Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Desenvolupament d'un sistema Hardware-In-the-Loop per a l'estudi de sistemes de control

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Gemma Zafra Ruiz

DIRECTOR: Marcos Quílez Figuerola

DATA: 16 de març del 2018

Resum

L'objectiu principal d'aquest projecte consisteix en desenvolupar i documentar un banc de proves que permeti estudiar el comportament de sistemes de control aplicats a diferents plantes mitjançant la tècnica de simulació Hardware-In-the-Loop (HIL). Aquest banc de proves servirà tant per a provar el funcionament de la planta a estudiar com per fer pràctiques de sistemes de control en l'àmbit docent. A més, es realitzarà fent servir dues eines software diferents (LabVIEW i Matlab) amb el propòsit d'explorar-les i comparar-les, especialment en la part de control i simulació i de interfície gràfica.

A continuació es detalla el contingut de cada capítol.

El primer capítol presenta l'estat de l'art en l'àrea de la simulació HIL i una breu descripció del funcionament d'un controlador PID i d'una targeta d'adquisició de dades (DAQ).

En el segon i tercer capítol s'introdueixen les diverses eines i materials necessaris per la realització del projecte. S'inclouen fonamentalment els programes LabVIEW i Matlab, la placa Arduino UNO i el dispositiu DAQ NI USB 6001.

El quart i cinquè capítol són el cos principal del projecte. El primer tracta l'etapa que involucra només una part software, en la qual es caracteritza una planta tèrmica i es creen dos simuladors molt semblants aptes per controlar qualsevol planta. El segon tracta l'etapa que consta d'una part hardware, a més de la de software. En aquesta fase es treballa únicament amb LabVIEW, es desenvolupa el banc de proves desitjat i es comprova el correcte funcionament.

En el sisè capítol es recullen les conclusions extretes del procés de desenvolupament del banc de proves i de la verificació del seu funcionament. També es comparen els dos softwares utilitzats i es fa una valoració de la tècnica HIL.

Finalment a l'últim capítol es comenten els possibles treballs que podrien realitzar-se en el futur com a continuació d'aquest.

Abstract

The main goal of this project is to develop a test bench to study the behaviour of control systems applied to different plants using the technique of the Hardware-In-the-Loop (HIL) simulation. This test bench will be useful both to test the operation of the plant to study and to carry out control system practices in the educational field. In addition, it will be done using two different software tools (LabVIEW and Matlab) in order to explore and compare them, especially in the part of control and simulation and graphical interface.

The content of each chapter is detailed below.

The first chapter presents the state of the art in the HIL simulation area is carried out and a brief description of the characteristics of a PID controller and a data acquisition device (DAQ) is done.

In the second and third chapter we introduce the tools and materials necessary to implement the project. The programs of LabVIEW and Matlab, the Arduino UNO board and the DAQ NI USB 6001 are included.

The fourth and fifth chapters are the main body of the project. The first one deals with the phase that involves only a software part, in which a thermal plant is characterized and two similar simulators are created to control any plant. The second deals with the phase that involves a hardware part, as well as software. This phase works only with LabVIEW, the desired test bench is developed and the correct operation is checked.

The sixth chapter contains the conclusions drawn from the development process of the test bench and the verification of its operation. The two software tools used are also compared and the HIL technique is evaluated.

Finally, in the last chapter we discuss the possible works that could be done in the future as a continuation of what has been done here.

Índex

Índex de figures

Índex de taules

Introducció

Aquest projecte sorgeix de la necessitat de simular i provar el funcionament de sistemes de control en aplicacions de l'àmbit de l'enginyeria de biosistemes. Dissenyar o simplement sintonitzar controladors PID directament connectats a una planta real sovint resulta inviable. A vegades, això és degut al fet de no disposar de la planta que es vol estudiar. Altres vegades, el risc de malmetre el biosistema degut a un error en els ajustos del controlador és el que desaconsella utilitzar la planta real. Per aquest motiu, disposar d'un banc de proves que emuli el comportament de la planta a controlar seria de gran utilitat.

Un dels camps de recerca de l'enginyeria de biosistemes és el cultiu de colònies de llevats i bacteris en estacions espacials. Aquest tipus de cultius fan un ús intensiu de sistemes de control. I és en aquest punt on l'enginyeria aeroespacial pot proporcionar eines de gran utilitat. En l'àmbit aeronàutic el control és un tema habitual i des de fa unes dècades s'ha introduït una tècnica de simulació anomenada Hardware-In-the-Loop (HIL) enfocada a la prova de sistemes complexos que presenta grans avantatges.

Per a verificar i validar un disseny aeronàutic es necessita fer proves exhaustives i fiables. Però, els sistemes cada cop són més complexos (p. ex. en un vehicle hi ha interacció entre sistemes elèctrics, mecànics, tèrmics...) i s'està arribant a un punt en el que les metodologies tradicionals de verificació i validació s'estan quedant obsoletes.

En un flux de treball tradicional, moltes vegades no es podia provar el disseny de sistemes de control fins a una etapa molt avançada del cicle de desenvolupament, quan els elements hardware del sistema estaven disponibles. Sovint, els problemes sorgits es resolien ajustant el sistema de control durant la integració final del sistema.

Però, en el cas dels sistemes complexos d'avui en dia, aquest flux de treball tradicional no és òptim ja que presenta una sèrie d'inconvenients. Per una banda, si no es descobreix l'error en les primeres etapes, la complexa interacció entre diferents subsistemes pot fer que sigui molt difícil rastrejar el problema i solucionar-lo i per tant, pot forçar a començar de nou el disseny. D'altra banda, la realització de proves amb una planta real complexa pot ser complicada o bé pot provocar danys en equips i riscos de seguretat al personal.

Els dissenyadors de sistemes són conscients que es necessiten nous mètodes de verificació i, de fet, estan adoptant el disseny basat en models, com és el cas de la simulació HIL. El disseny basat en models consisteix en simular la planta física (màquina) i provar el sistema de control en les primeres fases del cicle de desenvolupament, quan els errors detectats són més fàcils i més econòmics de corregir. Per tant, permet la verificació anticipada, la qual cosa disminueix el temps de disseny, abarateix els costos i millora la qualitat, la precisió i el rendiment general del sistema.

Exportar la tècnica de la simulació HIL a l'enginyeria de biosistemes obre un ventall de noves possibilitats. Disposar d'un banc de proves basat en la simulació HIL permetria experimentar amb un controlador real connectat a una simulació

de la planta que es vol estudiar i controlar sense necessitat de tenir-la físicament o sense el risc de malmetre-la.

L'objectiu principal d'aquest projecte consisteix en desenvolupar i documentar un banc de proves que permeti estudiar el comportament de sistemes de control aplicats a diferents plantes mitjançant la tècnica de simulació Hardware-In-the-Loop (HIL). Un requisit imprescindible del banc de proves implementat és que pugui ser utilitzat per fer pràctiques de sistemes de control en un àmbit docent universitari. Aquesta característica fa necessari que la documentació descrigui detalladament el procés de desenvolupament. A més, s'utilitzaran dues eines softwares diferents (LabVIEW i Matlab) amb el propòsit d'explorar-les i comparar-les, especialment en la part de control i simulació i de interfície gràfica.

1. Marc teòric

1.1. Simulació Hardware-In-the-Loop (HIL)

1.1.1. Definició

La simulació Hardware-In-the-Loop (HIL) és una tècnica que es fa servir per al desenvolupament i la prova de sistemes de control que s'utilitzen per al funcionament de màquines i sistemes complexos.

Per entendre com s'arriba a aquesta simulació es parteix del següent esquema:

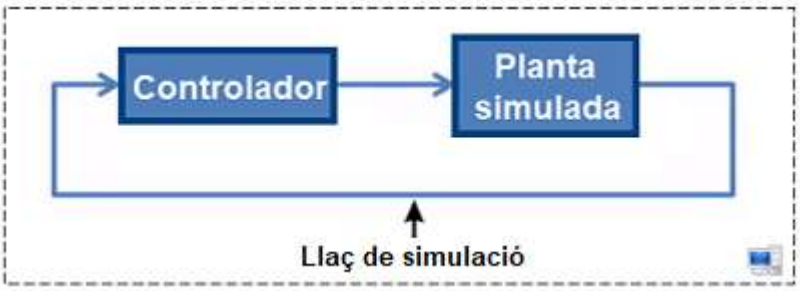


Fig. 1.1: Llaç de simulació

A la figura anterior es pot veure representada una simulació d'un sistema de control qualsevol, on tots els components estan simulats per ordinador. Llavors dins del llaç de simulació s'elimina el controlador simulat i s'afegeix un de real. D'aquí el nom de Hardware-In-the-Loop. Com que hi ha una part software (ordinador) i una part hardware (controlador) es necessita un dispositiu d'adquisició de dades (DAQ) que permeti la comunicació entre ambdues parts.

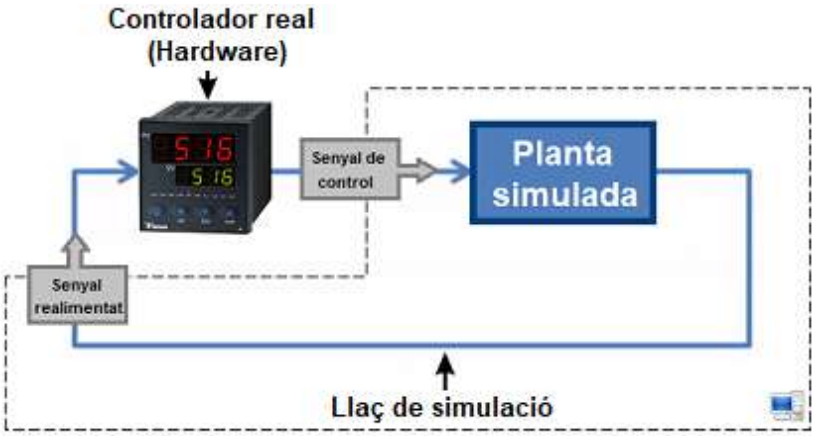


Fig. 1.2: Simulació HIL

1.1.2. Característiques

La simulació HIL es caracteritza per incloure una part software que és una simulació de la màquina o part física del sistema (planta) i una part hardware que sol ser la ECU (Unitat de Control Electrònic) que es vol provar com, per exemple, un controlador PID industrial. La planta es connecta normalment amb el sistema de control mitjançant actuadors i sensors i la unitat de control electrònic és la responsable de prendre decisions (operar els actuadors) en funció de les mesures proporcionades pels sensors.

Per aclarir el concepte es fa servir l'exemple d'un motor de cotxe. La simulació HIL inclouria una part hardware que seria la ECU del motor (encarregada de convertir les mesures del sensor en accions com ajustar la ingesta d'aire quan es prem l'accelerador) i una part software que simularia la planta (motor) amb senyals d'entrada i sortida reals com si el motor físic estigués present.

En la simulació més bàsica tant l'ECU com la planta són elements software, és a dir, estan programats per ordinador.

En la simulació tradicional l'ECU és la part software, la planta és la part hardware i es necessita un dispositiu DAQ que faci de interfície entre l'ordinador i els senyals reals del món exterior. L'entrada analògica del DAQ (AI) llegeix el senyal de sortida (y) i el converteix en digital per poder ser manipulat per l'ordinador i la sortida analògica del DAQ (AO) treu el senyal de control (u) per poder actuar sobre la planta real.

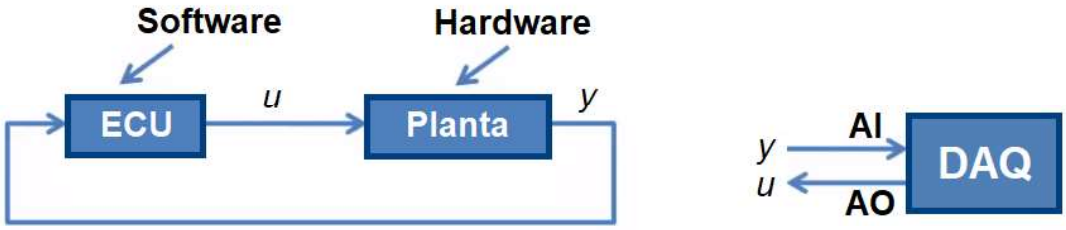


Fig. 1.3: DAQ en la simulació tradicional

En canvi en la simulació HIL es canvien completament els papers. L'ECU passa a ser la part hardware i la planta la part software. Per tant, l'entrada analògica en aquest cas serà el senyal de control i la sortida analògica el senyal de sortida de la planta.

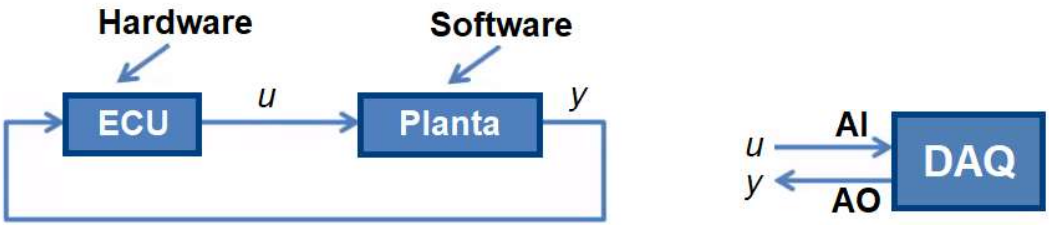


Fig. 1.4: DAQ en la simulació HIL

1.1.3. Avantatges

El principal objectiu de la simulació HIL és provar el dispositiu hardware en temps real amb una planta simulada abans d'implementar-lo a la planta real. El desenvolupament de la part de software no necessita esperar a tenir una planta física per programar i provar el codi. Els principals avantatges que comporta l'ús de la simulació HIL s'enumeren a continuació:

- Augment de la seguretat: En sistemes on la seguretat del personal és de màxima importància (com les grues elevadores), la simulació HIL és extremadament útil. La prova d'aquestes màquines és potencialment perillosa i requereix procediments de seguretat extensos. Mitjançant la simulació HIL, es poden executar proves que a la realitat destruirien la planta real o que posarien en risc la seguretat de les persones.

- Estalvi de temps: En moltes empreses, les màquines i els sistemes de control es desenvolupen de forma paral·lela, la qual cosa significa que només es poden trobar errors durant la posada en marxa de les dues parts. La simulació HIL, permet trobar aquests errors al principi i resoldre'ls de manera anticipada, i per tant, reduir el temps de sortida al mercat del producte.

- Reducció de costos: Per una banda la simulació HIL pot ser un veritable estalvi de diners quan es prova una maquinària costosa. D'altra banda estalvia molta feina durant la fase d'implementació on les hores de treball generalment són molt més costoses que durant les primeres etapes del disseny.

1.1.4. Etapes

Per dur a terme la simulació HIL és habitual seguir els següents passos:

- Simulació software: S'ha de desenvolupar un model matemàtic que representi la planta real. Un cop fet, s'ha de programar la planta simulada i s'ha de dissenyar el controlador. Tot això s'implementa per ordinador, per tant, aquest pas només involucra software i no necessita cap dispositiu DAQ.

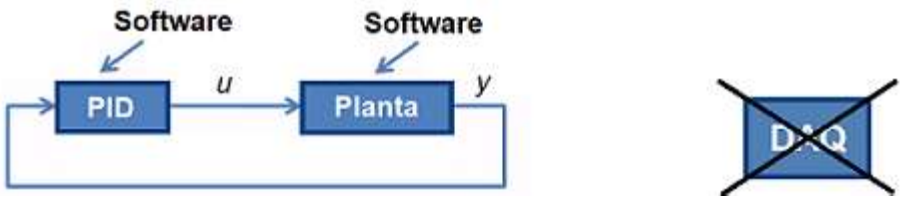


Fig. 1.5: Simulació HIL: etapa I

- Simulació HIL: S'ha de reemplaçar el PID simulat per un PID real i s'ha de provar aquest dispositiu amb la planta simulada anteriorment. Ara es té una part hardware i una part software, per tant, cal un dispositiu DAQ.

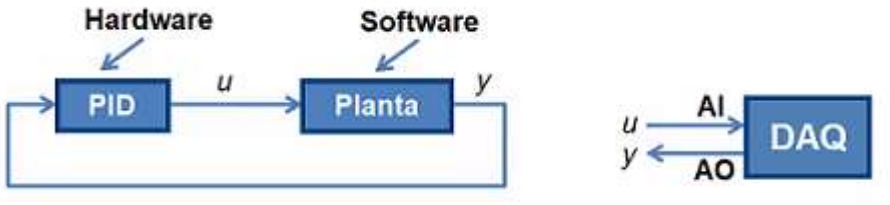


Fig. 1.6: Simulació HIL: etapa II

- Prova del sistema real: Finalment s'ha de substituir la planta simulada per la planta real i s'ha de comprovar que tot funcioni de la manera prevista. Com que aquest pas només involucra hardware no és necessari el dispositiu DAQ.

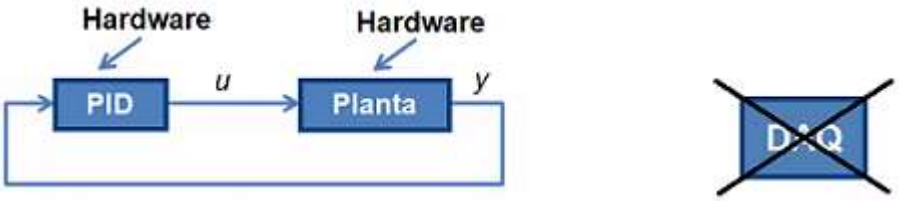


Fig. 1.7: Simulació HIL: etapa III

1.2. Controlador PID

1.2.1. Definició

Un controlador o regulador PID és un dispositiu que permet controlar un sistema en llaç tancat perquè assoleixi l'estat de sortida desitjat. És l'algorisme de control més utilitzat en la indústria i la seva popularitat es deguda en part al seu rendiment robust en un ampli rang de condicions de funcionament i en part també a la seva simplicitat.

Com el seu nom indica, el controlador PID està compost de tres coeficients bàsics (guany proporcional, guany integral i guany derivatiu) que cal sintonitzar per obtenir la resposta òptima. La idea bàsica del controlador PID és llegir un sensor i després calcular la sortida de l'actuador sumant les respostes proporcional, integral i derivativa.

La funció de transferència del controlador PID té la següent forma:

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \qquad \textbf{(1.1)}$$

A la figura 1.8 la variable *e(t)* representa l'error, la diferència entre el valor d'entrada desitjat *r(t)* i la sortida real *y(t)*. Aquest senyal d'error *e(t)* s'envia al controlador PID i el senyal *u(t)* és igual al guany proporcional (K_P) multiplicat per

l'error més el guany integral (K_I) multiplicat per la integral de l'error més el guany derivatiu (K_D) multiplicat per la derivada de l'error.

$$u = K_P\, e + K_I \int e\, dt + K_D \frac{de}{dt} \qquad \textbf{(1.2)}$$

Aquest senyal *u(t)* s'envia a la planta i s'obté la nova sortida *y(t)*. A la vegada, la nova sortida s'envia un altre cop al sensor per trobar el nou senyal d'error. El controlador pren aquest nou senyal d'error i així successivament.

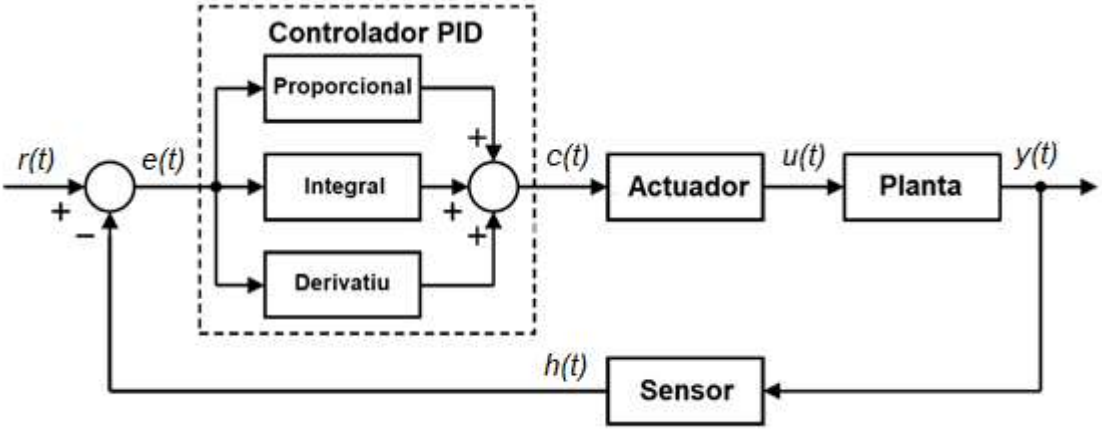


Fig. 1.8: Sistema de control en llaç tancat

1.3. Targeta d'adquisició de dades (DAQ)

És un dispositiu hardware que actua com una interfície entre un ordinador i senyals físics del món exterior. S'encarrega principalment de digitalitzar senyals analògics entrants perquè un ordinador pugui interpretar-los. Els tres components clau d'una targeta d'adquisició de dades són el circuit de condicionament de senyals, el convertidor analògic-digital i, finalment, el bus de l'ordinador.

