Titulació:
Enginyeria Superior Aeronàutica

Alumne (nom i cognoms):
Jordi Salvador Bernadí

Títol PFC:
Disseny del sistema de processament de dades en vol d'un coet espacial.

Director del PFC:
Miquel Sureda, Josefina López Herrera

Convocatòria de lliurament del PFC
Febrer 2011

Contingut d'aquest volum: **-Memòria-**

Memòria

Disseny del sistema de processament de dades en vol d'un coet espacial

Agraïments

A la meva tutora Josefina, que m'ha ajudat en tot moment i gràcies a ella he après moltes coses noves per mi.

Al meu tutor Miquel, que m'ha donat suport en el desenvolupament de la part aeronàutica.

A la Laia i la meva família, que m'han donat suport i ajudat al llarg de tot el projecte.

A tots moltes gràcies.

Índex:

1. Introducció

1.1. Objectiu del projecte

L'objectiu d'aquest projecte és el disseny i programació d'un software prototip per l'administració de senyals a bord d'un coet espacial destinat a òrbites baixes. El programa cal que s'executi en temps real i el seu algoritme sigui prou flexible com per poder-lo adaptar fàcilment a diferents configuracions de vehicle.

1.2. Abast del projecte

· Desenvolupament d'un prototip implementat en C++ que gestioni en temps real un nombre predeterminat de senyals característiques d'un coet espacial.

· Obtenció de dades d'un cas real amb que realitzar el processament. Es buscarà la col·laboració dintre del projecte Perseus per obtenir aquestes dades.

· Realització d'un simulador que representi de forma aproximada l'actuació d'un coet espacial a fi de generar les senyals d'interès a tractar. Es podran utilitzar indistintament les dades reals o les simulades, segons interessi al usuari.

· Creació d'una base de dades implementada en MySQL on emmagatzemar totes les dades processades pel programa, de forma que es pugui estudiar el comportament del sistema a posteriori.

· Creació dels *scripts* necessaris per generar les taules de la base de dades que s'hagin d'utilitzar en el programa prototip i que en permetin el correcte funcionament.

1.3. Especificacions bàsiques del projecte

· El programa prototip s'ha d'executar a temps real de manera que es compleixin sempre els terminis de lliurament de cada senyal enregistrada a la funció pertinent. Per això és necessari implementar funcions concurrents per optimitzar la rapidesa del processador, permetent l'execució de múltiples tasques a la vegada.

· S'haurà d'emmagatzemar les senyals simulades en una base de dades perquè el prototip desenvolupat pugui disposar d'aquestes. Aquesta estarà organitzada de tal manera que qualsevol usuari pugui consultar els valors guardats sabent en tot moment quan han estat enregistrats i de quina senyal es tracta.

· El programa haurà de llegir les senyals de la base de dades prèviament creada, simulant el temps d'adquisició dels sensors, administrar-la entre les diferents funcions de control que la requereixen i finalment emmagatzemar-la en una base de dades històrica a fi de poder analitzar el comportament del sistema a posteriori. Cal que l'administració del programa sigui tal que una funció sempre obtingui la dada més actualitzada de la senyal quan la demani.

· Les funcions de control no seran implementades, es simularà el temps de processament destinat a cada una d'elles.

· El software haurà d'anar eliminant les senyals més antigues que ja hagin estat processades i guardades, de forma que la memòria no quedi plena i s'utilitzi d'una forma eficient.

· El programa haurà de tenir implementats mètodes de tractament d'errors per poder detectar possibles problemes i corregir-los a posteriori. No s'inclou dintre de l'abast del projecte la detecció i correcció automàtica d'errors.

· El programa ha de disposar d'una estructura general prou flexible com per poder afegir i treure senyals y tasques a fi de poder-lo personalitzar per cada aplicació diferent i els seus requeriments concrets.

1.4. Justificació

Des de fa poc temps l'espai ha deixat de ser un lloc destinat només a grans projectes estatals per convertir-se en un món accessible a tothom. Cada cop més persones, tant de l'àmbit públic com privat, troben la forma d'accedir a l'espai d'una manera assequible per poder-hi desenvolupar les seves pròpies idees i projectes. Exemples d'aquests fets els podem trobar en el Space Ship One, primer vehicle espacial privat en realitzar un vol suborbital; en el Google Lunar X Prize, on 22 equips de tot el món estan competint per portar un robot a la Lluna; o en el treball fet per estudiants d'universitats d'arreu, que desenvolupen microsatèl·lits per aplicar els coneixements adquirits durant la carrera. En resposta a aquest fet l'agència espacial europea (ESA) ha engegat un programa entre universitats d'Europa, anomenat projecte Perseus, amb el qual es pretén dissenyar i construir un coet de baix cost per enviar una petita càrrega útil, principalment els microsatèl·lits comentats, a òrbites baixes. És dintre d'aquest entorn en el que es pretén emmarcar aquest projecte de final de carrera.

Un coet espacial és un sistema complex, format per varis subsistemes, tots ells interelacionats entre si. Aquest tipus d'aparells han de funcionar de forma autònoma i telecomandada, fet que implica obligatòriament l'autogestió de tots els subsistemes embarcats. Per poder realitzar el control d'un coet espacial cal administrar un elevat nombre de senyals, que poden anar des d'uns centenars fins a milers, provinent dels múltiples subsistemes que cal controlar. Es fa palès doncs la necessitat de disposar d'algun procediment per gestionar aquest torrent de senyals, per poder garantir que els algoritmes de control estan sempre treballant amb variables el més actualitzades possibles, concedint prioritat a aquelles tasques més crítiques. És per això que qualsevol coet ha de portar implementat un algoritme d'administració de senyals en temps real, problema que es pretén solucionar amb aquest projecte de final de carrera.

Tot i estar el treball enfocat al sistema d'un coet espacial, el software desenvolupat podria ser usat en múltiples aplicacions, doncs al tractar-se d'una solució flexible i polivalent no seria difícil adaptar-lo a altres aparells. Així, en el camp de l'aeronàutica, que és el que ens ocupa, podria ser utilitzat en la gestió de qualsevol sistema automàtic de gestió del vol, els quals cada cop tenen més importància en l'aviació comercial. També es podria utilitzar en un camp en clara expansió, que donarà molt de que parlar els pròxims anys, com és el dels vehicles aeris no tripulats, més coneguts com a UAV (Unmanned Aerial Vehicle), els quals han de gestionar un gran nombre de senyals al no disposar de cap

tripulant a bord, ja sigui per el control automàtic a bord o pel telecomandament des de terra.

Finalment afegir que s'espera que al tractar el problema des d'uns coneixements d'enginyeria aeronàutica es pugui crear una solució completament enfocada al producte final i estretament lligada al sistema físic a gestionar i operar.

2. Entorn del prototip

2.1. Coet espacial destinat a òrbites baixes

S'entén per coet espacial un vehicle capaç de transportar una càrrega útil definida fins a un punt determinat més enllà de l'atmosfera terrestre, gràcies a la creació d'energia cinètica a partir de l'expansió d'un conjunt de gasos.

El software que es pretén desenvolupar està enfocat a un coet com el que s'està desenvolupant dintre del projecte Perseus, destinat a òrbites baixes pel transport de nanosatèl·lits.

Les òrbites baixes són aquelles que es troben compreses entre 200 i 1200 km d'altura. Concretament interessa situar la càrrega útil en una òrbita polar circular[1] d'uns 250 km d'altura. L'avantatge d'aquestes òrbites és que són les més properes a la terra i per tant les que menys energia es necessita per fer-hi arribar una càrrega útil. Per contra al estar tant a prop reben un gran fregament amb l'atmosfera que escurça la seva vida en servei i el seu període de rotació ràpid fa que tinguin estones de disponibilitat curtes.

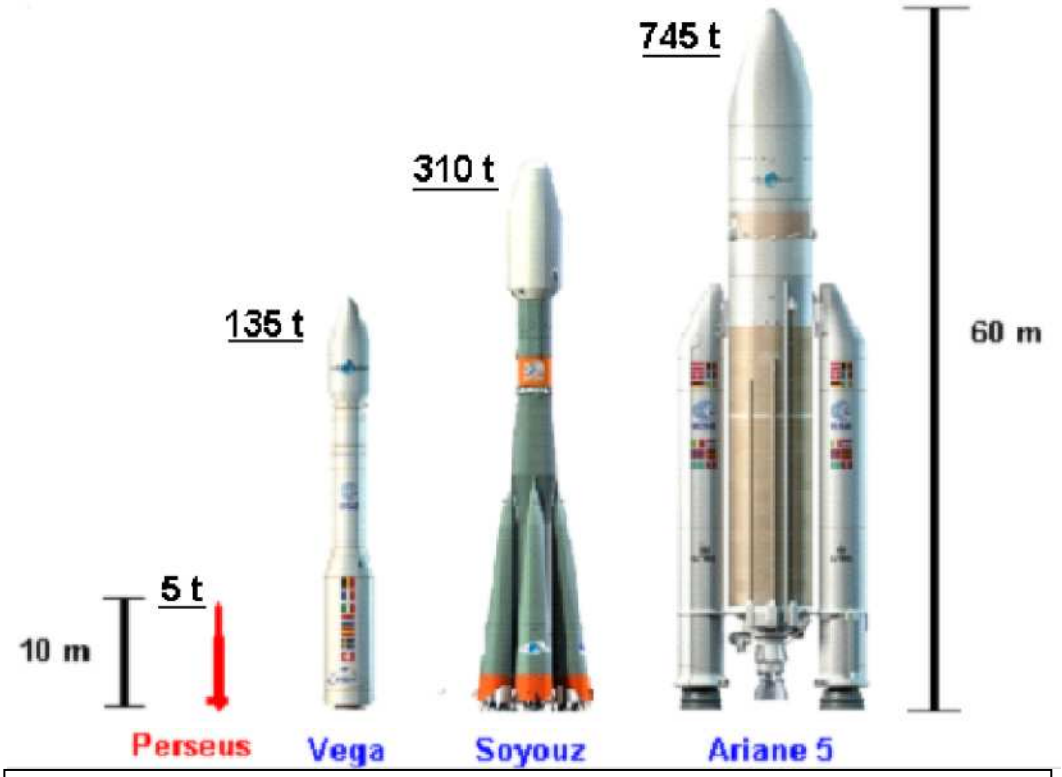


Figura 1. Comparació de mida i pes entre diferents coets espacials i el proposat pel projecte Perseus

[1] Òrbita polar circular: la traça d'aquesta creua en algun moment pels pols.

La càrrega útil del coet s'espera que sigui nanosatèl·lits d'una massa aproximada de 10 kg. Tradicionalment aquest tipus de vehicles espacials han tingut bàsicament finalitats pedagògiques i científiques, degut a la seva mida reduïda. Tot i així comencen a oferir les mateixes possibilitats que els de majors dimensions i s'espera que puguin acabar realitzant qualsevol tasca comercial.

Les configuracions que pot tenir un vehicle llançador d'aquestes característiques són múltiples i bastant diferents entre si. Per aquest motiu és necessari que el software es pugui adaptar fàcilment a cada una d'aquestes solucions tècniques, disposant d'uns algoritmes flexibles i modificables de forma senzilla, sense haver de refer tot el programa de nou cada cop.

Entre les possibles solucions proposades pel coet prototip destaquen les següents:

· Tipus de motor:

a) Propulsió líquida: Els combustibles es troben en estat líquid. Solució més complexa i cara però a la vegada és la que dóna més bones prestacions i rendiment.

b) Propulsió sòlida: Els combustibles es troben en estat sòlid. Solució més barata i senzilla però a la vegada és la que dóna menys prestacions.

c) Propulsió mixta: Un dels combustibles es troba en estat sòlid mentre que l'altre en estat líquid. Solució intermèdia entre les dues anteriors.

· Vehicle llançador:

a) Coet de varies etapes suborbital: Vehicle tipus coet durant tot el trajecte, amb una etapa inicial per situar-lo en l'alçada correcte i una segona etapa per ajustar els paràmetres de l'òrbita. La utilització de varies etapes permet una millor optimització de les toveres que es tradueix en un estalvi important de combustible.

b) Sistema de llançament aerotransportat: Una aeronau alçaria el coet fins a una certa altura per a continuació llençar-lo des d'allà. D'aquesta manera la part inicial del trajecte es realitzaria gràcies a la sustentació creada per l'avió, mètode molt més eficient que el del coet, que permet estalviar combustible. El coet llençat també podria disposar igualment de més d'una etapa.

c) Coet llençat des d'un globus: Igual que el mètode anterior, però en aquest cas es tractaria d'un globus d'heli que elevaria el coet fins a una certa alçada, estalviant així tot el combustible inicial.

· Control d'actitud i navegació:

a) Amb superfícies aerodinàmiques: Un conjunt de superfícies i timons deflactables permetria controlar l'actitud de l'aparell gràcies a la creació i control de forces aerodinàmiques. Aquest sistema només serviria per l'etapa de vol que s'executés a l'interior de l'atmosfera.

b) Amb motors coet secundaris: Un conjunt de motors coet més petits que l'original permetrien generar les forces necessàries per controlar l'actitud de l'aparell. Aquest sistema podria ser utilitzat tant a dins com a fora de l'atmosfera, però té l'inconvenient que consumeix combustible, afegint així més pes al vehicle.

c) Tovera balancejant: La tovera principal es pot inclinar un cert angle permetent així variar la direcció del vector impuls. Igual que l'anterior pot ser utilitzat tant a dins com a fora de l'atmosfera. Sol ser el més utilitzat en coets pel control de navegació.

Figura 2. A l'esquerra sistema de llançament aerotransportat, a la dreta coet suborbital

2.2. Sistema a gestionar

El sistema a gestionar inclou tot el procediment des de que un sensor envia una senyal fins que un actuador rep la resposta pertinent degut a la dada enregistrada. Això implica l'adquisició de la senyal des dels sensors, l'adaptació d'aquesta a un format genèric (a fi de poder-la administrar en el programa), el lliurament d'aquesta a la funció pertinent, l'aplicació de l'algoritme de control, l'adquisició de les respostes generades (amb la corresponent adaptació al format genèric) i, finalment, l'entrega d'aquesta als actuadors que la necessitin. Tot això cal fer-ho assegurant que en tot moment (en temps real) s'està treballant amb les dades més actualitzades possibles, que no es perden senyals significatives i que aquestes són lliurades a temps a les tasques corresponents. Cal doncs, a part de gestionar les senyals en si, administrar també una sèrie de variables de control que indiquin i registrin tots aquests factors, i que el software s'executi en temps real, controlant en tot moment el temps d'execució.

