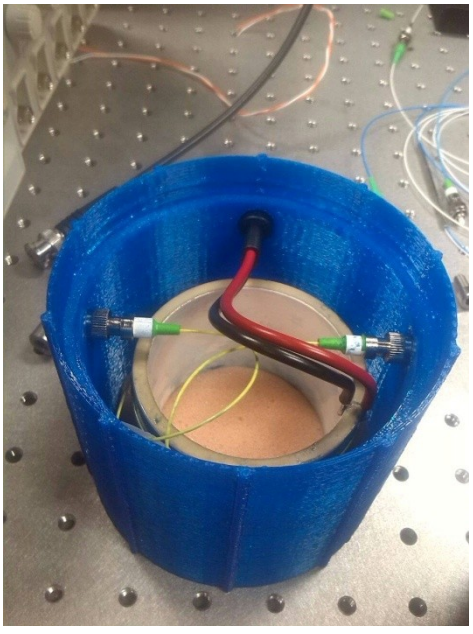
[19] LISA Pathfinder Data Management Unit, http://sci.esa.int/lisa-pathfinder/45879-data-management-unit/

APÈNDIX

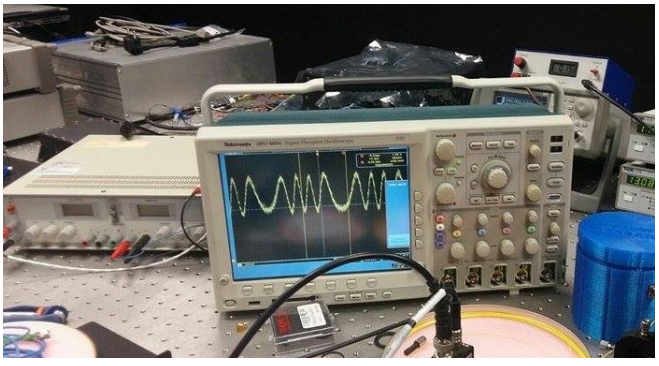
A1.1 Banc òptic d'interferometria



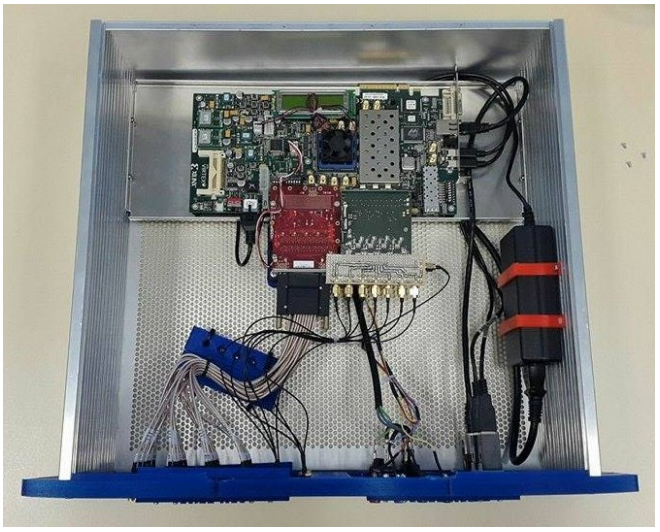
A1.2 Piezo



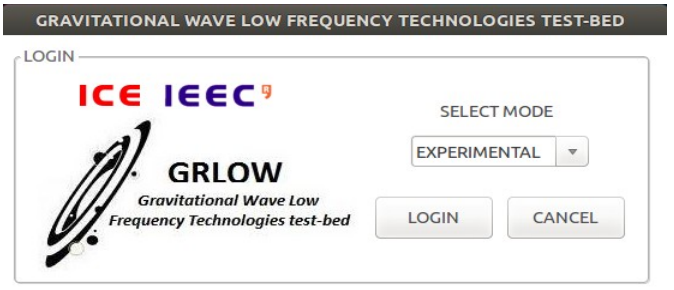
A1.3 Oscil·loscopi amb patró d'interferometria