2. Software utilitzat

Un cop entès el concepte de simulació HIL es busca quines eines hi ha disponibles per poder dur-la a terme. S'han seleccionat LabVIEW i Matlab ja que ambdues compleixen els requisits necessaris per a la realització d'aquest treball. A més, són eines que s'han utilitzat durant el grau universitari i per tant, existeix una familiarització prèvia que facilita l'aprenentatge futur. Es fa servir també l'entorn de desenvolupament (IDE) d'Arduino, per programar el controlador PID. A continuació es fa una breu descripció del funcionament bàsic de cadascuna.

2.1. LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) és un entorn de programació gràfic creat per National Instruments i dissenyat perquè enginyers i científics desenvolupin aplicacions de prova, control i mesura. Gràcies al seu llenguatge de programació G facilita molt la comprensió; els programes no s'escriuen, sinó que es dibuixen. A més, LabVIEW combina la potència de la programació gràfica amb dispositius hardware per simplificar i accelerar el desenvolupament de dissenys.

Els programes de LabVIEW s'anomenen instruments virtuals o VI ja que aparenten ser instruments físics com per exemple un oscil·loscopi. Quan es crea un nou VI s'obren dues finestres: una amb el fons gris (Panell Frontal) i una altra amb el fons blanc (Diagrama de Blocs). El panell frontal és on es crea la interfície gràfica d'usuari i el diagrama de blocs és on es programa tot el codi.

Pel propòsit del projecte es necessita, per una banda, el mòdul *LabVIEW Control Design and Simulation* per simular sistemes dinàmics i dissenyar controladors. Per altra banda, s'ha de descarregar el software *NI-DAQmx* perquè l'ordinador reconegui el dispositiu DAQ i aparegui la paleta *DAQmx – Data Acquisition* a LabVIEW. Tant el mòdul com el *driver* es poden descarregar fàcilment des de la pàgina web de National Instruments.

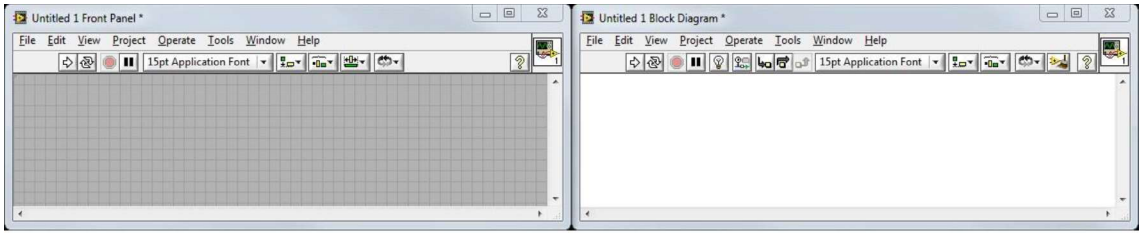


Fig. 2.1: Panell frontal i Diagrama de blocs

2.2. MATLAB/Simulink/GUIDE

MATLAB (abreviatura de MATrix LABoratory) és una eina de software matemàtic creada per The Mathworks que ofereix un entorn de desenvolupament integrat amb un llenguatge de programació propi (llenguatge M). És adequat per a la manipulació de matrius, la representació de dades i funcions, la creació de interfícies d'usuari i la comunicació amb programes amb altres llenguatges i amb altres dispositius hardware. És un software molt utilitzat en universitats i centres d'investigació i desenvolupament. El paquet de MATLAB disposa de dues eines addicionals que amplien les seves prestacions: Simulink i GUIDE.

Simulink és una eina de simulació de models dinàmics i genera arxius amb extensió .mdl (de "model"). Per començar a utilitzar Simulink s'ha de prémer el botó Simulink que hi ha a la barra d'eines o bé s'ha d'escriure "simulink" a la finestra de comandament. Llavors s'obre una nova finestra on es crea el model desitjat. Al botó de llibreria es troben els diferents blocs que es poden utilitzar, només cal arrossegar els blocs a la finestra en blanc. En el cas de Simulink, conté blocs especials *(Data Acquisition Toolbox)* que permeten la comunicació amb targetes d'adquisició de dades. El més útil és utilitzar Simulink juntament amb Matlab. Això vol dir que es fan servir variables en comptes de valors directament a Simulink. D'aquesta manera s'especifiquen les dades i els paràmetres de simulació des de Matlab sense haver de tocar el model de Simulink.

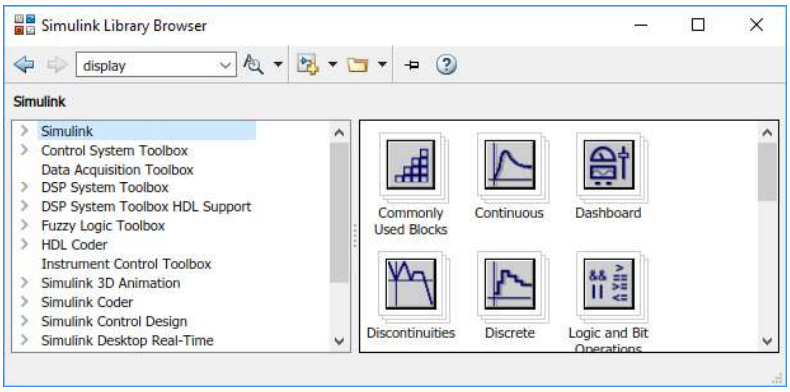


Fig. 2.2: Llibreria de Simulink

GUIDE (Graphical User Interface Development Environment) és una eina que permet dissenyar gràficament una interfície d'usuari i genera de manera automàtica el codi de MATLAB per construir la interfície, el qual es pot modificar per programar el comportament de la aplicació. La interfície gràfica està formada per dos arxius: un (.m) i un altre (.fig). L'arxiu .m és un arxiu de text on es mencionen totes les funcions que fan referència a tots els objectes presentats a l'arxiu .fig, i l'arxiu .fig és on es visualitza el disseny de la interfície gràfica. Per obrir GUIDE s'ha de teclejar la instrucció guide a la finestra de comandament i seguidament es presenta un quadre de diàleg amb diferents opcions. Un cop seleccionada l'opció, s'obre una finestra amb l'arxiu .fig on es troben les eines

necessàries per crear la GUI. Alguns exemples són els *axes* que serveixen per mostrar gràfics o imatges, el *push button* per crear botons, el *static text* per escriure un text, el *edit text* per crear també un text però que pugui ser editat per l'usuari, etc. Amb la realització d'una interfície gràfica d'usuari es pot modificar qualsevol paràmetre d'un sistema creat a Simulink.

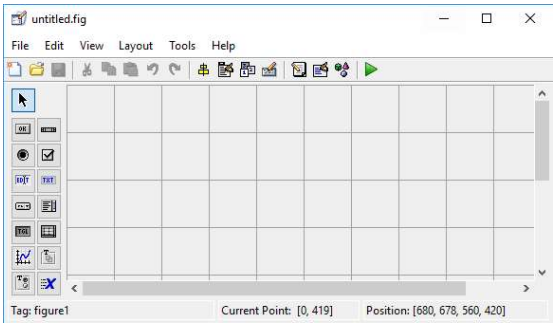


Fig. 2.3: GUI en blanc

2.3. Entorn de desenvolupament d'Arduino

Un entorn de desenvolupament, també conegut com a IDE (Integrated Development Environtment) és un programa informàtic format per un conjunt d'eines de programació. Pot dedicar-se exclusivament a un sol llenguatge de programació o bé pot utilitzar-se per a varis llenguatges. Consisteix en un editor de codi, un compilador, un depurador i un constructor de interfície gràfica. A més en el cas d'Arduino incorpora les eines per carregar el programa a la placa. En aquest cas es fa servir per crear el codi del PID i es pot descarregar l'última versió des de la pàgina web d'Arduino.

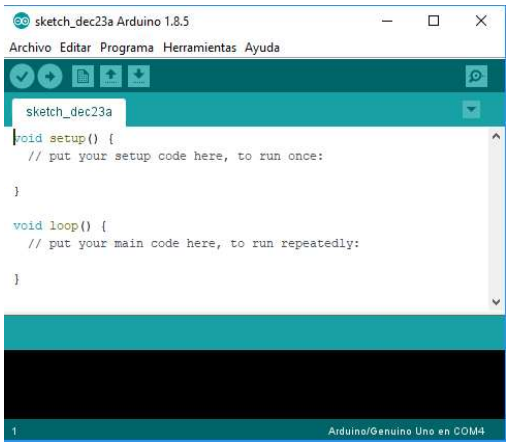


Fig. 2.4: Editor de codi Arduino

3. Hardware utilitzat

Per executar una simulació HIL també és necessària una part hardware que en aquest cas està formada principalment per la targeta d'adquisició de dades USB 6001 i la placa Arduino UNO.

3.1. NI USB 6001

És un dispositiu DAQ multifunció de baix cost que consta de 8 entrades analògiques, 2 sortides analògiques, 13 entrades/sortides digitals i un comptador de 32 bits. A més, té una coberta mecànica lleugera, es poden connectar fàcilment sensors i senyals gràcies a la connexió de terminal de cargol i s'alimenta per bus.

3.2. Controlador PID basat en Arduino

En un primer moment es tenia la idea d'utilitzar un PID comercial, però més endavant s'ha optat per construir i programar un PID basat en Arduino, ja que sembla una opció interesant, amb la qual es poden adquirir altres coneixements.

L'Arduino UNO és una placa amb un microcontrolador basat en l'ATmega328P i és la millor per començar amb l'electrònica i la codificació. Compta amb 14 pins digitals d'entrada/sortida (dels quals 6 es poden utilitzar com a sortides PWM), 6 entrades analògiques, un cristall de quars de 16 MHz, una connexió USB d'impressora, un connector d'alimentació, un encapçalament ICSP i un botó de reset. La connexió USB amb l'ordinador és necessària per programar la placa. Per alimentar-la es pot utilitzar el mateix cable USB o bé una font d'alimentació externa.

4. Etapa I: Simulació software

En aquesta primera fase es desenvolupa un model matemàtic que representa la planta tèrmica esmentada a la introducció. A continuació es programa la planta simulada i es dissenya el controlador. Finalment es crea una interfície gràfica d'usuari per a simular qualsevol planta de la qual es conegui la seva funció de transferència.

4.1. Caracterització de la planta

En aquest apartat es modela un sistema tèrmic com és el cas d'un bany termostàtic. L'objectiu de modelar un sistema, en definitiva, de caracteritzar-lo matemàticament, consisteix en descriure'l analíticament de la forma més senzilla possible, per posteriorment predir l'evolució d'aquest, analitzar l'efecte de la variació de paràmetres o de les entrades i poder realitzar el disseny d'un controlador. Es coneix com model matemàtic el conjunt d'expressions matemàtiques que descriuen les relacions existents entre les magnituds del sistema i que representen el comportament dinàmic del sistema.

4.1.1. Sistemes tèrmics

Els sistemes tèrmics són aquells que involucren l'intercanvi d'energia calorífica d'una substància a una altra. Les seves expressions segueixen la llei de conservació de l'energia: la quantitat de calor que s'aporta a un element menys la quantitat de calor que aquest desprèn és igual a la quantitat de calor acumulat per aquest. Els senyals d'entrada i sortida per a aquest tipus de sistemes són la temperatura *T(t)* i la potència calorífica o flux calorífic *q(t)*.

L'objectiu d'un sistema de control de temperatura és mantenir la temperatura de la planta a un valor corresponent al que proporciona el senyal de referència. Aquest senyal de referència és un nivell de voltatge que representa la temperatura desitjada per a la planta tèrmica. Dins de la planta hi ha un sensor de temperatura que produeix un voltatge que és acoblat al sumador juntament amb el senyal de referència. Aquest voltatge es compara amb el voltatge de referència, i si hi ha una discrepància (error), el controlador envia un senyal a l'element calefactor per fer que la temperatura de la planta adquireixi el valor desitjat.

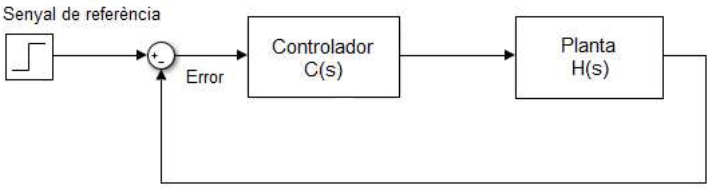


Fig. 4.1: Esquema sistema en llaç tancat

4.1.2. Variables tèrmiques

Com s'ha comentat a l'inici de l'apartat un bany termostàtic és un sistema tèrmic i com a tal, compleix la llei de conservació de l'energia: la diferència entre la quantitat de calor aportada a l'aigua i la quantitat de calor alliberada se li denomina calor neta:

$$q = q_i - q_o \tag{4.1}$$

Dos elements que es fan servir per descriure els processos de transmissió de la calor i d'acumulació de l'energia calorífica són la resistència tèrmica (R) i la capacitància tèrmica (C).

Dins de l'àmbit del modelat de sistemes tèrmics senzills, les transferències de calor només es produeixen per conducció i en menor mesura, per convecció. Ambdues formes poden ser expressades a través de la resistència tèrmica, que es defineix com el canvi en la diferència de temperatura dividit pel canvi en el flux calorífic.

$$R = \frac{dT}{dq} \tag{4.2}$$

El flux calorífic transmès de l'aigua a l'ambient serà igualat a la diferència de temperatura partit per la resistència tèrmica. Indica que la raó de transferència de calor en un cos és proporcional a la diferència de temperatura a través del cos.

$$q_o = \frac{Tamb - Taigua}{R} = \frac{T}{R} \tag{4.3}$$

La capacitància tèrmica mostra el nivell de capacitat que té una substància d'emmagatzemar energia tèrmica i es defineix com el canvi en la calor emmagatzemada dividit entre el canvi en la temperatura.

$$C = \frac{dq}{dT} \tag{4.4}$$

La potencia calorífica està definida per la capacitància tèrmica i la variació de la temperatura amb el temps, segons s'obté de la fórmula anterior:

$$q = C \cdot \frac{dT}{dt} \tag{4.5}$$

El flux calorífic no deixa de ser un concepte de potència, energia per unitat de temps. Però sol expressar-se en unitats diferents al watt, com són les kilocalories per segon.

A més, la capacitància tèrmica està relacionada amb la massa de la substància que emmagatzema l'energia tèrmica, m, i amb el seu calor específic, c:

$$C = mc \tag{4.6}$$

A continuació es resumeixen en una taula les principals variables dels sistemes tèrmics amb les corresponents unitats en el sistema internacional.

Taula 4.1: Variables dels sistemes tèrmics

Magnitud física	Abreviació	Unitats SI
Flux o potència calorífic/a	Q	kcal/s o kJ/s=kW
Temperatura	T	K
Calor específic	C	kcal/kg·K
Resistència tèrmica	R	K/W o K/kcal
Capacitància tèrmica	C	kcal/K o J/K

4.1.3. Funció de transferència planta

Per obtenir l'expressió de la planta tèrmica es comença representant el següent esquema que simula el bany termostàtic.

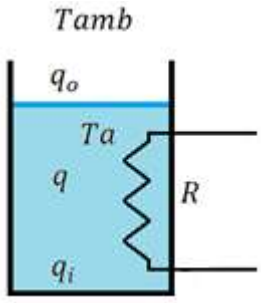


Fig. 4.2: Il·lustració planta tèrmica

L'equació diferencial que descriu el comportament de l'esquema il·lustrat a la figura 4.2 s'obté igualant les equacions (4.1) i (4.5):

$$q_i - q_o = \frac{dT}{dt}C \tag{4.7}$$

Substituint l'equació (4.3) a l'expressió anterior s'obté

$$q_i - \left(\frac{T}{R}\right) = \frac{dT}{dt}C \tag{4.8}$$

que es pot reescriure com

$$Rq_i - T = \frac{dT}{dt}RC \tag{4.9}$$

Aplicant la transformada de Laplace es converteixen les equacions diferencials lineals (en la variable t) en equacions algebraiques (en la variable complexa s).

$$Rq_i(s) - T(s) = T(s)sRC \tag{4.10}$$

Agrupant els termes s'obté la funció de transferència que relaciona q_i amb T

$$\frac{T(s)}{q_i(s)} = \frac{R}{RCs + 1} \tag{4.11}$$

S'observa que la funció de transferència descrita per l'equació (4.11) representa un sistema de primer ordre on RC és la constant de temps (τ) i R és el guany del sistema. La constant de temps del sistema defineix el temps que triga la resposta a l'esglaó a assolir el 63,2 % del seu valor final i el temps d'establiment es defineix com el que necessita el sistema per assolir el règim permanent. Es sol considerar pel temps d'establiment un error del 2% respecte el valor final, per tant, s'assumeix que la resposta a l'esglaó assoleix el seu valor final quan la constant de temps és 4τ (98,2 %).

Una vegada coneguda la forma de la funció de transferència queda per definir el valor de la constant de temps i del guany del sistema. Amb el propòsit de trobar-los s'ha dissenyat un model que representa el comportament de la planta:

$$T(t) = \left[Rq_i\left(1 - e^{-\frac{t}{RC}}\right) + (T_0 - T_{amb})e^{-\frac{t}{RC}}\right] + T_{amb} \qquad \textbf{(4.12)}$$

En aquesta expressió es tenen en compte les condicions inicials $(\mathrm{T_0}, \mathrm{T_{amb}})$ a diferència de l'expressió de la funció de transferència.

Per comprovar si aquest model té sentit físic, s'assumeix un bany termostàtic amb 1 litre d'aigua, una temperatura inicial de 70 ºC i una temperatura ambient de 25 ºC. La capacitat tèrmica de l'aigua és la multiplicació de la massa pel calor específic de l'aigua. La resistència tèrmica s'assumeix de 0,2 K/W i per últim la potència injectada és de 0 W.

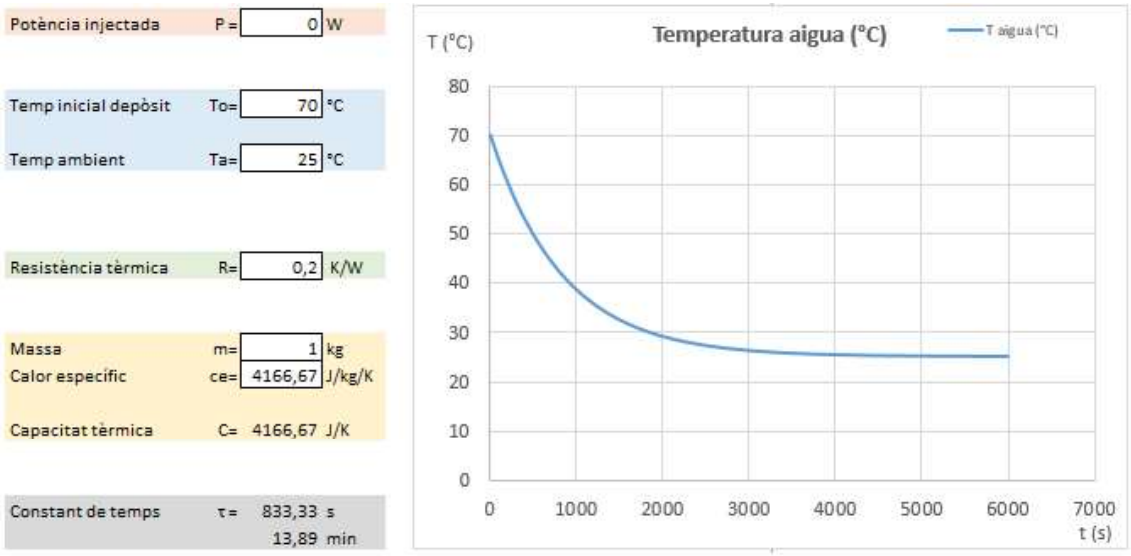


Fig. 4.3: Sistema tèrmic

S'obté una exponencial negativa que parteix de 70 ºC i assoleix pràcticament la temperatura ambient en una hora (3600 s). No és descabellat que 1 litre d'aigua a 70 ºC trigui a refredar-se a una temperatura ambient de 25 ºC aproximadament 1 h, per tant, el model plantejat és un bon candidat.

Es substitueix el guany pel valor de la resistència tèrmica i la constant de temps pel producte de la capacitat tèrmica i la resistència tèrmica i la planta final obtinguda és:

$$\frac{\boldsymbol{T(s)}}{\boldsymbol{q_i(s)}} = \frac{\boldsymbol{0,2}}{\boldsymbol{833,33s + 1}} \qquad \textbf{(4.13)}$$

A continuació es comprova que la resposta a l'esglaó de l'equació de la planta (4.13) coincideix amb l'equació (4.12). Primer de tot s'assumeixen condicions inicials nul·les a l'expressió (12) i s'aplica una potència de 1 W, com si s'apliqués un esglaó unitari. La resta de paràmetres es manté igual i s'obté el següent gràfic on l'eix d'abscisses representa el temps en segons i l'eix d'ordenades la temperatura en graus Celsius.