Es pot veure una representació del sistema en la figura 3. El lector observarà que tot i que el control s'ha inclòs dintre del sistema, el software desenvolupat no inclou els algoritmes de control, els quals són prou complexos com per ser un Projecte final de carrera en si cada un d'ells. En el seu lloc, simplement s'ha simulat un temps de processament estimat, deixant les indicacions pertinents per poder-hi implementar les funcions de control necessàries.

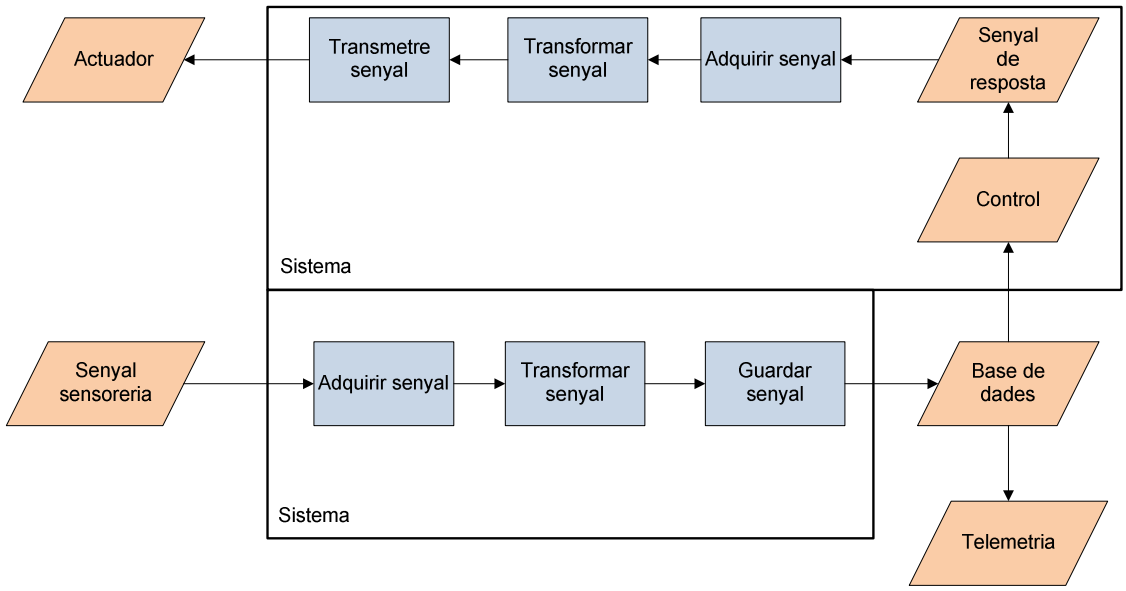


Figura 3. Diagrama de blocs del sistema general

En l'esquema de la figura 3 es pot veure que el sistema es troba dividit en dos. La representació és així degut a que la solució final contempla l'existència de dos programes principals independents, un dels quals adquireix les senyals des dels sensors i les guarda en la base de dades, mentre que l'altre agafa directament les dades d'aquesta base de dades pel seu tractament. Diferenciem doncs entre una funció principal de captura i una altre de tractament o control.

En l'esquema de la figura 3 també es poden veure dos actors que encara no s'havien introduït. El primer d'ells és la telemetria, que serveix d'enllaç entre l'estació terrestre i el coet. A través d'aquest sistema el coet no només envia el seu estat sinó que també rep senyals de control. És indispensable l'existència d'aquest mòdul en qualsevol aparell per disposar de l'estat d'aquest en temps real a l'estació terrestre, al mateix temps que permet corregir qualsevol anomalia de funcionament que es pugui detectar.

El segon actor és la base de dades, en la qual s'emmagatzemen les senyals gestionades. Això permet obtenir un registre de tota la missió, permetent un cop aquesta ha estat finalitzada analitzar-la i estudiar-la per incorporar millores per futures missions. Tot i que en el prototip es guarden les dades com si la base de dades es trobés a bord del coet, en un aparell real aquesta també es podria trobar a la base de terra, obtenint-les a partir de la telemetria, doncs en la majoria de casos el vehicle seria irrecuperable al no aterrar.

Cal destacar que es tracta d'un software crític, entenent que si el sistema falla es pot produir fàcilment un error catastròfic que podria suposar el final de la missió o fins hi tot la pèrdua de vides humanes. És per aquest motiu que cal introduir un sistema de tractament d'errors, de forma que aquests es puguin detectar a fi de donar una resposta el més ràpida possible als problemes que es poguessin produir. En el prototip només s'ha implementat un sistema capaç de trobar errors a fi de que l'usuari els pugui corregir en el codi font, deixant per una fase posterior la detecció i correcció automàtica d'aquests.

Com ja s'ha comentat l'ordre de magnitud de les senyals que cal gestionar en un coet pot anar des de uns centenars fins a milers[2]. Intentar abastar des d'un començament la gestió i administració d'aquest gran nombre de senyals comportaria segurament el fracàs. És per aquest motiu que per l'elaboració del prototip només s'han tingut en compte dos sistemes, el de control d'actitud i el de navegació, simplificant així el model inicial. Un cop realitzada i testada l'estructura base del software, no és un problema poder afegir més senyals, sent

[2] Un resum de les senyals més destacades i els sistemes als quals van associades es pot veure en el Annex 1.

necessari tan sols implementar els trossos de codi de la nova dada a tractar en els punts adients, doncs la flexibilitat del codi és un dels requisits inicials.

Els motius per escollir aquests dos sistemes no són trivials. En primer lloc cal tenir en compte que el sistema de control d'actitud és un dels pilars per garantir el bon funcionament de l'aparell. Si no s'aconsegueix garantir l'estabilitat del coet la trajectòria serà incontrolable, podent fins hi tot arribar a la pèrdua del vehicle a l'estavellar-se, i per tant la missió no es podrà portar a terme amb èxit. Estretament lligat amb aquest sistema hi ha el de navegació, que utilitza les mateixes senyals, sent l'encarregat de controlar la trajectòria de l'aparell. Un altre factor important per haver escollit aquests dos sistemes és que es troben implementats en qualsevol aeronau que disposi d'un pilot automàtic o d'un telecomandament, sent així el codi desenvolupat extensible a altres tipus d'aparells.

Es donarà en els següents punts una breu descripció dels sistemes en qüestió.

2.2.1. Sistema de control d'actitud

Aquest sistema s'encarrega de determinar i controlar l'actitud de l'aparell respecte a un sistema d'eixos inercials. Això ho fa a partir de les acceleracions y velocitats angulars a que està sotmès, amb les quals podem determinar els angles d'Euler, que són els angles de la rotació que passa les coordenades de eixos cos als eixos inercials de referència[3].

La mesura d'aquestes variables es fa utilitzant giroscopis, els quals obtenen directament les velocitats angulars. Si derivem temporalment aquesta es pot determinar l'acceleració angular. Com que el sensor és solidari a l'aparell s'obtenen les magnituds en eixos cos, a partir de les quals i coneixent els angles d'Euler inicials, es pot determinar com varia l'actitud del coet respecte al sistema de referència inercial al llarg del temps.

Aquest tipus de sensors presenten problemes d'error estàtic, el qual s'acumula amb el temps. Per solucionar-ho s'acostumen a incorporar també magnetòmetres que corregeixen aquests errors a partir de la mesura del camp magnètic terrestre. És necessari com a mínim un giroscopi i magnetòmetre per cada un dels tres eixos, tot i que el més normal és que se'n incloguin més, formant per exemple un tetraedre, a fi de disposar de redundància i poder filtrar millor els errors.

[3] Veure definicions dels diferents sistemes de coordenades en el Annex 2

Tot aquest sistema sol anar implementat en les anomenades IMU (Inertial Mesurement Unit), les quals ja realitzen tota la tasca de integració, filtratge i correcció d'errors donant directament com a sortida les velocitats i acceleracions angulars de l'aparell. Pel disseny del prototip es considera que el control interactua directament amb la IMU.

El sistema disposa de diferents opcions a l'hora d'implementar els actuadors:

· Superfícies aerodinàmiques. S'utilitzen en coets de mida reduïda i que operin en l'atmosfera. Es col·loquen en els laterals amb l'objectiu d'enrederir el centre de pressió, fet que dota d'una certa estabilitat el coet. Poden disposar de parts mòbils, les quals quan es deflacten produeixen canvis en les forces aerodinàmiques d'aquestes superfícies, podent així controlar els moments a que es veu sotmès l'aparell, de la mateixa manera que es realitza en els avions. Aquesta solució es veu limitada a etapes que actuïn dintre de l'atmosfera, doncs un cop aquesta es fa menys densa perden efectivitat, sent completament inútils en el vuit, ja que estan basats en les forces aerodinàmiques.

· Tovera balancejant. Utilitzant dos actuadors hidràulics disposats perpendicularment es pot variar la inclinació de la tovera, permet controlar la direcció del vector impuls i per tant també el del moment que aquest crea. És el mètode més utilitzat actualment en grans coets.

· Motors auxiliars laterals. permeten generar impulsos petits en comparació al motor principal però que generen moments prou importants com per controlar l'actitud de l'aparell.

En l'esquema de la figura 4 es pot veure un diagrama de blocs senzill del sistema i els seus actors principals.

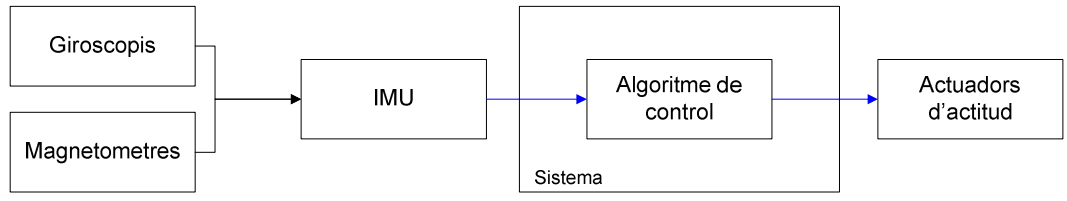


Figura 4. Diagrama de blocs del sistema de control d'actitud

Les línies representades en blau son els fluxos de dades que ha de gestionar i supervisar el programa. Com es pot veure el control del sistema queda inclòs dintre del software d'administració de senyals. No està dintre de l'abast d'aquest projecte el disseny i programació d'un algoritme de control, podent ser en si mateix un Projecte Final de Carrera per cada un dels diferents sistemes, de tal

manera que simplement s'indica en el codi del programa els llocs on caldria implementar aquest algoritme.

Les senyals que es tractaran en el prototip son les pròpies del control d'actitud:

- Velocitat angular en eixos cos, vector de tres components.
- Acceleració angular en eixos cos, vector de tres components.
- Angles d'Euler, vector de tres components.

Les senyals destinades al control del funcionament i operació del sistema no es tenen en compte.

2.2.2. Sistema de navegació

Aquest sistema és el que s'encarrega de determinar i controlar la trajectòria que segueix el coet. Per fer-ho parteix de la posició relativa de l'aparell respecte a un sistema de referència inercial i de la velocitat i acceleració lineal.

La velocitat i l'acceleració lineal s'obtenen amb acceleròmetres, els quals mesuren directament l'acceleració de l'aparell, a partir de la integració temporal de la qual podem trobar la velocitat. Aquest procés sol anar integrat també dintre de la IMU, de forma que aquesta té directament com a sortida les acceleracions i velocitats tant linears com angulars de l'aparell en eixos cos. Si es torna a integrar la velocitat es pot trobar la posició. Aquesta també pot ser controlada per GPS, a partir dels quals també es fa possible trobar tant velocitats com acceleracions.

El control d'aquest sistema es realitza amb els mateixos sistemes que els explicats en el cas del control d'actitud. La diferència que hi ha entre els dos és que amb el control d'actitud es busca mantenir l'aparell estable dintre d'uns angles d'Euler determinats, mentre que amb el sistema de navegació el que es fa és modificar aquests angles d'Euler de forma controlada i determinada per seguir la trajectòria desitjada. Queda palès doncs que els dos sistemes estan estretament lligats, sent en realitat el control d'actitud un llaç intern al de navegació que s'ha d'executar a una freqüència major.

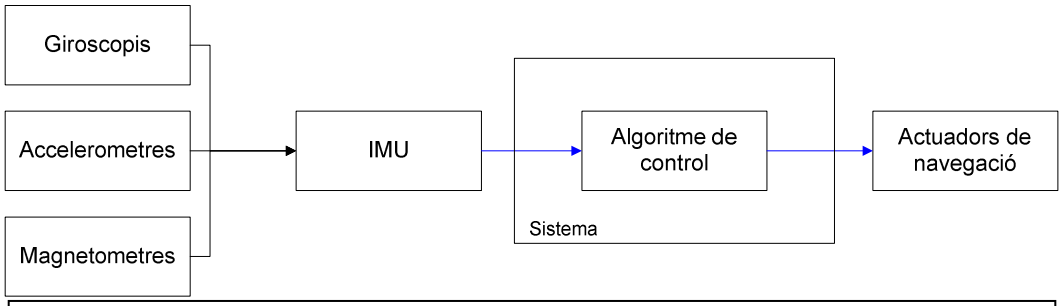


Figura 5. Diagrama de blocs del sistema de control de navegació

Les senyals que es tractaran en el prototip son les pròpies del control d'actitud:

- Velocitat en eixos cos, vector de tres components.
- Acceleració en eixos cos, vector de tres components.
- Posició respecte sistema inercial, vector de tres components.
- Velocitat en eixos inercials, vector de tres components.
- Acceleració en eixos inercials, vector de tres components.