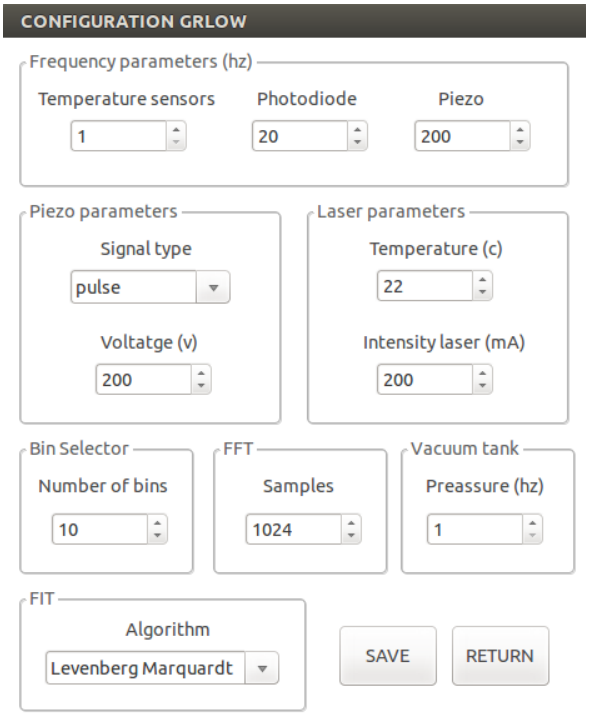
A2 FPGA



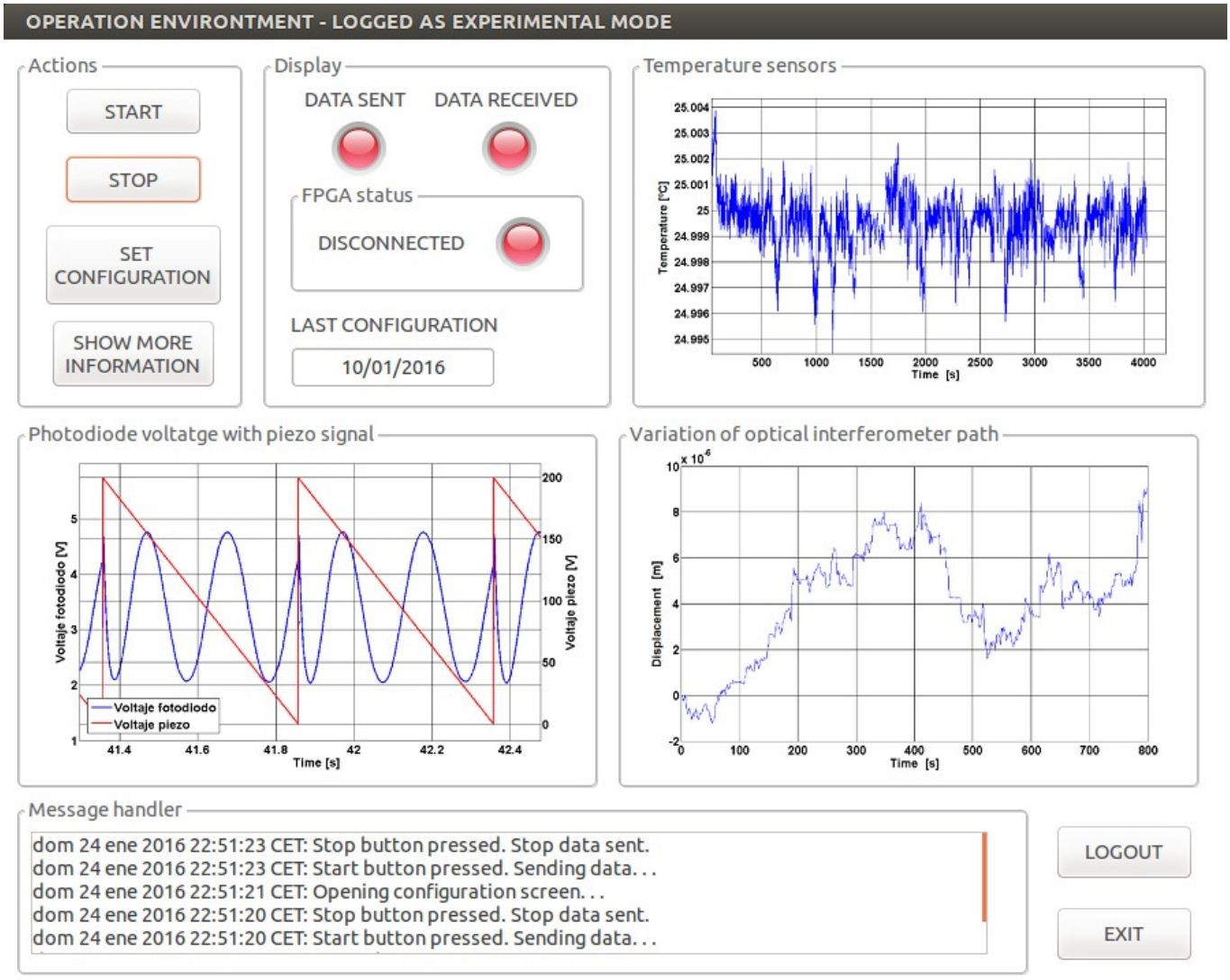
A3.1 Pantalla de login



A3.2 Pantalla de configuració



A3.3 PANTALLA PRINCIPAL DE CONTROL



A3.4 PANTALLA DE VISUALITZACIÓ I CONFIGURACIÓ DE LA INFORMACIÓ D'ANÀLISI