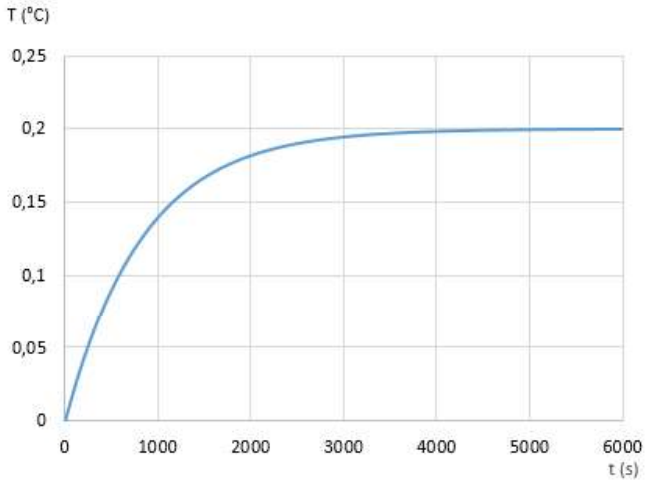


Fig. 4.4: Gràfica de l'expressió 4.12

D'altra banda, s'utilitza Matlab per obtenir la resposta a l'esglaó de l'equació (4.13). Per això es defineix la funció de transferència i s'aplica la funció *step* que per defecte mostra la resposta en llaç obert suposant una entrada unitària.

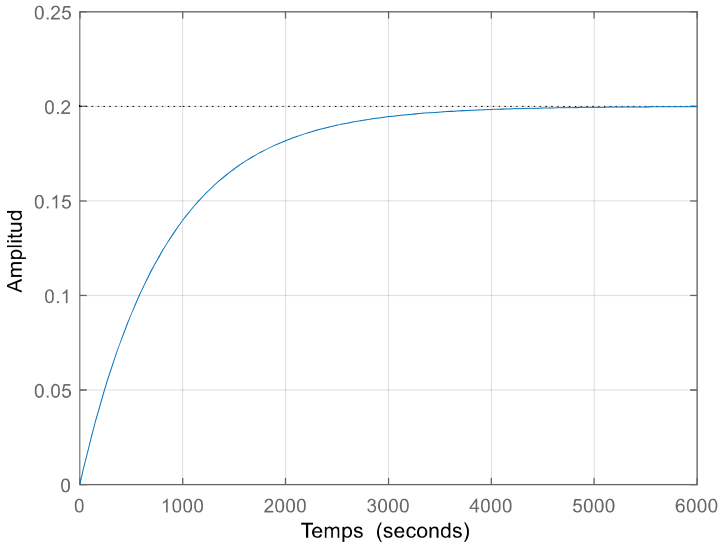


Fig. 4.5: Resposta de la planta a Matlab

Com es pot observar les dues respostes són idèntiques i per tant, ja es té la planta caracteritzada.

4.2. Simulació de la planta amb LabVIEW

4.2.1. Representació del sistema de control

A continuació s'implementa l'esquema típic d'un sistema de control de temperatura a LabVIEW. Aquesta transició és realment senzilla ja que la programació a LabVIEW és molt semblant a la versió esquemàtica en paper.

Primer de tot s'obre la paleta *Simulation* i es selecciona *Control & Simulation Loop*. Aquest requadre és necessari si es vol utilitzar qualsevol de les funcions de la paleta *Simulation*. Es comença seleccionant les funcions *Summation* (el comparador), *Transfer Function*, *Simulation Time Waveform* i *PID*. Fent clic dret al bloc *Transfer Function* i a *Configuration* s'introdueixen els valors de la constant de temps (833,33 s) i el guany (0,2) trobats a l'apartat anterior. Es fa el mateix amb el *PID* i es selecciona la forma paral·lela, deixant els valors que apareixen per defecte (Ki=1, Kp=1, Kd=0). Fent les unions corresponents s'obté l'esquema d'un sistema de control:

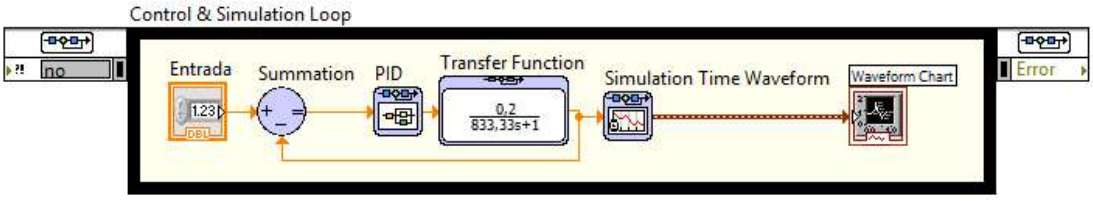


Fig. 4.6: Sistema de control a LabVIEW

4.2.2. Creació de Simulation Subsystems

Els *Simulation Subsystems* serveixen per estructurar el codi i són molt útils quan s'està treballant amb sistemes de simulació grans. Es caracteritzen per tenir el fons del diagrama de blocs de color, en comptes de ser blanc. En aquest cas es creen dos *Simulation Subsystems*: la planta i el PID més el comparador. Primer es selecciona la funció de transferència, es clica a *Edit* i a *Create Simulation Subsystem* i es realitza la mateixa operació amb el conjunt del comparador i el PID. A continuació s'obre el subsistema del PID més el comparador, es fa clic dret sobre el bloc de PID i a *Configuration* es selecciona l'opció Terminal per a cada paràmetre del PID. Aquest canvi permet que les diferents constants siguin controls i es puguin canviar des del panell frontal. Per acabar, s'amplia el requadre dels paràmetres de simulació (a l'esquerra) per fer que es puguin introduir el temps inicial i final de simulació i es canvia el *Solver Method* a *Runge-Kutta 23 (variable)*. Per executar la simulació sense restriccions de temps, s'ha de comprovar que l'opció *Synchronize Loop to Timing Source* està desactivada.

D'aquesta forma la simulació s'executa el més ràpid possible i no en temps real. A la figura 4.7 es veu el programa obtingut amb els dos subsistemes. El subsistema Planta té una entrada (el senyal que surt del controlador PID) i una sortida (la resposta del sistema), mentre que el subsistema Comparador+PID té cinc entrades (la consigna, els guanys proporcional, integral i derivatiu i la resposta del sistema) i una sortida (el senyal de control que actua sobre la planta).

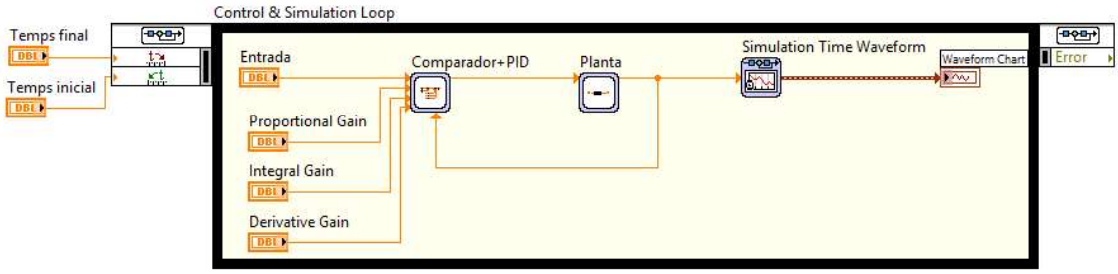


Fig. 4.7: Subsistemes de simulació

Si s'executa el programa anterior assumint una entrada d'amplitud 1 s'obté una resposta subamortiguada (oscil·la cada cop menys) que triga bastant en estabilitzar-se. Evidentment no és una resposta adequada per a la planta tèrmica, més endavant s'escolliran valors més apropiats pel PID.

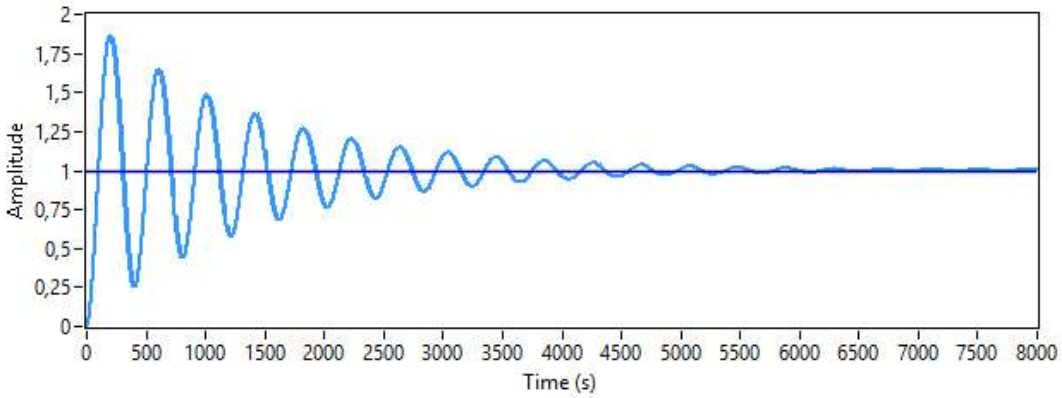


Fig. 4.8: Resposta del sistema

4.2.2. Disseny del PID

Tot i no ser un objectiu del projecte sembla important tenir una idea de com s'han d'escollir els valors d'un PID. En la regulació del PID, un 80% del treball el realitza el guany proporcional, un 15% el guany integral i el 5% restant, quan és necessari, el guany derivatiu. A l'hora d'ajustar aquests paràmetres per controlar correctament una planta es fa servir el següent mètode:

1. S'ajusten els guanys integral i derivatiu a 0.

2. S'augmenta el guany proporcional poc a poc fins aconseguir apropar-se al valor de la consigna, tot i que el sistema sigui inestable. En el cas de que el sistema sigui inestable es redueix el guany proporcional fins que el sistema torni a ser estable, independentment de no haver assolit el valor de la consigna.
3. Si la resposta del sistema està per sota o per sobre del valor de la consigna s'augmenta el guany integral vigilant de no tornar el sistema inestable.
4. Si disminuint el guany integral el sistema segueix sent inestable s'augmenta lleugerament el guany derivatiu.

Es parteix del programa il·lustrat a la figura 4.7, on es simulava un sistema de control de llaç tancat amb un controlador PID, una planta tèrmica i realimentació unitària. A partir d'aquí, es defineix un valor de consigna igual a 1 i s'augmenta progressivament el guany proporcional.

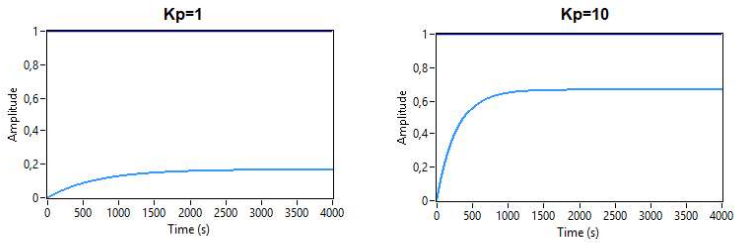


Fig. 4.9: Resposta amb Kp=1 i Kp=10

Amb una acció proporcional petita Kp=1 s'obté una resposta lenta que triga 2500 segons en assolir la temperatura final i l'error és molt gran, de més de 0,8 ºC. A mesura que s'augmenta el guany proporcional, l'error disminueix i la velocitat de la resposta augmenta. Amb un guany proporcional Kp=10 el sistema és una mica més ràpid, triga 1500 segons en assolir la temperatura permanent. Així mateix l'error es redueix a una mica menys de 0,4ºC. Amb guanys superiors s'aconsegueix reduir encara més l'error i una major velocitat de la resposta, però com que no es necessita una resposta molt ràpida es deixa el guany proporcional en 10. Per a millorar aquesta resposta s'incorpora un nou control que elimini l'error i que és el control integral.

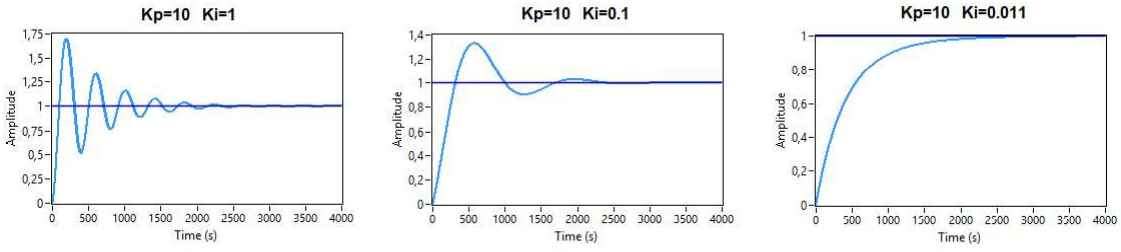


Fig. 4.10: Resposta amb Ki=1, Ki=0,1 i Ki=0,011

Es comença amb una acció integral Ki=1 i s'obté una resposta que assoleix el valor de consigna però amb moltes oscil·lacions. Per tant, s'ha de disminuir el valor del guany integral. Amb un guany de Ki=0,1 encara segueix oscil·lant, però en canvi amb un guany de Ki=0,011 la resposta deixa d'oscil·lar.

Fins ara s'ha implementat un controlador PI. Com que la resposta que ens proporciona aquest controlador és suficient, no cal afegir un controlador derivatiu al sistema. S'ha de tenir present que no és necessari implementar els tres controladors, com més senzill sigui el controlador, millor. Si no es tenen unes especificacions de disseny es pot concloure que la següent equació del controlador PI és adequada:

$$\boldsymbol{PI(s) = 10 + \frac{0.011}{s} = \frac{10s + 0.011}{s}} \qquad \textbf{(4.14)}$$

La resposta ampliada del nostre sistema es mostra a la figura 4.11.

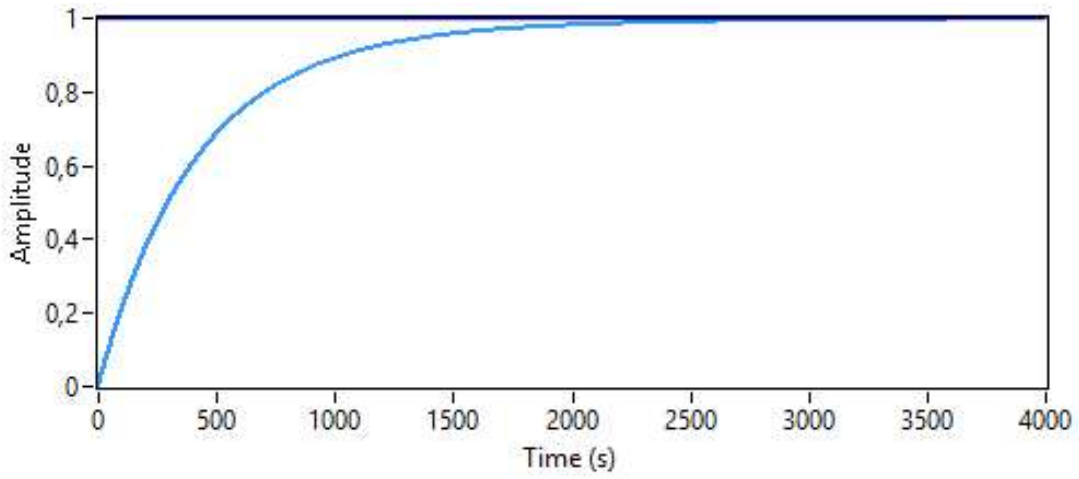


Fig. 4.11: Resposta desitjada Kp=10 i Ki=0,011

4.3. Aplicació a LabVIEW

En aquest apartat es crea una aplicació a LabVIEW que serveix per simular sistemes de control en llaç tancat. Es pot introduir la planta que es desitgi (màxim tercer ordre), dissenyar el controlador per aquesta planta, introduir realimentacions i veure en tot moment la resposta del sistema, l'error que li entra al controlador, el senyal de control que actua sobre la planta i el valor de consigna. En definitiva, pretén ser una ajuda per aquelles persones que estan aprenent control.

4.3.1. Control & Simulation Loop

En primer lloc es crea un sistema de control semblant al de la figura 4.6, però amb petits canvis. Primer es col·loquen les funcions conegudes *Summation, Transfer Function, Simulation Time Waveform i PID,* dins del *Control & Simulation Loop*. A continuació s'afegeix una segona funció de transferència que representa la realimentació i un segon gràfic. El següent és fer que els guanys del PID i les funcions de transferència s'introdueixin per un terminal, fent clic dret al bloc i després a *Configuration*. Per acabar s'arrossega un nou bloc anomenat *Build Array* per a poder mostrar en un dels gràfics de simulació diferents senyals a la vegada. En concret, en el primer gràfic es representen la consigna, el senyal d'error i la sortida i el segon gràfic mostra el senyal de l'actuador (el que surt del controlador). S'ha fet aquesta divisió perquè els valors que adopta el senyal del controlador solen ser molt diferents a la resta i d'aquesta manera es facilita la visualització.

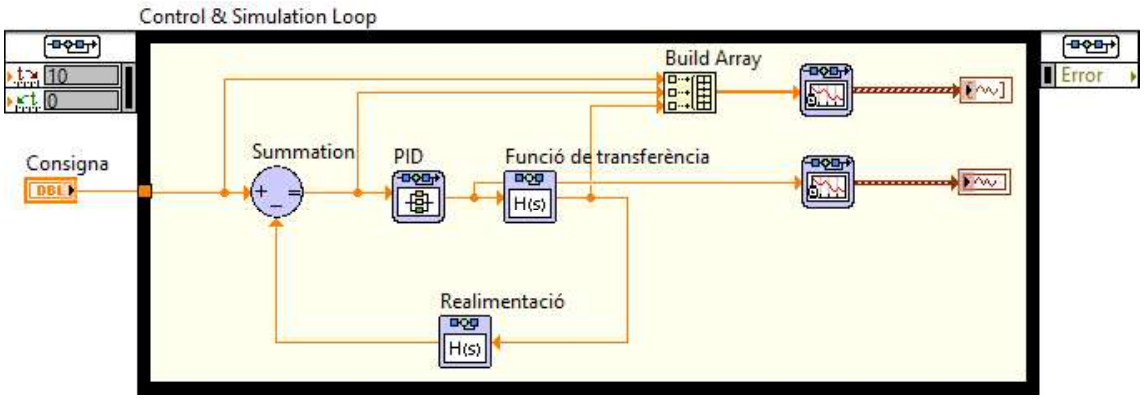


Fig. 4.12: Control & Simulation Loop de qualsevol sistema

Dins del *Control & Simulation Loop* només es col·loquen els blocs destinats a la simulació ja que si es comença a afegir altres blocs la simulació es torna molt més lenta.

4.3.2. Subsistemes VI

En segon lloc es creen diferents subsistemes VI per organitzar i clarificar el programa principal on es fa la interfície gràfica.

Paràmetres planta SubVI

Amb aquest subsistema és pretén que l'usuari introdueixi la planta desitjada i vegi la seva resposta a l'esglaó i l'equació de la funció de transferència. Es comença col·locant una estructura anomenada *Case*, que és similar a la instrucció If existent en llenguatges de programació basats en text. Aquesta estructura té dos o més casos i el *Case Selector* és el que determina quin cas s'executa. Es cableja un valor de tipus enumerat al *Case Selector* per escollir quin tipus de planta es desitja: planta de primer ordre, planta de segon ordre o planta de qualsevol ordre.

- Planta de primer ordre: Dins del *Case* s'afegeix un bloc VI anomenat *CD Construct Special TF Model* i es deixa l'opció per defecte (1st Order). Es creen dos controls que són els paràmetres que caracteritzen un sistema de primer ordre: el guany i la constant de temps. La sortida d'aquest bloc d'una banda es deixa com a dos indicadors (TF1,TF) i d'altra banda es connecta a un segon bloc VI anomenat *CD Draw Transfer Function Equation*, que a la sortida representa gràficament l'equació de la funció de transferència de la planta. Per últim, fora del *Case* es col·loca un bloc VI anomenat *CD Step Response* que a la sortida ploteja la resposta a l'esglaó en llaç obert de la planta.