Les senyals destinades al control del funcionament i operació del sistema no es tenen en compte.

2.3. Altres sistemes on aplicar el prototip

Al tractar-se d'un software polivalent per diverses configuracions de coet permet al mateix temps que s'adapti a sistemes completament diferents. A continuació es resumirà breuement quins són els més importants:

· Gestió de dades en un vehicle espacial: De la mateixa manera que els coets, qualsevol altre vehicle espacial (ja sigui un satèl·lit, una estació espacial, una sonda planetària...) han de complir amb els mateixos requeriments en autogestió dels sistemes embarcats, doncs en la majoria dels casos es tracta d'aparells no tripulats el control dels quals es fa de forma telecomandada des d'estacions terrestres, havent de funcionar el vehicle la major part del temps de forma completament autònoma, amb la necessitat de controlar la posició, l'actitud, els motors, els sistemes de control tèrmic... Fins i tot en els casos en que hi ha tripulants a bord, com en el cas de l'estació espacial internacional, aquests estan a l'espai per la realització de tasques científiques, no per estar pendents tota l'estona del correcte funcionament dels sistemes, necessitant també un grau elevat d'automatització. Per aquests motius el software es podria implementar en qualsevol plataforma espacial per gestionar les senyals pertinents.

· Gestió de dades en el pilot automàtic d'aeronaus tripulades: Les aeronaus actuals porten totes incorporades pilots automàtics que permeten governar l'aparell de forma autònoma facilitant així la feina dels pilots. Per això és necessari gestionar un nombre considerable de senyals referents a la posició, actitud y condicions de vol de l'aparell, a fi de poder controlar l'aeronau de la forma més efectiva i eficient possible sense la necessitat de cap intervenció humana. Per tant el software presentat també podria ser utilitzat en aquest cas. Al mateix temps, a part del pilot automàtic, el control de tots els sistemes a bord de les aeronaus es troba centralitzat en la cabina de control, on el pilot pot gestionar-los de forma relativament simple i intuïtiva. La gestió de dades que implica també podria ser administrada pel software que es planteja en aquest projecte.

· Gestió de dades en UAS: Un UAS és un sistema aeri no tripulat (de les sigles en anglès de Unmanned Aerial System). Es tracta d'aeronaus de dimensions més reduïdes que els vehicles aeris tripulats al no haver de transportar cap persona a bord. El seu control es realitza amb l'ús d'un pilot automàtic i de forma telecomandada des d'una estació a terra. Per que això sigui possible, igual que en el cas anterior, el vehicle ha de saber en tot moment quines son les seves condicions de vol, posició i actitud, a fi de poder-les controlar i mantenir dintre dels marges establerts. A més per poder controlar l'aparell des de terra és necessari tenir control sobre la càrrega útil, conèixer l'estat de les bateries i el

seu consum, condicions ambientals... Tot aquest flux de dades fa necessària una correcte gestió de els senyals. Així doncs el software desenvolupat durant el projecte també podria ser utilitzat en aquests sistemes.

· Gestió de dades en estacions terrestres: En qualsevol estació terrestre aj sigui pel control del trànsit aeri, de vehicles espacials o de UAS, arriben un flux de dades importants que cal organitzar de la millor forma possible perquè el controlador, ja sigui una persona o una computadora, pugui realitzar la seva feina amb la major facilitat possible. Així doncs també és necessària la implementació d'un software per la gestió de les senyals com el desenvolupat.

Així doncs, tot i que la intenció inicial del software és que estigui destinat a una plataforma tipus coet, el prototip té en realitat aplicacions en tot el camp aeronàutic, com es desprèn de la llista anterior. Això es així degut a que la complexitat dels seus sistemes fa que hi hagi un elevat nombre de senyals a gestionar, fent necessària la implementació d'algun mètode d'administració de dades per garantir el correcte tractament d'aquestes.

3. Disseny del programa prototip

3.1. Requeriments

· Execució en temps real. Cal lliurar les senyals a les funcions corresponents dintre d'un temps determinat a fi que s'estigui treballant amb les variables el més actualitzades possibles per poder donar la resposta més adient a l'estat del sistema. És per això que és necessari que el programa s'executi en temps real. Això comporta disposar de funcions que permetin conèixer el temps relatiu i el temps absolut al programa a fi de coordinar temporalment les diverses tasques. També és necessari que les funcions s'executin de forma concurrent doncs moltes d'elles s'han de processar a la vegada per poder arribar als terminis establerts.

· Protecció de variables. Com que les funcions s'executen de forma concurrent cal establir mètodes de protecció de les variables globals, de forma que una variable no estigui sent actualitzada a la vegada per dues funcions diferents, fet que podria comportar errors d'execució del programa, com per exemple la lectura de senyals no actualitzades.

· Coordinació de funcions concurrents. Cal establir mètodes per coordinar les tasques de les diverses funcions de forma que les interaccions entre elles es realitzin correctament de forma optimitzada. De la mateixa manera cal garantir que les tasques comencen i acaben en els moments adequats.

· Codi flexible i personalitzable. Interessa que el programa es pugui adaptar a plataformes i necessitats diferents sense un esforç important. Per això cal que el codi disposi de una base genèrica en la que es puguin afegir mòduls independents per cada tipus o grup de senyals.

· Optimització de la memòria. Cal optimitzar els recursos de memòria per poder minimitzar la dimensió d'aquesta. Per això es necessari disposar d'un buffer d'una capacitat limitada en el qual s'emmagatzemen de forma temporal les senyals i s'eliminen d'aquest un cop han estat processades.

· Anàlisis i verificació del prototip. Un cop executat el processament cal assegurar que aquest ha realitzat la seva funció correctament. Per això s'haurà d'establir un registre on es mostrin les senyals de captura junt amb la senyal de control corresponent i els temps de captura, processat i emmagatzemat d'aquestes. Així és pot veure si totes les senyals han complert els terminis desitjats.

· Base de dades ordenada. Cal que la base de dades on es presenten les senyals estigui correctament ordenada a fi que l'usuari pugui classificar les dades per tipus, temps relatiu o identificador.

· Tractament d'errors. Cal establir protocols per detectar errors de funcionament de computació a fi de poder-los reparar o establir mesures correctores quan succeeixen.

3.2. Eines de treball

Les eines escollides pel desenvolupament del prototip han estat:

· Llenguatge de programació C++: S'ha escollit aquest llenguatge al ser el que s'ha estudiat durant tota la formació universitària i per tant del que més coneixements se'n disposava. Compleix amb els requisits necessaris doncs permet la programació tant en alt com en baix nivell, la generació de funcions concurrents gestionades per semàfors i la protecció de variables globals. Com a complidor s'ha utilitzat el Bloodshed Dev-C++, doncs es tracta d'un programa open-source gratuït amb el que s'ha treballat sempre a l'escola.

· Base de dades MySQL: S'ha escollit aquesta base de dades al tractar-se d'un programari open-source gratuït, a part de ser una de les més utilitzades i per tant de les que és més fàcil obtenir-ne referències. Permet una fàcil interacció amb el llenguatge C++.

3.3. Dades emprades

Inicialment es pretenia utilitzar dades reals d'algun experiment o simulació realitzades en el projecte Perseus[4] impulsat pel CNES, a fi de poder integrar els resultats obtinguts pel prototip en aquest. Per desgràcia, per problemes burocràtics, no s'ha pogut disposar d'aquestes dades ni de la col·laboració desitjada, de forma que s'ha optat per programar un senzill simulador[5] que enregistra en una base de dades les senyals necessàries per l'execució del prototip.

Les senyals que s'han generat han estat les següents:

· Posició del coet respecte un sistema de referència inercial, tres components (x, y, z).

· Velocitat del coet respecte un sistema de referència inercial, tres components (v_x, v_y, v_z).

· Acceleració del coet respecte un sistema de referència inercial, tres components (a_x, a_y, a_z).

· Velocitat del coet en eixos cos, tres components (v_{bx}, v_{by}, v_{bz}).

· Acceleració del coet en eixos cos, tres components (a_{bx}, a_{by}, a_{bz}).

· Velocitat angular del coet respecte els eixos cos, tres components (ω_x, ω_y, ω_z).

· Acceleració angular del coet respecte els eixos cos, tres components (α_x, α_y, α_z).

· Angles d'Euler (rotació entre els sistema de referència inercial i els eixos cos), tres components (ε_x, ε_y, ε_z).

[4] Veure [12] i [13]
[5] Veure annex 3.

Les senyals s'han emmagatzemat en la base de dades utilitzant el següent format:

Camp	Tipus	Descripció
ide	double	Identificació numèric de la senyal. Va de menor a major per ordre de captura.
type_code	varchar(6)	Identificació del tipus de senyal[6].
signal_value_x	double	Valor de la senyal en la coordenada x.
signal_value_y	double	Valor de la senyal en la coordenada y.
signal_value_z	double	Valor de la senyal en la coordenada z.
relative_time	double	Temps relatiu de captura de la senyal des del inici del procés.

De cada tipus de senyal se'n emmagatzemen dos valors:

La primera es la senyal de control, que es tracta de la referència necessària pels algoritmes de control per poder executar la seva tasca. Representen per exemple els valors del pla de vol de l'aparell, les maniobres predefinides, els valors de les ordres de rectificació enviades des de l'estació terrestre, etc, és a dir, les referències desitjades a partir de les quals l'algoritme de control pot trobar l'error en que es troba l'aparell a fi de corregir-lo. En el prototip, a fi de simplificar-lo, només s'utilitzen les senyals de control que hi hauria guardades en la base de dades internes del coet (pla de vol, maniobres definides i valors límits operatius). No es tenen en compte possibles correccions enviades des d'un aparell remot, tot i que en el sistema real existirien. Aquestes es podria simular en cas de necessitat de forma senzilla amb la inserció en les taules de control dels registres que es volen modificar. Les senyals de control es guarden en taules amb el nom *t_sim_"type_code"_control* (per exemple, en el cas de la posició, és t_sim_pos_control). Aquestes es guarden des del programa Simulador_control.exe que es pot trobar en el suport digital adjunt amb aquest treball.

En segon lloc hi ha les senyals de captura. Aquestes serien les dades obtingudes des dels sensors, és a dir, els valors reals enregistrats per l'aparell. En el simulador es creen a partir d'afegir a les senyals de control un cert error aleatori proporcional al seu valor. Les senyals de captura es guarden en taules amb el nom *t_sim_"type_code"* (per exemple, en el cas de la posició, és t_sim_pos). Aquestes es guarden des del programa Simulador_capture.exe que es pot trobar en el suport digital adjunt amb aquest treball, el qual representa el procés d'adquisició de senyals des dels sensors, tal i com es veurà en el següent apartat.

[6] Veure annex 4.

3.4. Processament de dades

Com ja s'ha comentat anteriorment, el prototip consta de dos programes principals. El primer d'ells és el procés de captura, que s'encarrega d'adquirir les senyals des dels sensors i guardar-les a la base de dades. És en aquesta on el segon programa, el procés de control o tractament, llegeix les dades necessàries i les guarda en un buffer al que poden accedir totes les tasques. Es pot veure un esquema simplificat d'aquest funcionament en la figura 6.

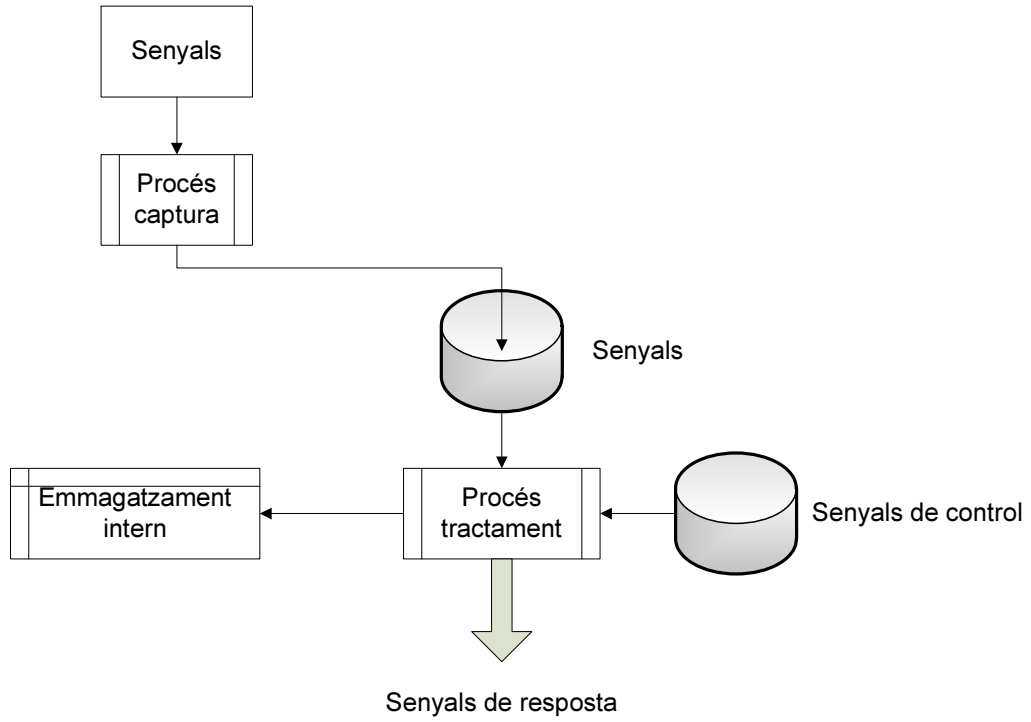


Figura 6. Diagrama de blocs del sistema d'adquisició de lles senyals

Un aspecte important de l'execució del programa és la sincronització temporal dels senyals de control i les de captura per poder-les comparar. Per això suposem que cada tipus de senyal té un període de mostreig t_s predefinit de forma que el registre 'n' d'una determinada taula de control es correspongui amb el valor de la senyal a l'instant $t_s \cdot n$. Es simula aquest període agafant temps de espera entre cada inserció de dades en la base de dades.