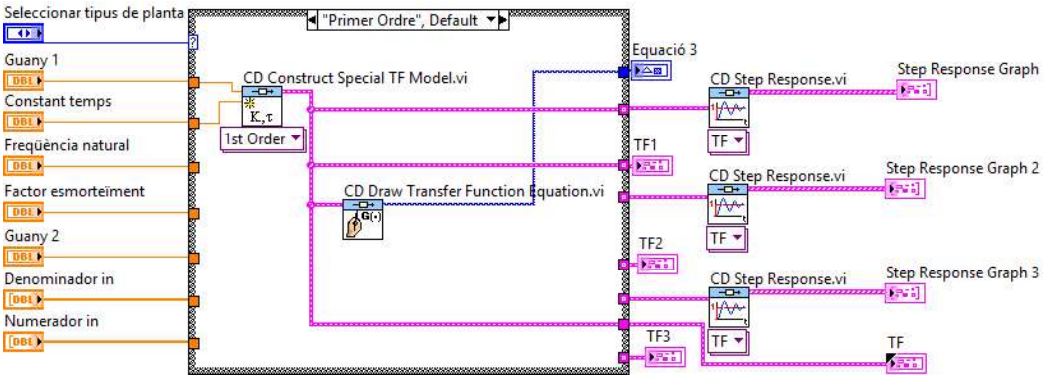


Fig. 4.13: Codi Subsistema Paràmetres planta (Primer Ordre)

- Planta de segon ordre: Es segueixen els mateixos passos, però ara en el bloc VI *CD Construct Special TF Model* es selecciona l'opció 2nd Order i es creen tres controls que són: la freqüència natural, el factor d'esmorteïment i el guany. La sortida d'aquest bloc d'una banda es deixa com a dos indicadors (TF2,TF) i d'altra banda es connecta al bloc VI *CD Draw Transfer Function Equation*. Per últim es representa la resposta a l'esglaó en llaç obert de la planta.

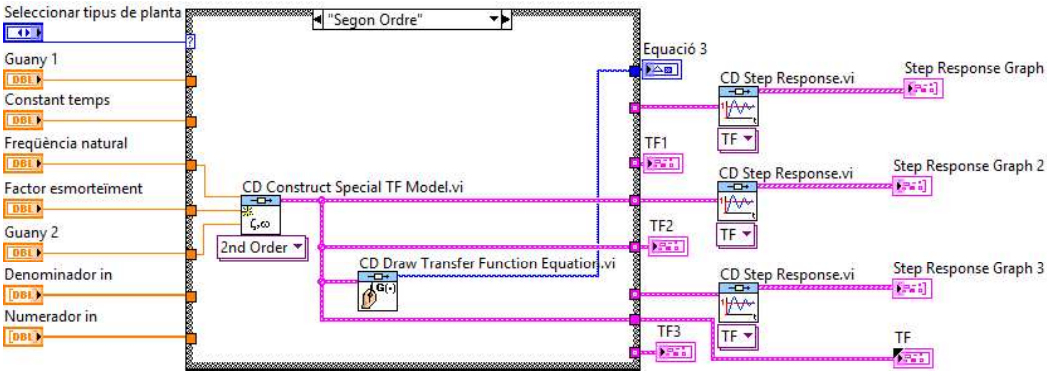


Fig. 4.14: Codi Subsistema Paràmetres planta (Segon Ordre)

- Planta de qualsevol ordre: En aquest cas s'utilitza el bloc VI *CD Construct Transfer Function Model* que crea una funció de transferència a partir de dos controls: numerador i denominador. La sortida d'aquest bloc es torna a deixar com a dos indicadors (TF3,TF) i es connecta al bloc VI *CD Draw*

Transfer Function Equation. Per acabar es representa la resposta a l'esglaó en llaç obert de la planta.

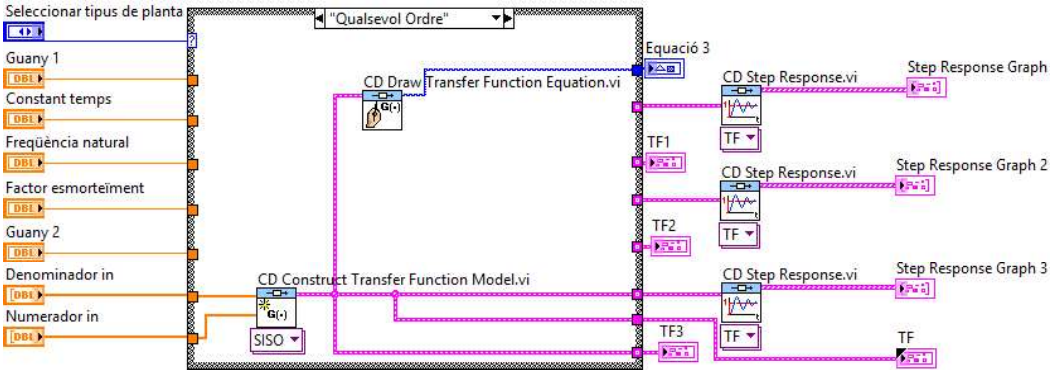


Fig. 4.15: Codi Subsistema Paràmetres planta (Qualsevol Ordre)

Finalment es crea el subsistema clicant a *Edit»Create SubVI.* Es pot comprovar com el subsistema té 8 entrades i 8 sortides.

Paràmetres controlador SubVI

Aquest subsistema s'encarrega de que l'usuari introdueixi els guanys del PID i vegi l'equació de la funció de transferència del controlador. Es comença afegint un bloc VI anomenat *CD Construct PID Model* que construeix la funció de transferència del PID. Es selecciona l'opció de *Parallel (Continuous)* i llavors es creen els controls que són els diferents guanys. Una sortida és l'indicador TFC i l'altre va directa al bloc VI *CD Draw Transfer Function Equation* que mostra l'equació del PID. Per últim es crea el subsistema com s'ha fet abans.

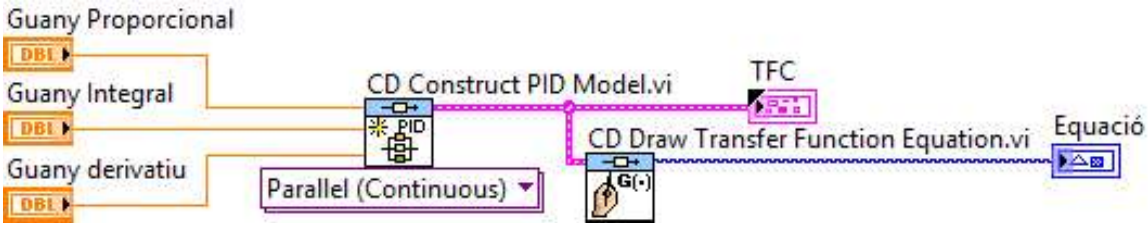


Fig. 4.16: Codi subsistema Paràmetres controlador

Paràmetres realimentació SubVI

Aquest subsistema és el més senzill de tots, s'encarrega de que l'usuari introdueixi el numerador i el denominador de la realimentació i vegi la seva equació. L'únic necessari és el bloc VI anomenat *CD Construct Transfer Function Model* i el bloc VI *CD Draw Transfer Function Equation.*

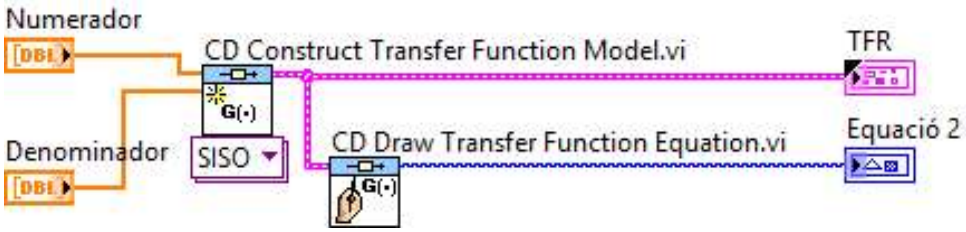


Fig. 4.17: Codi subsistema Paràmetres realimentació

FunciónTransferènciaTotal SubVI

Aquest subsistema calcula la funció de transferència total del sistema, mostra l'equació del sistema i dibuixa el diagrama de pols i zeros del sistema. Aquí es torna a utilitzar l'estructura *Case* que executa diferents ordres segons el tipus de planta que es seleccioni. Dins del *Case* es col·loca el bloc VI *CD Series* que multiplica la funció de transferència del controlador (TFC) per la funció de transferència de la planta seleccionada (TF1 en el cas de Primer Ordre, TF2 en el cas de Segon Ordre i TF3 en el cas de Qualsevol Ordre). A la sortida del *Case* s'afegeix un bloc VI anomenat *CD Feedback* que calcula la funció de transferència total del sistema tenint en compte la realimentació (TFR). Finalment es mostra l'equació de la funció de transferència total del sistema i gràcies al bloc VI *CD Pole-Zero Map* es representa el diagrama de pols i zeros del sistema i s'indiquen els valors dels pols i zeros.

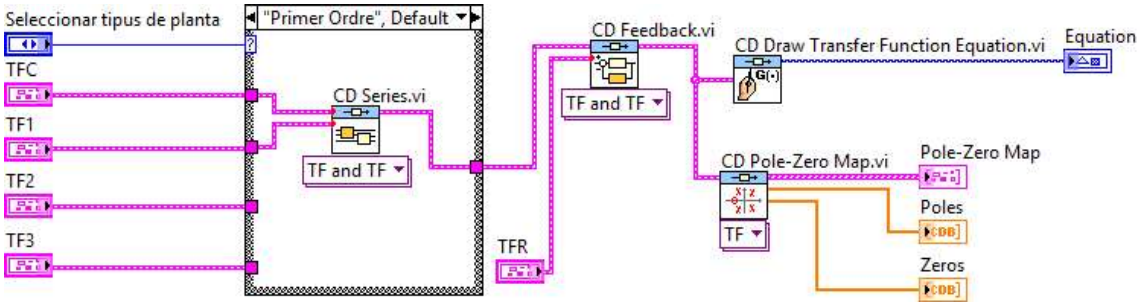


Fig. 4.18: Codi subsistema FunciónTransferènciaTotal

Un cop arribat a aquest punt es comença a donar forma al programa principal. Primer es creen dues estructures *While*. En una s'hi posa la part de simulació i a l'altra tota la resta d'elements. La idea és fer que la simulació s'iniciï quan ja s'han introduït tots els paràmetres necessaris. A la figura 4.19 es poden observar les connexions necessàries entre el *Control & Simulation Loop* i els subsistemes PID, Planta i Realimentació, així com alguns controls que defineixen paràmetres de simulació. Al *While* interior s'ha col·locat el botó d'Iniciar, en prémer el botó es surt del bucle interior i comença la simulació en el bucle exterior. Al While exterior es col·loca una constant FALSE perquè la simulació no s'aturi.

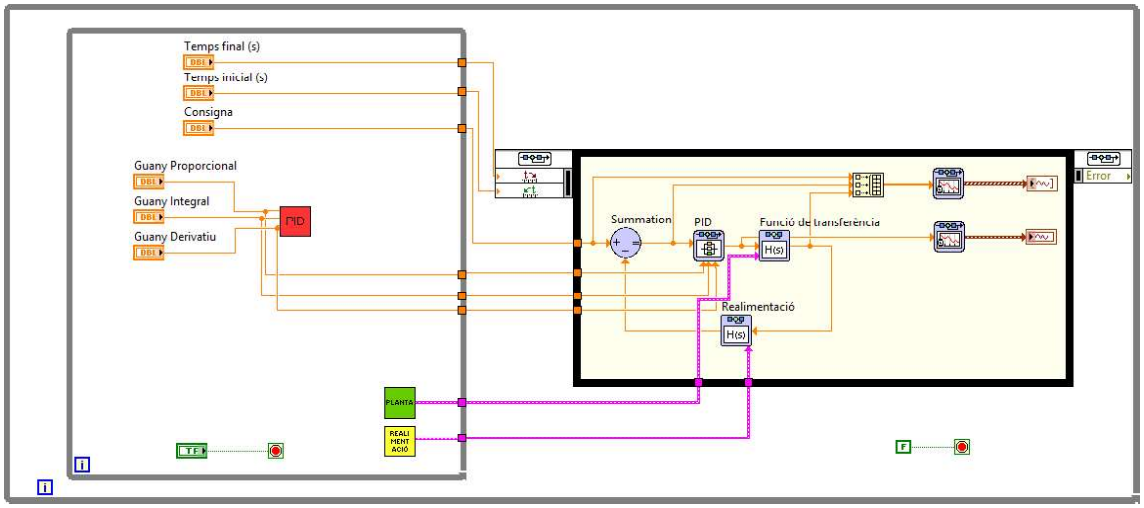


Fig. 4.19: Connexions Subsistemes-Control & Simulation Loop

4.3.3. Disseny del panell frontal

Per a dissenyar la interfície gràfica d'usuari es comença a treballar al panell frontal. En primer lloc s'introdueix el títol i se li dona la forma desitjada. Per donar color es fa servir la *Tools Palette* i per introduir títols, línies i requadres cal anar a *Controls Palette»Decorations.* Com es pot veure a la figura 4.20, es divideix en tres requadres: el primer de tot es destina a la planta, el gran de la dreta mostra les gràfiques de simulació i l'últim serveix pel controlador, la realimentació i les característiques generals del sistema.

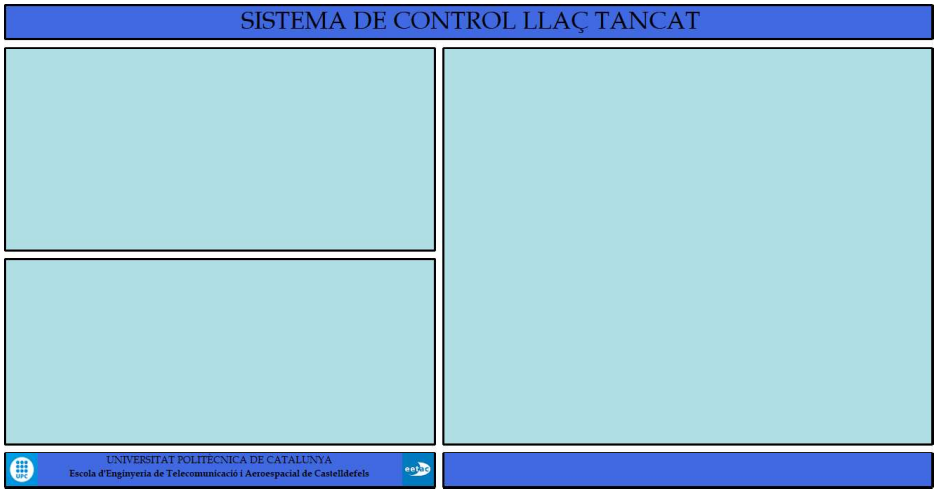


Fig. 4.20: Disseny inicial de la GUI a LabVIEW

Per estalviar espai es fa servir el *Tab Control* que permet afegir les pestanyes que siguin necessàries. A més, es creen dos nous botons a part del ja comentat. Per aturar la simulació es col·loca dins del *Control & Simulation Loop* un bloc anomenat *Halt Simulation* que va connectat a un botó anomenat aturar. L'altre serveix per tancar LabVIEW i es connecta a un *Case* dins del qual s'afegeix un missatge que pregunta a l'usuari si està segur de tancar. En cas afirmatiu surt del *Case* i es connecta a un *Application Control* que tanca LabVIEW.

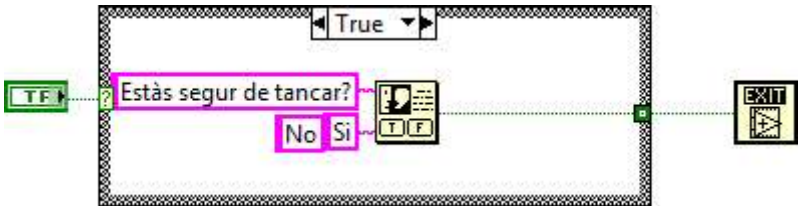


Fig. 4.21: Codi botó tancar

Un cop dissenyat el panell frontal, amb els corresponents controls i indicadors, s'han de dur a terme les connexions en el diagrama de blocs per a que el programa es pugui executar. A continuació es pot veure el resultat final del panell frontal.

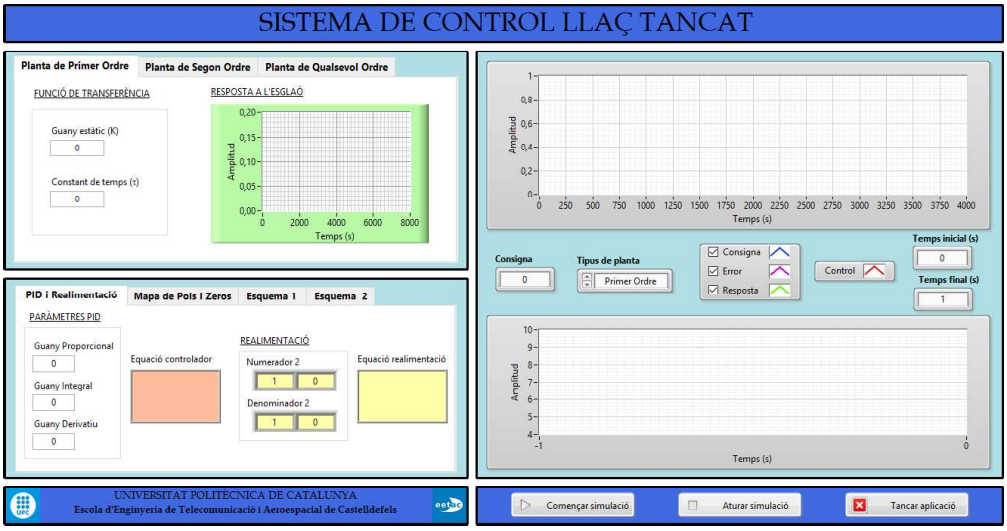


Fig. 4.22: Disseny final de la GUI a LabVIEW

Per netejar els gràfics de forma programàtica cada vegada que s'executa el programa es fa servir un *Property Node*, fent clic dret al gràfic seleccionant *Create»Property Node»History Data*. Un cop creat, es fa clic dret, es selecciona *Change to Write* i es crea una constant que val 0. Això escriu un vector de dades al gràfic que el neteja al ser executat.

Fig. 4.23: Property Node de les gràfiques

Una altra opció molt útil és la de reiniciar tots els valors dels controls del panell frontal als seus valors per defecte. Primer es defineixen els valors per defecte d'alguns controls com, per exemple, el de la realimentació que per defecte es posa que és 1. Després, es selecciona *VI Server Reference* i *Invoke Node* de la paleta *Application Control.* Es connecta la sortida de la referència de la funció *VI Server Reference* a la referència del *Invoke Node*. Es fa clic dret a la paraula method de l'*Invoke Node* i es selecciona *Methods» Default Values» Reinitialize All to Default.* Finalment de la paleta *Application Control* es selecciona *Close Reference* i es cableja a la sortida del node de propietat.

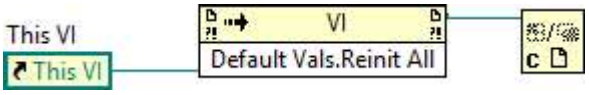


Fig. 4.24: Invoke Node per reiniciar als valors per defecte

Per donar per finalitzada aquesta interfície gràfica d'usuari es crea una aplicació mitjançant la qual es pugui executar el programa sense haver d'obrir LabVIEW. Per fer-ho es clica a *Tools* i a *Build Application (.EXE) from VI* i automàticament es crea un projecte nou i el respectiu executable.

4.4. Aplicació a Matlab

En aquest apartat és reprodueix el programa creat anteriorment amb LabVIEW, però mitjançant Matlab. És evident que no es pot recrear exactament la mateixa interfície però la finalitat és la mateixa: simular sistemes de control en llaç tancat i jugar amb els components que el formen (controlador, planta i realimentació).

4.4.1. Simulink

Es comença utilitzant Simulink per a simular un sistema de control amb realimentació i un controlador PID. Primer de tot, s'afegeixen dos blocs de funció de transferència (un per la planta i l'altre per a la realimentació), el bloc de controlador PID, el bloc comparador i una constant que representa la consigna del sistema. Un cop feta la unió d'aquests blocs, es col·loca un bloc anomenat *Mux* que combina les seves entrades per formar finalment un vector. Això és molt útil si es vol representar gràficament més d'un senyal. A continuació es col·loquen dos blocs anomenats *Scope* que permeten representar els senyals. El *Scope* connectat al *Mux* mostra el senyal de consigna, el senyal d'error i el senyal de resposta i el segon *Scope* mostra el senyal del controlador. Per acabar l'esquema es seleccionen quatre blocs anomenats *To Workspace* que agafen els diferents senyals i escriuen les dades en el workspace.

Ara que ja es té l'esquema muntat es canvien els valors dels diferents blocs per variables fent doble clic sobre el bloc. El valor de la constant es canvia per consigna, els paràmetres del controlador es canvien per kp, ki i kd

respectivament, els blocs de funció de transferència es canvien per NumP, DenP, NumR i DenR i els blocs To Workspace per c, e, u i y. Això permet modificar els valors dels blocs de Simulink des de Matlab. Finalment per a poder mostrar el gràfic del *Scope* en un plot a Matlab s'ha d'accedir a *Configuration Properties»Logging* i seleccionar *Log data to workspace*.

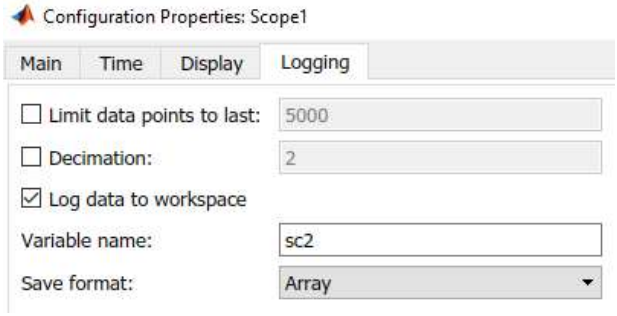


Fig. 4.25: Propietats de configuració del bloc Scope

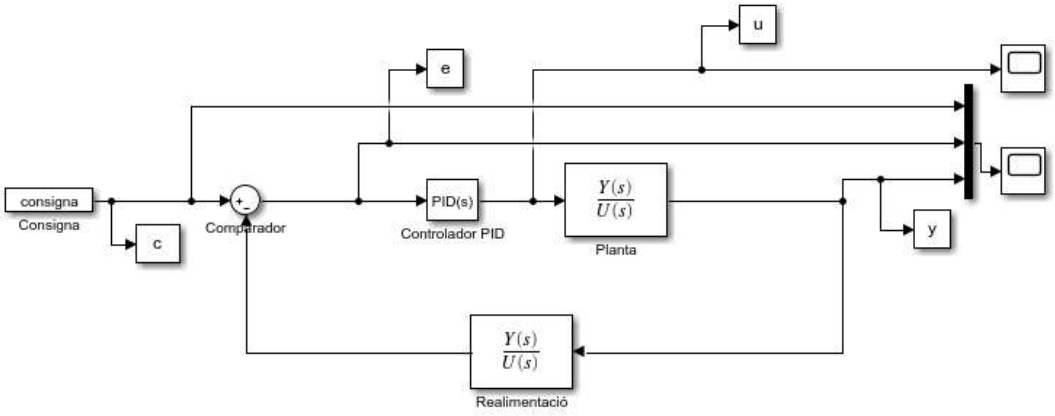


Fig. 4.26: Esquema sistema de control a Simulink

4.4.2. Disseny

Primer es crea un nou *GUI Blank* per començar a editar la interfície gràfica. S'utilitza el control *Panel* per organitzar la interfície en requadres, de forma similar a com s'ha fet a LabVIEW. Per canviar el color es fa doble clic sobre el panell i automàticament s'obre el *Property Inspector* on es pot editar la seva aparença. En aquest cas s'han creat quatre requadres, a més del requadre que conté el títol de la interfície, el requadre amb els logos de la universitat i el requadre de botons a la part inferior. Per mostrar text en qualsevol requadre es fa servir el control *Static Text*, per carregar imatges o bé representar gràfiques s'utilitza el *Axes*, per col·locar botons el *Push Button*, per introduir i editar text el *Edit Text*. Aquests són els més utilitzats però també hi ha d'altres, tots ells es poden personalitzar mitjançant l'editor de propietats. També es pot canviar el nom amb el qual apareix el control a l'arxiu .m, simplement editant el camp *Tag*

de l'editor de propietats. A continuació s'explica més detalladament el contingut dels requadres interiors.

El requadre de l'esquerra està format per un esquema del sistema de control on es poden introduir la planta i la realimentació desitjada. Al centre es troba el panell del controlador que permet escollir els guanys del PID i al costat escriu quin tipus de controlador és. La part inferior consta de dues gràfiques: la resposta a l'esglaó de la planta en llaç obert i el mapa de pols i zeros del sistema. Un dels requadres del centre indica els valors dels paràmetres més destacats de la resposta del sistema i l'altre permet introduir el valor de la consigna, el temps final de simulació, l'opció de mostrar la llegenda i la quadrícula dels gràfics i un botó per guardar les dades en Excel. Finalment, el requadre de la dreta mostra les dues gràfiques obtingudes mitjançant la simulació. A la figura 4.27 es pot veure el disseny de la interfície.

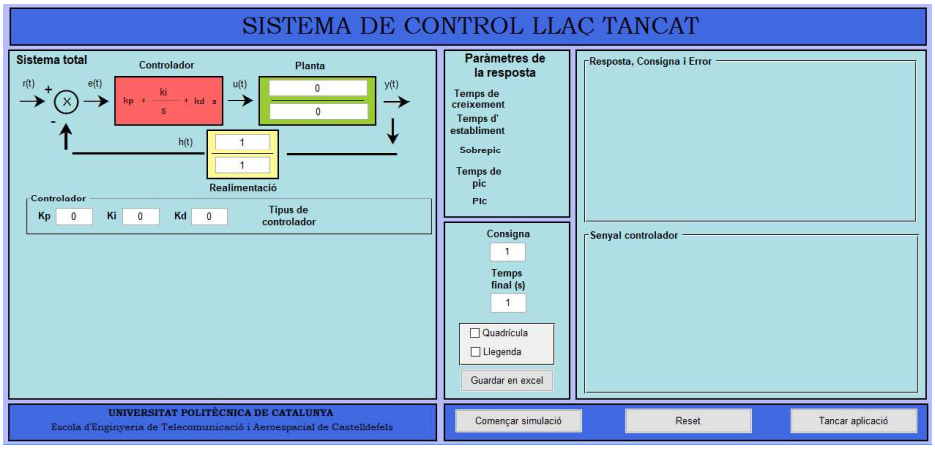


Fig. 4.27: Disseny final de la GUI a Matlab

4.4.3. Codi

Un cop tots els controls estan en posició s'editen les funcions de trucada *(Callback)* de cadascun d'ells, escrivint el codi de Matlab a l'arxiu .m que s'executa quan el control és utilitzat. A continuació s'explica el codi utilitzat per a cada acció.

Imatges

Per mostrar imatges a la interfície gràfica és necessari tenir ubicades les imatges a la carpeta on es troben els arxius .m i .fig del programa. Després s'utilitza la funció *imread()* que llegeix les imatges i la funció *imshow()* que mostra les imatges, en aquest cas als *Axes* corresponents: axes(handles.upc) i axes(handles.eetac). Aquest codi es col·loca a la funció d'obertura del programa que ja està creada per defecte.

```
function GUISistema_OpeningFcn(hObject, eventdata, handles, varargin)
%Mostra imatges en axes
a=imread('upc.png');   %Llegeix la imatge
axes(handles.upc)      %Carrega la imatge en upc
imshow(a);             %Presenta la imatge
```

```
b=imread('eetac.png');
axes(handles.eetac)
imshow(b);
```

Botó inici de la simulació

Un cop es prem el botó de simulació s'han de dur a terme moltes accions. Primer de tot, s'ha de mostrar de quin tipus de controlador es tracta (P, PI, PD o PID) i s'han de mostrar els diferents guanys al requadre vermell del controlador. Per a això s'utilitzen el comandament *get* que serveix per obtenir dades introduïdes per l'usuari i *set* que serveix per assignar dades. La funció *str2double* transforma el valor ingressat per l'usuari de format *string* a format *double* o el que és el mateix, de paraula a número.

```
function botoinici_Callback(hObject, eventdata, handles)

%Mostra el tipus de controlador (P, PI, PD o PID)
if (str2double(get(handles.kp,'String'))~=0 &&
str2double(get(handles.ki,'String'))==0 &&
str2double(get(handles.kd,'String'))==0);
    set(handles.pid,'String','Proporcional P');
end
if (str2double(get(handles.kp,'String'))~=0 &&
str2double(get(handles.ki,'String'))~=0 &&
str2double(get(handles.kd,'String'))==0);
    set(handles.pid,'String','Proporcional-Integral PI');
end
if (str2double(get(handles.kp,'String'))~=0 &&
str2double(get(handles.ki,'String'))==0 &&
str2double(get(handles.kd,'String'))~=0);
    set(handles.pid,'String','Proporcional-Derivatiu PD');
end
if (str2double(get(handles.kp,'String'))~=0 &&
str2double(get(handles.ki,'String'))~=0 &&
str2double(get(handles.kd,'String'))~=0);
    set(handles.pid,'String','Proporcional Integral Derivatiu PID');
end

%Omple el requadre Controlador
set(handles.KP, 'String', str2double(get(handles.kp,'String')));
set(handles.KI, 'String', str2double(get(handles.ki,'String')));
set(handles.KD, 'String', str2double(get(handles.kd,'String')));
```

Per a poder dur a terme la simulació es creen les variables de Simulink que obtenen el valor introduït per l'usuari gràcies al comandament *get*. En aquest cas apareix una nova funció *str2num* que converteix un *string* en un *array*. Per defecte, Simulink busca els paràmetres de les variables en el *base workspace*. Les variables que es creen es troben al *function workspace*, on no està buscant Simulink. Per tant, s'han de traslladar les variables al *base workspace*. Mitjançant la funció *save* es guarden totes les variables del *workspace* actual (el de la funció) en un fitxer .mat i la funció *evalin* carrega totes les variables del fitxer al *base workspace*. La simulació s'inicia amb la funció *sim()*, es crea una matriu amb les dades dels senyals c, e, u i y i finalment es dibuixen els senyals obtinguts de Simulink als *Axes* corresponents.

```
%Paràmetres de la simulació
consigna=str2double(get(handles.consigna,'String'));
kp=str2double(get(handles.kp,'String'));
ki=str2double(get(handles.ki,'String'));
kd=str2double(get(handles.kd,'String'));
NumP=str2num(get(handles.NumP,'String'));
DenP=str2num(get(handles.DenP,'String'));
NumR=str2num(get(handles.NumR,'String'));
DenR=str2num(get(handles.DenR,'String'));
t_stop=str2double(get(handles.tf,'String')); %Temps final de simulació
(s)
save('test.mat','consigna', 'kp', 'ki', 'kd', 'NumP', 'DenP', 'NumR',
'DenR', 't_stop');
evalin('base','load(''test.mat'')')
load_system('Plantaqo'); %Carrega invisiblement el model de Simulink
options=simset('solver', 'ode23'); %Solver utilitzat per a la
simulació
sim('Plantaqo', t_stop, options); %Inici de la simulació
A=[e c y u]; % Es crea una matriu
save('test.mat','A');
evalin('base','load(''test.mat'')')
axes(handles.response);
plot(sc1(:,1), sc1(:,2), 'b',sc1(:,1), sc1(:,3),'m', sc1(:,1),
sc1(:,4),'g', 'LineWidth', 1.5);
xlabel('Temps(s)', 'FontSize',8)
ylabel('Amplitud', 'FontSize',8)
axes(handles.controlador);
plot(sc2(:,1), sc2(:,2), 'r', 'LineWidth', 1.5);
xlabel('Temps(s)', 'FontSize',8)
ylabel('Amplitud', 'FontSize',8)
```

El següent és mostrar la resposta a l'esglaó de la planta i el mapa de pols i zeros del sistema. Per aconseguir-ho es calcula la funció de transferència de la planta amb la funció *tf()* i a continuació s'utilitza la funció *stepplot()* per personalitzar la gràfica de la resposta a l'esglaó. Per obtenir la funció total de transferència del sistema s'han de calcular també les funcions de transferència del controlador PID i de la realimentació. Llavors es multipliquen els blocs en sèrie (planta i controlador) amb la funció *series()* i es calcula el feedback tenint en compte la realimentació. A continuació es fa sevir una funció que torna els paràmetres de la resposta del sistema total i alguns d'aquests es mostren a la interfície. Per últim es representa el mapa de pols i zeros amb la funció *pzplot()* que també permet personalitzar la gràfica.

```
TF=tf(NumP,DenP); %Funció de transferència de la planta
axes(handles.step);
opt = timeoptions;
opt.Title.String='Resposta de la planta';
opt.Title.FontWeight='bold';
opt.Title.FontSize=11;
opt.XLabel.String='Temps';
opt.YLabel.String='Amplitud';
h1=stepplot(TF, opt, 'g'); %Representa la step response
C=pid(kp,ki,kd); %Funció de transferència del controlador
R=tf(NumR,DenR); %Funció de transferència de la realimentació
sys=series(TF,C); %Sèrie del controlador i la planta
systotal=feedback(sys,R); %Funció transferència del sistema
info=stepinfo(systotal); %Info paràmetres resposta sistema
save('test1.mat','TF','info', 'C', 'sys', 'R', 'systotal');
```

```
evalin('base','load(''test1.mat'')')
set(handles.tc, 'String', info.RiseTime);
set(handles.te, 'String', info.SettlingTime);
set(handles.s, 'String', info.Overshoot);
set(handles.tp, 'String', info.PeakTime);
set(handles.p, 'String', info.Peak);
axes(handles.polezero);
opt = pzoptions;
opt.Title.String='Mapa pols i zeros';
opt.Title.FontWeight='bold';
opt.Title.FontSize=11;
opt.XLabel.String='Eix real';
opt.YLabel.String='Eix imaginari';
h2 = pzplot(systotal, opt, 'b'); % Representa el mapa de pols i zeros
```

L'última part del botó d'inici consisteix en desactivar les opcions d'un checkbox que es veu més endavant.

```
%Deshabilita el checkbox
set(handles.checkboxlegend, 'Value', 0);
set(handles.checkboxgrid, 'Value', 0);
```

Botó de reset

El que es pretén amb aquest botó és posar tots els controls, gràfiques, etc, al seu valor inicial mitjançant el comandament *set*.

```
function botoreset_Callback(hObject, eventdata, handles)
set(handles.kp, 'String', 0);
set(handles.ki, 'String', 0);
set(handles.kd, 'String', 0);
set(handles.KP, 'String', 'kp');
set(handles.KI, 'String', 'ki');
set(handles.KD, 'String', 'kd');
set(handles.NumP, 'String', 0);
set(handles.DenP, 'String', 0);
set(handles.NumR, 'String', 1);
set(handles.DenR, 'String', 1);
set(handles.pid, 'String', '');
set(handles.tc, 'String', '');
set(handles.te, 'String', '');
set(handles.s, 'String', '');
set(handles.tp, 'String', '');
set(handles.p, 'String', '');
set(handles.consigna, 'String', 0);
set(handles.tf, 'String', 0);
cla(handles.response);
cla(handles.controlador);
cla(handles.step);
cla(handles.polezero);
set(handles.checkboxlegend, 'Value', 0);
set(handles.checkboxgrid, 'Value', 0);
```

Botó de tancar

En prémer el botó tancar es mostra un missatge a l'usuari preguntant si està segur de tancar i en cas afirmatiu, es tanca el programa.

```
function bototancar_Callback(hObject, eventdata, handles)
opc=questdlg('Estàs segur de sortir?',...
```

```
    'Tancar',...
    'Si', 'No', 'No');
if strcmp(opc, 'Si')
   delete(handles.figure1);
end
return
```

Checkboxes

Quan es selecciona el checkbox de la quadrícula es mostra la quadrícula de les gràfiques de la simulació i quan es selecciona el checkbox de llegenda es mostra la llegenda de la gràfica que té diversos senyals.

```
function checkboxgrid_Callback(hObject, eventdata, handles)
if (get(hObject,'Value'))==1
    axes(handles.response);
    grid on;
    axes(handles.controlador);
    grid on;
else
    axes(handles.response);
    grid off;
    axes(handles.controlador);
    grid off;
end

function checkboxlegend_Callback(hObject, eventdata, handles)
if (get(hObject,'Value'))==1
    axes(handles.response);
    legend('Consigna','Error', 'Resposta')
else
    legend off
end
```

Botó guardar

És l'últim que queda per programar. Aquest botó permet crear un arxiu Excel anomenat DadesSistema.xls amb les dades de les quatre gràfiques del requadre dret.

```
function botoguardar_Callback(hObject, eventdata, handles)
consigna=str2double(get(handles.consigna,'String'));
kp=str2double(get(handles.kp,'String'));
ki=str2double(get(handles.ki,'String'));
kd=str2double(get(handles.kd,'String'));
NumP=str2num(get(handles.NumP,'String'));
DenP=str2num(get(handles.DenP,'String'));
NumR=str2num(get(handles.NumR,'String'));
DenR=str2num(get(handles.DenR,'String'));
t_stop=str2double(get(handles.tf,'String'));
save('test.mat','consigna', 'kp', 'ki', 'kd', 'NumP', 'DenP', 'NumR',
'DenR', 't_stop');
evalin('base','load(''test.mat'')')
load_system('Plantaqo');
options=simset('solver', 'ode23');
sim('Plantaqo', t_stop, options);
A=[e c y u]; % Es crea una matriu
xlswrite('DadesSistema.xls', A);
```

4.5. Resultats

Un cop creades les dues interfícies gràfiques es comprova que els resultats són els mateixos. Per raons de comparació les simulacions numèriques amb LabVIEW i amb Simulink han estat executades utilitzant el mateix mètode de resolució (algorisme de Runge-Kutta d'ordre 23). Es simula per exemple la planta tèrmica juntament amb el controlador PI que s'havia implementat en apartats anteriors i amb realimentació unitària. Es selecciona un temps final de 4000 segons i es prem el botó d'iniciar. Com es pot observar a les figures 4.28 i 4.29 les gràfiques coincideixen.

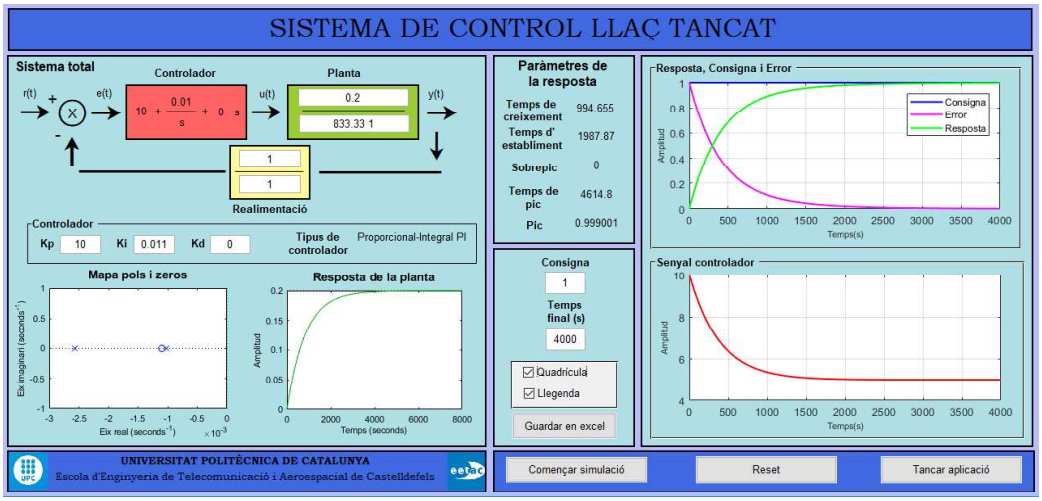


Fig. 4.28: Resposta del sistema a Matlab

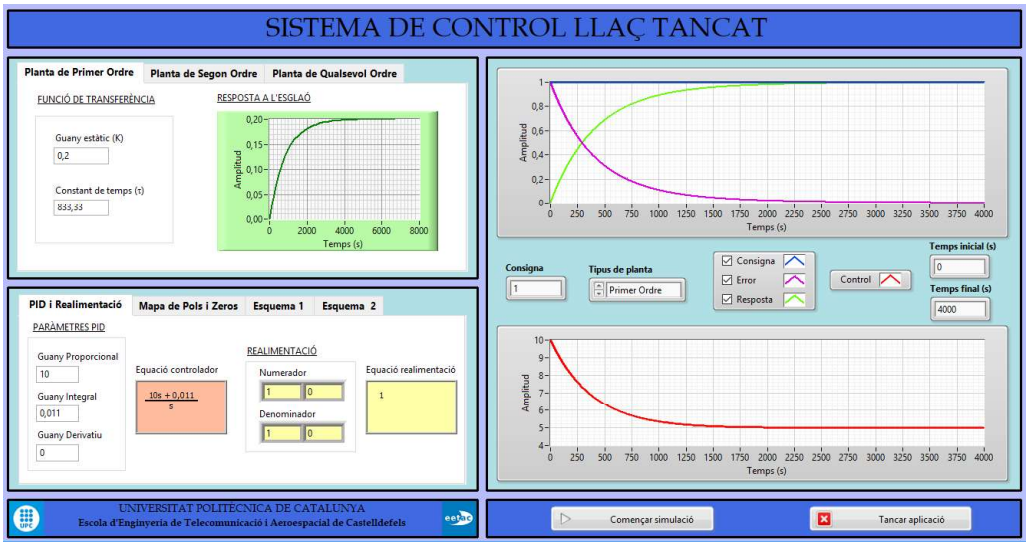


Fig. 4.29: Resposta del sistema a LabVIEW

Pel que fa el senyal de resposta del sistema assoleix el valor de consigna al voltant dels 2000 segons i no té sobreimpuls. El gràfic del senyal del controlador parteix d'una amplitud de 10 i decreix fins a estabilitzar-se també a aproximadament 2000 segons a una amplitud de 5. Gràcies al mapa de pols i zeros es veu que el sistema consta de dos pols simples a s=-0,0026 i s=-0,0010 i un zero a s=-0,0011. Per tant, es pot confirmar que el sistema no és oscil·lant (no té pols complexos), és un sistema estable ja que no té cap pol situat a la part dreta del pla s i és un sistema amb una resposta lenta degut a que els pols estan molt a prop de l'origen.