3.4.1. Procés de captura de senyals

En el prototip implementat no es disposa dels sensors, doncs com ja s'ha comentat interactuar directament amb aquests no entra dins l'abast del projecte, sent a més aquesta una comunicació que dependria del tipus i marca de sensor utilitzat. Per aquest motiu el que fa aquest procés es simular el procés d'adquisició i la freqüència de mostreig, guardant les senyals en les taules de simulació corresponents comentades en l'apartat anterior.

Així aquestes taules representen la base de dades que es trobaria a bord del coet, des de la qual el procés de tractament llegeix les dades que es van adquirint. En el prototip real aquesta base de dades podria ser tant interna com externa, en aquest segon cas incloent la necessitat d'un protocol de comunicació més complex.

3.4.2. Procés de tractament de senyals

Aquest és el procés que adquireix les dades de la base de dades de bord i les guarda en el buffer, a fi que totes les tasques concurrents puguin disposar de les senyals necessàries.

Com ja s'ha comentat, la base de dades a bord no ha estat implementada, però es simula la seva existència en el procés de lectura, realitzant el mateix temps d'adquisició mencionada en el procés de captura de senyals. Les dades es llegeixen des de les taules de simulació, considerant aquestes com si fossin les que s'han anat generant a la base de dades de bord. Així s'aconsegueix que el sistema es comporti com si les dades disponibles anessin apareixen cada cert període de temps, aconseguint el mateix comportament que el sistema real sense haver de fixar els mecanismes de comunicació amb la base de dades que són desconeguts, tal i com ja s'ha vist anteriorment.

El procés de tractament de les senyals en el prototip és el següent:

· A l'inici del procés o cada cop que el buffer està ple, el fil de captura llegeix 'n' registres dels següent cursor i es guarden en el buffer. Gràficament es pot representar de la següent manera:

Element	Senyal control	Senyal captura	Tractat
1	Valor 1		N
2	Valor 2		N
...	...		...
n-1	Valor n-1		N
n	Valor n		N

· El fil d'administració va llegint de forma continua des de la base de dades els nous registres adquirits pel procés de captura i els guarda en el buffer. Aquí es suposa que mai hi haurà més registres nous que el número màxim d'elements del buffer. Hi ha una variable global destinada a marcar quina ha estat la última senyal que s'ha guardat en el buffer.

Element	Senyal control	Senyal captura	Tractat
1	Valor 1	Valor 1	N
2	Valor 2		N
...	...		...
n-1	Valor n-1		N
n	Valor n		N

· Paral·lelament el fil de control va realitzant el tractament de les senyals, actualitzant el camp relatiu a si el registre ha estat tractat i aplicant els algoritmes de tractament corresponents. Hi ha una variable global per tipus de senyal que indica quin és el següent element del buffer a tractar.

Element	Senyal control	Senyal captura	Tractat
1	Valor 1	Valor 1	S
2	Valor 2	Valor 2	N
...	...		...
n-1	Valor n-1		N
n	Valor n		N

· Quan el buffer està ple elimina els elements que ja han estat tractats i els reordena. En el model actual es considera que el buffer està ple quan s'ha guardat en ell el paràmetre n, podent ser aquest ajustat per l'usuari. També actualitza la variable destinada a controlar la ultima posició en la que s'ha guardat una senyal i la que indica següent element del buffer a tractar.

El buffer quan està ple quedaria doncs de la següent manera:

Element	Senyal control	Senyal captura	Tractat
1	Valor 1	Valor 1	S
2	Valor 2	Valor 2	S
...	...	...	...
n-1	Valor n-1	Valor n-1	N
n	Valor n	Valor n	N

I un cop reordenat:

Element	Senyal control	Senyal captura	Tractat
n-1	Valor n-1	Valor n-1	N
n	Valor n	Valor n	N
...	...	...	...

2n-1			N
2n			N

· A continuació tornaria a començar el procés de nou, repetint-se fins a la finalització del programa.

Element	Senyal control	Senyal captura	Tractat
n-1	Valor n-1	Valor n-1	N
n	Valor n	Valor n	N
...	...	...	...
2n-1	Valor 2n-1		N
2n	Valor 2n		N

Les actualitzacions de les variables globals i del buffer s'han protegit sempre amb l'ús de mutex per evitar que dues funcions concurrents puguin disposar de una variable en el mateix moment, fet que comportaria problemes de sincronització i execució.

3.5. Historial

Totes les dades que es processen es guarden un altre cop en la base de dades, a fi de poder analitzar a posteriori si el prototip s'ha comportat de la forma esperada. Això es realitza amb l'execució al final de tot el procés del programa Historial.exe[7] el qual realitza una còpia de seguretat de totes les taules utilitzades. Es guarden per una banda les senyals de control en les taules t_histsec_"type_code"_control (per exemple, en el cas de la posició, seria t_histsec_pos_control) i per l'altre les senyals de captura en les taules t_histsec_"type_code" (per exemple, en el cas de la posició, seria simplement t_histsec_pos). L'estructura de les taules utilitzades en la base de dades historial és la següent:

Camp	Tipus	Descripció
ide	double	Identificació numèric de la senyal. Va de menor a major per ordre de captura.
type_code	varchar(6)	Identificació del tipus de senyal[8].
signal_value_x	double	Valor de la senyal en la coordenada x.
signal_value_y	double	Valor de la senyal en la coordenada y.
signal_value_z	double	Valor de la senyal en la coordenada z.
relative_time	double	Temps relatiu de captura de la senyal des del inici del procés.

A més, al final del procés de tractament, s'ha implementat una funció d'historial que guarda els paràmetres de processament, a fi de poder fer un seguiment dels temps d'execució i poder comprovar que aquests són correctes. Aquesta funció representaria el que seria en el model real l'enviament de dades a través del sistema de telemetria i/o el procés d'emmagatzemar les senyals en la base de dades a bord de l'aparell. Cal destacar però que el sistema de telemetria real actuaria de forma independent al programa principal (tal i com ja s'ha vist anteriorment en la figura 3), mentre que en el prototip s'ha implementat el guardat de dades de l'historial al final de tot el processat.

En aquest cas les taules on es guarden les senyals tenen els camps següents:

Camp	Tipus	Descripció
ide	double	Identificació numèric de la senyal. Va de menor a major per ordre de captura.
ide_c	double	Identificació numèric de la senyal de control associada. Ha de coincidir amb ide si s'ha realitzat el processament correctament.

[7] Adjunt amb el suport digital annexat amb el treball.
[8] Veure annex 4.

type_code	varchar(6)	Identificació del tipus de senyal[9].
signal_value_x	double	Valor de la senyal en la coordenada x.
signal_value_y	double	Valor de la senyal en la coordenada y.
signal_value_z	double	Valor de la senyal en la coordenada z.
relative_time	double	Temps relatiu de captura de la senyal des del inici del procés.
cap_time	varchar(30)	Hora i data de captura de la senyal.
pr_time	varchar(30)	Hora i data de processament de la senyal.
st_time	varchar(30)	Hora i data del guardat de la senyal a la base de dades.

Com es pot veure a la taula anterior a partir de les tres últimes variables podem veure si els temps que passen entre la captura de la senyal, el processat d'aquesta i el posterior guardat es troben entre els límits desitjats. En cas de no ser així el programa permet la variació de certs paràmetres per poder-ho ajustar a les necessitats de cada aparell en concret, ja sigui rebaixant els temps per aconseguir un control més fi o augmentant-los per poder estalviar capacitat de computació.

La nomenclatura d'aquestes taules és *t_hist_"type_code"* (per exemple, pel cas de la posició seria t_hist_pos). En aquest cas només es guarda per la senyal de captura, quedant la identificació de la de control inclosa dintre d'aquesta.

L'emmagatzemen de totes les dades en la base de dades es realitza al final de l'execució del programa, per no interferir així en el procés principal, fet que provocaria una pèrdua de potència de computació. Així, un cop s'ha acabat d'executar, totes les senyals que s'han utilitzat es guarden en la base de dades perquè es puguin consultar a posteriori. Aquest programa no te requeriments de temps reals i no és una aplicació crítica, de forma que no cal que compleixi els requisits que això suposaria.

Per la generació de la base de dades de l'historial s'ha programat un *script* anomenat *CreateTablesHist.sql* que es poden trobar en el suport digital que s'ha annexat junt amb el treball.

[9] Veure annex 4.

3.6. Components

3.6.1. Procés de tractament

Com ja s'ha vist anteriorment el prototip es relaciona constantment amb la base de dades, des de on llegeix les taules de les senyals de captura i de control, guardant les dades processades en l'historial.

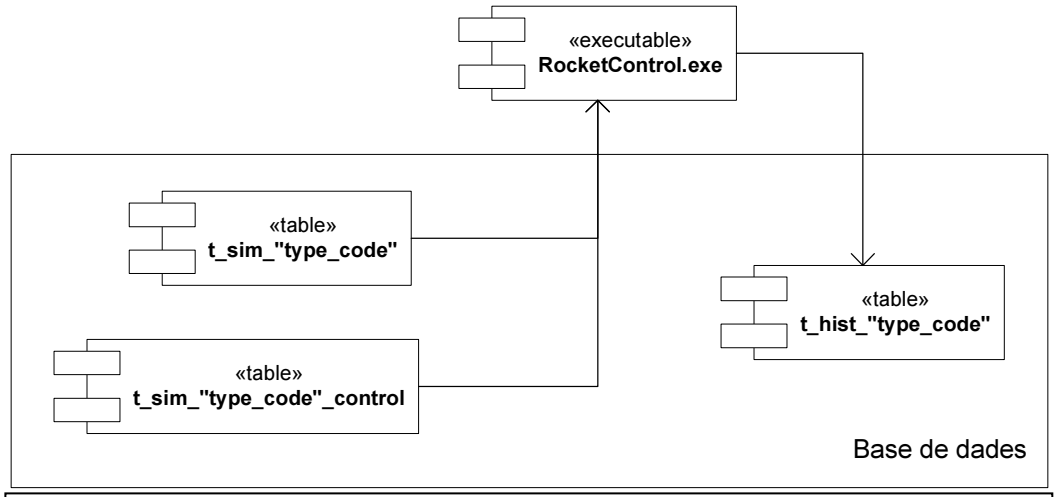


Figura 7. Diagrama de components de la base de dades i el procés de control

Per poder realitzar totes les comandes de comunicació amb la base de dades el programa ha d'utilitzar la llibreria especial lmysql.dll. De la mateixa manera es necessita la biblioteca lpthreadGC2.dll per poder realitzar tota la programació concurrent. Aquestes s'han de trobar indicades a la línia de comandes del enllaçador quan aquest es compila.

El programa es troba estructurat en diversos codis fonts, a fi de simplificar la seva estructura i fer-la més intuïtiva. Això facilita la modificació de parts concretes del programa sense haver de retocar el codi sencer, aconseguint així un software flexible i adaptable a diverses configuracions.

Per aconseguir-ho el software s'ha organitzat en el següents codis:

Main.cpp: És on hi ha escrit el programa principal des del qual es criden i gestionen totes les altres funcions. Bàsicament es troba dividit en tres parts: la primera en la que s'inicialitza tot el sistema i s'estableix la connexió amb la base de dades, la segona que consisteix en un bucle definit que determina la duració del programa i la tercera en la que es finalitzen totes les tasques, es guarda l'historial i es tanca el sistema.

Common.cpp: Aquí és on es defineixen les funcions i variables globals que s'utilitzen en els altres codis, com per exemple les encarregades de obtenir els temps del sistema.

Capture.cpp: Aquest codi conté els algoritmes encarregats de gestionar la captura de les senyals des de la base de dades. En ell es simula el temps d'adquisició de les senyals i es guarden en el buffer utilitzant l'estructura pertinent.

Control.cpp: Aquest codi conté els algoritmes encarregats de gestionar el control de les senyals. En ell es simula el temps de processament degut al llaç de control i es generen les senyals de resposta

Bufferadm.cpp: Aquest codi conté els algoritmes encarregats de gestionar el buffer. És l'encarregat de reordenar-lo quan està ple, esborrant les senyals que ja han estat processades i guardades, podent estalviar així memòria.

Histadm.cpp: Aquest codi conté els algoritmes encarregats de gestionar el guardat de dades en l'historial. S'encarrega de guardar en la base de dades les senyals que ja han estat processades, per poder realitzar a posteriori un anàlisi detallat del funcionament del programa, a fi de millorar-lo i detectar-ne possibles errors.

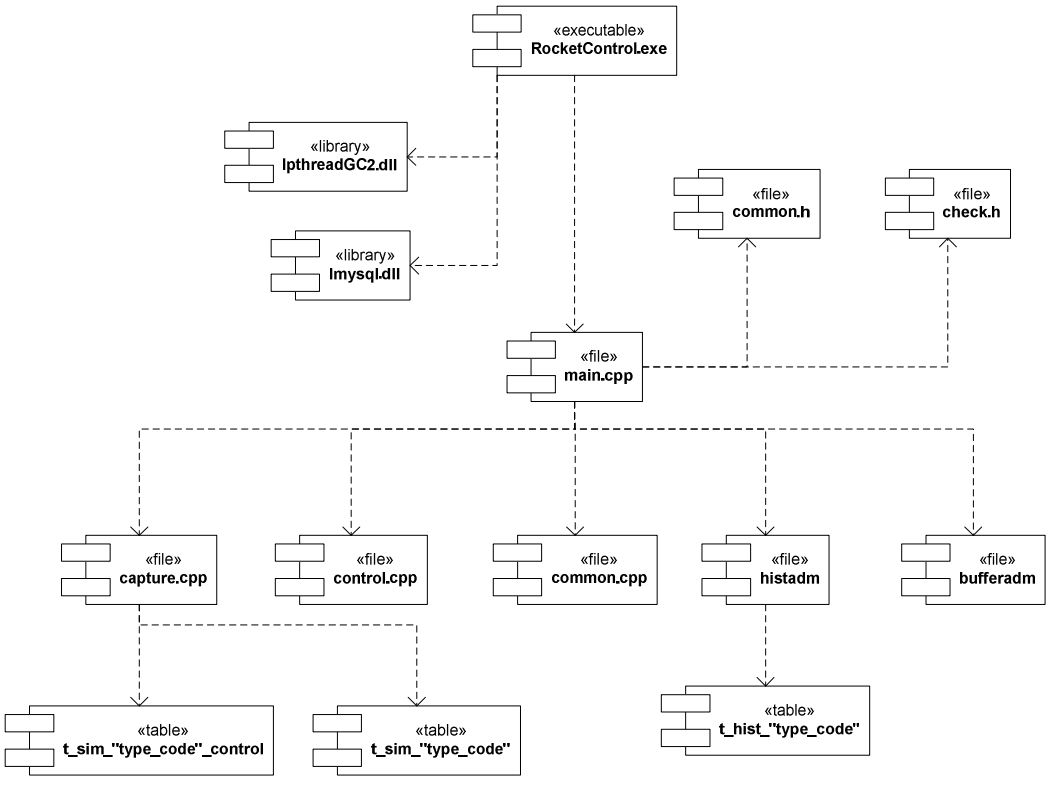


Figura 8. Diagrama de components del procés de tractament

Tots els codis es troben lligats gràcies al arxiu common.h, el qual crida totes les biblioteques i declara les variables i funcions globals que s'utilitzen en els diversos codis.