5. Etapa II: Simulació HIL

En aquesta segona fase s'entra en matèria de la simulació HIL: es reemplaça el PID simulat anteriorment per un PID implementat amb un microcontrolador que controla i actua sobre la planta tèrmica mitjançant un sistema d'adquisició de dades (DAQ). El primer pas és programar el PID amb la placa Arduino UNO, després es prova el funcionament del DAQ USB-6001 i finalment s'agrupen tots els elements per dur a terme la simulació del sistema. A més, tal i com s'ha fet en l'etapa anterior es dissenya una interfície gràfica. Aquesta vegada no es fan servir les dues eines emprades a l'apartat anterior sinó que es decideix efectuar la simulació HIL mitjançant LabVIEW per la seva senzillesa i facilitat a l'hora de fer interfícies gràfiques. De totes maneres, es podria fer servir sense cap problema Matlab/Simulink i s'obtindria el mateix resultat.

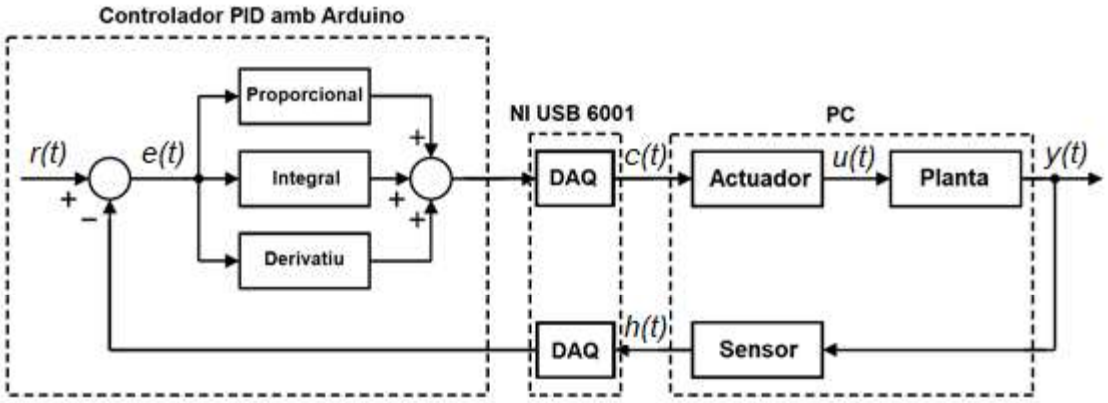


Fig. 5.1: Esquema de la simulació HIL

5.1. Controlador PID amb Arduino

L'objectiu d'aquest apartat és implementar un sistema de control PID basat en una placa Arduino. Aquest PID ha de ser capaç de comparar el valor del senyal de sortida de la planta amb el valor del *setpoint* o consigna per decidir quanta potència ha de subministrar-li a la planta.

La majoria d'automatismes no són capaços de proporcionar una autèntica sortida analògica que variï la seva amplitud de manera continua i Arduino UNO no és una excepció. L'únic que poden proporcionar és una sortida digital de -Vcc o Vcc, en el cas d'Arduino 0 V i 5 V. Per solucionar aquesta limitació s'utilitza una tècnica que consisteix en activar una sortida digital durant un temps i mantenir-la desactivada durant la resta. La mitjana de la tensió de sortida és igual al valor analògic desitjat. Aquesta tècnica s'anomena PWM (Pulse Width Modulation) i està definida per la freqüència que és la quantitat de polsos per segon i

l'anomenat *Duty Cicle*, que determina el percentatge de temps que el pols està actiu.

Per implementar un control PID a la placa Arduino s'ha utilitzat la llibreria PID, que es pot descarregar des de la pàgina web d'Arduino. La llibreria PID conté les següents funcions:

- *PID():* Crea un controlador PID connectat a l'entrada, la sortida i el *setpoint* especificats. L'algorisme PID es troba en forma paral·lela.
- *Compute():* Conté l'algorisme PID i es crida una vegada en cada bucle.
- *SetMode():* Especifica si el PID ha d'estar activat (automàtic) o desactivat (manual).
- *SetOutputLimits():* El controlador PID està dissenyat per variar la seva sortida dins d'un rang determinat. De manera predeterminada, aquest rang és de 0 a 255, però gràcies a aquesta funció es pot modificar.
- *SetTunings():* Els paràmetres de sintonització (kp, ki, kd) determinen el comportament dinàmic del PID. Aquesta funció serveix per canviar aquests paràmetres durant l'execució.
- *SetSampleTime():* Determina la freqüència amb què s'avalua l'algorisme PID. El valor predeterminat és 200 ms.
- *SetControllerDirection():* Aquesta funció especifica quin tipus de procés està connectat amb el PID.

A la pàgina web d'Arduino es poden consultar diferents exemples de PID, en aquest cas ens hem basat en el que porta per nom *RelayOuput*. A continuació es comenta en detall el codi programat. Se li ha afegit un led que s'encén quan el relé està activat i s'apaga quan està desactivat.

```
#include <PID_v1.h> //Inclou biblioteques externes

#define PIN_INPUT 0 //El compilador substitueix el nom PIN_INPUT per 0

#define RELAY_PIN 3

#define LED_PIN 2

//Inicialitza les variables que es connecten al PID

double Setpoint, Input, Output;

//Es defineixen els guanys del PID

double Kp=10, Ki=0,01, Kd=0;

//Es crea un controlador PID en forma paral·lela

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

//Es crea una constant (mida de la finestra)

int WindowSize = 1000; //WindowSize són els mil·lisegons que té cada
cicle PWM

unsigned long windowStartTime; // windowStartTime és el temps en el que
ha començat l'actual cicle PWM

void setup()

{
```

```
  pinMode(RELAY_PIN,OUTPUT); //Configura RELAY_PIN (pin 3) com a sortida
  pinMode(LED_PIN,OUTPUT); //Configura LED_PIN (pin 3) com a sortida
  windowStartTime = millis(); //La función millis()torna el número de
mil·lisegons desde que la placa Arduino va començar a executar el sketch


  //Declara el valor de la consigna
  Setpoint = 500;
  //Indica el rang de la sortida del PID (entre 0 i WindowSize)
  myPID.SetOutputLimits(0, WindowSize);
  //Activa el PID en mode automàtic
  myPID.SetMode(AUTOMATIC);
  Serial.begin(9600); //Configura el port sèrie a 9600 bps
}
void loop()
{
  Input = analogRead(PIN_INPUT); //Llegeix el valor del PIN_INPUT (0)
  Serial.print("IN: ");
  Serial.println(Input); //Envia el valor analògic d'entrada al port
sèrie
  myPID.Compute(); //Executa l'algorisme PID
  Serial.println();
  Serial.print("OUT: ");
  Serial.println(Output); //Envia el valor de la sortida del PID al port
sèrie
//Activa o desactiva el pin de sortida (3) en funció de la sortida del
PID
  if ((millis() - windowStartTime )> WindowSize)
  { //Hora de canviar de finestra
    windowStartTime += WindowSize;
  }
  if (Output < (millis() - windowStartTime)) {
    digitalWrite(RELAY_PIN, LOW); //Diposita en el 'RELAY_PIN' un valor
LOW (0 V)
    digitalWrite(LED_PIN, LOW);
  }
  else {
    digitalWrite(RELAY_PIN, HIGH); //Deposita en el 'RELAY_PIN' un valor
HIGH (5 V)
    digitalWrite(LED_PIN, HIGH);
```

```
  }
  delay(1);
}
```

El *setpoint* es defineix tenint en compte la *Window Size*, per tant en aquest cas el *setpoint* pot adoptar valors des de 0 a 1000 ms. Per saber l'equivalència del *setpoint* en volts es fa servir la següent expressió:

$$\frac{V_{max}\ (V)}{1000\ (ms)} = \frac{V_{desitjada}\ (V)}{Setpoint\ (ms)} \qquad \textbf{(5.1)}$$

El voltatge màxim és el que ens dóna Arduino com a màxim (5 V) i el voltatge desitjat és el que representa la temperatura desitjada. Per exemple, si s'utilitza un sensor LM35 i es vol obtenir una temperatura de 100 ºC, el valor desitjat de voltatge seria 1 V ja que en un LM35 cada grau Celsius equival a 10 mV. A més en aquest cas, el *setpoint* seria la cinquena part de 1000 ms, és a dir, 200 ms.

5.2. Prova de la targeta d'adquisició de dades

Abans de començar a programar el simulador HIL es comprova que la targeta d'adquisició de dades USB-6001 funciona correctament mitjançant un senzill programa a LabVIEW. Primer de tot s'agafa el dispositiu DAQ i es cableja l'entrada analògica 0 amb la sortida analògica 0 tal i com mostra la figura 5.2.

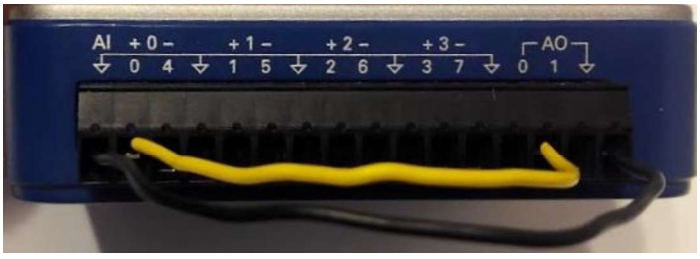


Fig. 5.2: Connexions AI0-AO0

A continuació s'obre LabVIEW i es col·loca un *While Loop* amb el respectiu botó d'aturada. Per accedir als terminals del DAQ només cal col·locar un bloc VI Express anomenat *DAQ Assist* dins de la paleta de *Measurement I/O*. Un cop s'arrossega aquest bloc es pot seleccionar l'opció de generar senyals o bé adquirir-les. En primer lloc es selecciona l'opció *Generate Signals* i seguidament *Analog Output*, *Voltage i ao0* que permet generar un voltatge pel port de sortida analògic 0. Finalment apareix una finestra on es poden configurar diferents

paràmetres com les unitats o el rang del voltatge de sortida. En aquest apartat, es selecciona *1 Sample (On Demand)* per aconseguir que el dispositiu generi una mostra contínuament fins que s'aturi el bucle. En segon lloc es col·loca de nou el bloc *DAQ Assist* i aquesta vegada es selecciona l'opció *Acquire Signals»Analog Input»Voltage»ai0*. Tota la resta és exactament igual. Per acabar el programa és connecta un control (Sortida analògica) al bloc que genera senyals i un indicador (Entrada analògica) al bloc que adquireix el senyal. Per comprovar que el dispositiu DAQ funciona correctament el valor de la entrada analògica ha de coincidir amb el valor que s'introdueixi manualment a la sortida analògica. Es posa en marxa el programa i s'entra un valor de 4 a la sortida analògica. Com es veu a la imatge inferior l'indicador entrada analògica ens mostra el mateix valor.

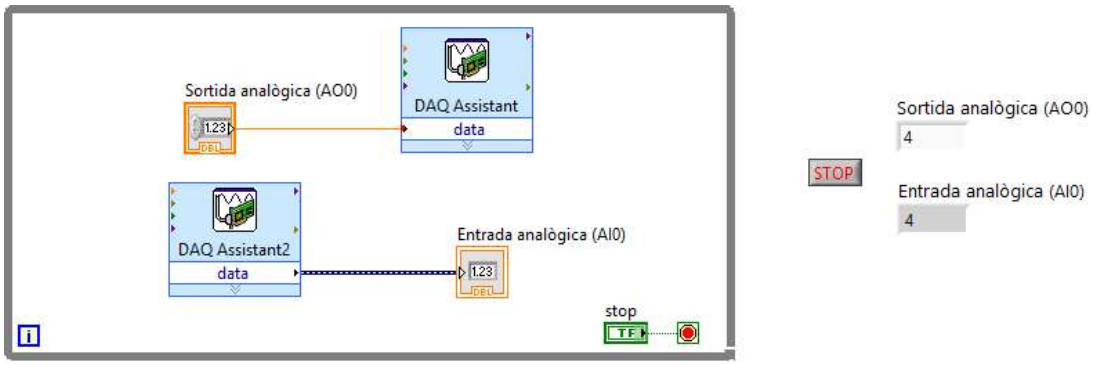


Fig. 5.3: Prova del DAQ a LabVIEW

5.3. Aplicació a LabVIEW

Per programar la simulació HIL s'ha de tenir molt clar el sistema. A continuació s'il·lustra un esquema gràfic de com es fan les connexions entre els diferents blocs.

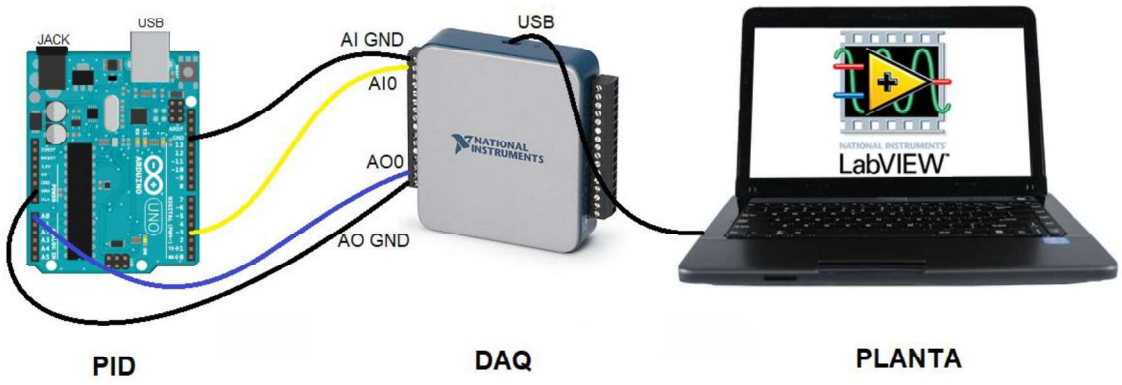


Fig. 5.4: Cablejat entre el PC, el DAQ i l'Arduino

El primer element d'esquerra a dreta és el controlador PID. L'Arduino pot estar alimentat mitjançant el cable USB connectat a l'ordinador o bé utilitzant una font d'alimentació externa connectada al jack i, un cop alimentat, es carrega el programa desitjat a la placa pel port USB. El següent element és la targeta d'adquisició de dades que, per una banda es connecta a la planta simulada a l'ordinador mitjançant el cable USB i per l'altra al controlador. La sortida de la planta passa pel cable USB fins al dispositiu DAQ que converteix el valor digital en analògic (AO0) i el transporta fins al pin analògic 0 de l'Arduino. Aquest executa l'algorisme del PID, compara el valor que li arriba amb el *setpoint* i calcula el senyal de control que ha de proporcionar. Com s'ha vist a l'apartat 5.1, s'utilitza el pin digital 3 per crear un senyal PWM que oscil·la entre 0 V i 5 V. El pin 3 es connecta a l'entrada analògica (AI0) del dispositiu DAQ que la converteix en digital per a que pugui ser interpretada per l'ordinador i arriba a la planta mitjançant de nou el cable USB.

Un cop entès el funcionament, es pren el programa creat a l'apartat 4.3 com a punt de partida, ja que l'estructura general és molt semblant. Es comença esborrant l'interior del *Control & Simulation Loop* i el bucle interior. Primer es col·loquen un bloc *DAQ Assist* per a generar senyals i un altre per adquirir senyals tal i com s'havia fet a l'apartat 5.2. Aquests blocs van connectats als anomenats *Convert From Dynamic Data* i *Convert To Dynamic Data* que es troben a la paleta de *Signal Manipulation.* A continuació s'arrossega el bloc de funció de transferència i s'introdueix la planta tèrmica que es vol simular. Per últim, s'afegeixen els gràfics que mostren el senyal actuador i el senyal de sortida de la planta.

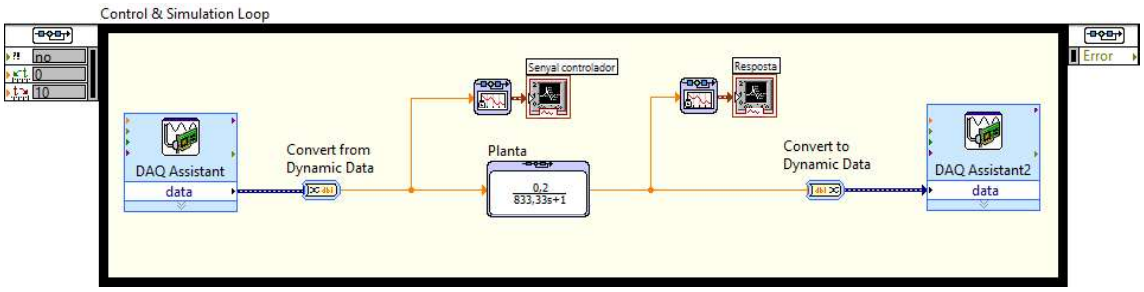


Fig. 5.5: Control & Simulation Loop del HIL

Per defecte el *Control & Loop Simulation* s'executa el més ràpid possible. Per al correcte funcionament de la simulació HIL és necessari que s'executi el *Simulation Loop* com si la simulació s'executés en temps real. Això s'aconsegueix configurant els paràmetres de simulació. En primer lloc es selecciona un mètode de resolució numèrica d'equacions diferencials de pas fix com per exemple el *Solver Method Runge-Kutta 1 (Euler)* i es selecciona un *Step Size (s)* de 0,01. A continuació s'accedeix a la pestanya paràmetres de temps, s'habilita l'opció *Synchronize Loop to Timing Source* i automàticament es marca el *Timing Source* a 1kHz Clock. Finalment s'especifica el *Period* i es relaciona amb el *Step Size.* La unitat del període canvia depenent de la font de temps que es seleccioni en el *Source Type*. El valor per defecte és 1000 ms, però es canvia a 10 ms, de forma que sigui equivalent als 0,01 s del *Step Size* i per tant, s'executi

la simulació en temps real. Cal dir que el Step Size especifica l'interval entre les vegades que el *ODE Solver* avalua el model i actualitza la sortida i el *Period* especifica la quantitat de temps que transcorre entre dues iteracions consecutives del *Control & Simulation Loop.* Si el *Period* fos el doble del *Step Size*, la simulació s'executaria a la meitat del temps real i, al contrari, la simulació s'executaria el doble que el temps real.

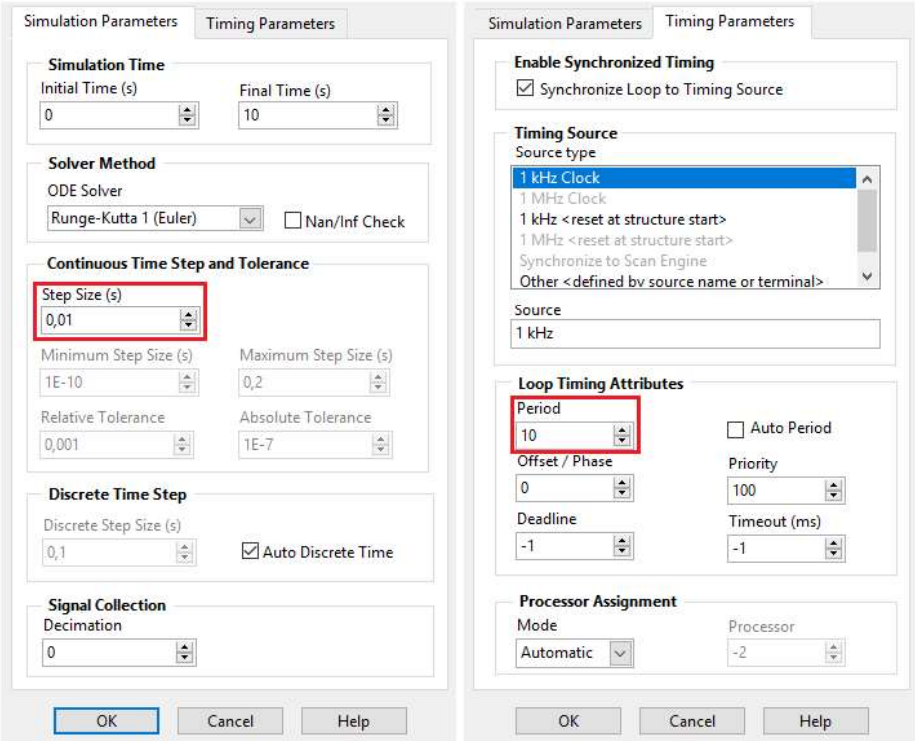


Fig. 5.6: Paràmetres de la simulació

Es posa en marxa el programa de la figura 5.5 amb tot el hardware necessari i s'executa. Una cosa important a tenir en compte és el botó RESET de la placa Arduino. Aquest botó serveix per a que el programa que té carregat comenci a executar-se des de l'inici novament i triga aproximadament 1 segon en córrer el programa. Per tant, abans de començar la simulació és imprescindible prémer el botó de RESET i esperar 1 segon. Això evita que el control integral acumuli valors molt alts deguts a un error massa elevat durant molt de temps (efecte *windup*). En aquest cas el sistema es satura, el control integral no pot fer la seva funció i el resultat de la simulació HIL no és l'esperat.

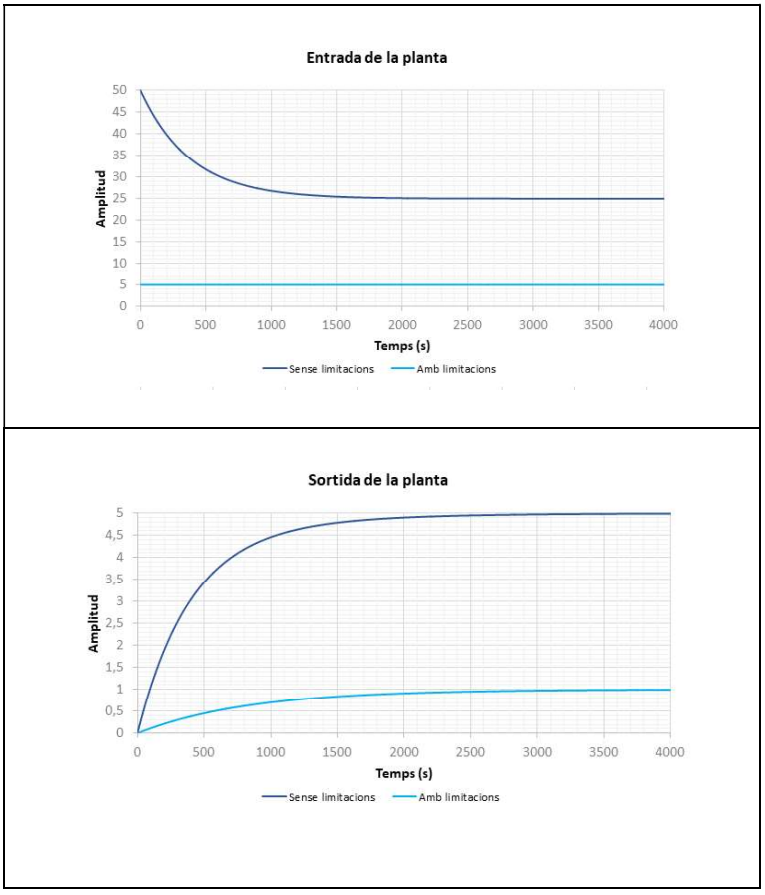
Quan s'executa per primera vegada s'observa que el senyal del controlador que hauria d'oscil·lar entre 0 V (quan està desactivat) i 5 V (quan està activat), no té valors exactes. Això pot ser degut a diversos factors com per exemple l'error o soroll de quantificació produït pel convertidor analògic-digital, que el port USB de l'ordinador mai dóna exactament 5 V, etc. També s'observa que si es toca el dispositiu o els cables varia el senyal del controlador. Per aquest motiu es col·loca un comparador al programa anterior per a que si el senyal és menor de 2,5 adopti el valor de 0 i en cas contrari, el valor de 5. A més, s'afegeix un botó que permet aturar la simulació quan es desitgi gràcies al bloc *Halt Simulation*.

Arribats a aquest punt, es dissenya una simple interfície gràfica on es pugui veure el senyal del controlador i el senyal de resposta de la planta, partint de la base de la creada en l'apartat anterior. Cal esmenar que tot està normalitzat, és a dir, no hi ha unitats i a la sortida de la planta no s'ha d'utilitzar cap sensor. Per obtenir unitats de temperatura (ºC) a la sortida de la planta s'hauria de col·locar un factor per a que l'entrada de la planta fos potència (W).

5.4. Saturació i límits del controlador PID

En els sistemes reals existeixen limitacions que redueixen la capacitat del controlador per aconseguir la resposta desitjada. Si s'aplica al sistema en llaç tancat un senyal de referència esglaó unitari r(t)=5 produeix una demanda instantània de 50 a la sortida del controlador, però el controlador implementat amb Arduino dóna com a màxim un valor de 5, per tant entra en saturació. En el disseny lineal del controlador que s'havia dut a terme en el quart capítol no s'havia tingut en compte aquest fenomen. Per fer quadrar la simulació amb la realitat es col·loca un bloc anomenat *Saturation* a la sortida del bloc *PID* i es defineix un límit inferior de 0 i un límit superior de 5. A continuació es mostren dues gràfiques on es pot comprovar la diferència que hi ha si es tenen compte els límits d'actuació o no.

Taula 5.1: Respostes amb i sense els límits d'actuació



Comparant la resposta del sistema amb i sense limitacions es veu que degut a la limitació a l'entrada de la planta s'obté una pitjor resposta que no assoleix el valor de referència desitjat. De fet, si el valor de la consigna és superior a 1 la resposta mai arriba al valor de consigna, es queda estancat a 1.

5.5. Resultats

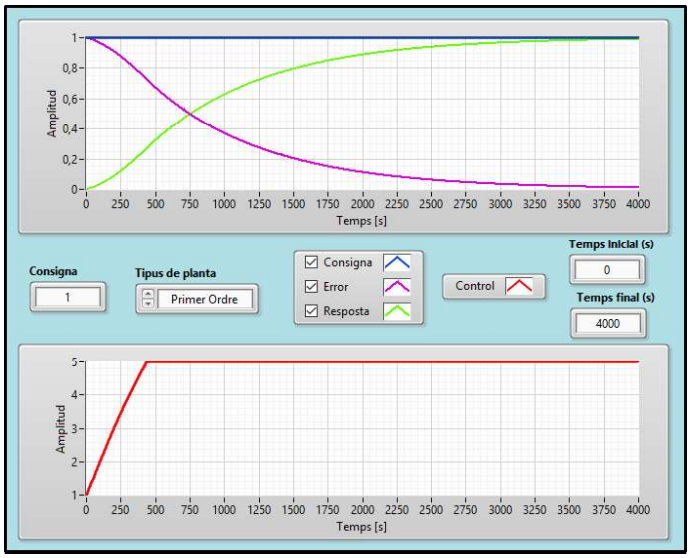
En aquest apartat s'analitza el funcionament del resultat global del projecte, és a dir, del banc de proves basat en la simulació HIL. A continuació es proven diferents valors de consigna i de PID per a la mateixa planta. Els gràfics obtinguts es comparen amb els obtinguts gràcies al simulador creat a la primera etapa. Primer es presenta el cas en el que el PID es satura i després el cas on el PID no es satura en cap moment.

Controlador que es satura

Fent servir el simulador s'han buscat uns guanys per al PID i un valor de consigna específics per a que el controlador entri en saturació. A la taula de sota es mostren aquests valors.

Taula 5.2: Valors PID saturat

PID			Consigna	Setpoint
K_p	K_i	K_d	1	200
1	0,011	0		



A la figura següent es mostren les gràfiques obtingudes amb el simulador. Pel que fa la resposta es pot observar que presenta una petita oscil·lació al principi i que assoleix el valor de consigna als 3400 segons. Pel que fa el controlador PID parteix d'1 i augmenta ràpidament el seu valor fins a quedar saturat una mica abans dels 500 segons.

Fig. 5.7: Simulació PID saturat

A continuació es mostra la resposta obtinguda amb el simulador HIL (Blava) i es compara amb la resposta simulada (Verda). Es pot comprovar com les dues respostes són idèntiques.

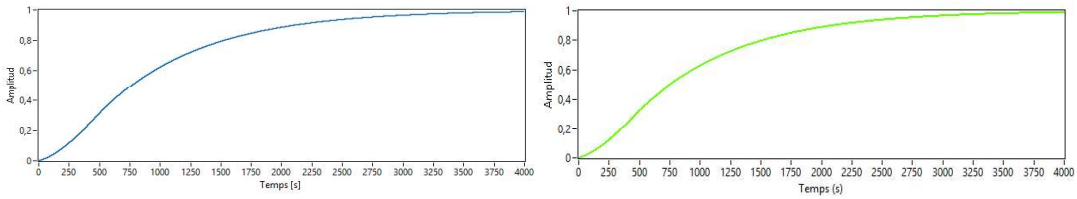


Fig. 5.8: Comparació resposta HIL i simulada

De fet, si s'exporten les dades a Excel i es superposen ambdues gràfiques la diferència és pràcticament inexistent com es pot veure a la figura 5.9.

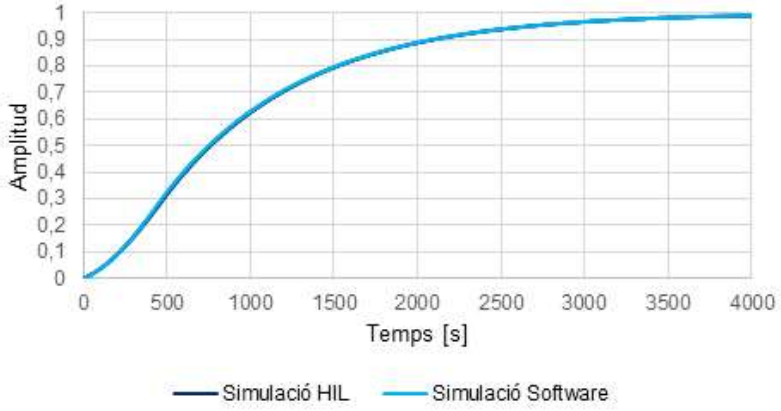


Fig. 5.9: Resposta Simulada VS Resposta HIL

Si es superposen els valors que prenen el controlador PID Simulat i el PID amb Arduino sí que es veu una lleugera diferència cap al final.

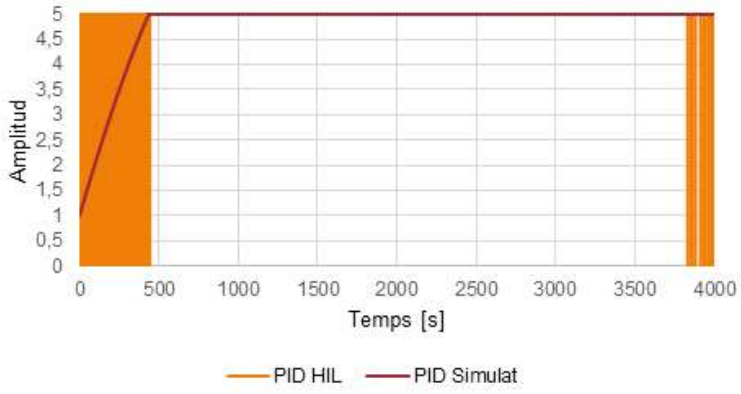


Fig. 5.10: PID Simulat VS PID HIL

En aquesta gràfica no es pot apreciar però, com s'ha explicat anteriorment, el PID HIL representa un senyal PWM que està un temps activat (5) i la resta del temps desactivat (0) en cada cicle i la mitjana equival al valor analògic del PID simulat. Es veu que durant aproximadament els primers 500 segons el senyal PID HIL adopta els valors de 5 i 0, depenent del valor mitjà que es vol aconseguir. En canvi, una mica abans dels 500 segons, el controlador es satura (no pot donar més de 5) i el PID HIL només adopta el valor de 5, això vol dir que el senyal PWM està tot el cicle activat, i el valor mitjà és 5. Finalment a la darrera part de la simulació es pot apreciar com el senyal PWM no es manté tota l'estona a 5. Ampliant aquesta zona es pot veure com només en determinats cicles s'adopta el valor de 0, però durant un temps mínim, el que vol dir que el valor mitjà serà gairebé de 5. Les figures que hi ha a sota mostren el senyal del PID HIL i la resposta HIL en el interval que va dels 3820 segons als 4000. Com que s'ha fet un zoom, a la gràfica de la resposta HIL es pot apreciar com es produeix una baixada mentre el PID està a 0 i una pujada quan està a 5.

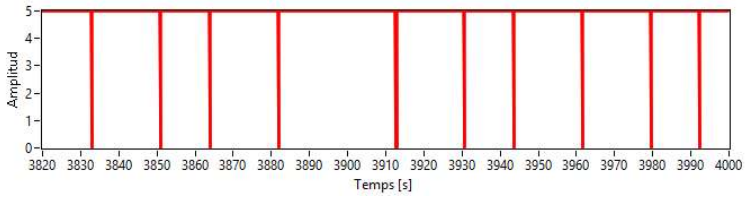


Fig. 5.11: PID HIL interval 3820 s-4000 s

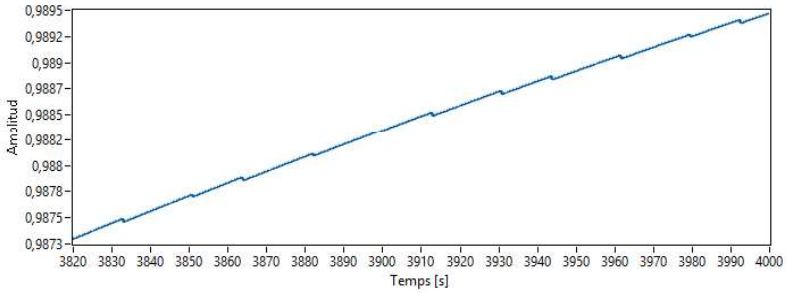


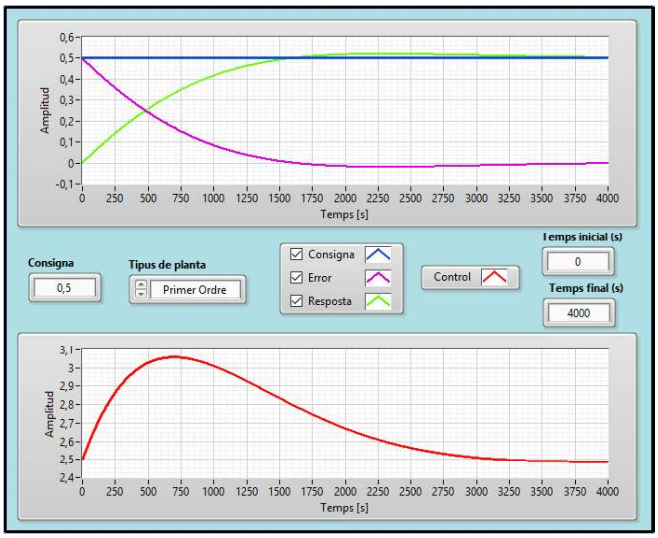
Fig. 5.12: Resposta HIL interval 3820 s-4000 s

Controlador que no es satura

Aquesta vegada s'han buscat uns guanys per al PID i un valor de consigna específics per a que el controlador no entri en cap moment en saturació. A la taula de sota es mostren aquests valors.

Taula 5.3: Valors PID no saturat

PID			Consigna	Setpoint
K_p	K_i	K_d	0,5	100
5	0,01	0		



A la figura 5.13 es mostren les gràfiques obtingudes amb el simulador. Pel que fa la resposta es pot observar com oscil·la una mica abans d'estabilitzar-se, presentant un lleuger sobreimpuls als 2250 segons. Pel que fa el controlador PID no arriba a saturar-se mai, parteix de 2,5 i oscil·la fins a gairebé 3,1 per finalitzar un altre cop a 2,5.

Fig. 5.13: Simulació PID no saturat

A continuació es mostra la resposta obtinguda amb el simulador HIL (Blava) i es compara amb la resposta simulada (Verda). Es pot comprovar com, de nou, les dues respostes són idèntiques.

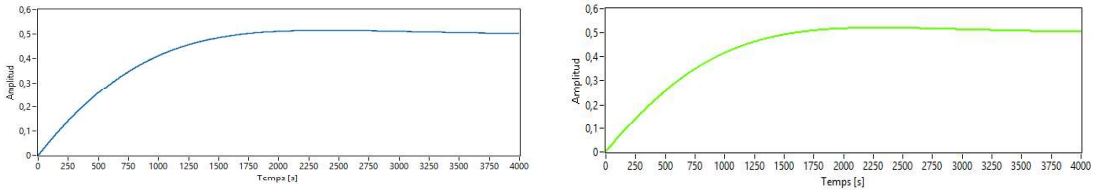


Fig. 5.14: Comparació resposta HIL i simulada

Exportant les dades a Excel i superposant ambdues gràfiques es veu com la diferència és molt petita.

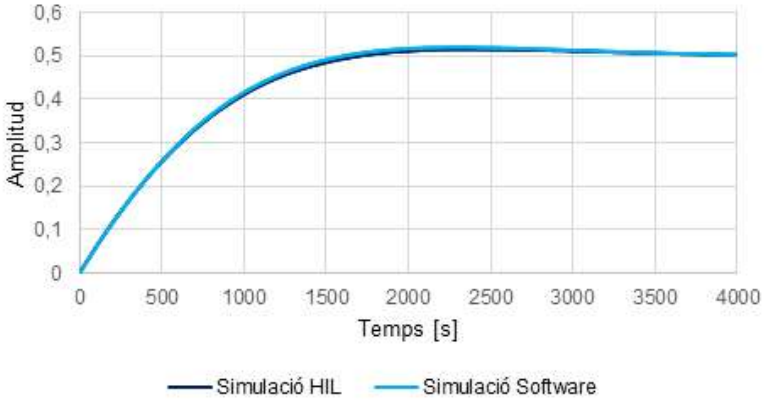


Fig. 5.15: Resposta Simulada VS Resposta HIL

En aquest cas com que el PID mai es satura es pot calcular el valor mitjà d'algun cicle del senyal PWM per comprovar si coincideix amb el valor analògic del PID simulat. Es selecciona l'interval en el qual es produeix el valor mínim del PID i s'analitzen els paràmetres d'un cicle. Mesurant és veu que el cicle complert és d'aproximadament 1000 ms, tal i com s'havia definit la Window Size i que està actiu aproximadament 492 ms, és a dir, el valor mitjà sería 2,46. Comparant-lo amb la gràfica 5.17 es comprova com el valor s'apropa al valor del PID simulat.

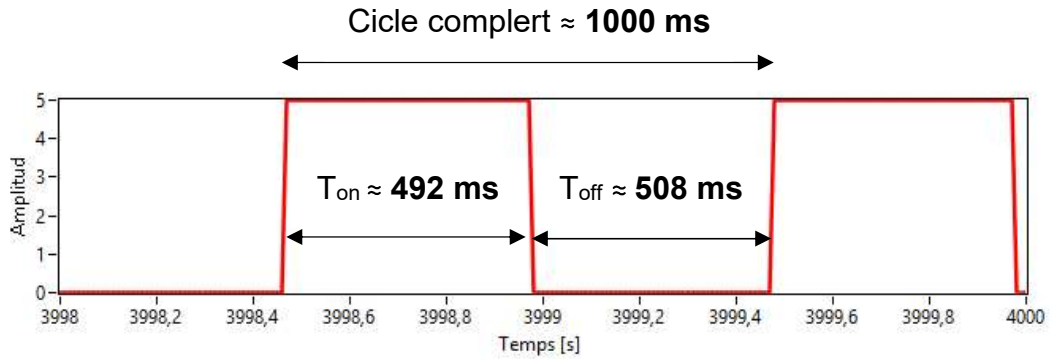


Fig. 5.16: PID HIL interval 3998 s-4000 s

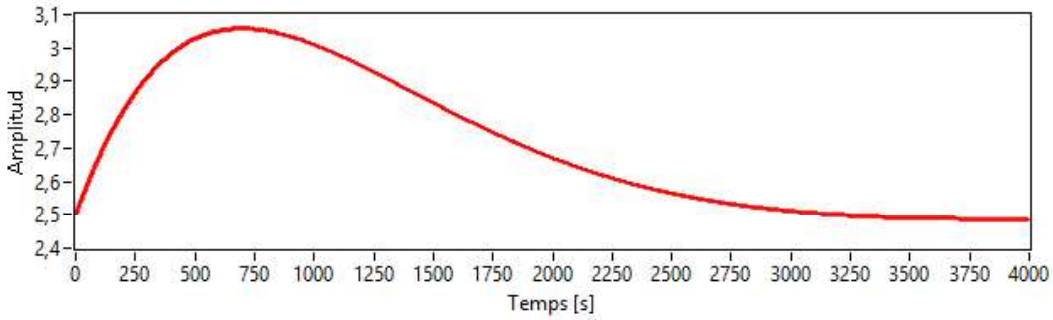


Fig. 5.17: PID Simulat

Per tant s'ha pogut comprovar que el simulador HIL funciona correctament, permet simular qualsevol sistema de control i és prou acurat. A partir d'aquí, es pot analitzar el sistema en profunditat, buscar els valors més òptims del PID, afegir pertorbacions per veure com respon la plana, etc.