Finalment comentar que també s'utilitza l'arxiu check.h, el qual inclou les funcions de tractament d'errors. Gràcies a l'ús d'aquest es pot saber quina tasca esta fallant i quin és l'error retornat per aquesta.

3.6.2. Procés de captura

Com ja s'ha comentat aquest procés simula l'adquisició de senyals dels sensors i les guarda en la base de dades, per tant es manté en constant relació amb aquesta:

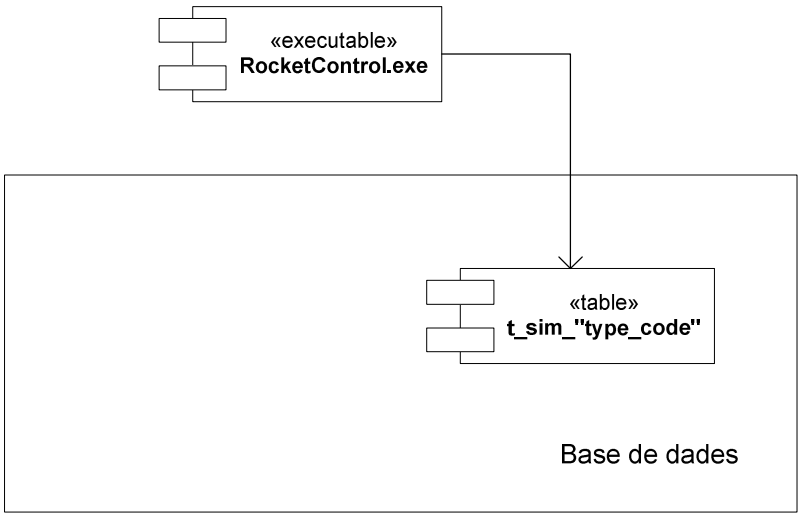


Figura 9. Diagrama de components de la base de dades i el procés de captura

Igual que en el procés anterior, per poder realitzar totes les comandes de comunicació amb la base de dades el programa ha d'utilitzar la llibreria especial lmysql.dll.

L'estructura del programa del procés de captura és relativament senzilla, constant tant sols d'un codi font que executa tota la tasca de simulació tal i com es pot veure en la figura 10.

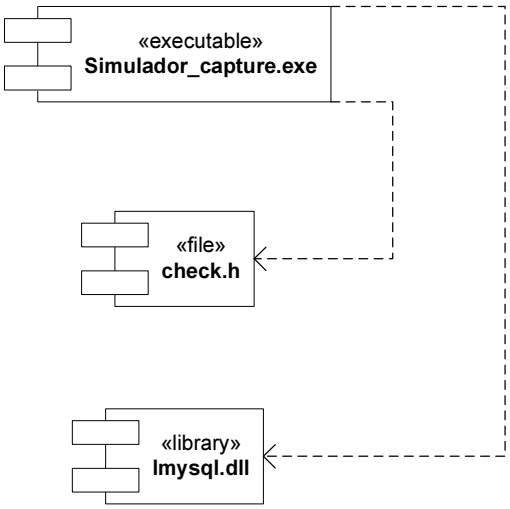


Figura 10. Diagrama de components del procés de captura

S'ha inclòs el fitxer check.h que permet el tractament d'errors per detectar possibles mals funcionaments del programa en cas de necessitat.

3.7. Estructures i variables emprades

Per tal de que el programa s'executi correctament s'ha utilitzat un nombre elevat de variables, ja siguin de control, de senyalització, de seguiment, de magatzem temporal de dades, etc.

A continuació es comentaran tant sols les estructures més significatives. Es pot veure un llistat de totes les variables emprades en el annex 5.

Variables principals del procés de tractament:

Nom:	**signal_cap**	
Tipus:	estructura	
Descripció:	Guarda les senyals capturades des de la base de dades	
Camps:		
ide:	Tipus:	double
	Descripció:	Identificació de la senyal
type_code	Tipus:	Char [6]
	Descripció:	Codi identificatiu del tipus de senyal
signal_value_x	Tipus:	double
	Descripció:	Valor de la primera coordenada de la senyal
signal_value_y	Tipus:	double
	Descripció:	Valor de la segona coordenada de la senyal
signal_value_z	Tipus:	double
	Descripció:	Valor de la tercera coordenada de la senyal
cap_time	Tipus:	Char [30]
	Descripció:	Data i hora en la que la senyal ha estat capturada
pr_time	Tipus:	Char [30]
	Descripció:	Data i hora en la que la senyal ha estat processada
rel_time	Tipus:	double
	Descripció:	Temps relatiu des del inici del processament fins al moment en que s'ha capturat la senyal

Nom:	**signal_control**	
Tipus:	estructura	
Descripció:	Guarda les senyals de control	
Camps:		
ide:	Tipus:	double
	Descripció:	Identificació de la senyal
type_code	Tipus:	Char [6]
	Descripció:	Codi identificatiu del tipus de senyal
signal_value_x	Tipus:	double
	Descripció:	Valor de la primera coordenada de la senyal
signal_value_y	Tipus:	double
	Descripció:	Valor de la segona coordenada de la senyal
signal_value_z	Tipus:	double

	Descripció:	Valor de la tercera coordenada de la senyal
rel_time	Tipus:	double
	Descripció:	Temps relatiu des del inici del processament fins al moment en que s'ha guardat la senyal

Nom:	**buffer3**	
Tipus:	estructura	
Descripció:	Guarda les senyals de control i de captura del buffer	
Camps:		
capture:	Tipus:	signal_cap
	Descripció:	Guarda les senyals capturades
control	Tipus:	signal_control
	Descripció:	Guarda les senyals de control
treated	Tipus:	bool
	Descripció:	Indica si la senyal en qüestió ha estat tractada

Nom:	**hist3**	
Tipus:	estructura	
Descripció:	Guarda les senyals de control i de captura del historial	
Camps:		
capture:	Tipus:	signal_cap
	Descripció:	Guarda les senyals capturades
ide_c:	Tipus:	double
	Descripció:	Identificació de la senyal de control associada
st_time	Tipus:	Char [30]
	Descripció:	Data i hora en el que la senyal ha estat enviada pel sistema de telemetria
sent	Tipus:	bool
	Descripció:	Indica si la senyal en qüestió ha estat enviada

En el procés de control s'inicialitza una estructura buffer3 per cada senyal a tractar, de forma que el maneig d'aquest sigui independent per cada dada, les quals poden tenir diferents requisits temporals de gestió, saturant més o menys ràpidament el buffer en qüestió. La mida d'aquest queda definit per la variable Nbuf[10], la qual pot ser regulada en el fitxer common.h. Està pendent la implementació d'una mida de buffer independent per cada tipus de senyal.

També s'inicialitza una estructura hist3 per cada tipus de senyal. Aquestes serveixen per guardar de forma temporal les dades de l'historial, per al final del programa guardar-les totes de cop en la base de dades. Com ja s'ha comentat la utilització d'aquest mètode permet simular de forma més veraç el sistema real, sense perdre potència de computació degut a l'emmagatzema't de les senyals de forma continua en la base de dades.

[10] Veure annex 5.

Variables principals del procés de captura:

Nom:	**signal_cap**	
Tipus:	estructura	
Descripció:	Guarda les senyals capturades des de la base de dades	
Camps:		
ide:	Tipus:	double
	Descripció:	Identificació de la senyal
type_code	Tipus:	Char [6]
	Descripció:	Codi identificatiu del tipus de senyal
signal_value_x	Tipus:	double
	Descripció:	Valor de la primera coordenada de la senyal
signal_value_y	Tipus:	double
	Descripció:	Valor de la segona coordenada de la senyal
signal_value_z	Tipus:	double
	Descripció:	Valor de la tercera coordenada de la senyal
cap_time	Tipus:	Char [30]
	Descripció:	Data i hora en la que la senyal ha estat capturada
rel_time	Tipus:	double
	Descripció:	Temps relatiu des del inici del processament fins al moment en que s'ha capturat la senyal

En el procés de captura es genera una estructura signal_cap per cada tipus de senyal, doncs cada una tindrà un sensor diferent associat, permetent així el maneig de les dades obtingudes de cada un de forma independent.

3.8. Algoritmes utilitzats en el procés de tractament

En aquest apartat s'exposarà el funcionament bàsic dels diferents algoritmes que s'han implementat en el prototip del procés de tractament a partir bàsicament dels seus diagrames d'activitat.

3.8.1. Programa principal

En primer lloc cal entendre quina és la mecànica global del programa principal i com està organitzat. Es pot veure el diagrama d'activitat d'aquest sistema en la figura 11.

El primer que es realitza és la connexió entre el prototip i la base de dades. S'estableix una connexió per cada tipus de variable, permetent així tractar cada senyal de forma independent a les altres. Per fer-ho s'utilitza el vector myData[i][11] de tipus MYSQL, on i és el numero màxim de senyals a tractar. Així cada dada te assignada una posició segons el seu codi identificatiu[12].

Un cop establerta la connexió s'inicialitzen una sèrie de variables globals:

· Variables semàfors: S'utilitzen per mantenir els fils (cada una de les funcions concurrents) en espera al inici del programa i poder-les adormir i despertar al llarg de l'execució del bucle principal, permetent així el control des del principal. Hi ha un semàfor per cada fil en execució.

· Variables mutex (mutual exclusion): S'utilitzen per protegir les variables globals de forma que dues funcions concurrents no puguin disposar a la vegada d'una mateixa dada, fet que podria comportar errors d'execució del programa, com per exemple la lectura de senyals no actualitzades.

· Buffer: S'inicialitza un buffer per cada senyal, a fi de poder-les gestionar de forma independent. En aquest s'aniran emmagatzemant temporalment les diverses variables de forma que totes les funcions que les requereixin les puguin llegir directament d'ell sense haver d'establir una connexió amb la base de dades cada vegada. Això permet agilitzar l'execució del programa consumint menys recursos.

· Altres variables de control: S'inicialitzen altres variables globals utilitzades per indicar estats de funcions, lectures del buffer, paràmetres processats...

[11] Veure annex 5
[12] Veure annex 4

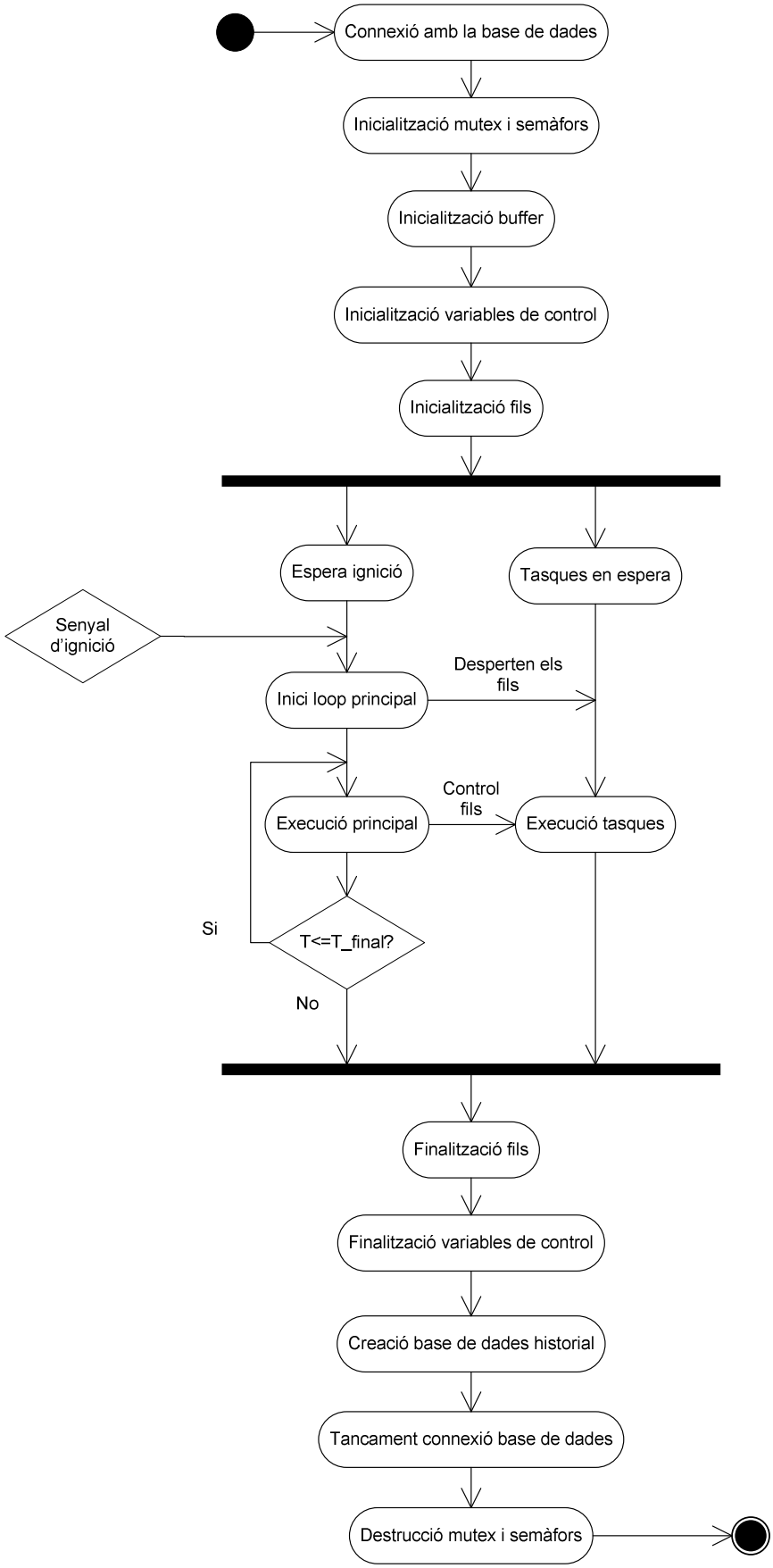


Figura 11. Diagrama d'activitat del programa principal

Un cop s'han inicialitzat totes les variables es creen els diversos fils necessaris per la correcte execució del programa:

· Fils de captura: Es crea un fil per cada variable que s'ha de capturar amb la funció associada de captura pertinent. Aquestes s'encarregaran de simular el temps d'adquisició de les senyals des de la base de dades i d'emmagatzemar-les correctament en el buffer.