6. Conclusions

El resultat final d'aquest projecte ha estat un banc de proves programat amb LabVIEW que permet la simulació HIL per a estudiar el funcionament de sistemes de control (hardware) aplicats a qualsevol planta que es pugui modelar a partir de la seva funció de transferència. Però prèviament també s'ha creat un simulador que permet estudiar el comportament del sistema de control PID que es vulgui. Per tant, el conjunt d'ambdues eines és útil tant per a un públic professional, que pot anar més enllà canviant el tipus de planta, com per a un públic més principiant que estigui aprenent els fonaments del control de sistemes. Pel que fa el funcionament del banc de proves s'ha comprovat que compleix la seva funció correctament i que la resposta de la simulació software coincideix amb la resposta obtinguda introduint el controlador hardware al llaç de simulació (HIL).

Pel que fa la tècnica de la simulació HIL, proporciona un escenari segur i eficient on l'enginyer es pot focalitzar en la funcionalitat del controlador sense prendre cap risc ni fer malbé la planta. A més, permet detectar i diagnosticar falles en la fase inicial del cicle de desenvolupament del producte, que es tradueixen en un estalvi de costos. Per tant, tot i que té el seu origen a l'àmbit aeronàutic hem comprovat que pot ser una eina força útil en altres camps de la enginyeria i de la indústria.

Tant LabVIEW com Matlab són eines molt completes on el nivell de detall el marca la duració del projecte i no les limitacions del programa. Pel que fa la part de control s'han trobat molt poques diferències entre el mòdul *Control Design and Simulation* de LabVIEW i Simulink, ja que ambdues consten dels blocs necessaris per a construir qualsevol esquema de sistema de control. A més, el funcionament és molt similar, per no dir que és idèntic. En relació a la part visual és més atractiu LabVIEW, ja que fa servir colors. Quan interessa fer simulacions amb una interfície gràfica d'usuari o bé és necessària la interacció amb un dispositiu hardware, s'ha observat que LabVIEW és més pràctic i està més destinat a aquest propòsit. Matlab no optimitza l'espai, ja que no permet col·locar pestanyes, tampoc permet copiar imatges, sinó que s'han de carregar. A més, s'ha de programar manualment l'acció que es durà a terme quan es prem un botó.

Fent una valoració més personal diria que valoro molt positivament l'aprenentatge aconseguit buscant la solució a les dificultats i problemes que han sorgit durant el transcurs del projecte. Sobretot m'he adonat de la dificultat que suposa aplicar a problemes reals els coneixements teòrics. Quan es treballa amb sistemes reals no tot és tan fàcil com en els models teòrics ideals. Em refereixo per exemple a limitacions que presenten els controladors reals, com ara la saturació de la seva sortida. L'experiència ha estat molt positiva: m'ha permès ampliar els coneixements de LabVIEW i Matlab, manipular un hardware totalment nou per a mi com és una targeta d'adquisició de dades i conèixer la simulació Hardware-In-the-Loop de la qual no havia sentit mai a parlar.

7. Projectes futurs

Aquest projecte pot servir com a punt de partida per a nous projectes relacionats amb aquesta temàtica. A continuació s'introdueixen algunes propostes.

- **Verificació de la simulació HIL en un sistema real**

  En aquest projecte no s'ha abordat l'últim pas de la simulació HIL que consisteix en col·locar el controlador en una planta real i comprovar el seu funcionament. Per tant, és un treball que es podria realitzar fent servir una planta real.

- **Optimització del control PID**

  L'objectiu d'aquest projecte no era crear un control PID robust i optimitzat, però pot servir de base per a dissenyar i implementar millores per aconseguir un bon control PID d'aquesta planta tèrmica, que tingui en compte per exemple l'efecte *windup*,

- **Sistemes i entorns més complexos**

  En aquest document s'ha volgut mostrar en què consisteix la simulació Hardware-In-the-Loop i per això s'ha utilitzat una planta senzilla. Per tant, es proposa com un possible treball futur utilitzar el mètode de la simulació HIL per estudiar i analitzar sistemes més complexos representatius de situacions reals.

- **Comparació entre LabVIEW i Simulink per a aplicacions de control**

  Per últim, un altre possible treball que es pot dur a terme és realitzar un estudi comparatiu entre aquests dos softwares aprofundint en l'àmbit del control. Per exemple, simular un mateix sistema de control per saber quin és el més eficient, el més ràpid, comparar totes les funcionalitats, etc.

Bibliografia

[1] *Arduino - Home*. Obtingut de https://www.arduino.cc/

[2] Halvorsen, H.-P. *The Technical Guy - a Blog about Technology*. Obtingut de https://www.halvorsen.blog/

[3] *MathWorks - Makers of MATLAB and Simulink*. Obtingut de https://www.mathworks.com/

[4] *National Instruments: Sistemas de Pruebas, Medidas y Embebidos - National Instruments*. Obtingut de http://www.ni.com/es-es.html

[5] Ogata, K. (2010). *Ingeniería de control moderna.* Madrid: Pearson.

[6] *Picuino - Home*. Obtingut de www.picuino.com

ANNEXOS

TÍTOL DEL TFG: Desenvolupament d'un sistema Hardware-In-the-Loop per a l'estudi de sistemes de control.

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Gemma Zafra Ruiz

DIRECTOR: Marcos Quílez Figuerola

DATA: 16 de març del 2018

Panell frontal del Simulador a LabVIEW

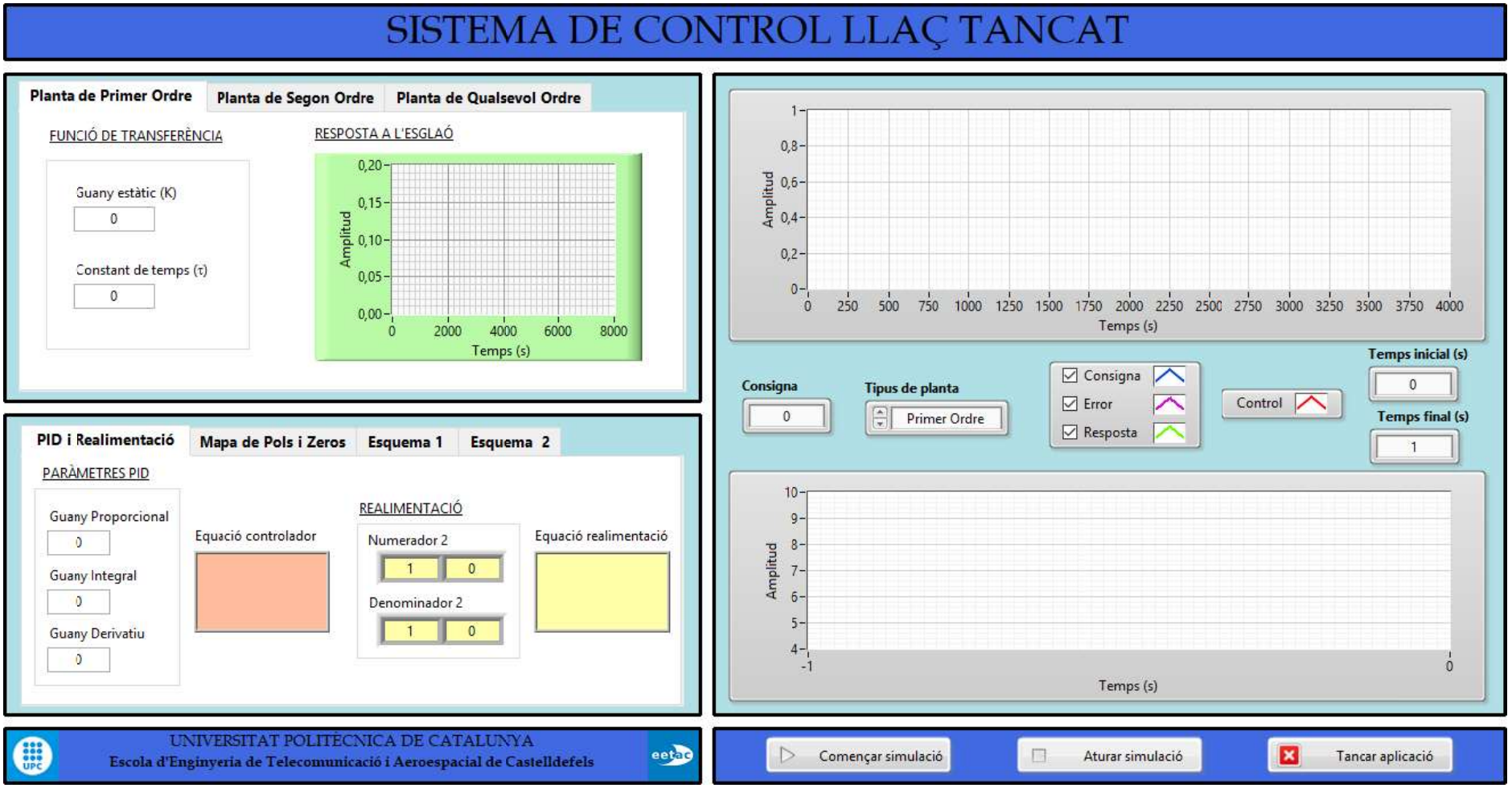
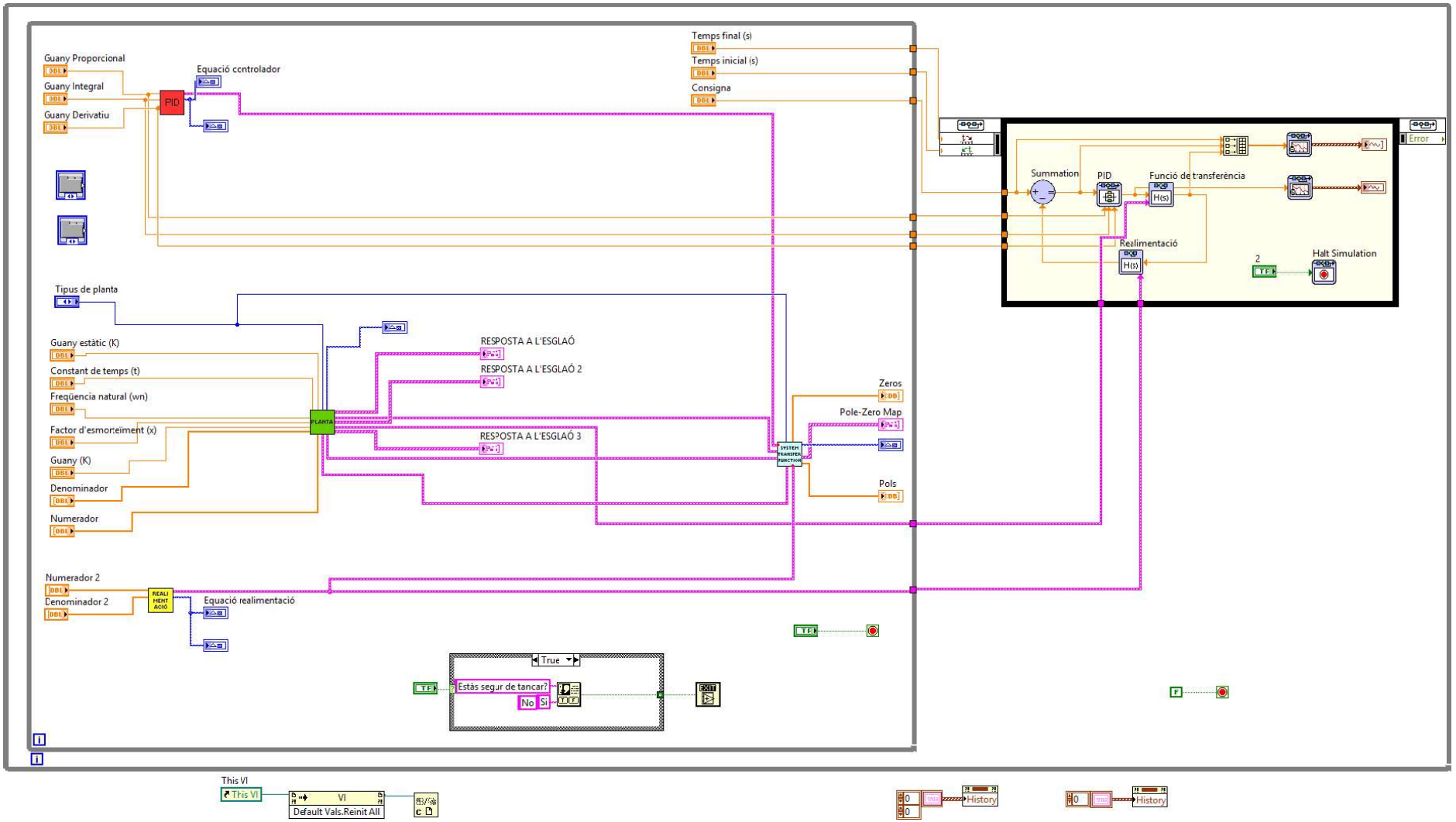
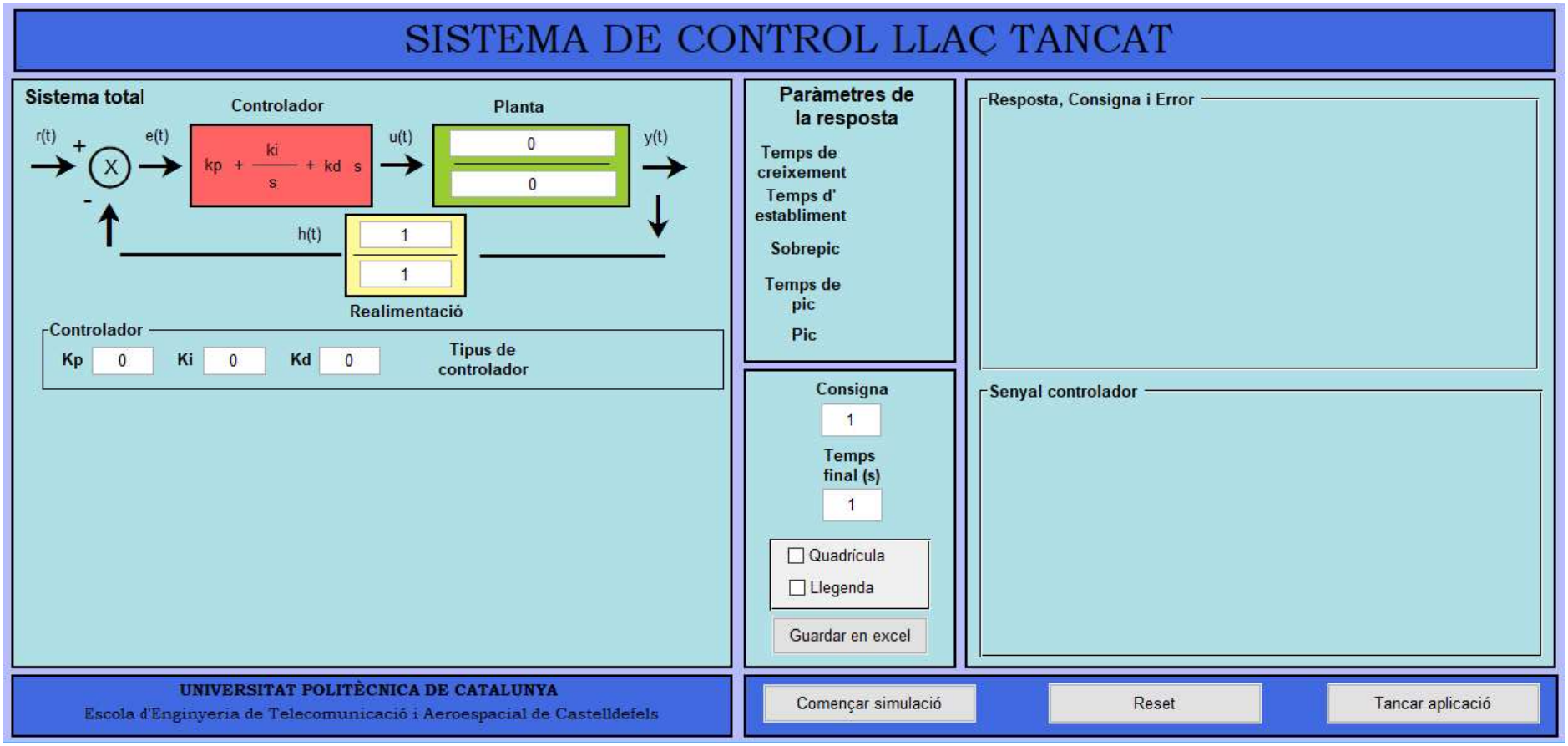


Diagrama de blocs del Simulador a LabVIEW



Panell frontal del Simulador a Matlab



Panell frontal del Simulador HIL a LabVIEW

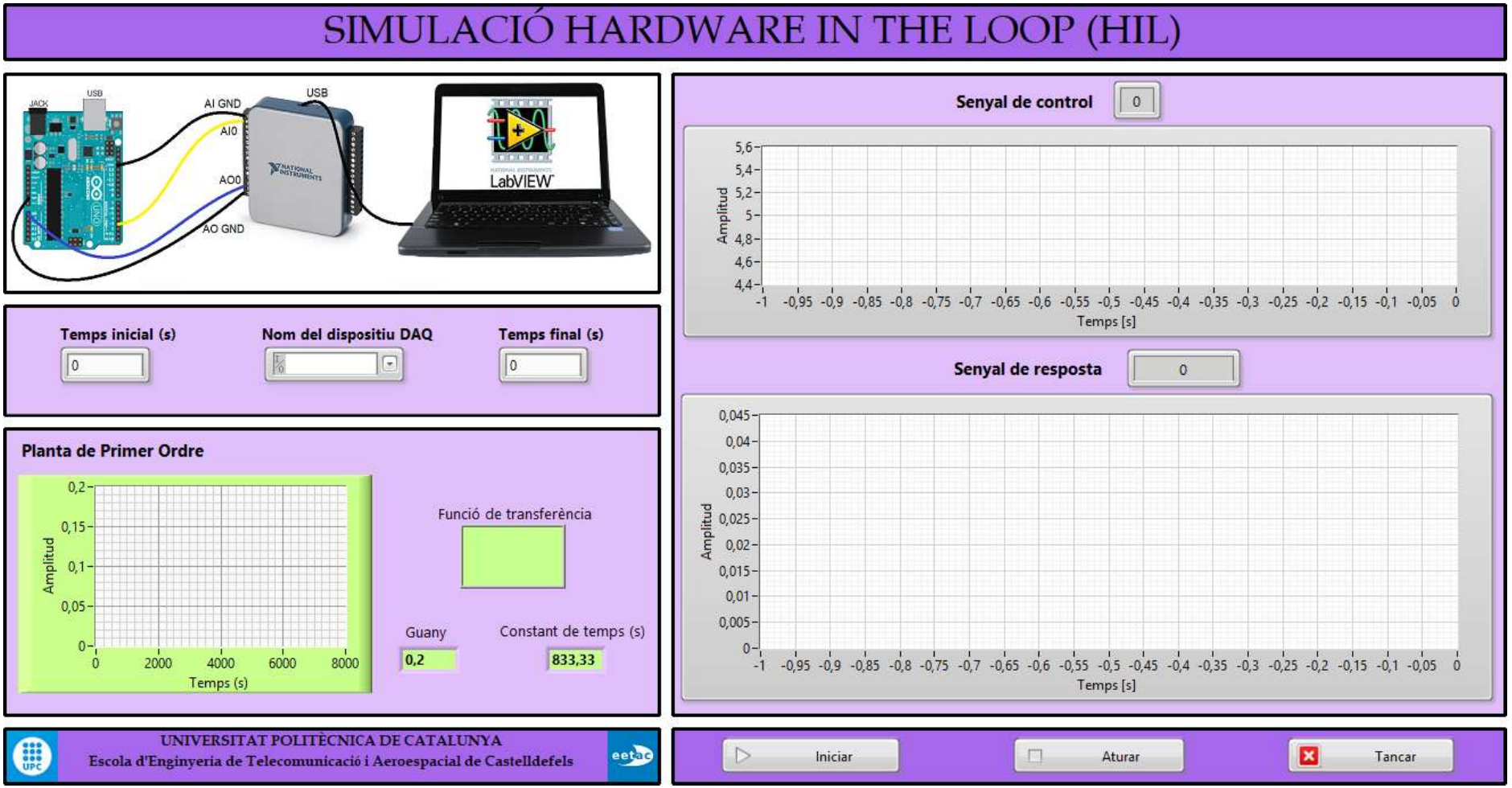


Diagrama de blocs del Simulador HIL a LabVIEW