· Fils de control: Es crea un fil per cada algoritme de control que s'ha d'executar amb la corresponent funció de control associada. Aquests s'encarregaran de simular el temps de processament del llaç de control així com de realitzar els tractaments de la senyal necessaris pertinents per cada funció.

· Fils de resposta: Es crea un fil per cada senyal de resposta que s'ha de gestionar amb la corresponent funció associada. Aquests s'encarregaran de l'adquisició de les senyals de resposta produïdes en el llaç de control i el lliurement d'aquestes en els respectius actuadors.

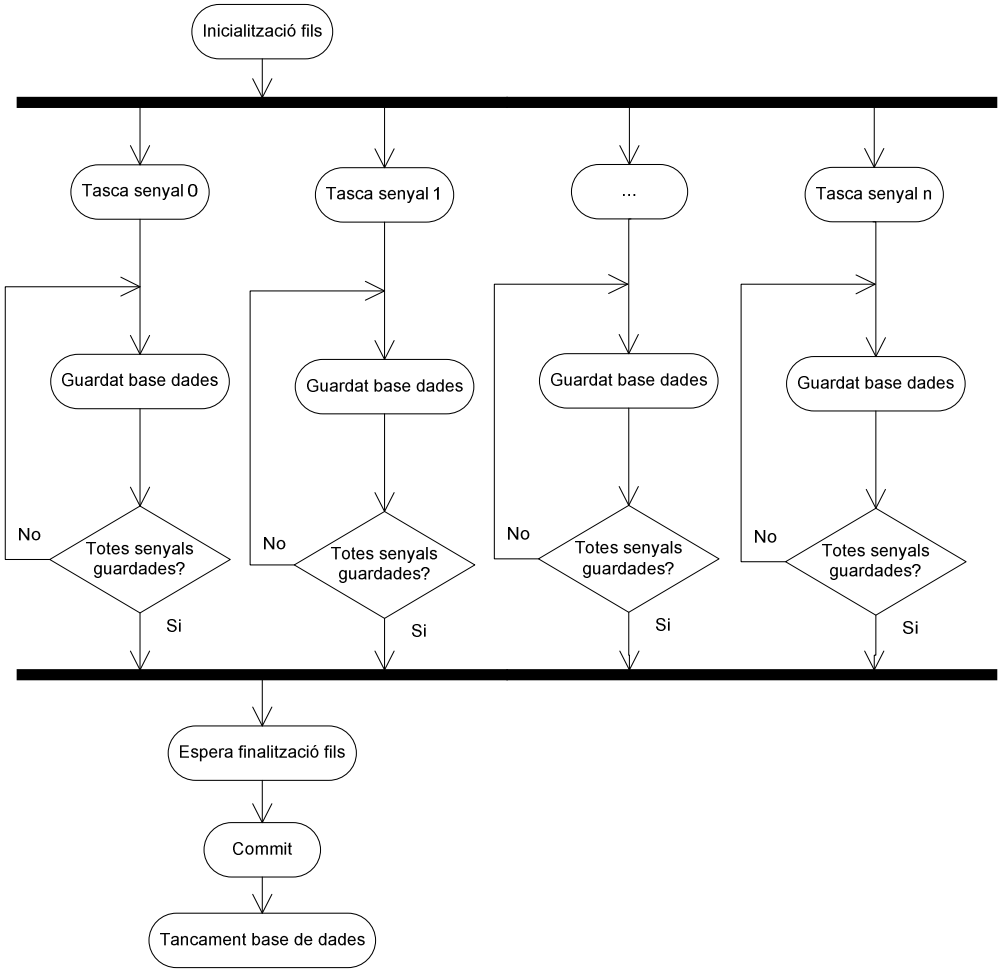


Figura 12. Diagrama d'activitat del procés de guardat de l'historial

Amb la utilització dels semàfors es mantenen totes les funcions adormides fins que l'usuari envia la senyal d'ignició. A partir d'aquí es comença a executar tot el programa fins que s'arriba al temps final predeterminat, que marcaria la duració de la missió real.

Quan s'arriba al temps final d'execució s'espera a la finalització de totes les funcions. A continuació es procedeix a guardar en la base de dades totes les senyals de les taules d'historial temporal. S'aprofita la mateixa connexió generada per la lectura de les dades de captura i control. Per agilitzar el procés és realitza de forma concurrent amb un fil per cada tipus de senyal. Un cop s'acaba el procés es tanca la connexió amb la base de dades. Es pot veure el diagrama d'activitat del procés en la figura 12.

Finalment es procedeix a la destrucció de les variables mutex i semàfor, per finalitzar tot seguit el programa.

Cal esmentar també que la connexió amb la base de dades s'estableix amb la opció de *autocommit* desactivada. Aquesta opció fa que el procés sigui més ràpid permetent emmagatzemar totes les dades de cop enlloc d'una per una. Això es fa al final del programa quan les condicions de processament ja no són crítiques, dintre del procés de l'historial.

3.8.2. Tasca de captura

Al inici del programa principal es genera un fil de captura per cada una de les senyals que cal administrar. Aquests fils simulen l'adquisició de senyals obtingudes des de la base de dades de bord i en fan la seva administració.

La funció s'engega quan és despertada pel semàfor vinculat amb el tipus de senyal. Això permet controlar totes les tasques de captura des del programa principal, sobretot pensant en la futura implantació d'un sistema d'administració de prioritats.

El primer que fa és llegir des de la base de dades una certa quantitat de senyals de control a la vegada, optimitzant així aquest procés, guardant-les en el format genèric de l'estructura *signal_control*. A continuació la funció va adquirint les senyals de captura, simulant un cert temps d'espera dependent de la freqüència de mostreig de cada sensor. Aquestes les guarda en el format genèric de l'estructura *signal_cap*.

Després de la captura de cada senyal la funció retorna el control al programa principal. Aquest procés es repeteix fins que el buffer està ple, moment en el que s'engega la funció d'administració del buffer per a continuació tornar a agafar un

nou paquet de dades de control, iniciant de nou el procés. Aquest es va repetint fins que el programa arriba a la seva fi.

El fil de captura disposa d'una estructura switch que permet la selecció de la funció adequada depenent del tipus de dada que s'està tractant. Això permet treure i afegir senyals de forma relativament fàcil sense haver de modificar l'estructura principal, aconseguint així un software flexible i personalitzable a cada sistema que s'hagi d'implementar.

Finalment comentar que sempre que es modifica el buffer des d'aquesta funció es bloqueja la variable mutex associada per assegurar que cap altre fil n'està fent ús de forma indeguda.

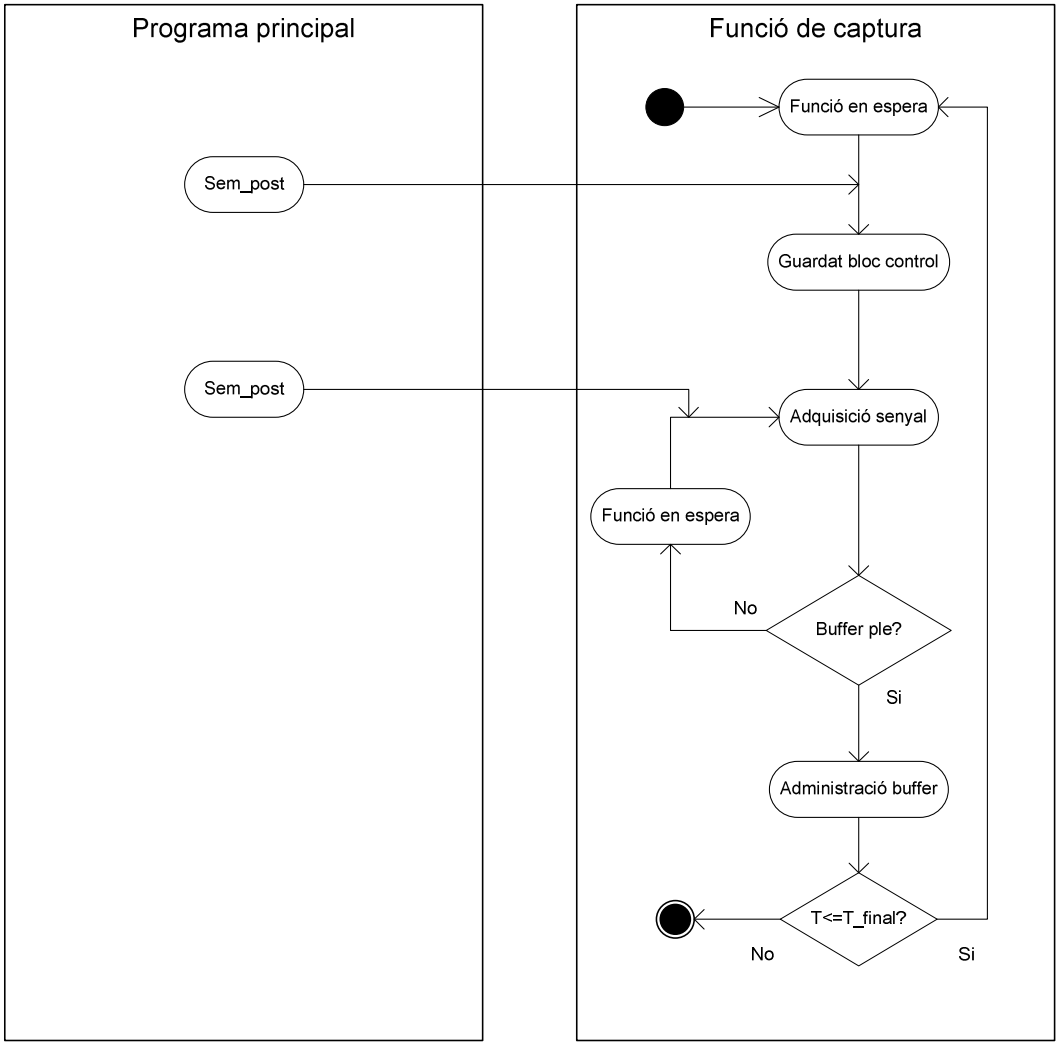


Figura 13. Diagrama d'activitat del procés de captura

3.8.3. Administració del buffer

Com ja s'ha comentat, es treballa amb un buffer limitat, doncs la gran quantitat de dades que pot arribar a administrar el sistema real obligaria a disposar d'una memòria enorme i impracticable per poder guardar totes les senyals. Per tant es necessari disposar d'una funció que administri el buffer, reordenant-lo i esborrant els paràmetres ja processats quan aquest està ple.

La funció es crida des del fil de captura quan aquest detecta que el buffer té emmagatzemats un cert nombre de dades predeterminat. De forma resumida, el que fa l'algoritme és, en primer lloc, guardar totes les dades que encara no han estat processades en una taula temporal, per a continuació reescriure-les per ordre començant en la primera posició del buffer, eliminant les senyals ja processades. Finalment canvia el valor de les senyals de control de posició del buffer[13] perquè s'adeqüin amb la nova configuració d'aquest.

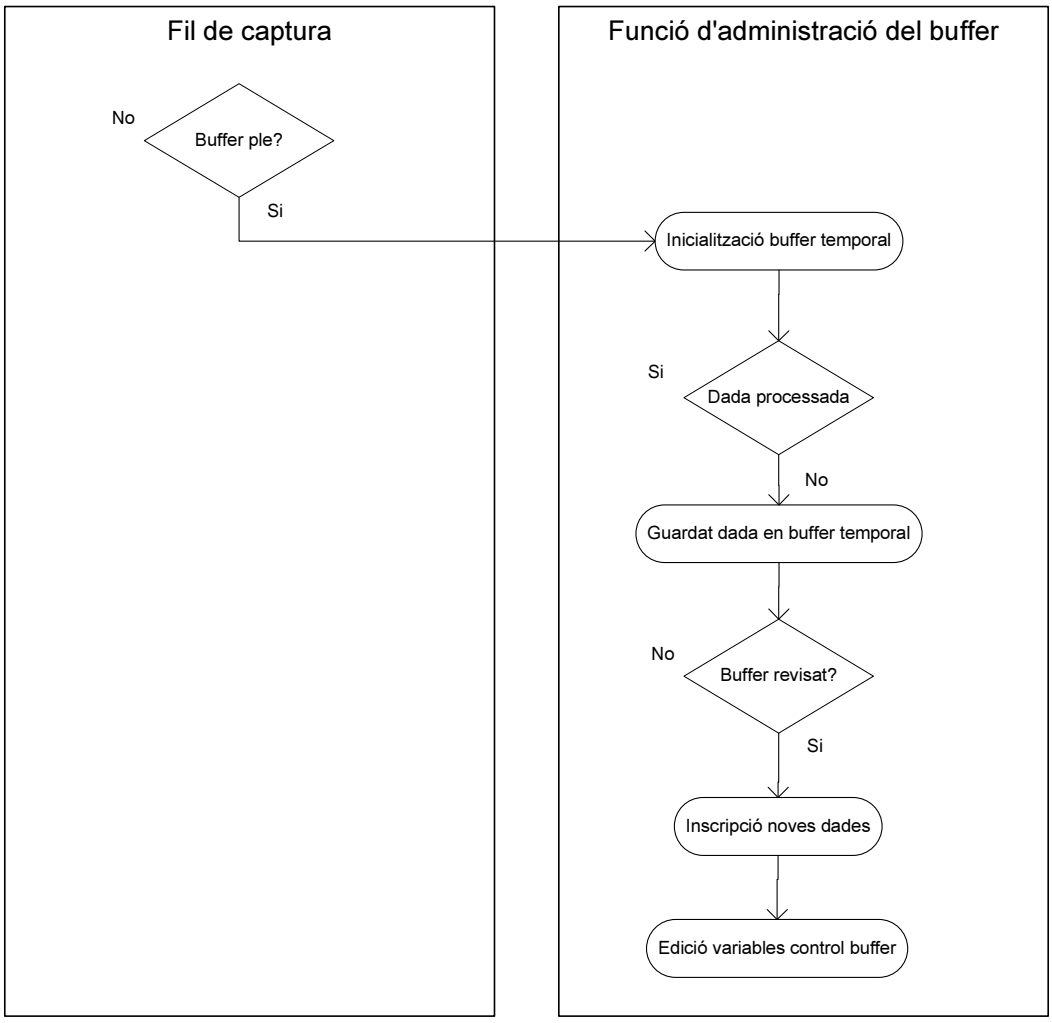


Figura 14. Diagrama d'activitat del procés d'administració del buffer

[13] Veure annex 8.

Com es pot veure el conjunt dels processos realitzats entre la tasca de captura i l'administració del buffer són les que acaben realitzant el processat de les senyals explicat en el apartat *3.4.2. Procés de tractament de senyals*.

3.8.4. Tasca de control

Al inici del programa principal es genera un fil de control per cada una de les senyals que cal administrar. Aquests fils simulen el processament dels algoritmes de control i la generació de les respostes pertinents.

La funció s'engega quan és despertada pel semàfor vinculat amb el tipus de senyal. Això permet controlar totes les tasques de captura des del programa principal, sobretot pensant en la futura implantació de un sistema d'administració de prioritats. Actualment el sistema dóna permís d'execució de forma immediata.

El programa llegeix les senyals directament del buffer i les guarda en variables internes dins de cada funció. D'aquesta forma el buffer es manté bloquejat el mínim de temps possible, podent executar així el llaç de control de forma independent a la resta del programa, ja que cada cop que la funció llegeix una senyal del buffer s'ha de bloquejar la variable mutex associada perquè cap altre fil la utilitzi de forma indeguda. Un cop la dada ha estat guardada en el format necessari es crida a la funció d'historial, que simula l'actualització dels diferents paràmetres en la base de dades.

Com que els algoritmes propis de control (tipus PID per exemple) no han estat implementats, doncs com s'ha comentat anteriorment la programació de cada un d'ells podria ser un Projecte Final de Carrera en si, el que es fa es simular un cert temps de processament. Aquest temps s'ha fixat en el doble que el d'adquisició de les senyals.

Quan acaba el llaç de control es torna a bloquejar el buffer per emmagatzemar les senyals de resposta corresponents, les qual són enviades als actuadors pertinents.

S'ha implementat una estructura tipus switch de manera que es pot canviar l'algoritme de cada senyal per separat sense haver de modificar l'estructura general del programa.

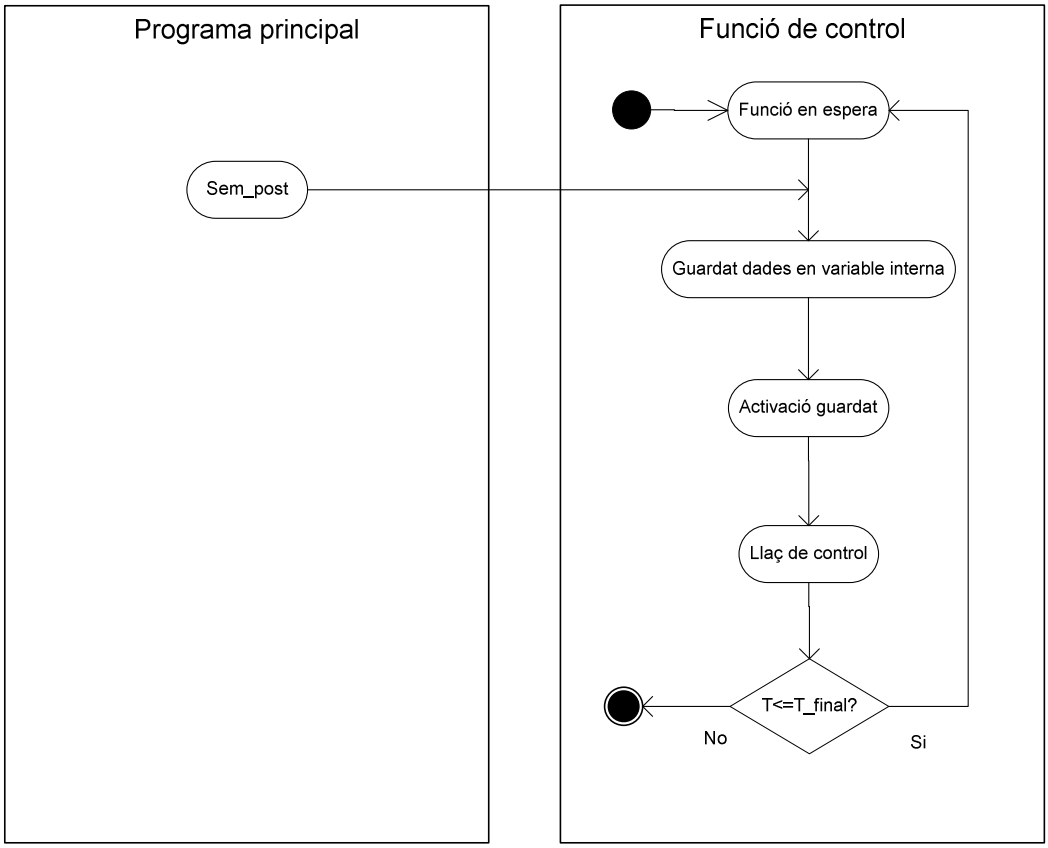


Figura 15. Diagrama d'activitat de l'algoritme de control

3.8.5. Historial

El que fa aquesta funció es guardar les senyals desitjades en una taula provisional abans d'emmagatzemar totes les dades en la base de dades històrica al final del programa. Com ja s'ha explicat amb anterioritat s'utilitza aquest sistema per simular de la forma més veraç possible el sistema real.

Igual que en els casos anteriors s'ha implementat una estructura tipus switch per poder afegir o treure senyals sense haver de modificar l'estructura general del programa.

3.9. Algoritmes utilitzats en el procés de captura

En aquest apartat, de la mateixa manera que s'ha fet en l'anterior, s'exposarà el funcionament bàsic dels diferents algoritmes que s'han implementat en el prototip del procés de captura.

Es pot veure el diagrama d'activitat de l'algoritme utilitzat en la figura 16 en la pàgina següent.

En primer lloc el programa inicialitza totes les variables necessàries per la seva execució, les més destacades de les quals són:

· Taules de les senyals: permeten emmagatzemar de forma temporal en una estructura signal_cap els valors simulats adquirits dels sensors. En cada iteració del programa aquestes es modifiquen un cop han estat degudament guardades en la base de dades.

· Cadenes de caràcters per la comunicació amb la base de dades: aquestes permeten crear de forma automàtica les consignes que interactuen amb la base de dades a fi de poder guardar de forma automàtica les senyals en aquesta.

· Constants de simulació: són les que defineixen les condicions de vol. És important que coincideixin amb les utilitzades en el programa simulador que genera les senyals de control[14]. Aquestes constants poden ser modificades per obtenir diferents comportaments de missió.

A continuació es realitza la connexió amb la base de dades, a través d'una estructura tipus MYSQL anomenada MyData. Aquesta connexió s'estableix amb la opció de *autocommit* desactivada. Aquesta opció fa que el procés sigui més ràpid permetent emmagatzemar totes les dades de cop enlloc d'una per una.

En aquest moment el programa ja es troba inicialitzat i entra en un bucle definit que acaba quan s'arriba al temps final d'execució del programa.

En aquest el programa va simulant de forma continuada l'adquisició de les diverses senyals capturades a partir de les senyals de control, afegint un error de mesura que l'usuari pot personalitzar segons el tipus de senyal i les necessitats del sistema. Les dades es guarden primer en les taules de senyals temporals, que s'utilitzen per executar tot el procés de simulació, el qual és anàleg a l'utilitza't en el simulador de control[15].

[14] simulador_control.exe
[15] Veure annex 3 per més informació

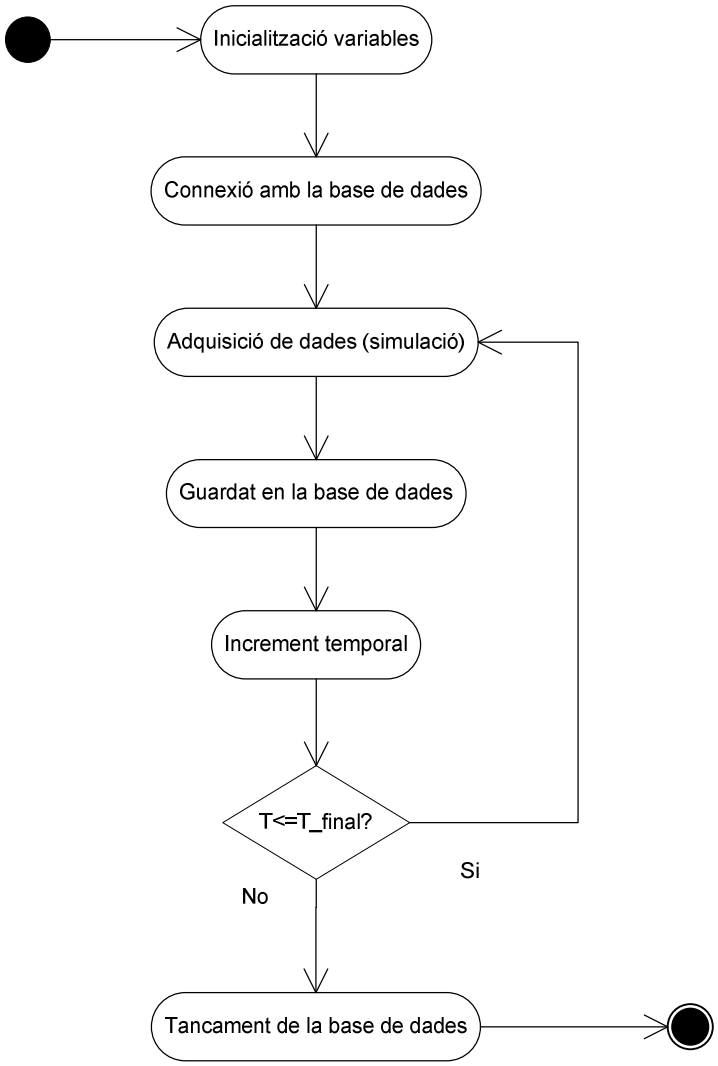


Figura 16. Diagrama d'activitat del procés de captura

Per tal de realitzar tot el procés de simulació el programa té implementades una sèrie de funcions que permeten el tractament directe de les diverses matrius i rotacions que s'han de realitzar en els canvis de coordenades.

Quan el programa arriba al temps final d'execució es finalitza el llaç principal. Aquest moment ve determinat per una constant inicial que l'usuari pot modificar segons les necessitats del seu sistema.

A continuació el programa procedeix al tancament de la connexió amb la base de dades, moment en què es realitza la funció de *commit* perquè quedin així tots els paràmetres enregistrats.

El programa disposa d'una estructura tipus switch que permet afegir i treure senyals de forma senzilla. Així el codi pot ser fàcilment personalitzat segons la plataforma on hagi de ser utilitzat, obtenint un codi flexible.

4. Resultats

La principal eina utilitzada per poder comprovar la correcte execució del prototip ha estat la base de dades de l'historial i les comandes impreses per pantalla que serien el seu equivalent.

La comprovació s'ha realitzat sobre la variable de posició, però per cap motiu en especial, podent haver estat realitzada sobre una altra senyal. S'ha establert en la simulació un període de mostreig de 100 ms i per tant, el temps de processament del llaç de control és de 200 ms (doncs s'estableix com el doble).

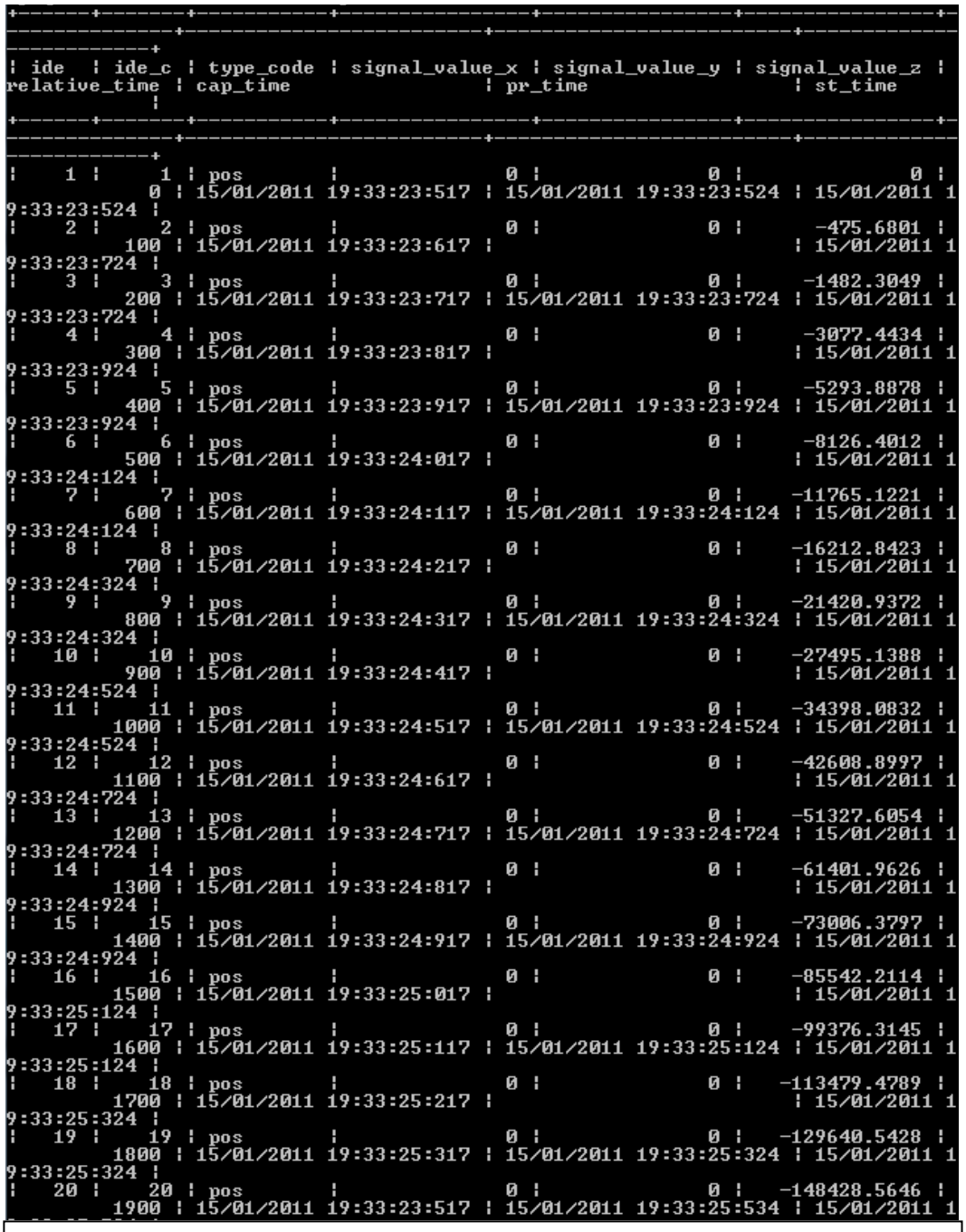
En la figura 17 es presenten les dades recollides en una execució. Per facilitar-ne la lectura s'ha elaborat la següent taula resumida amb les primeres mostres obtingudes:

ide	ide_c	type_code	cap_time	pr_time	st_time
1	1	pos	19:33:23:517	19:33:23:524	19:33:23:524
2	2	pos	19:33:23:617		
3	3	pos	19:33:23:717	19:33:23:724	19:33:23:724
4	4	pos	19:33:23:817		
5	5	pos	19:33:23:917	19:33:23:924	19:33:23:924
6	6	pos	19:33:24:017		
7	7	pos	19:33:24:117	19:33:24:124	19:33:24:124
8	8	pos	19:33:24:217		
9	9	pos	19:33:24:317	19:33:24:324	19:33:24:324
10	10	pos	19:33:24:417		

En primer lloc es pot veure directament que totes les dades queden classificades com a tipus pos (posició)[16]. Es confirma que un del requeriments, que era el fet de poder identificar sempre el tipus de senyal, es compleix. Això permet a l'usuari buscar dades pel seu tipus fàcilment

Per altra banda s'observa que l'identificador de la senyal capturada (ide) y el de la senyal de control (ide_c) coincideixen. Això indica que el procés de tractament de senyals[17] s'ha executat correctament, agrupant cada senyal de captura amb el seu homòleg de control, és a dir, les que tenen el mateix temps relatiu. Per tant es pot concloure que el tractament de la senyal s'ha executat de forma correcte.

[16] Veure Annex 4
[17] Descrit en el apartat *3.4.2. Procés de tractament de senyals*



ide	ide_c	type_code	signal_value_x	signal_value_y	signal_value_z	relative_time	cap_time	pr_time	st_time
1	1	pos	0	0	0	0	15/01/2011 19:33:23:517	15/01/2011 19:33:23:524	15/01/2011 19:33:23:524
2	2	pos	0	0	-475.6801	100	15/01/2011 19:33:23:617		15/01/2011 19:33:23:724
3	3	pos	0	0	-1482.3049	200	15/01/2011 19:33:23:717	15/01/2011 19:33:23:724	15/01/2011 19:33:23:724
4	4	pos	0	0	-3077.4434	300	15/01/2011 19:33:23:817		15/01/2011 19:33:23:924
5	5	pos	0	0	-5293.8878	400	15/01/2011 19:33:23:917	15/01/2011 19:33:23:924	15/01/2011 19:33:23:924
6	6	pos	0	0	-8126.4012	500	15/01/2011 19:33:24:017		15/01/2011 19:33:24:124
7	7	pos	0	0	-11765.1221	600	15/01/2011 19:33:24:117	15/01/2011 19:33:24:124	15/01/2011 19:33:24:124
8	8	pos	0	0	-16212.8423	700	15/01/2011 19:33:24:217		15/01/2011 19:33:24:324
9	9	pos	0	0	-21420.9372	800	15/01/2011 19:33:24:317	15/01/2011 19:33:24:324	15/01/2011 19:33:24:324
10	10	pos	0	0	-27495.1388	900	15/01/2011 19:33:24:417		15/01/2011 19:33:24:524
11	11	pos	0	0	-34398.0832	1000	15/01/2011 19:33:24:517	15/01/2011 19:33:24:524	15/01/2011 19:33:24:524
12	12	pos	0	0	-42608.8997	1100	15/01/2011 19:33:24:617		15/01/2011 19:33:24:724
13	13	pos	0	0	-51327.6054	1200	15/01/2011 19:33:24:717	15/01/2011 19:33:24:724	15/01/2011 19:33:24:724
14	14	pos	0	0	-61401.9626	1300	15/01/2011 19:33:24:817		15/01/2011 19:33:24:924
15	15	pos	0	0	-73006.3797	1400	15/01/2011 19:33:24:917	15/01/2011 19:33:24:924	15/01/2011 19:33:24:924
16	16	pos	0	0	-85542.2114	1500	15/01/2011 19:33:25:017		15/01/2011 19:33:25:124
17	17	pos	0	0	-99376.3145	1600	15/01/2011 19:33:25:117	15/01/2011 19:33:25:124	15/01/2011 19:33:25:124
18	18	pos	0	0	-113479.4789	1700	15/01/2011 19:33:25:217		15/01/2011 19:33:25:324
19	19	pos	0	0	-129640.5428	1800	15/01/2011 19:33:25:317	15/01/2011 19:33:25:324	15/01/2011 19:33:25:324
20	20	pos	0	0	-148428.5646	1900	15/01/2011 19:33:23:517	15/01/2011 19:33:25:534	15/01/2011 19:33:25:534

Figura 17. Registres de la base de dades de l'historial

Si s'observa la columna del temps de captura (cap_time) es pot comprovar que les senyals efectivament han estat capturades cada 100 ms, tal i com s'havia fixat el període de mostreig.

La següent columna fa referència al temps de processament de les senyals. S'observa que la diferència de temps entre el moment de captura i el de processament es tant sols de 7 ms, fet que indica que quan el llaç de control inicia la seva execució disposa d'una senyal que ha estat capturada fa tant sols 7

ms. Si es compara aquest valor amb el fixat de processament de 200 ms, es pot veure que la relació és molt petita, sent per tant un valor acceptable. Cal remarcar que els 7ms ha estat la pitjor de les marques observades, estant el temps comprès normalment entre 3ms i 7ms de retard. La conclusió és que la sincronització entre la captura i el control de les senyals s'executa de forma adient.

Pot resultar estrany a primera vista el veure que la meitat de les dades no han estat processades. Això és degut al simple fet que el llaç de control de processament és el doble de lent que el d'adquisició de dades, de forma que per cada procés que ha fet l'algoritme de control s'han obtingut dos nous registres, dels quals només se'n aprofita un. En tot cas no s'ha de pensar que aquests valors no caldria agafar-los, doncs poden ser utilitzats en el cas de tenir implementat un filtre Kalman.

Finalment es pot observar com l'actualització de la informació en la base de dades pel posterior guardat en l'historial es realitza pràcticament de forma immediata al processament de la senyal.

Així doncs, a partir dels resultats es pot concloure que el processat desitjat ha estat implementat de forma correcte i sincronitzada.

En el suport digital annexat amb el treball es poden trobar tots els executables comentats, així com els scripts necessaris per la base de dades. En el annex 6 hi ha especificat el procés que cal seguir per executar el prototip. En aquest s'hi ha implementat una sèrie de impressions de pantalla que permeten seguir tot el procés en temps real.

5. Conclusions

Com ja es desprenia en l'apartat anterior, s'ha aconseguit resoldre els principals requeriments:

· Execució en temps real: S'han implementat fils concurrents per les tasques principals (control i captura) de forma que cada un dels llaços de control així com l'adquisició de les senyals es pugui realitzar de forma independent i a la vegada.

· Protecció de variables. Gràcies a l'ús de variables mutex (mutal exclusion) s'ha protegit el buffer de forma que no pugui ser actualitzat per més d'una funció a la vegada.

· Coordinació de funcions concurrents. Amb l'ús de semàfors totes les funcions són controlades des del programa principal. A més hi ha diferents variables de control que permeten coordinar els processos.

· Codi flexible i personalitzable. Gràcies a l'ús d'estructures switch en una base genèrica permet modificar el programa afegint o treien senyals sense haver de modificar tot el programa.

· Optimització de la memòria. Utilitzant un buffer limitat i una funció per administrar-lo ha permès minimitzar la memòria necessària per executar correctament el programa.

· Anàlisis i verificació del prototip. Gràcies a la creació d'una base de dades historial on es guarden totes les senyals es pot procedir a analitzar a posteriori si el software s'ha executat correctament.

· Base de dades ordenada. S'ha dotat a totes les variables d'un nom que les defineix, un identificador que les ordena i el temps relatiu en què han estat preses, permetent classificar les variables de diverses formes segons sigui necessari.

· Tractament d'errors. S'han establert diferents controls que permeten identificar els errors que s'han produït a fi de reparar-los.

Tot i complir els requeriments plantejats al inici, encara hi ha una sèrie se punts que es podrien millorar per disposar d'un prototip encara més adequat:

· Sistema de prioritats. Implementar un sistema que administri de forma intel·ligent les senyals segons quines siguin les condicions de vol i/o estat del processador, donant més prioritat a aquelles tasques més crítiques quan la

capacitat de processament és limitada o quan s'està produint algun error de funcionament en el sistema.

· Sistema de tractament d'errors intel·ligent. Actualment el tractament d'errors requereix de l'acció humana per corregir-se. En una següent fase es podria implementar algoritmes de tractament d'errors automàtics que permetessin al programa corregir els seus propis errors.

· Coordinació real entre els dos programes principals. Actualment la sincronització entre els dos programes principals es simulada a fi de simplificar la seva relació. Es podria buscar la forma d'establir una sincronització real amb la finalitat de millorar el processament d'aquest.

Finalment cal comentar que el següent pas lògic per avançar el projecte seria la seva implementació en un sistema real. Començar directament amb un coet podria ser complicat, sobretot degut al cost econòmic que suposaria i al fet que es un sistema amb moltes senyals per administrar.

Així doncs, abans de implementar el resultat en un coet espacial, seria convenient provar-ho en alguna plataforma més senzilla i econòmica, com podria ser un UAS. Això permetria la realització de múltiples proves del sistema programat sense un cost econòmic alt, al ser aquesta una plataforma recuperable. Per tal de guiar lleugerament aquest següent pas, s'ha inclòs tant en el pressupost com en el plec de condicions (adjunts en aquest treball) referències a aquesta opció.

6. Bibliografia

[1] George P. Sutton, Oscar Biblarz. Rocket Propulsion Elements. 7th edition, Wiley Interscience (2001).

[2] Varis autors. Space Mission Analysis and design. 3th edition, Space Technology Library (2007).

[3] Varis autors. Spacecraft Systems Engineering. 3th edition, Wiley (2003).

[4] Ed Chester. Apunts de l'assignatura Vehicles espacials I. ETSEIAT 2009.

[5] Ed Chester. Apunts de l'assignatura Vehicles espacials II. ETSEIAT 2010.

[6] Josefina Lopez. Apunts de l'assignatura Software crític en temps real. ETSEIAT 2009.

[7] Pàgina web MySQL. http://www.mysql.com/ (últim accés Desembre 2010).

[8] Pàgina web Bloodshed Software. http://www.bloodshed.net/devcpp.html (últim accés Desembre 2010).

[9] MySQL 5.0 Reference Manual (2010).

[10] William F. Riley, Leroy D. Sturges. Ingeniería mecànica, Dinámica. Editorial Reverté (2001).

[11] Miquel Sureda. Apunts de l'assignatura Mecànica de vol. ETSEIAT (2008).

[12] Pàgina web noticia CNES sobre el projecte Perseus, http://www.cnes.fr/web/CNES-fr/4652-perseus-la-creativite-des-jeunes-au-profit-des-technologies-spatiales.php (últim accés Desembre 2010).

[13] Pàgina web noticia CNES sobre el projecte Perseus, http://www.cnes.fr/web/CNES-en/7134-press-releases.php?item=1313 (últim accés Desembre 2010).

[14] Entrevista a Javier Ventura-Traveset (Director de la Oficina de Comunicació i Educació de la ESA a Espanya) sobre la 1a xarxa mundial de nanosatèl·lits liderada des de Vigo, http://www.farodevigo.es/gran-vigo/2010/03/20/red-nanosatelites-estudiantes-aplicaciones-empresariales/421901.html (últim accés Desembre 2010).

[15] José Torres Riera. Nanosatélites, microsatélites y minisatélites: una alternativa de futuro. Resum conferència.